Oracle® Solaris 11.1 の管理: Oracle Solaris ゾーン、Oracle Solaris 10 ゾーン、およびリソース管理

Part No: E36684–04
2013 年 5 月

ORACLE®

130606@25097

目次

はじめに 23

パート I **Oracle Solaris** のリソース管理 27

1 リソース管理の紹介 29
リソース管理の概要 29
リソースの分類 30
リソース管理の制御メカニズム 31
リソース管理構成 32
非大域ゾーンとの相互動作 32
リソース管理機能を使用する場合 33
サーバーを統合する場合 33
大規模で多様なユーザーが利用するシステムをサポートする場合 34
リソース管理の設定 (タスクマップ) 34

2 プロジェクトとタスク (概要) 37
プロジェクトとタスクの機能 37
プロジェクト識別子 38
ユーザーのデフォルトプロジェクトの判定 38
`useradd` および `usermod` コマンドによるユーザー属性の設定 39
`project` データベース 40
PAM サブシステム 40
ネームサービス構成 40
ローカルの `/etc/project` ファイルの形式 41
NIS のプロジェクト構成 43
LDAP のプロジェクト構成 43
タスク識別子 44

プロジェクトとタスクで使用するコマンド 45

3 プロジェクトとタスクの管理 47
プロジェクトとタスクの管理 (タスクマップ) 47
コマンドとコマンドオプションの例 48
プロジェクトとタスクで使用するコマンドオプション 48
プロジェクトとタスクでの cron と su の使用 50
プロジェクトの管理 51
▼ プロジェクトを定義して現在のプロジェクトを表示する方法 51
▼ /etc/project ファイルからプロジェクトを削除する方法 53
/etc/project ファイルの内容を検証する方法 54
プロジェクトのメンバーシップ情報を取得する方法 54
▼ 新しいタスクを作成する方法 55
▼ 実行中のプロセスを新しいタスクに移動する方法 55
プロジェクト属性の編集と検証 56
▼ 属性と属性値をプロジェクトに追加する方法 56
▼ 属性値をプロジェクトから削除する方法 56
▼ リソース制御属性をプロジェクトから削除する方法 57
▼ プロジェクトの属性と属性値を置換する方法 57
▼ リソース制御属性の既存の値を削除する方法 57

4 拡張アカウンティング (概要) 59
拡張アカウンティングの紹介 59
拡張アカウンティングの動作 60
拡張可能な形式 61
exacct レコードとその形式 61
ゾーンがインストールされている Oracle Solaris システムでの拡張アカウンティングの使用 62
拡張アカウンティング構成 62
拡張アカウンティングの起動および持続的な有効化 62
レコード 63
拡張アカウンティングで使用されるコマンド 63
libexacct に対する Perl インタフェース 64

5 拡張アカウンティングの管理 (タスク) 67
拡張アカウンティング機能の管理 (タスクマップ) 67
拡張アカウンティング機能の使用 68
▼ フロー、プロセス、タスク、およびネットワークコンポーネントの拡張アカウンティングを起動する方法 68
拡張アカウンティングステータスを表示する方法 69
使用可能なアカウンティングリソースを表示する方法 69
▼ プロセス、タスク、フロー、およびネットワーク管理のアカウンティングを停止する方法 70
libexacct に対する Perl インタフェースの使用 71
exacct オブジェクトの内容を再帰的に出力する方法 71
新しいグループレコードを作成してファイルに書き込む方法 73
exacct ファイルの内容を出力する方法 73
Sun::Solaris::Exacct::Object->dump() からの出力例 74

6 リソース制御 (概要) 75
リソース制御の概念 75
リソース制限とリソース制御 76
プロセス間通信とリソース制御 76
リソース制御の制約メカニズム 77
プロジェクトの属性メカニズム 77
リソース制御と属性の構成 77
使用可能なリソース制御 78
ゾーン規模のリソース制御 81
単位のサポート 83
リソース制御値と特権レベル 85
リソース制御値に対応付けられた大域アクションと局所アクション 85
リソース制御のフラグとプロパティー 87
リソース制御の実行 89
リソース制御イベントの大域監視 89
リソース制御の適用 90
動作中のシステム上のリソース制御値を一時的に更新する 90
ログステータスの更新 90
リソース制御の更新 90
リソース制御で使用するコマンド 91

7 リソース制御の管理 (タスク) 93
リソース制御の管理 (タスクマップ) 93
リソース制御の設定 94
▼ プロジェクト内の各タスクの最大 LWP 数を設定する方法 94
▼ プロジェクトに複数の制御を設定する方法 95
prctl コマンドの使用 96
▼ prctl コマンドを使ってデフォルトのリソース制御値を表示する方法 96
▼ prctl コマンドを使って特定のリソース制御の情報を表示する方法 99
▼ prctl を使って値を一時的に変更する方法 99
▼ prctl を使ってリソース制御値を下げる方法 100
▼ prctl を使ってプロジェクトの制御値を表示、置換、確認する方法 100
rctladm の使用 100
rctladm を使用する方法 100
ipcs の使用 101
ipcs を使用する方法 101
容量に関する警告 102
▼ Web サーバーに十分な CPU 容量が割り当てられているかどうかを判定する方法 102

8 公平配分スケジューラ (概要) 103
スケジューラの紹介 104
CPU 配分の定義 104
CPU 配分とプロセスの状態 105
CPU 配分と使用効率 105
CPU 配分の例 106
例 1: CPU にバインドされた 2 つの プロセスが各プロジェクトに存在する場合 106
例 2: プロジェクト間に競合がない場合 107
例 3: 一方のプロジェクトが実行されない場合 107
FSS の構成 108
プロジェクトとユーザー 108
CPU 配分の構成 108
FSS とプロセッサセット 110
FSS とプロセッサセットの例 110
FSS とほかのスケジューリングクラスの併用 112
システムのスケジューリングクラスの設定 113

ゾーンがインストールされているシステムでのスケジューリングクラス 113
FSS で使用するコマンド 113

9 公平配分スケジューラの管理 (タスク) 115
公平配分スケジューラの管理 (タスクマップ) 115
FSS の監視 116
▼ システムの CPU 使用量をプロジェクトごとに監視する方法 116
▼ プロセッサセット内の CPU 使用量をプロジェクトごとに監視する方法 117
FSS の構成 117
システム上のスケジューラクラスのリスト表示 117
▼ FSS をデフォルトのスケジューラクラスにする方法 118
▼ プロセスを TS クラスから FSS クラスに手動で移動する方法 118
▼ プロセスをすべてのユーザークラスから FSS クラスに手動で移動する方法 119
▼ プロジェクトのプロセスを FSS クラスに手動で移動する方法 119
スケジューラのパラメータを調整する方法 119

10 リソース上限デーモンによる物理メモリーの制御 (概要) 121
リソース上限デーモンの紹介 121
リソース上限制御のしくみ 122
プロジェクトの物理メモリーの使用率を制限する属性 122
rcapd の構成 123
ゾーン環境がインストールされているシステムでのリソース上限デーモンの使用 123
メモリー上限実行しきい値 124
上限値の決定 124
rcapd の動作間隔 126
rcapstat によるリソース使用効率の監視 127
rcapd とともに使用されるコマンド 129

11 リソース上限デーモンの管理 (タスク) 131
常駐セットサイズ上限の設定 131
▼ プロジェクトに rcap.max-rss 属性を追加する方法 131
▼ projmod コマンドを使用してプロジェクトに rcap.max-rss 属性を追加する方法 132
リソース上限デーモンの構成と使用 (タスクマップ) 132

rcapadm によるリソース上限デーモンの管理 133
▼ メモリー上限実行しきい値を設定する方法 133
▼ 動作間隔を設定する方法 133
▼ リソース上限制御を有効にする方法 134
▼ リソース上限制御を無効にする方法 134
▼ ゾーンに一時的なリソース上限を指定する方法 135
rcapstat による報告の生成 135
上限とプロジェクトの情報の報告 135
プロジェクトの RSS の監視 136
プロジェクトの作業セットサイズの決定 137
メモリー使用効率とメモリー上限実行しきい値の報告 138

12 リソースプール (概要) 139
リソースプールの紹介 140
動的リソースプールの紹介 141
リソースプールと動的リソースプールの有効化/無効化について 141
ゾーンで使用されるリソースプール 141
リソースプールを使用する場合 142
リソースプールのフレームワーク 143
/etc/pooladm.conf の内容 144
プールのプロパティー 144
システム上でのプールの実装 145
project.pool 属性 146
SPARC: 動的再構成の処理とリソースプール 146
プール構成の作成 146
動的構成の直接操作 147
poold の概要 148
動的リソースプールの管理 148
構成の制約と目標 149
構成の制約 149
構成の目標 150
poold のプロパティー 152
構成可能な poold の機能 153
poold の監視間隔 154
poold のログ情報 154

ログの場所 156
logadm によるログ管理 156
動的リソース割り当てのしくみ 156
使用可能なリソースについて 156
使用可能なリソースの特定 157
リソース不足の特定 158
リソース使用効率の判定 158
制御違反の特定 158
適切な改善操作の決定 159
poolstat によるプール機能とリソース使用効率の監視 159
poolstat の出力 160
poolstat の動作間隔の調整 160
リソースプール機能で使用するコマンド 161

13 リソースプールの作成と管理 (タスク) 163
リソースプールの管理 (タスクマップ) 163
プール機能の有効化と無効化 165
▼ svcadm を使用してリソースプールサービスを有効にする方法 165
▼ svcadm を使用してリソースプールサービスを無効にする方法 166
▼ svcadm を使用して動的リソースプールサービスを有効にする方法 166
▼ svcadm を使用して動的リソースプールサービスを無効にする方法 168
▼ pooladm を使ってリソースプールを有効にする方法 168
▼ pooladm を使ってリソースプールを無効にする方法 169
プールの構成 169
▼ 静的構成を作成する方法 169
▼ 構成の変更方法 170
▼ プールをスケジューリングクラスに対応付ける方法 172
▼ 構成の制約を設定する方法 174
▼ 構成の目標を定義する方法 175
▼ poold のログレベルを設定する方法 177
▼ poolcfg でコマンドファイルを使用する方法 177
リソースの転送 178
▼ プロセッサセット間で CPU を移動する方法 178
プール構成の起動と削除 178
▼ プール構成を起動する方法 178

▼構成を確定する前に構成を検証する方法 179
▼プール構成を削除する方法 179
プール属性の設定とプールへの結合 180
▼プロセスをプールに結合する方法 180
▼タスクまたはプロジェクトをプールに結合する方法 180
▼プロジェクトの project.pool 属性を設定する方法 181
▼project 属性を使ってプロセスを別のプールに結合する方法 181
poolstat を使ってプールに関連付けられているリソースについて統計情報を報告する 182
poolstat のデフォルトの出力を表示する 182
特定の間隔で複数の報告を生成する 182
リソースセットの統計情報を報告する 182

14 リソース管理の構成例 185
統合前の構成 185
統合後の構成 186
構成の作成 186
構成の表示 188

パート II Oracle Solaris ゾーン 193

15 Oracle Solaris ゾーンの紹介 195
ゾーンの概要 196
このリリースの Oracle Solaris ゾーンについて 197
読み取り専用の solaris 非大域ゾーン 200
ipkg ゾーンの solaris ゾーンへの変換について 200
ブランドゾーンについて 200
ブランドゾーンで実行中のプロセス 201
このリリースで使用できる非大域ゾーン 202
ゾーンを使用する場合 202
ゾーンのしくみ 204
ゾーンの機能別のサマリー 205
非大域ゾーンの管理のしくみ 206
非大域ゾーンの作成のしくみ 206
非大域ゾーンの状態モデル 207

非大域ゾーンの特性 210
非大域ゾーンでのリソース管理機能の使用 211
ゾーン関連の SMF サービス 211
非大域ゾーンの監視 212
非大域ゾーンによって提供される機能 212
システムのゾーンの設定 (タスクマップ) 213

16 非大域ゾーンの構成 (概要) 217
ゾーンのリソースについて 217
ゾーン管理での権利プロファイルと役割の使用 218
インストール前の構成処理 218
ゾーンのコンポーネント 218
ゾーンの名前とパス 218
ゾーンの自動ブート 218
読み取り専用ルートゾーンの `file-mac-profile` プロパティー 219
`admin` リソース 219
`dedicated-cpu` リソース 219
`capped-cpu` リソース 220
スケジューリングクラス 220
物理メモリーの制御と `capped-memory` リソース 221
`rootzpool` リソース 222
`zpool` リソースの自動追加 224
ゾーンネットワークインタフェース 224
ゾーンでマウントされるファイルシステム 229
ファイルシステムのマウントと更新 230
ゾーン内のホスト ID 231
非大域ゾーンの `/dev` ファイルシステム 231
非大域ゾーンのリムーバブル `lofi` デバイス 231
非大域ゾーン内でのディスク形式のサポート 232
構成可能な特権 232
リソースプールの関連付け 233
ゾーン規模のリソース制御の設定 233
ゾーンのコメントの追加 237
`zonecfg` コマンドの使用 237
`zonecfg` のモード 238

zonecfg の対話型モード 238
zonecfg のコマンドファイルモード 241
ゾーン構成データ 241
リソースタイプとプロパティー 241
リソースタイプのプロパティー 247
ゾーンの構成例 258
Tecla コマンド行編集ライブラリ 259

17 非大域ゾーンの計画と構成 (タスク) 261
非大域ゾーンの計画と構成 (タスクマップ) 261
現在のシステム設定の評価 265
必要なディスク容量 265
ゾーンサイズの制限 266
ゾーンのホスト名およびネットワーク要件の決定 266
ゾーンのホスト名 266
共有 IP ゾーンのネットワークアドレス 267
排他的 IP ゾーンのネットワークアドレス 268
ファイルシステムの構成 268
非大域ゾーン構成の作成、改訂、および削除 (タスクマップ) 269
ゾーンを構成、検証、および確定する 269
▼ ゾーンの構成方法 270
次に進む手順 276
複数のゾーンを構成するスクリプト 276
▼ 非大域ゾーンの構成を表示する方法 280
zonecfg コマンドを使用してゾーン構成を変更する 281
▼ ゾーン構成内のリソースタイプを変更する方法 281
▼ ゾーン構成内のプロパティーをクリアーする方法 282
▼ ゾーンの名前を変更する方法 282
▼ 専用のデバイスをゾーンに追加する方法 283
▼ 大域ゾーンの zone.cpu-shares を設定する方法 284
zonecfg コマンドを使用してゾーン構成を元に戻す、または削除する 284
▼ ゾーン構成を元に戻す方法 284
▼ ゾーン構成を削除する方法 286

18 非大域ゾーンのインストール、停止処理、停止、アンインストール、クローニングについて (概要) 287
ゾーンのインストールと管理の概要 287
ゾーンの構築 288
ゾーンのインストール方法 290
zoneadmd デーモン 292
zsched ゾーンスケジューラ 293
ゾーンアプリケーション環境 293
ゾーンの停止処理、停止、リブート、およびアンインストールについて 293
ゾーンの停止処理 293
ゾーンを停止する 294
ゾーンをリブートする 294
ゾーンのブート引数 294
ゾーンの autoboot 設定 295
ゾーンのアンインストール 296
非大域ゾーンのクローニングについて 296

19 非大域ゾーンのインストール、ブート、停止処理、停止、アンインストール、およびクローニング (タスク) 297
ゾーンのインストール (タスクマップ) 298
ゾーンのインストールとブート 299
▼ (オプション) インストール前に構成済みのゾーンを検証する方法 299
▼ 構成済みのゾーンをインストールする方法 300
▼ インストールされた非大域ゾーンの UUID を取得する方法 302
▼ インストールした非大域ゾーンに不完全のマークを付ける方法 303
▼ (オプション) インストール済みのゾーンを準備完了状態に移行する方法 303
▼ ゾーンのブート方法 304
▼ ゾーンをシングルユーザーモードでブートする方法 305
次に進む手順 305
非大域ゾーンの停止処理、停止、リブート、アンインストール、クローニング、および削除 (タスクマップ) 306
ゾーンの停止処理、停止、リブート、およびアンインストール 307
▼ ゾーンを停止処理する方法 307
▼ ゾーンの停止方法 307
▼ ゾーンをリブートする方法 308
▼ ゾーンをアンインストールする方法 309

同一システム上での非大域ゾーンのクローニング 310
▼ ゾーンをクローンする方法 310
非大域ゾーンの移動 312
▼ 共有ストレージ上にないゾーンを移動する方法 312
システムから非大域ゾーンを削除する 313
▼ 非大域ゾーンを削除する方法 313

20 非大域ゾーンへのログイン (概要) 315
zlogin コマンド 315
ゾーンの内部構成 316
対話式システム構成ツール 317
ゾーン構成プロファイルの例 318
非大域ゾーンへのログイン方法 322
ゾーンコンソールログイン 322
ユーザーログインの方法 323
フェイルセーフモード 323
リモートログイン 323
対話型モードと非対話型モード 324
対話型モード 324
非対話型モード 324

21 非大域ゾーンへのログイン (タスク) 325
初期ゾーンブートおよびゾーンログインの手順 (タスクマップ) 325
ゾーンへのログイン 326
▼ 構成プロファイルを作成する方法 326
▼ ゾーンコンソールにログインして初期ゾーン構成を行う方法 327
▼ ゾーンコンソールへのログイン方法 328
▼ 対話型モードを使用してゾーンにアクセスする方法 328
▼ 非対話型モードを使用してゾーンにアクセスする方法 329
▼ 非大域ゾーンから抜ける方法 329
▼ フェイルセーフモードを使用してゾーンに入る方法 330
▼ zlogin を使用してゾーンを停止処理する方法 330
サービスの有効化 331
現在のゾーンの名前を出力する 331

22 ゾーンの移行と **zonep2vchk** ツールについて 333
Physical-To-Virtual の概念と Virtual-to-Virtual の概念 333
移行方針の選択 333
zonep2vchk ツールを使用したシステムの移行の準備 335
zonep2vchk ツールについて 335
解析のタイプ 337
生成される情報 338

23 **Oracle Solaris** システムの移行と非大域ゾーンの移行 **(**タスク**)** 339
別のマシンへの非大域ゾーンの移行 339
ゾーンの移行について 339
▼ ZFS アーカイブを使用して非大域ゾーンを移行する方法 340
使用できないマシンからのゾーンの移行 343
非大域ゾーンへの Oracle Solaris システムの移行 343
solaris 非大域ゾーンへの Oracle Solaris システムの移行について 343
▼ zonep2vchk を使用したソースシステムのスキャン 344
▼ ネットワークデバイス上にシステムイメージのアーカイブを作成する方法 344
▼ ターゲットシステム上のゾーンを構成する方法 345
▼ ターゲットシステムへのゾーンのインストール 346

24 ゾーンがインストールされている **Oracle Solaris 11.1** システムでの自動インストールおよびパッケージ 347
Oracle Solaris 11.1 リリースを実行しているシステムのイメージパッケージングシステムソフトウェア 347
ゾーンのパッケージの概要 348
パッケージとゾーンについて 349
ゾーンがインストールされているシステムへのパッケージの追加について 350
大域ゾーンでの pkg の使用 350
非大域ゾーンでの pkg install コマンドの使用 350
カスタム AI マニフェストを使用したゾーンへのパッケージの追加 350
ゾーン内でのパッケージの削除について 352
パッケージ情報の照会 352
インストール済みのゾーンがあるシステムでのプロキシ構成 352
大域ゾーンでのプロキシの構成 353
https_proxy および http_proxy を使用した system-repository プロキシの

オーバーライド 353
ゾーンの並列更新 354
ゾーンの状態がパッケージの操作に与える影響 354

25 Oracle Solaris ゾーンの管理 (概要) 357
大域ゾーンの可視性とアクセス 358
ゾーン内でのプロセス ID の可視性 358
ゾーン内のシステム監視機能 359
zonestat ユーティリティーを使用したアクティブなゾーンの統計情報の報告 359
fsstat ユーティリティーを使用した非大域ゾーンの監視 360
非大域ゾーンのノード名 360
ゾーンでの NFS サーバーの実行 361
ファイルシステムと非大域ゾーン 361
-o nosuid オプション 361
ゾーン内でのファイルシステムのマウント 362
ゾーン内でのファイルシステムのアンマウント 364
セキュリティーの制限およびファイルシステムの動作 364
NFS クライアントとして機能する非大域ゾーン 367
ゾーン内での mknod の使用禁止 367
ファイルシステムの行き来 367
大域ゾーンから非大域ゾーンにアクセスする際の制限 368
共有 IP 非大域ゾーンにおけるネットワーク 369
共有 IP ゾーンの区分化 369
共有 IP ネットワークインタフェース 370
同一マシン上の共有 IP ゾーン間の IP トラフィック 370
共有 IP ゾーンでの Oracle Solaris IP フィルタ 370
共有 IP ゾーン内の IP ネットワークマルチパス 371
排他的 IP 非大域ゾーンにおけるネットワーク 371
排他的 IP ゾーンの区分化 372
排他的 IP データリンクインタフェース 372
同一マシン上の排他的 IP ゾーン間の IP トラフィック 372
排他的 IP ゾーンにおける Oracle Solaris IP フィルタ 373
排他的 IP ゾーン内の IP ネットワークマルチパス 373
非大域ゾーンでのデバイスの使用 373
/dev および /devices 名前空間 373

排他使用のデバイス 374
デバイスドライバの管理 374
非大域ゾーンで動作しないか、変更されるユーティリティー 375
非大域ゾーンでのアプリケーションの実行 375
非大域ゾーンで使用されるリソース制御 376
ゾーンがインストールされている システムでの公平配分スケジューラ 377
大域ゾーンまたは非大域ゾーン内の FSS 配分分割 377
ゾーン間の配分均衡 377
ゾーンがインストールされているシステムでの拡張アカウンティング 377
非大域ゾーン内の特権 378
ゾーン内での IP セキュリティーアーキテクチャーの使用 383
共有 IP ゾーン内の IP セキュリティーアーキテクチャー 383
排他的 IP ゾーン内の IP セキュリティーアーキテクチャー 383
ゾーン内での Oracle Solaris 監査の使用 383
ゾーン内のコアファイル 384
非大域ゾーン内での DTrace の実行 384
ゾーンがインストールされている Oracle Solaris システムのバックアップについて 385
ループバックファイルシステムのディレクトリのバックアップ 385
大域ゾーンからのシステムのバックアップ 385
システム上の非大域ゾーンを個別にバックアップ 385
Oracle Solaris ZFS バックアップの作成 386
非大域ゾーン内でバックアップするデータの決定 387
アプリケーションデータのみのバックアップ 387
一般的なデータベースバックアップ操作 387
テープによるバックアップ 388
非大域ゾーンの復元について 388
ゾーンがインストールされているシステムで使用するコマンド 389

26 Oracle Solaris ゾーンの管理 (タスク) 395
ppriv ユーティリティーの使用 395
▼ 大域ゾーンでの Oracle Solaris の特権を一覧表示する方法 396
▼ 非大域ゾーンの特権セットの表示方法 396
▼ 非大域ゾーンの特権セットを冗長出力で表示する方法 397
非大域ゾーンでの zonestat ユーティリティーの使用 397
▼ zonestat ユーティリティーを使用して、CPU およびメモリーの使用効率のサマ

リーを表示する方法 398
▼ zonestat ユーティリティーを使用して、デフォルトの pset について報告する方法 399
▼ zonestat を使用した合計および最大使用効率の報告 399
▼ 排他的 IP ゾーンでのネットワーク帯域幅の使用効率を取得する方法 400
すべてのゾーンのゾーン別の fstype 統計情報の報告 401
▼ -z オプションを使用して特定のゾーンのアクティビティーを監視する方法 401
▼ すべてのゾーンのゾーン別の fstype 統計情報を表示する方法 402
非大域ゾーン内での DTrace の使用 402
▼ DTrace を使用する方法 402
非大域ゾーンの SMF サービスのステータスの確認 403
▼ コマンド行から SMF サービスのステータスを確認する方法 403
▼ ゾーン内から SMF サービスのステータスを確認する方法 403
稼働中の非大域ゾーン内でファイルシステムをマウントする 404
▼ LOFS を使用してファイルシステムをマウントする方法 404
▼ ZFS データセットを非大域ゾーンに委任する方法 405
大域ゾーン内の特定のファイルシステムへのアクセス権を非大域ゾーンに追加する 407
▼ 非大域ゾーンで CD または DVD メディアにアクセスする権限を追加する方法 407
ゾーンがインストールされている Oracle Solaris システムでの IP ネットワークマルチパスの使用 409
▼ 排他的 IP 非大域ゾーンで IP ネットワークマルチパスを使用する方法 409
▼ IP ネットワークマルチパス機能を共有 IP 非大域ゾーンに拡張する方法 409
排他的 IP 非大域ゾーンでのデータリンクの管理 410
▼ dladm show-linkprop の使用方法 410
▼ dladm を使用して、一時的なデータリンクを割り当てる方法 412
▼ dladm reset-linkprop の使用方法 412
ゾーンがインストールされている Oracle Solaris システムでの公平配分スケジューラの使用 413
▼ prctl コマンドを使用して大域ゾーンの FSS 配分を設定する方法 413
▼ ゾーンの zone.cpu-shares 値を動的に変更する方法 414
ゾーン管理での権利プロファイルの使用 414
▼ Zone Management プロファイルを割り当てる方法 414
ゾーンがインストールされている Oracle Solaris システムのバックアップ 414
▼ ZFSsend を使用してバックアップを実行する方法 415
▼ ゾーン構成のコピーを出力する方法 415
非大域ゾーンの再作成 416

▼ 非大域ゾーンを個別に再作成する方法 416

27 不変ゾーンの構成と管理 417
読み取り専用ゾーンの概要 417
読み取り専用ゾーンの構成 418
`zonecfg` の `file-mac profile` プロパティー 418
`zonecfg add dataset` リソースポリシー 419
`zonecfg add fs` リソースポリシー 419
読み取り専用ゾーンの管理 419
`zoneadm list -p` の表示 420
書き込み可能なルートファイルシステムを持つ読み取り専用ゾーンをブートするオプション 420

28 **Oracle Solaris** ゾーンで発生するさまざまな問題のトラブルシューティング 423
排他的 IP ゾーンがデバイスを使用しているために `dladm reset-linkprop` が失敗する 423
ゾーン構成内に不正な特権セットが指定されている 423
ゾーンが停止しない 424

パート **III** **Oracle Solaris 10** ゾーン 425

29 **Oracle Solaris 10** ゾーンの紹介 427
`solaris10` ブランドについて 427
`solaris10` ゾーンのサポート 429
Oracle Solaris 10 ゾーンの SVR4 パッケージおよびパッチ 429
`solaris10` ブランドゾーンでのパッケージおよびパッチの使用について 429
リモートからのパッケージ操作およびパッチ操作の実行について 430
NFS クライアントとして機能する非大域ゾーン 431
ゾーンの一般的な概念 431
このリリースの Oracle Solaris 10 ゾーンについて 432
操作に関する制限事項 432
Oracle Solaris 10 ゾーンのネットワーク 432
ネイティブ非大域ゾーンがインストールされている場合 434

30 Oracle Solaris 10 システムの評価とアーカイブの作成 435
ソースシステムおよびターゲットシステムの前提条件 435
Oracle Solaris 10 のパッケージツールおよびパッチツールの有効化 435
ターゲットシステムへの必須 Oracle Solaris パッケージのインストール 435
zonep2vchk ユーティリティーを使用して移行されるシステムの評価 436
Oracle Solaris 10 システムのみ: zonep2vchk ユーティリティーの取得 436
Oracle Solaris 10 システムをゾーンに直接移行するためのイメージの作成 437
▼ flarcreate を使用してイメージを作成する方法 437
▼ flarcreate を使用して特定のデータを除外する方法 438
ほかのアーカイブ作成方法 439
ホスト ID のエミュレーション 439

31 (オプション) Oracle Solaris 10 ゾーンへの Oracle Solaris 10 native 非大域ゾーンの移行 441
アーカイブに関する考慮事項 441
solaris10 ゾーン移行プロセスの概要 441
solaris10 ゾーンの切り離しおよび接続について 442
solaris10 ブランドゾーンの移行 442
Oracle Solaris 10 システムの既存のゾーンの移行 443
▼ 既存のネイティブ非大域ゾーンを移行する方法 443

32 solaris10 ブランドゾーンの構成 447
事前構成タスク 447
デフォルトで構成に含まれているリソース 447
solaris10 ブランドゾーンで構成されるデバイス 447
solaris10 ブランドゾーンで定義される特権 448
solaris10 ブランドゾーンの構成処理 448
ターゲットゾーンの構成 449
▼ 排他的 IP solaris10 ブランドゾーンの構成方法 449
▼ 共有 IP solaris10 ブランドゾーンの構成方法 451

33 solaris10 ブランドゾーンのインストール 455
ゾーンのインストールイメージ 455
システムイメージの種類 455
イメージ sysidcfg のステータス 455

solaris10 ブランドゾーンのインストール 456
インストーラオプション 456
▼ solaris10 ブランドゾーンをインストールする方法 457

34 ゾーンのブート、ログイン、ゾーンの移行 459
solaris10 ブランドゾーンのブートについて 459
イメージ sysidcfg プロファイル 459
▼ solaris10 ブランドゾーンの内部構成 461
▼ solaris10 ブランドゾーンをブートする方法 461
別のホストへの solaris10 ブランドゾーンの移行 462

用語集 463

索引 467

はじめに

このドキュメントは、Oracle Solaris オペレーティングシステムの管理に関する情報の大部分について説明するドキュメントセットの一部です。このマニュアルでは、オペレーティングシステムがすでにインストールされており、使用する予定のネットワークソフトウェアが設定済みであることを前提としています。

このリリースの新機能については、197 ページの「このリリースの Oracle Solaris ゾーンについて」で説明しています。

Oracle Solaris ゾーンについて

Oracle Solaris ゾーン製品とは、アプリケーション用の機能が完備された実行環境のことです。ゾーンは、アプリケーションからプラットフォームリソースへの仮想マッピングを提供します。ゾーンを使用すると、Oracle Solaris オペレーティングシステムの単一のインスタンスを複数のゾーンで共有しているにもかかわらず、アプリケーションコンポーネントを互いに隔離できます。Oracle Solaris リソースマネージャーの製品コンポーネントは、通常、「リソース管理機能」と呼ばれ、作業負荷として受け取られるリソースの量を割り当てることができます。

ゾーンは、CPU などのリソースの消費量に制限を設けます。ゾーン内で実行されるアプリケーションの処理要件の変化に応じて、これらの制限を拡張することもできます。

追加の隔離として、不変ゾーンと呼ばれる、読み取り専用ルートを保持するゾーンを構成できます。

Oracle Solaris 10 ゾーンについて

Oracle Solaris 10 ゾーンは、`solaris10` ブランド非大域ゾーンとも呼ばれ、BrandZ テクノロジを使用して Oracle Solaris 11 オペレーティングシステム上で Oracle Solaris 10 アプリケーションを実行します。アプリケーションは、非大域ゾーンによって提供される安全な環境内で、変更されることなく実行されます。これにより、Oracle Solaris 10 システムを使用してアプリケーションの開発、テスト、および配備を行うことができます。これらのブランドゾーン内で実行される作業負荷は、カーネルに加えられた機能拡張を利用して、Oracle Solaris 11 リリース上でのみ利用できる一部の革新的技術を使用できます。

この製品を使用するには、パート III「Oracle Solaris 10 ゾーン」を参照してください。

Oracle Solaris Trusted Extensions システム上での Oracle Solaris ゾーンの使用について

Oracle Solaris Trusted Extensions システム上でゾーンを使用する方法については、『Trusted Extensions 構成と管理』の第 13 章「Trusted Extensions でのゾーンの管理」を参照してください。Oracle Solaris Trusted Extensions システムでは `labeled` ブランドしかブートできないことに注意してください。

Oracle Solaris クラスタのゾーンクラスタ

ゾーンクラスタとは、Oracle Solaris クラスタソフトウェアの 1 つの機能です。ゾーンクラスタのすべてのノードが、`cluster` 属性を使用して非大域 `solaris` ゾーンとして構成されます。その他のブランドタイプは許可されません。ゾーンによって提供される隔離を使用して、グローバルクラスタ上と同様に、サポートされるサービスをゾーンクラスタ上で実行できます。詳細は、『Oracle Solaris Cluster システム管理』を参照してください。

Oracle Solaris リソースマネージャー

リソース管理を利用すると、利用可能なシステムリソースを、アプリケーションがどのように使用するかを制御できます。パート I「Oracle Solaris のリソース管理」を参照してください。

対象読者

このマニュアルは、Oracle Solaris リリースが稼働しているシステムの管理者を対象としています。このドキュメントを利用するにあたっては、UNIX のシステム管理について少なくとも 1-2 年の経験が必要です。

Oracle サポートへのアクセス

Oracle のお客様は、My Oracle Support を通じて電子的なサポートを利用することができます。詳細は、http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info を参照してください。聴覚に障害をお持ちの場合は、http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs を参照してください。

表記上の規則

次の表では、このドキュメントで使用される表記上の規則について説明します。

表 P–1　表記上の規則

字体	意味	例
AaBbCc123	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コード例を示します。	.login ファイルを編集します。 ls -a を使用してすべてのファイルを表示します。 machine_name% you have mail.
AaBbCc123	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して示します。	machine_name% **su** Password:
aabbcc123	プレースホルダ: 実際に使用する特定の名前または値で置き換えます。	ファイルを削除するには、 rm *filename* と入力します。
AaBbCc123	書名、新しい単語、および強調する単語を示します。	『ユーザーズガイド』の第 6 章を参照してください。 キャッシュは、ローカルに格納されるコピーです。 ファイルを保存しないでください。 注: いくつかの強調された項目は、オンラインでは太字で表示されます。

コマンド例のシェルプロンプト

Oracle Solaris OS に含まれるシェルで使用する、UNIX のデフォルトのシステムプロンプトとスーパーユーザープロンプトを次に示します。コマンド例に示されるデフォルトのシステムプロンプトは、Oracle Solaris のリリースによって異なります。

表 P–2　シェルプロンプト

シェル	プロンプト
Bash シェル、Korn シェル、および Bourne シェル	`$`
Bash シェル、Korn シェル、および Bourne シェルのスーパーユーザー	`#`
C シェル	`machine_name%`
C シェルのスーパーユーザー	`machine_name#`

特権および管理権限に関する情報の取得

役割と管理権限の詳細は、『Oracle Solaris 11.1 の管理: セキュリティーサービス』のパート III「役割、権利プロファイル、特権」を参照してください。

パート I

Oracle Solaris のリソース管理

このパートでは、Oracle Solaris のリソース管理について説明します。Solaris のリソース管理を使用すると、利用可能なシステムリソースをアプリケーションでどのように使用するかを制御できます。

◆ ◆ ◆ 第 1 章

1 リソース管理の紹介

Oracle Solaris のリソース管理機能を利用すると、アプリケーションが利用可能なシステムリソースをどのように使用するかを制御できます。次のような制御が可能になります。

- プロセッサ時間などのコンピュータリソースを割り当てます
- 割り当てたリソースの使用状況を監視し、必要に応じて調整します
- 解析、課金、および容量計画のために拡張アカウンティング情報を生成します

この章の内容は次のとおりです。

- 29 ページの「リソース管理の概要」
- 33 ページの「リソース管理機能を使用する場合」
- 34 ページの「リソース管理の設定 (タスクマップ)」

リソース管理の概要

今日のコンピューティング環境では、システム上で実行されるアプリケーションによって生成されるさまざまな作業負荷に柔軟に対応できる能力が求められます。「作業負荷」とは、1 つのアプリケーションまたは一連のアプリケーションが持つすべてのプロセスの集まりです。リソース管理機能を使用しない場合、Oracle Solaris オペレーティングシステムは、新しいアプリケーションの要求に動的に適応することによって作業負荷の要求に対応しています。このデフォルトの動作は、通常、システムのすべてのアクティビティーに対してリソースへのアクセスを同等に与えることを意味します。リソース管理機能を使用すると、作業負荷を個別に扱うことができるようになります。次のような制御が可能になります。

- 特定のリソースへのアクセスを制限します
- 優先順位に基づいて作業負荷にリソースを提供します
- 作業負荷を互いに分離します

作業負荷が相互に影響し合うことによる性能の低下を最小限に抑える機能と、リソースの使用状況や使用効率を監視する機能を総称して「リソース管理機能」とい

います。リソース管理機能は、いくつかのアルゴリズムの集合として実装されます。これらのアルゴリズムは、アプリケーションがその実行過程で発行する一連のリソース要求を処理します。

リソース管理機能を使用すると、さまざまな作業負荷に対してオペレーティングシステムのデフォルトの動作を変更できます。この場合の「動作」とは、主に、アプリケーションが 1 つ以上のリソース要求をシステムに発行したときに、オペレーティングシステムのアルゴリズムが行う一連の決定処理のことです。リソース管理機能は、次の目的で使用できます。

- あるアプリケーションに対して、リソースの割り当てを拒否したり、ほかのアプリケーションに許可されているよりも大きい割り当てを与えたりします
- 特定の割り当てを個別にではなく一括して行います

リソース管理機能を使用できるようにシステムを構成すると、いくつかの目的が達成できます。次のような制御が可能になります。

- アプリケーションが際限なくリソースを浪費するのを防止します
- 外部イベントに基づいてアプリケーションの優先順位を変更します
- 一連のアプリケーションにリソースの使用を保証する一方で、システムの使用効率を最大限に高めます

リソース管理機能をシステム構成に組み込むときは、次のような作業が事前に必要です。

- システム上で競合する作業負荷の特定
- 競合しない作業負荷と、主要な作業負荷に影響を与えるような性能要件を伴った作業負荷との区別

競合しない作業負荷と競合する作業負荷を特定したら、システムの能力が許す範囲内で、業務サービスへの影響を最小限に抑えたリソース構成を構築できます。

効率のよいリソース管理を実現するために、Oracle Solaris システムには制御メカニズム、通知メカニズム、および監視メカニズムが用意されています。これらの機能の多くは、`proc(4)` ファイルシステム、プロセッサセット、スケジューリングクラスなどの既存メカニズムの拡張機能として提供されます。その他の機能はリソース管理に固有です。これらの機能については、以降の章で説明します。

リソースの分類

リソースは、アプリケーションの動作を変更する目的で操作されるコンピューティングシステムのあらゆる側面を意味します。つまり、リソースは、アプリケーションが暗黙的または明示的に要求する機能です。この機能が拒否または制限された場合は、堅固に作成されているアプリケーションの実行速度が低下します。

リソースの分類は、リソースの識別とは対照的に、多くの基準に基づいて行うことができます。たとえば、リソースは、暗黙的な要求か明示的な要求か、時間 (CPU 時間など) に基づいた要求か時間に無関係な要求 (割り当てられた CPU 配分など) か、などを基準に行うことができます。

通常、スケジューラに基づいたリソース管理は、アプリケーションが暗黙的に要求するリソースに適用されます。たとえば、実行を継続するときは、アプリケーションは暗黙的に追加の CPU 時間を要求します。ネットワークソケットにデータを書き込むときは、アプリケーションは暗黙的に帯域幅を要求します。暗黙的に要求されるリソースの総使用量に対して制約を設けることができます。

帯域幅または CPU サービスのレベルを明示的に折衝できるように、インタフェースを追加することもできます。追加スレッドの要求のように明示的に要求されるリソースは、制約によって管理できます。

リソース管理の制御メカニズム

Oracle Solaris オペレーティングシステムでは、制約、スケジューリング、および区分という 3 種類の制御メカニズムを使用できます。

制約メカニズム

制約を使用すると、管理者やアプリケーション開発者は、作業負荷が使用する特定のリソースの消費にいくつかの制限を設定できます。制限を設定すると、リソースの消費シナリオを簡単にモデル化できます。また、制限を設定することにより、無秩序にリソースを要求してシステムの性能や可用性に悪影響を及ぼす可能性がある悪質なアプリケーションを制御できます。

制約は、アプリケーションに制限を課します。アプリケーションとシステムの関係は、アプリケーションが動作できないところまで悪化してしまう可能性があります。そのような事態を回避する方法の 1 つは、リソースに関する動作が不明なアプリケーションに対する制約を徐々に強めていくことです。第 6 章「リソース制御 (概要)」で説明するリソース制御が、制約メカニズムを提供します。新たに作成するアプリケーションであれば、リソースの制約をアプリケーションが認識するようにすることもできます。ただし、すべての開発者がこの機能を使用するとは限りません。

スケジューリングメカニズム

スケジューリングとは、一定の間隔で割り当てを決定することです。この決定は、予測可能なアルゴリズムに基づいて行われます。現在割り当てられているリソースを必要としないアプリケーションは、ほかのアプリケーションが使用できるように、そのリソースを解放します。スケジューリングに基づいてリソース管理を行うと、リソースに余裕がある構成の場合は使用効率を最大限にできると同時

に、リソースが限界まで、あるいは過剰に使用されている場合には、割り当てを制御できます。スケジューリングのアルゴリズムにより、「制御」という用語の意味が決まります。場合によっては、スケジューリングアルゴリズムは、すべてのアプリケーションがリソースにある程度アクセスできることを保証します。第 8 章「公平配分スケジューラ (概要)」で説明する公平配分スケジューラ (FSS) は、アプリケーションが制御された方法でCPU リソースにアクセスするように管理します。

区分メカニズム

区分は、作業負荷をシステム上で使用可能なリソースの一部に結合 (バインド) するために使用されます。リソースと結合することにより、作業負荷は常に一定量のリソースを使用できることが保証されます。第 12 章「リソースプール (概要)」で説明するリソースプール機能を使用すれば、作業負荷をマシンの特定の部分に制限できます。

区分を使用する構成では、システム全体が過剰使用されるのを防ぐことができます。ただし、この方法では、高い使用効率の達成は難しくなります。プロセッサなど、予約済みのリソースに結合されている作業負荷がアイドル状態になっている場合でも、別の作業負荷がそのリソースを使用することはできないためです。

リソース管理構成

リソース管理構成の一部をネットワークのネームサービスに置くことができます。この機能により、管理者は、リソース管理制約をマシンごとに排他的に適用するのではなく、マシンの集合に対して一括して適用できます。関連する作業は共通識別子を共有でき、その作業の使用状況はアカウンティングデータに基づいて表形式で表すことができます。

リソース管理構成と作業負荷識別子の詳細については、第 2 章「プロジェクトとタスク (概要)」に記載されています。これらの識別子をアプリケーションのリソース使用状況と結び付ける拡張アカウンティング機能は、第 4 章「拡張アカウンティング (概要)」に記載されています。

非大域ゾーンとの相互動作

リソース管理機能をゾーンと組み合わせて使用すると、アプリケーション環境をさらに細かく調整できます。リソース管理機能とゾーンの相互動作については、このガイドの該当セクションで説明します。

リソース管理機能を使用する場合

リソース管理機能を使用して、アプリケーションが必要な応答時間を確保できるようにします。

また、リソース管理機能により、リソースの使用効率を向上させることができます。使用状況を分類して優先付けすることにより、オフピーク時に余ったリソースを効率よく使用でき、処理能力を追加する必要がなくなります。また、負荷の変動が原因でリソースを無駄にすることもなくなります。

サーバーを統合する場合

リソース管理機能は、多くのアプリケーションを 1 台のサーバー上で統合できる環境で使用するともっとも高い効果を発揮します。

多数のマシンの管理は複雑でコストがかかるため、より大規模で拡張性の高いサーバーにアプリケーションを統合することが望まれます。個々の作業負荷を別々のシステムで実行して、そのシステムのリソースへの完全なアクセスを与える代わりに、リソース管理ソフトウェアを使用すれば、システム内の作業負荷を分離できます。リソース管理を使用すると、1 つの Oracle Solaris システムで複数の異なるアプリケーションを実行して制御することにより、システムの総保有コスト (TCO) を低減することができます。

インターネットサービスやアプリケーションサービスを提供する場合は、リソース管理を使用すると、次のことが可能になります。

- 1 台のマシンに複数の Web サーバーを配置します。各 Web サイトのリソース消費を制御し、各サイトをほかのサイトで起こる可能性のある過剰使用から保護します。
- 欠陥のある CGI (Common Gateway Interface) スクリプトが CPU リソースを浪費するのを防止します。
- 不正な動作をするアプリケーションによって引き起こされる、仮想メモリーのリークを防止します。
- 顧客のアプリケーションが、同じサイトで実行されている別の顧客のアプリケーションの影響を受けないようにします。
- 同一マシン上で異なるレベルまたはクラスのサービスを提供します。
- 課金目的でアカウンティング情報を取得します。

大規模で多様なユーザーが利用するシステムをサポートする場合

リソース管理機能は、特に、教育機関のように大規模で多様なユーザーが利用するシステムでその効果を発揮します。さまざまな作業負荷が混在している場合は、特定のプロジェクトに高い優先順位を与えるようにソフトウェアを構成できます。

たとえば、大きな証券会社のトレーダは、データベースのクエリーや計算を実行するために、一時的に高速なアクセスが必要になる場合があります。一方、ほかのユーザーの作業負荷は、一定しています。トレーダのプロジェクトに、作業量に応じてより高い処理能力を割り当てれば、トレーダは必要とする応答性を確保できます。

また、リソース管理機能は、シン (thin) クライアントシステムをサポートするのにも適しています。これらのプラットフォームは、フレームバッファーとスマートカードのような入力デバイスを持つステートレスなコンソールを備えています。実際の処理は共有サーバー上で行われるため、タイムシェアリング型の環境とみなすことができます。リソース管理機能を使ってサーバー上のユーザーを分離してください。こうすることで、過剰な負荷を引き起こしたユーザーがハードウェアリソースを占有し、システムを使用するほかのユーザーに重大な影響を与えることがなくなります。

リソース管理の設定 (タスクマップ)

次のタスクマップに、システム上にリソース管理を設定するための手順の全体的な概要を示します。

タスク	説明	参照先
システム上の作業負荷を特定し、各作業負荷をプロジェクト別に分類します。	`/etc/project` ファイル、NIS マップ、または LDAP ディレクトリサービス内にプロジェクトエントリを作成します。	40 ページの「`project` データベース」
システム上の作業負荷に優先順位を付けます。	どのアプリケーションが重要かを判定します。重要な作業負荷にはリソースへの優先的なアクセスが必要になる場合があります。	サービスの目的を考慮してください。

タスク	説明	参照先
システム上で実際のアクティビティーを監視します。	パフォーマンスツールを使用して、システムで実行されている作業負荷の現在のリソース消費量を表示します。その上で、特定のリソースへのアクセスを制限する必要があるかどうか、あるいは特定の作業負荷をほかの作業負荷から分離する必要があるかどうかを判定できます。	cpustat(1M)、 iostat(1M)、mpstat(1M)、 prstat(1M)、 sar(1)、 および vmstat(1M) のマニュアルページ
システムで実行されている作業負荷を一時的に変更します。	変更可能な設定値を決めるには、Oracle Solaris システムで使用できるリソース制御を参照します。タスクまたはプロセスが実行している間は、コマンド行から値を更新できます。	78 ページの「使用可能なリソース制御」、85 ページの「リソース制御値に対応付けられた大域アクションと局所アクション」、90 ページの「動作中のシステム上のリソース制御値を一時的に更新する」、および rctladm(1M) と prctl(1) のマニュアルページ
project データベースまたはネームサービスプロジェクトデータベース内のプロジェクトエントリごとにリソース制御とプロジェクト属性を設定します。	/etc/project ファイルまたはネームサービスプロジェクトデータベース内の各プロジェクトエントリには、リソース制御または属性を 1 つ以上含めることができます。これらのリソース制御は、そのプロジェクトに属するタスクとプロセスを制約します。リソース制御で指定する各しきい値に対しては、その値に達したときに行われるアクションを 1 つ以上対応付けることができます。 リソース制御は、コマンド行インタフェースを使って設定できます。	40 ページの「project データベース」、41 ページの「ローカルの /etc/project ファイルの形式」、78 ページの「使用可能なリソース制御」、85 ページの「リソース制御値に対応付けられた大域アクションと局所アクション」、および第 8 章「公平配分スケジューラ (概要)」
プロジェクト内のプロセスの集合が消費する物理メモリーの容量に上限を設けます。	リソース上限デーモンは、/etc/project ファイルでプロジェクトの rcap.max-rss 属性に指定されたとおり、物理メモリーのリソース上限を制限します。	40 ページの「project データベース」および第 10 章「リソース上限デーモンによる物理メモリーの制御 (概要)」
リソースプール構成を作成します。	リソースプールは、プロセッサなどのシステムリソースを区分する手段を提供し、リブート時にもそのパーティションを保持します。/etc/project ファイルの各エントリに project.pool 属性を 1 つ追加できます。	40 ページの「project データベース」および第 12 章「リソースプール (概要)」

タスク	説明	参照先
公平配分スケジューラ (FSS) をデフォルトのシステムスケジューラとして設定します。	単一の CPU システムまたはプロセッサセット内のすべてのユーザープロセスが同じスケジューリングクラスに属するようにします。	117 ページの「FSS の構成」および `dispadmin(1M)` のマニュアルページ
拡張アカウンティング機能を起動し、タスクまたはプロセスに基づきリソース消費を監視して記録します。	拡張アカウンティングデータを使って現在のリソース制御を評価し、将来の作業負荷のための容量要件を計画します。システム全体の総使用状況を追跡できます。複数のシステムに渡って相互に関連しあう作業負荷について完全な使用統計を取得するために、プロジェクト名は複数のマシンで共有できます。	68 ページの「フロー、プロセス、タスク、およびネットワークコンポーネントの拡張アカウンティングを起動する方法」および `acctadm(1M)` のマニュアルページ
(オプション) 構成をさらに調整する必要がある場合は、引き続きコマンド行から値を変更できます。値は、タスクまたはプロセスの実行中でも変更できます。	既存のタスクに対しては、プロジェクトを再起動しなくても、変更を一時的に適用できます。満足のいく性能が得られるまで値を調整します。次に、`/etc/project` ファイルまたはネームサービスのプロジェクトデータベースで現在の値を更新します。	90 ページの「動作中のシステム上のリソース制御値を一時的に更新する」および `rctladm(1M)` と `prctl(1)` のマニュアルページ
(オプション) 拡張アカウンティングデータを取得します。	アクティブなプロセスおよびタスクの拡張アカウンティングレコードを書き込みます。作成されるファイルは、計画、チャージバック、および課金のために使用できます。`libexacct` への Perl (Practical Extraction and Report Language) インタフェースを使用して、報告および抽出用のカスタムスクリプトを作成することもできます。	`wracct(1M)` のマニュアルページおよび 64 ページの「`libexacct` に対する Perl インタフェース」

◆ ◆ ◆ 第 2 章

2 プロジェクトとタスク (概要)

この章では、Oracle Solaris のリソース管理のプロジェクトおよびタスクの機能について説明します。プロジェクトとタスクは、作業負荷にラベル付けを行い、ほかの作業負荷と区別するために使用されます。

この章の内容は次のとおりです。

- 37 ページの「プロジェクトとタスクの機能」
- 38 ページの「プロジェクト識別子」
- 44 ページの「タスク識別子」
- 45 ページの「プロジェクトとタスクで使用するコマンド」

プロジェクトとタスクの機能の使用方法については、第 3 章「プロジェクトとタスクの管理」を参照してください。

プロジェクトとタスクの機能

作業負荷の応答性を最適化するには、まず解析対象のシステム上で実行中の作業負荷を特定できなければなりません。この情報は、プロセス指向の手法とユーザー指向の手法のどちらか一方だけを使用して取得できるものではありません。Oracle Solaris システムには、作業負荷を区別して識別するための「プロジェクト」と「タスク」という 2 つの追加機能があります。「プロジェクト」は、関連した作業に対してネットワーク全体で適用される管理識別子を与えます。「タスク」は、プロセスのグループを、作業負荷のコンポーネントを表す管理しやすいエンティティーにまとめます。

`project` ネームサービスデータベースで指定された制御は、プロセス、タスク、およびプロジェクトに対して設定されます。プロセスの制御とタスクの制御は `fork` および `settaskid` システムコールを通して継承されるので、これらの制御は同じプロジェクト内に作成されるすべてのプロセスとタスクに継承されます。これらのシステムコールについては、`fork(2)` および `settaskid(2)` のマニュアルページを参照してください。

実行中のプロセスは、そのプロセスのプロジェクトメンバーシップまたはタスクメンバーシップに基づいて、Oracle Solaris の標準コマンドを使用して操作できます。拡張アカウンティング機能は、プロセスとタスクの両方の使用状況について報告を作成し、各レコードに管理用プロジェクト識別子のタグを付けることができます。この処理により、オフラインで行う作業負荷解析作業をオンラインでの監視作業と関連付けることができます。プロジェクト識別子は、`project` ネームサービスデータベースを介して複数のマシンで共有できます。したがって、最終的には、複数のマシン上で実行される (つまり複数のマシンにわたる) 関連した作業負荷のリソース消費をすべてのマシンについて解析できます。

プロジェクト識別子

プロジェクト識別子は、関連する作業を特定するために使用される管理識別子です。プロジェクト識別子は、ユーザー識別子やグループ識別子と同様な、作業負荷に付けられているタグと考えることができます。ユーザーまたはグループは 1 つ以上のプロジェクトに所属できます。プロジェクトは、ユーザー (またはユーザーグループ) が参加することを許可されている作業負荷を表すために使用します。このメンバーシップは、たとえば、使用状況や初期リソース割り当てに基づく課金の根拠となります。ユーザーにはデフォルトのプロジェクトを割り当てる必要がありますが、ユーザーが起動するプロセスは、ユーザーが属しているプロジェクトであれば、どのプロジェクトにでも関連付けることができます。

ユーザーのデフォルトプロジェクトの判定

システムにログインするには、そのユーザーにデフォルトのプロジェクトが割り当てられている必要があります。ユーザーは、そのプロジェクトで指定されたユーザーリストやグループリストに含まれていない場合でも、自動的にそのデフォルトプロジェクトのメンバーになります。

システム上の各プロセスはプロジェクトのメンバーシップを所有しているため、ログインやその他の初期処理にデフォルトのプロジェクトを割り当てるアルゴリズムが必要です。アルゴリズムについては、`getprojent`(3C) のマニュアルページを参照してください。システムは、手順に従ってデフォルトプロジェクトを判定します。デフォルトプロジェクトが見つからない場合、ユーザーのログインまたはプロセスの開始要求は拒否されます。

システムは、次の手順でユーザーのデフォルトプロジェクトを判定します。

1. ユーザーが、`/etc/user_attr` 拡張ユーザー属性データベースで定義されている `project` 属性のエントリを持っている場合は、その `project` 属性の値がデフォルトプロジェクトになります。`user_attr(4)` のマニュアルページを参照してください。

2. user.*user-id* という名前のプロジェクトが project データベースにある場合は、そのプロジェクトがデフォルトプロジェクトになります。詳細は、project(4) のマニュアルページを参照してください。
3. group.*group-name* というプロジェクトが project データベースにあり、*group-name* がユーザーのデフォルトのグループ名 (passwd ファイルで指定されている) である場合は、そのプロジェクトがデフォルトプロジェクトになります。passwd ファイルについては、passwd(4) のマニュアルページを参照してください。
4. project データベースに default という特別なプロジェクトがある場合は、そのプロジェクトがデフォルトプロジェクトになります。

このロジックは getdefaultproj() ライブラリ関数が提供します。詳細は、getprojent(3PROJECT) のマニュアルページを参照してください。

useradd および usermod コマンドによるユーザー属性の設定

次のコマンドに -K オプションと *key=value* ペアを付けて使用すると、ローカルファイル内のユーザー属性を設定できます。

useradd　ユーザーのデフォルトプロジェクトを設定する

usermod　ユーザー情報を変更する

ローカルファイルには、次のようなものがあります。

- /etc/group
- /etc/passwd
- /etc/project
- /etc/shadow
- /etc/user_attr

NIS などのネットワークネームサービスを使ってローカルファイルに追加エントリを補足する場合、これらのコマンドでは、ネットワークネームサービスで指定された情報を変更することはできません。ただし、これらのコマンドを使用すると、次の内容を外部の「ネームサービスデータベース」と照合して検証できます。

- ユーザー名 (または役割) の一意性
- ユーザー ID の一意性
- 指定されたグループ名の存在

詳細は、useradd(1M)、usermod(1M)、および user_attr(4) のマニュアルページを参照してください。

project データベース

プロジェクトのデータは、ローカルファイル、ドメインネームシステム (DNS)、ネットワーク情報サービス (NIS) のプロジェクトマップ、または LDAP (Lightweight Directory Access Protocol) ディレクトリサービスに保存できます。/etc/project ファイルまたはネームサービスは、ログイン時、および PAM (プラグイン可能認証モジュール) によるアカウント管理に対する要求があったときに使用され、ユーザーをデフォルトのプロジェクトに結合します。

注 – プロジェクトデータベース内のエントリに対する更新は、/etc/project ファイルまたはネットワークネームサービスのデータベース表現のどちらに対するものであっても、現在アクティブなプロジェクトには適用されません。更新内容は、login コマンドまたは newtask コマンドが使用されたときに、プロジェクトに新たに参加するタスクに適用されます。詳細は、login(1) および newtask(1) のマニュアルページを参照してください。

PAM サブシステム

アカウント情報の変更や設定を行う操作には、システムへのログイン、rcp または rsh コマンドの呼び出し、ftp の使用、su の使用などがあります。アカウント情報の変更や設定が必要な場合は、設定可能なモジュール群を使用して、認証、アカウント管理、資格管理、セッション管理などを行います。

PAM の概要については、『Oracle Solaris 11.1 の管理: セキュリティーサービス』の第 14 章「プラグイン可能認証モジュールの使用」を参照してください。

ネームサービス構成

リソース管理は、ネームサービス project データベースをサポートします。project データベースが格納されている場所は、/etc/nsswitch.conf ファイルで定義されます。デフォルトでは files が最初に指定されていますが、ソースは任意の順序で指定できます。

```
project: files [nis] [ldap]
```

プロジェクト情報に対して複数のソースが指定されている場合、nsswitch.conf ファイルによりルーチンは最初に指定されているソースで情報を検索し、次にその後に続くソースを検索します。

/etc/nsswitch.conf ファイルについては、『Oracle Solaris Administration: Naming and Directory Services』の第 2 章「Name Service Switch (Overview)」および nsswitch.conf(4) を参照してください。

ローカルの /etc/project ファイルの形式

nsswitch.conf ファイルで project データベースのソースとして files を選択した場合、ログインプロセスはプロジェクト情報を /etc/project ファイルで検索します。詳細は、projects(1) および project(4) のマニュアルページを参照してください。

project ファイルには、システムによって認識されるプロジェクトごとにエントリが 1 行ずつ、次の形式で記述されています。

```
projname:projid:comment:user-list:group-list:attributes
```

フィールドの定義は次のとおりです。

projname プロジェクト名。英数字、下線 (_)、ハイフン (-)、およびピリオド (.) から成る文字列を指定します。ピリオドは、オペレーティングシステムに対して特殊な意味を持つプロジェクト用に予約されており、ユーザーのデフォルトプロジェクトの名前にだけ使用できます。*projname* にコロン (:) や改行文字を使用することはできません。

projid システムでプロジェクトに割り当てられる一意の数値 ID (PROJID)。*projid* フィールドの最大値は UID_MAX (2147483647) です。

comment プロジェクトの説明。

user-list プロジェクトへの参加を許可されたユーザーをコンマで区切ったリスト。

このフィールドではワイルドカードを使用できます。アスタリスク (*) を指定した場合、すべてのユーザーにプロジェクトへの参加を許可します。感嘆符に続けてアスタリスクを指定した場合 (!*)、すべてのユーザーのプロジェクトへの参加を拒否します。感嘆符 (!) に続けてユーザー名を指定した場合、指定されたユーザーのプロジェクトへの参加を拒否します。

group-list プロジェクトへの参加を許可されたユーザーのグループをコンマで区切ったリスト。

このフィールドではワイルドカードを使用できます。アスタリスク (*) を指定した場合、すべてのグループにプロジェクトへの参加を許可します。感嘆符に続けてアスタリスクを指定した場合 (!*)、すべてのグ

ループのプロジェクトへの参加を拒否します。感嘆符 (!) に続けてグループ名を指定した場合、指定されたグループのプロジェクトへの参加を拒否します。

attributes リソース制御など、名前と値の対をセミコロンで区切ったリスト (第 6 章「リソース制御 (概要)」を参照)。*name* は、オブジェクトに関連する属性を指定する任意の文字列です。また *value* はその属性のオプション値です。

`name[=`*value*`]`

名前と値の組で、名前に使用できるのは、文字、数字、下線、ピリオドに制限されます。リソース制御 (rctl) のカテゴリとサブカテゴリの区切り文字にはピリオドを使用します。属性名の最初の文字は英字にする必要があります。名前については大文字と小文字は区別されます。

値を、コンマと括弧を使用して構造化することにより、優先順位を設定できます。

セミコロンは、名前と値の組を区切るために使用されます。セミコロンは値の定義には使用できません。コロンは、プロジェクトフィールドを区切るために使用されます。コロンは値の定義には使用できません。

注 – このファイルを読み取るルーチンは、無効なエントリを検出すると停止します。このため、無効なエントリの後に指定されているプロジェクトの割り当てはどれも実行されません。

次に、デフォルトの `/etc/project` ファイルの例を示します。

```
system:0::::
user.root:1::::
noproject:2::::
default:3::::
group.staff:10::::
```

次に、デフォルトの `/etc/project` ファイルの最後にプロジェクトエントリを追加した例を示します。

```
system:0::::
user.root:1::::
noproject:2::::
default:3::::
group.staff:10::::
user.ml:2424:Lyle Personal:::
booksite:4113:Book Auction Project:ml,mp,jtd,kjh::
```

`/etc/project` ファイルにリソース制御と属性を追加することもできます。

- プロジェクトのリソース制御を追加する方法については、94 ページの「リソース制御の設定」を参照してください。
- リソース上限デーモン (rcapd(1M) に記載) を使用して物理メモリーリソースの上限をプロジェクトに対して定義する方法については、122 ページの「プロジェクトの物理メモリーの使用率を制限する属性」を参照してください。
- プロジェクトのエントリに project.pool 属性を追加する方法については、186 ページの「構成の作成」を参照してください。

NIS のプロジェクト構成

NIS を使用している場合は、NIS プロジェクトマップ内でプロジェクトを検索するように、/etc/nsswitch.conf ファイルで指定できます。

```
project: nis files
```

NIS マップの project.byname と project.bynumber はどちらも、次に示すように /etc/project ファイルと同じ形式です。

```
projname:projid:comment:user-list:group-list:attributes
```

詳細については、『Oracle Solaris Administration: Naming and Directory Services』の第 5 章「Network Information Service (Overview)」を参照してください。

LDAP のプロジェクト構成

LDAP を使用している場合は、LDAP project データベースでプロジェクトを検索するように、/etc/nsswitch.conf ファイルで指定できます。

```
project: ldap files
```

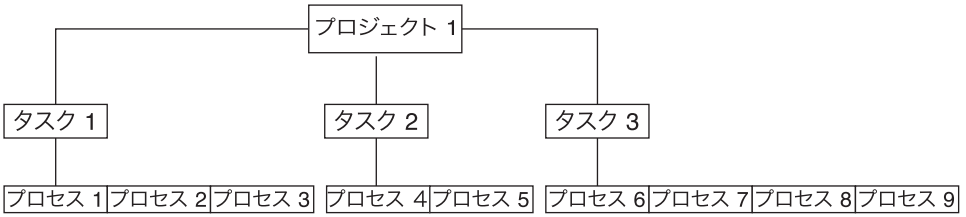
LDAP については、『Oracle Solaris Administration: Naming and Directory Services』の第 9 章「Introduction to LDAP Naming Services (Overview)」を参照してください。LDAP データベースでのプロジェクトエントリのスキーマの詳細は、『Oracle Solaris Administration: Naming and Directory Services』の「Oracle Solaris Schemas」を参照してください。

タスク識別子

プロジェクトへのログインが成功するたびに、ログインプロセスを含む新しい「タスク」が作成されます。タスクは、時間をかけて行われる一連の作業を表すプロセスです。また、タスクは「作業負荷のコンポーネント」と考えることもできます。各タスクには、自動的にタスク ID が割り当てられます。

各プロセスは 1 つのタスクのメンバーであり、各タスクは 1 つのプロジェクトに関連付けられています。

図 2–1 プロジェクトとタスクのツリー



シグナル送信のようなプロセスグループ上のすべての操作も、タスクでサポートされています。タスクを「プロセッサセット」に結合して、スケジューリングの優先順位とクラスを設定することにより、タスク内の現在のプロセスとそれに続くすべてのプロセスを変更することもできます。

プロジェクトへの参加が発生するたびに、タスクが作成されます。タスクは、次のアクション、コマンド、および関数によって作成されます。

- ログイン
- `cron`
- `newtask`
- `setproject`
- `su`

次のいずれかの方法で、最終的なタスクを作成できます。これ以降は、新しいタスクを作成しようとすると失敗します。

- `newtask` コマンドに `-F` オプションを付けて実行します。
- `project` ネームサービスデータベースで、プロジェクトに `task.final` 属性を設定します。`setproject` によってそのプロジェクト内に作成されるすべてのタスクに、`TASK_FINAL` フラグが設定されます。

詳細は、`login(1)`、`newtask(1)`、`cron(1M)`、`su(1M)`、および `setproject(3PROJECT)` のマニュアルページを参照してください。

拡張アカウンティング機能は、プロセスのアカウンティングデータを提供できます。データはタスクレベルで集計されます。

プロジェクトとタスクで使用するコマンド

次の表に示すコマンドは、プロジェクトとタスクの機能を管理するための主要なインタフェースとなります。

マニュアルページ	説明
projects(1)	ユーザーのプロジェクトメンバーシップを表示します。project データベース内のプロジェクトを一覧表示します。特定のプロジェクトについて情報を表示します。プロジェクト名が指定されていない場合は、すべてのプロジェクトについて情報を表示します。projects コマンドに -l オプションを付けて実行すると、詳細情報が出力されます。
newtask(1)	ユーザーのデフォルトのシェルまたは指定されたコマンドを実行し、指定されたプロジェクトが所有する新しいタスクに実行コマンドを配置します。また、newtask は、実行中のプロセスに結合するタスクとプロジェクトを変更するためにも使用できます。-F オプションを付けて実行すると、最終的なタスクを作成できます。
projadd(1M)	/etc/project ファイルに新しいプロジェクトエントリを追加します。projadd コマンドは、ローカルシステム上にだけプロジェクトエントリを作成します。projadd は、ネットワークネームサービスから提供される情報は変更できません。 デフォルトファイル /etc/project 以外のプロジェクトファイルを編集する場合に使用できます。project ファイルの構文検査機能を提供します。プロジェクト属性の検証と編集を行います。倍率値をサポートします。
projmod(1M)	ローカルシステム上のプロジェクトの情報を変更します。projmod は、ネットワークネームサービスから提供される情報は変更できません。ただし、このコマンドは、外部のネームサービスに対してプロジェクト名とプロジェクト ID の一意性を確認します。 デフォルトファイル /etc/project 以外のプロジェクトファイルを編集する場合に使用できます。project ファイルの構文検査機能を提供します。プロジェクト属性の検証と編集を行います。新しい属性の追加、属性の値の追加、または属性の削除に使用できます。倍率値をサポートします。 -A オプションとともに使用すると、プロジェクトデータベース内にあるリソース制御値をアクティブなプロジェクトに適用できます。project ファイル内に定義されている値に一致しない、既存の値は削除されます。

マニュアルページ	説明
projdel(1M)	ローカルシステムからプロジェクトを削除します。projdel は、ネットワークネームサービスから提供される情報は変更できません。
useradd(1M)	デフォルトプロジェクトの定義をローカルファイルに追加します。-K *key*=*value* オプションを付けて実行すると、ユーザー属性を追加したり置換したりできます。
userdel(1M)	ローカルファイルからユーザーのアカウントを削除します。
usermod(1M)	システムにあるユーザーのログイン情報を変更します。-K *key*=*value* オプションを付けて実行すると、ユーザー属性を追加したり置換したりできます。

◆ ◆ ◆ 第 3 章

プロジェクトとタスクの管理

この章では、Oracle Solaris のリソース管理のプロジェクトおよびタスクの機能の使用方法について説明します。

この章の内容は次のとおりです。

- 48 ページの「コマンドとコマンドオプションの例」
- 51 ページの「プロジェクトの管理」

プロジェクトとタスクの機能の概要については、第 2 章「プロジェクトとタスク (概要)」を参照してください。

注 – これらの機能をゾーンがインストールされている Oracle Solaris システムで使用する場合は、非大域ゾーンでこれらのコマンドを実行すると、プロセス ID を受け取るシステムコールインタフェースを通して、同じゾーン内のプロセスだけが表示されます。

プロジェクトとタスクの管理 (タスクマップ)

タスク	説明	参照先
プロジェクトとタスクで使用するコマンドとオプションの例を表示します。	タスク ID とプロジェクト ID を表示し、システムで現在実行されているプロセスとプロジェクトについて各種の統計情報を表示します。	48 ページの「コマンドとコマンドオプションの例」
プロジェクトを定義します。	`/etc/project` ファイルにプロジェクトエントリを追加し、そのエントリの値を変更します。	51 ページの「プロジェクトを定義して現在のプロジェクトを表示する方法」

タスク	説明	参照先
プロジェクトを削除します。	/etc/project ファイルからプロジェクトエントリを削除します。	53 ページの「/etc/project ファイルからプロジェクトを削除する方法」
project ファイルまたはプロジェクトデータベースを検証します。	/etc/project ファイルの構文を検査します。または、外部のネームサービスと照合してプロジェクト名およびプロジェクト ID の一意性を確認します。	54 ページの「/etc/project ファイルの内容を検証する方法」
プロジェクトのメンバーシップ情報を取得します。	起動中のプロセスの現在のプロジェクトメンバーシップを表示します。	54 ページの「プロジェクトのメンバーシップ情報を取得する方法」
新しいタスクを作成します。	newtask コマンドを使用して、特定のプロジェクトに新しいタスクを作成します。	55 ページの「新しいタスクを作成する方法」
実行中のプロセスを別のタスクとプロジェクトに関連付けます。	指定されたプロジェクト内の新しいタスク ID にプロセス番号を関連付けます。	55 ページの「実行中のプロセスを新しいタスクに移動する方法」
プロジェクト属性を追加し、操作します。	プロジェクトデータベースの管理コマンドを使用して、プロジェクト属性の追加、編集、検証、および削除を行います。	56 ページの「プロジェクト属性の編集と検証」

コマンドとコマンドオプションの例

このセクションでは、プロジェクトとタスクで使用するコマンドとオプションの例を示します。

プロジェクトとタスクで使用するコマンドオプション

ps コマンド

タスクおよびプロジェクトの ID を表示するには、ps コマンドに -o オプションを付けて実行します。たとえば、プロジェクト ID を表示するには、次のように入力します。

```
# ps -o user,pid,uid,projid
USER PID   UID  PROJID
jtd  89430 124  4113
```

id コマンド

ユーザーおよびグループ ID に加えて、現在のプロジェクト ID を表示するには、id コマンドに -p オプションを付けて実行します。*user* オペランドを指定した場合、そのユーザーの通常のログインに関連付けられたプロジェクトが表示されます。

```
#  id -p
uid=124(jtd) gid=10(staff) projid=4113(booksite)
```

pgrep コマンドと pkill コマンド

特定のリスト内のプロジェクト ID と一致するプロセスだけを表示するには、pgrep コマンドと pkill コマンドに -J オプションを付けて実行します。

```
# pgrep -J projidlist
# pkill -J projidlist
```

特定のリスト内のタスク ID と一致するプロセスだけを表示するには、pgrep コマンドと pkill コマンドに -T オプションを付けて実行します。

```
# pgrep -T taskidlist
# pkill -T taskidlist
```

prstat コマンド

システムで現在実行中のプロセスとプロジェクトのさまざまな統計情報を表示するには、prstat コマンドに -J オプションを付けて実行します。

```
% prstat -J
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
 12905 root     4472K 3640K cpu0    59    0   0:00:01 0.4% prstat/1
   829 root       43M   33M sleep   59    0   0:36:23 0.1% Xorg/1
   890 gdm        88M   26M sleep   59    0   0:22:22 0.0% gdm-simple-gree/1
   686 root     3584K 2756K sleep   59    0   0:00:34 0.0% automountd/4
     5 root       0K    0K sleep   99  -20   0:02:43 0.0% zpool-rpool/138
  9869 root       44M   17M sleep   59    0   0:02:06 0.0% poold/9
   804 root     7104K 5968K sleep   59    0   0:01:28 0.0% intrd/1
   445 root     7204K 4680K sleep   59    0   0:00:38 0.0% nscd/33
   881 gdm      7140K 5912K sleep   59    0   0:00:06 0.0% gconfd-2/1
   164 root     2572K 1648K sleep   59    0   0:00:00 0.0% pfexecd/3
   886 gdm      7092K 4920K sleep   59    0   0:00:00 0.0% bonobo-activati/2
    45 netcfg   2252K 1308K sleep   59    0   0:00:00 0.0% netcfgd/2
   142 daemon   7736K 5224K sleep   59    0   0:00:00 0.0% kcfd/3
    43 root     3036K 2020K sleep   59    0   0:00:00 0.0% dlmgmtd/5
   405 root     6824K 5400K sleep   59    0   0:00:18 0.0% hald/5
PROJID    NPROC  SWAP   RSS MEMORY      TIME  CPU PROJECT
     1        4 4728K   19M   0.9%   0:00:01 0.4% user.root
     0      111  278M  344M    17%   1:15:02 0.1% system
    10        2 1884K 9132K   0.4%   0:00:00 0.0% group.staff
     3        3 1668K 6680K   0.3%   0:00:00 0.0% default

Total: 120 processes, 733 lwps, load averages: 0.01, 0.00, 0.00
```

システムで現在実行中のプロセスとタスクのさまざまな統計情報を表示するには、prstat コマンドに -T オプションを付けて実行します。

```
% prstat -T
       PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
 12907 root     4488K 3588K cpu0    59    0   0:00:00 0.3% prstat/1
   829 root       43M   33M sleep   59    0   0:36:24 0.1% Xorg/1
   890 gdm        88M   26M sleep   59    0   0:22:22 0.0% gdm-simple-gree/1
  9869 root       44M   17M sleep   59    0   0:02:06 0.0% poold/9
     5 root        0K    0K sleep   99  -20   0:02:43 0.0% zpool-rpool/138
   445 root     7204K 4680K sleep   59    0   0:00:38 0.0% nscd/33
   881 gdm      7140K 5912K sleep   59    0   0:00:06 0.0% gconfd-2/1
   164 root     2572K 1648K sleep   59    0   0:00:00 0.0% pfexecd/3
   886 gdm      7092K 4920K sleep   59    0   0:00:00 0.0% bonobo-activati/2
    45 netcfg   2252K 1308K sleep   59    0   0:00:00 0.0% netcfgd/2
   142 daemon   7736K 5224K sleep   59    0   0:00:00 0.0% kcfd/3
    43 root     3036K 2020K sleep   59    0   0:00:00 0.0% dlmgmtd/5
   405 root     6824K 5400K sleep   59    0   0:00:18 0.0% hald/5
   311 root     3488K 2512K sleep   59    0   0:00:00 0.0% picld/4
   409 root     4356K 2768K sleep   59    0   0:00:00 0.0% hald-addon-cpuf/1
TASKID    NPROC  SWAP   RSS MEMORY      TIME  CPU PROJECT
  1401        2 2540K 8120K   0.4%   0:00:00 0.3% user.root
    94       15   84M  162M   7.9%   0:59:37 0.1% system
   561        1   37M   24M   1.2%   0:02:06 0.0% system
     0        2    0K    0K   0.0%   0:02:47 0.0% system
    46        1 4224K 5524K   0.3%   0:00:38 0.0% system
Total: 120 processes, 733 lwps, load averages: 0.01, 0.00, 0.00
```

注 – -J オプションと -T オプションを一緒に使用することはできません。

プロジェクトとタスクでの cron と su の使用

cron コマンド

cron コマンドは、settaskid を発行し、実行を要求したユーザーの適切なデフォルトプロジェクトを使用して、cron、at、および batch の各ジョブが別のタスクで実行されるようにします。また、at および batch コマンドは、現在のプロジェクト ID を取得して at ジョブを実行するときにプロジェクト ID が復元されるようにします。

su コマンド

su コマンドは、ログインのシミュレーションの一環として、新しいタスクを作成することによってターゲットユーザーのデフォルトプロジェクトに参加します。

su コマンドを使用してユーザーのデフォルトプロジェクトを切り替えるには、次のように入力します。

```
# su - user
```

プロジェクトの管理

▼ プロジェクトを定義して現在のプロジェクトを表示する方法

次の例は、`projadd` コマンドを使用してプロジェクトエントリを追加し、`projmod` コマンドを使用してそのエントリを変更する方法を示します。

1 **root** になるか、同等の役割になります。

2 **`projects -l`** を使用して、システムのデフォルトの **`/etc/project`** ファイルを表示します。

```
# projects -l
system
        projid : 0
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
user.root
        projid : 1
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
noproject
        projid : 2
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
default
        projid : 3
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
group.staff
        projid : 10
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
```

3 *booksite* という名前のプロジェクトを追加します。追加したプロジェクトを *mark* という名前のユーザーにプロジェクト **ID** 番号 *4113* で割り当てます。

```
# projadd -U mark -p 4113 booksite
```

4 再度 **/etc/project** ファイルを表示します。

```
# projects -l
system
        projid : 0
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
user.root
        projid : 1
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
noproject
        projid : 2
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
default
        projid : 3
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
group.staff
        projid : 10
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
booksite
        projid : 4113
        comment: ""
        users  : mark
        groups : (none)
        attribs:
```

5 **comment** フィールドにプロジェクトを説明するコメントを追加します。

```
# projmod -c 'Book Auction Project' booksite
```

6 **/etc/project** ファイルに加えた変更を確認します。

```
# projects -l
system
        projid : 0
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
user.root
        projid : 1
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
```

```
noproject
        projid : 2
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
default
        projid : 3
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
group.staff
        projid : 10
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
booksite
        projid : 4113
        comment: "Book Auction Project"
        users  : mark
        groups : (none)
        attribs:
```

参照　プロジェクト、タスク、およびプロセスをプールに結合する方法については、180 ページの「プール属性の設定とプールへの結合」を参照してください。

▼ /etc/project ファイルからプロジェクトを削除する方法

次の例は、projdel コマンドを使ってプロジェクトを削除する方法を示します。

1. **root** になるか、同等の役割になります。
2. **projdel** コマンドを使ってプロジェクト *booksite* を削除します。

   ```
   # projdel booksite
   ```

3. **/etc/project** ファイルを表示します。

   ```
   # projects -l
   system
           projid : 0
           comment: ""
           users  : (none)
           groups : (none)
           attribs:
   user.root
           projid : 1
           comment: ""
           users  : (none)
           groups : (none)
   ```

```
        attribs:
noproject
        projid : 2
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
default
        projid : 3
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
group.staff
        projid : 10
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
```

4 ユーザー名 *mark* でログインして、**projects** と入力し、このユーザーに割り当てられているプロジェクトを表示します。

```
# su - mark
# projects
default
```

/etc/project ファイルの内容を検証する方法

編集オプションが指定されていない場合、projmod コマンドは project ファイルの内容を検証します。

NIS マップを検証するには、次のように入力します。

```
# ypcat project | projmod -f –
```

/etc/project ファイルの構文を検査するには、次のように入力します。

```
# projmod -n
```

プロジェクトのメンバーシップ情報を取得する方法

-p フラグを付けて id コマンドを使用し、起動中のプロセスの現在のプロジェクトメンバーシップを表示します。

```
$ id -p
uid=100(mark) gid=1(other) projid=3(default)
```

▼ 新しいタスクを作成する方法

1 次の例では、宛先となるプロジェクト *booksite* のメンバーとしてログインします。

2 *booksite* プロジェクト内に新しいタスクを作成します。 それには、システムのタスク **ID** を取得するための **-v** (冗長) オプションを指定して **newtask** コマンドを実行します。

```
machine% newtask -v -p booksite
16
```

newtask を実行すると、指定したプロジェクト内に新しいタスクが作成され、そのタスクにユーザーのデフォルトのシェルが置かれます。

3 起動中のプロセスの現在のプロジェクトメンバーシップを表示します。

```
machine% id -p
uid=100(mark) gid=1(other) projid=4113(booksite)
```

今度は、プロセスが新しいプロジェクトのメンバーになっています。

▼ 実行中のプロセスを新しいタスクに移動する方法

次の例は、実行中のプロセスを別のタスクと新しいプロジェクトに関連付ける方法を示します。この操作を実行するには、root ユーザーになるか、必要な権利プロファイルを持っているか、またはプロセスの所有者で新しいプロジェクトのメンバーである必要があります。

1 **root** になるか、同等の役割になります。

注 – プロセスの所有者または新しいプロジェクトのメンバーであれば、この手順は省略できます。

---

2 *book_catalog* プロセスのプロセス **ID** を取得します。

```
# pgrep book_catalog
    8100
```

3 プロセス *8100* を、新しいタスク **ID** を使って *booksite* プロジェクトに関連付けます。

```
# newtask -v -p booksite -c 8100
    17
```

-c オプションは、newtask が指定された既存のプロセスに対して動作することを指定します。

4 タスクとプロセス **ID** の対応を確認します。

```
# pgrep -T 17
    8100
```

プロジェクト属性の編集と検証

プロジェクトデータベースの管理コマンド projadd および projmod を使用して、プロジェクト属性を編集できます。

-K オプションは、属性の置換リストを指定します。属性はセミコロン (;) で区切られます。-K オプションを -a オプションとともに使用すると、その属性または属性値が追加されます。-K オプションを -r オプションとともに使用すると、その属性または属性値が削除されます。-K オプションを -s オプションとともに使用すると、その属性または属性値が置換されます。

▼ 属性と属性値をプロジェクトに追加する方法

プロジェクトの属性に値を追加するには、projmod コマンドに -a オプションと -K オプションを付けて実行します。属性が存在しない場合は、新たに作成されます。

1 **root** になるか、同等の役割になります。

2 プロジェクト **myproject** 内に *task.max-lwps* リソース制御属性を値なしで追加します。プロジェクトに加わるタスクでは、その属性にシステム値だけが設定されます。

```
# projmod -a -K task.max-lwps myproject
```

3 その後、プロジェクト **myproject** 内の *task.max-lwps* に値を追加できます。この値は、特権レベル、しきい値、およびしきい値に達したときのアクションから成ります。

```
# projmod -a -K "task.max-lwps=(priv,100,deny)" myproject
```

4 リソース制御は複数の値を持つことができるので、同じオプションを使用して、既存の値リストに別の値を追加できます。

```
# projmod -a -K "task.max-lwps=(priv,1000,signal=KILL)" myproject
```

複数の値はコンマで区切られます。task.max-lwps エントリはこの時点で次のようになっています。

```
task.max-lwps=(priv,100,deny),(priv,1000,signal=KILL)
```

▼ 属性値をプロジェクトから削除する方法

この手順では次の値を使用します。

```
task.max-lwps=(priv,100,deny),(priv,1000,signal=KILL)
```

1 **root** になるか、同等の役割になります。

2 プロジェクト *myproject* 内のリソース制御 **task.max-lwps** から属性値を削除するには、**projmod** コマンドに **-r** オプションと **-K** オプションを付けて実行します。

```
# projmod -r -K "task.max-lwps=(priv,100,deny)" myproject
```

task.max-lwps が次のように複数の値を持っているとします。

```
task.max-lwps=(priv,100,deny),(priv,1000,signal=KILL)
```

この場合は、最初に一致する値が削除されます。したがって、結果は次のようになります。

```
task.max-lwps=(priv,1000,signal=KILL)
```

▼ リソース制御属性をプロジェクトから削除する方法

リソース制御 task.max-lwps をプロジェクト *myproject* から削除するには、projmod コマンドに -r オプションと -K オプションを付けて実行します。

1 **root** になるか、同等の役割になります。

2 属性 **task.max-lwps** とそのすべての値を、プロジェクト *myproject* から削除します。

```
# projmod -r -K task.max-lwps myproject
```

▼ プロジェクトの属性と属性値を置換する方法

プロジェクト *myproject* 内のリソース制御 task.max-lwps の属性値を別の値で置換するには、projmod コマンドに -s オプションと -K オプションを付けて実行します。属性が存在しない場合は、新たに作成されます。

1 **root** になるか、同等の役割になります。

2 **task.max-lwps** の現在の値を、次に示す新しい値で置換します。

```
# projmod -s -K "task.max-lwps=(priv,100,none),(priv,120,deny)" myproject
```

結果は次のようになります。

```
task.max-lwps=(priv,100,none),(priv,120,deny)
```

▼ リソース制御属性の既存の値を削除する方法

1 **root** になるか、同等の役割になります。

2 **task.max-lwps** の現在の値をプロジェクト *myproject* から削除するには、次のように入力します。

```
# projmod -s -K task.max-lwps myproject
```

◆ ◆ ◆ 第 4 章

拡張アカウンティング (概要)

プロジェクトおよびタスク機能 (第 2 章「プロジェクトとタスク (概要)」で説明) を使って作業負荷にラベル付けを行い、分離することにより、作業負荷ごとのリソース消費を監視できます。「拡張アカウンティング」サブシステムを使用すると、プロセスとタスクの両方について詳細なリソース消費統計情報を取得できます。

この章の内容は次のとおりです。

- 59 ページの「拡張アカウンティングの紹介」
- 60 ページの「拡張アカウンティングの動作」
- 62 ページの「拡張アカウンティング構成」
- 63 ページの「拡張アカウンティングで使用されるコマンド」
- 64 ページの「libexacct に対する Perl インタフェース」

拡張アカウンティングの使用を開始するには、68 ページの「フロー、プロセス、タスク、およびネットワークコンポーネントの拡張アカウンティングを起動する方法」に進んでください。

拡張アカウンティングの紹介

拡張アカウンティングサブシステムは、行われた作業の対象プロジェクトの使用状況レコードにラベル付けします。また、拡張アカウンティングを IPQoS (Internet Protocol Quality of Service、IP サービス品質) フローアカウンティングモジュール (『Oracle Solaris 11.1 での IP サービス品質の管理』の第 5 章「フローアカウンティングの使用と統計情報の収集 (タスク)」に記載) と組み合わせて、システム上のネットワークフロー情報を取得することもできます。

リソース管理メカニズムを適用する前に、まず、さまざまな作業負荷がシステムに対して行うリソース消費要求の特徴をつかむ必要があります。Oracle Solaris オペ

レーティングシステムの拡張アカウンティング機能には、次のものに関するシステムおよびネットワークのリソース消費量を柔軟に記録するための方法が備わっています。

- タスク。
- プロセス数。
- IPQoS `flowacct` モジュールで提供されるセレクタ。詳細は、`ipqos`(7IPP) のマニュアルページを参照してください。
- ネットワーク管理。`dladm`(1M) および `flowadm`(1M) を参照してください。

システムの使用状況をリアルタイムで計測するオンライン監視ツールとは異なり、拡張アカウンティング機能を使用すると、使用状況の履歴を調べることができます。その上で、将来の作業負荷の容量要件を算定できます。

拡張アカウンティングのデータを使用すれば、リソースの課金、作業負荷の監視、容量計画などの目的でソフトウェアを開発したり購入したりできます。

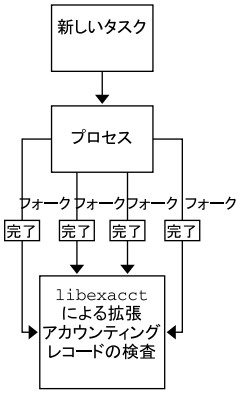
拡張アカウンティングの動作

Oracle Solaris オペレーティングシステムの拡張アカウンティング機能は、バージョン番号が付けられた拡張可能なファイル形式を使用してアカウンティングデータを格納します。このデータ形式を使用するファイルは、添付のライブラリ `libexacct` (`libexacct`(3LIB) のマニュアルページを参照) で提供される API を使ってアクセスまたは作成できます。作成されたファイルは、拡張アカウンティング機能を使用できる任意のプラットフォーム上で解析でき、データを容量計画や課金に使用できます。

拡張アカウンティングを起動すると、`libexacct` API で調べることができる統計情報が収集されます。`libexacct` は、`exacct` ファイルを前後どちらの方向からでも検査できます。API は、カーネルが作成するファイルだけでなく、`libexacct` によって生成されたサードパーティーのファイルもサポートします。`libexacct` に対する Perl (Practical Extraction and Report Language) インタフェースが用意されています。これを使えば、報告および抽出用のカスタムスクリプトを開発できます。64 ページの「`libexacct` に対する Perl インタフェース」を参照してください。

たとえば、拡張アカウンティングを有効にすると、タスクは、自分のメンバープロセスの総リソース使用状況を追跡します。タスクのアカウンティングレコードは、そのタスクの完了時に書き込まれます。実行中のプロセスやタスクについて中間レコードを書き込むこともできます。タスクの詳細については、第 2 章「プロジェクトとタスク (概要)」を参照してください。

図 4–1 拡張アカウンティング起動時のタスクの追跡



拡張可能な形式

拡張アカウンティングの形式は、レガシーシステムのアカウンティングソフトウェアの形式よりも拡張可能性が非常に高いです。拡張アカウンティングでは、システムアカウンティングメトリックスのシステムへの追加や削除をシステムの解放時またはシステムの操作中に行うことができます。

注 – 拡張アカウンティングと古いシステムのアカウンティングソフトウェアの両方をシステム上で同時に起動できます。

---

exacct レコードとその形式

exacct レコードを作成するルーチンは、次の 2 つの目的で使用できます。

- サードパーティーの exacct ファイルを作成できるようにします。
- putacct システムコールを使用してカーネルアカウンティングファイルに埋め込むためのタグ付けレコードを作成できるようにします (getacct(2) を参照)。

  注 – Perl インタフェースから putacct システムコールを利用することもできます。

この形式では、すべての変更を明示的なバージョン変更にしなくても、さまざまな形式のアカウンティングレコードを取得できます。アカウンティングデータを使用するアプリケーションは、認識不可能なレコードを無視するように作成する必要があります。

libexacct ライブラリは、ファイルを exacct 形式に変換し、その形式のファイルを生成します。このライブラリは、exacct 形式のファイルに対するインタフェースとしてサポートされている唯一のインタフェースです。

注 – getacct、putacct、wracct の各システムコールは、フローには適用されません。IPQoS フローアカウンティングの構成時には、カーネルによってフローレコードが作成され、ファイルに書き込まれます。

ゾーンがインストールされている Oracle Solaris システムでの拡張アカウンティングの使用

拡張アカウンティングサブシステムを大域ゾーンで実行した場合、非大域ゾーンを含むシステム全体の情報が収集および報告されます。また、大域管理者または zonecfg ユーティリティーを使用して適切な承認が許可されたユーザーは、リソースの消費量をゾーン単位で決めることもできます。詳細は、377 ページの「ゾーンがインストールされているシステムでの拡張アカウンティング」を参照してください。

拡張アカウンティング構成

ディレクトリの /var/adm/exacct は、拡張アカウンティングのデータを配置するための標準の場所です。acctadm コマンドを使用すると、プロセスやタスクのアカウンティングデータファイルの格納場所を変更できます。詳細は、acctadm(1M) のマニュアルページを参照してください。

拡張アカウンティングの起動および持続的な有効化

acctadm コマンド (acctadm(1M) に記載) は、Oracle Solaris サービス管理機能 (SMF) サービス (smf(5) に記載) を使用して拡張アカウンティングを起動します。

拡張アカウンティングの構成は、SMF リポジトリに格納されます。この構成は、ブート時にサービスインスタンス (アカウンティングタイプごとに 1 つのサービスインスタンス) によって元に戻されます。個々の拡張アカウンティングタイプは、次のように SMF サービスの別々のインスタンスによって表されます。

svc:/system/extended-accounting:flow
　フローアカウンティング

svc:/system/extended-accounting:process
　プロセスアカウンティング

svc:/system/extended-accounting:task
　タスクアカウンティング

svc:/system/extended-accounting:net
　ネットワークアカウンティング

acctadm(1M) を使用して拡張アカウンティングを有効にすると、対応するサービスインスタンスが現在有効でない場合に有効にされます。この結果、拡張アカウンティング構成は次回のブート時に元に戻されます。同様に、この構成によってあるサービスに対するアカウンティングが無効にされると、対応するサービスインスタンスも無効にされます。このインスタンスは、必要に応じて acctadm によって有効または無効にされます。

あるリソースに対する拡張アカウンティングを持続的に有効にするには、次のようにコマンドを実行します。

```
# acctadm -e resource_list
```

resource_list は、リソースまたはリソースグループのコンマ区切りリストです。

レコード

acctadm コマンドは、/var/adm/exacct 内の既存のファイルに新しいレコードを追加します。

拡張アカウンティングで使用されるコマンド

コマンド	説明
acctadm(1M)	拡張アカウンティング機能の各種属性の変更、拡張アカウンティングの停止と起動を行います。また、プロセス、タスク、フロー、およびネットワークを追跡するためのアカウンティング属性を選択するためにも使用します。
wracct(1M)	アクティブなプロセスおよびタスクの拡張アカウンティングアクティビティーを書き込みます。
lastcomm(1)	直前に呼び出されたコマンドを表示します。lastcomm では、標準アカウンティングプロセスのデータまたは拡張アカウンティングプロセスのデータのどちらかを使用できます。

タスクやプロジェクトに関連するコマンドについては、48 ページの「コマンドとコマンドオプションの例」を参照してください。IPQoS フローアカウンティングについては、ipqosconf(1M) のマニュアルページおよび『Oracle Solaris 11.1 での IP サービス品質の管理』の第 5 章「フローアカウンティングの使用と統計情報の収集 (タスク)」を参照してください。

libexacct に対する Perl インタフェース

Perl インタフェースによって、exacct フレームワークで作成されたアカウンティングファイルを読み取ることのできる、Perl スクリプトを作成できます。exacct ファイルを作成する Perl スクリプトも作成できます。

このインタフェースの機能は、ベースとなる C 言語の API と同様です。可能な場合は、ベースとなる C 言語の API から取得したデータを Perl データタイプとして表示します。このインタフェースを使用すると、データへのアクセスが容易になり、またバッファーでのパックとアンパックの操作が不要になります。さらに、あらゆるメモリー管理が Perl ライブラリによって実行されます。

各種のプロジェクト、タスク、exacct 関連機能はいくつかのグループに分けられます。各機能グループは、別々の Perl モジュールに配置されます。各モジュールは、Oracle Solaris の標準の Sun::Solaris:: Perl パッケージ接頭辞で始まります。Perl exacct ライブラリが提供するクラスはすべて、Sun::Solaris::Exacct モジュールの下にあります。

配下の libexacct(3LIB) ライブラリは、exacct 形式のファイル、カタログタグ、および exacct オブジェクトに対する操作を実行します。exacct オブジェクトは、次の 2 つのタイプに分けられます。

- アイテム (単一データ値 [スカラー])
- グループ (項目のリスト)

次の表に各モジュールの概要を示します。

モジュール (空白文字は使用不可)	説明	参照先
Sun::Solaris::Project	このモジュールは、次のプロジェクト操作関数にアクセスする機能を提供します。getprojid(2)、 endprojent(3PROJECT)、fgetprojent(3PROJECT)、getdefaultproj(3PROJECT)、getprojbyid(3PROJECT)、getprojbyname(3PROJECT)、getprojent(3PROJECT)、getprojidbyname(3PROJECT)、inproj(3PROJECT)、project_walk(3PROJECT)、setproject(3PROJECT)、 および setprojent(3PROJECT) です。	Project(3PERL)
Sun::Solaris::Task	このモジュールは、タスク操作関数である gettaskid(2) と settaskid(2) にアクセスする機能を提供します。	Task(3PERL)

モジュール (空白文字は使用不可)	説明	参照先
Sun::Solaris::Exacct	最上位レベルの exacct モジュール。このモジュールは、exacct 関連のシステムコールである getacct(2)、putacct(2)、および wracct(2) にアクセスする機能を提供します。このモジュールは、libexacct(3LIB) ライブラリ関数である ea_error(3EXACCT) にアクセスする機能も提供します。exacct EO_*、EW_*、EXR_*、P_*、および TASK_* マクロのすべてに対応する定数も、このモジュールで提供されます。	Exacct(3PERL)
Sun::Solaris::Exacct:: Catalog	このモジュールは、exacct カタログタグ内のビットフィールドにアクセスする、オブジェクト指向型メソッドを提供します。このモジュールによって、EXC_*、EXD_*、および EXD_* マクロの定数にもアクセスできます。	Exacct::Catalog(3PERL)
Sun::Solaris::Exacct:: File	このモジュールは、次の libexacct アカウンティングファイル関数にアクセスする、オブジェクト指向型メソッドを提供します。ea_open(3EXACCT)、ea_close(3EXACCT)、ea_get_creator(3EXACCT)、ea_get_hostname(3EXACCT)、ea_next_object(3EXACCT)、ea_previous_object(3EXACCT)、および ea_write_object(3EXACCT) です。	Exacct::File(3PERL)
Sun::Solaris::Exacct:: Object	このモジュールは、個々の exacct アカウンティングファイルオブジェクトにアクセスする、オブジェクト指向型メソッドを提供します。exacct オブジェクトは、該当する Sun::Solaris::Exacct::Object サブクラスに与えられた、隠された参照として表されます。このモジュールはさらに、アイテムかグループかのオブジェクトタイプに分けられます。このレベルで、ea_match_object_catalog(3EXACCT) および ea_attach_to_object(3EXACCT) 関数にアクセスするメソッドがあります。	Exacct::Object(3PERL)
Sun::Solaris::Exacct:: Object::Item	このモジュールは、独立した exacct アカウンティングファイルアイテムにアクセスする、オブジェクト指向型メソッドを提供します。このタイプのオブジェクトは、Sun::Solaris::Exacct::Object から継承します。	Exacct::Object::Item(3PERL)

モジュール (空白文字は使用不可)	説明	参照先
Sun::Solaris::Exacct::Object::Group	このモジュールは、独立した exacct アカウンティングファイルグループにアクセスする、オブジェクト指向型メソッドを提供します。このタイプのオブジェクトは、Sun::Solaris::Exacct::Object から継承します。これらのオブジェクトによって、ea_attach_to_group(3EXACCT) 関数にアクセスできます。グループ内のアイテムは Perl 配列として表されます。	Exacct::Object::Group(3PERL)
Sun::Solaris::Kstat	このモジュールは、kstat 機能に対する Perl のタイハッシュインタフェースを提供します。このモジュールの使用例については、Perl で記述された /bin/kstat を参照してください。	Kstat(3PERL)

表で説明したモジュールの使用例については、71 ページの「libexacct に対する Perl インタフェースの使用」を参照してください。

◆ ◆ ◆ 第 5 章

拡張アカウンティングの管理 (タスク)

この章では、拡張アカウンティングサブシステムを管理する方法について説明します。

拡張アカウンティングサブシステムの概要については、第 4 章「拡張アカウンティング (概要)」を参照してください。

拡張アカウンティング機能の管理 (タスクマップ)

タスク	説明	参照先
拡張アカウンティング機能を起動します。	拡張アカウンティングを使用して、システムで実行されている各プロジェクトのリソース消費を監視します。「拡張アカウンティング」サブシステムを使用すると、タスク、プロセス、およびフローの履歴データを取り込むことができます。	68 ページの「フロー、プロセス、タスク、およびネットワークコンポーネントの拡張アカウンティングを起動する方法」
拡張アカウンティングのステータスを表示します。	拡張アカウンティング機能のステータスを調べます。	69 ページの「拡張アカウンティングステータスを表示する方法」
使用可能なアカウンティングリソースを表示します。	システム上の使用可能なアカウンティングリソースを表示します。	69 ページの「使用可能なアカウンティングリソースを表示する方法」
フロー、プロセス、タスク、およびネットのアカウンティングインスタンスを停止します。	拡張アカウンティング機能をオフにします。	70 ページの「プロセス、タスク、フロー、およびネットワーク管理のアカウンティングを停止する方法」

タスク	説明	参照先
拡張アカウンティング機能に対する Perl インタフェースを使用します。	Perl インタフェースを使用して、報告および抽出用のカスタムスクリプトを作成します。	71 ページの「libexacct に対する Perl インタフェースの使用」

拡張アカウンティング機能の使用

ユーザーは、次の管理対象のアカウンティングタイプに対する権利プロファイルがあれば、拡張アカウンティングの管理 (アカウンティングの開始、アカウンティングの停止、およびアカウンティング構成パラメータの変更) を行うことができます。

- 拡張アカウンティングのフロー管理
- プロセス管理
- タスク管理
- ネットワーク管理

▼ フロー、プロセス、タスク、およびネットワークコンポーネントの拡張アカウンティングを起動する方法

タスク、プロセス、フロー、およびネットワークコンポーネントの拡張アカウンティング機能を起動するには、`acctadm` コマンドを使用します。`acctadm` の最後に付けられたオプションのパラメータは、このコマンドが、拡張アカウンティング機能のフロー、プロセス、システムタスク、またはネットワークのいずれのアカウンティングコンポーネントに作用するかを示します。

注 – 役割には、認証と特権コマンドが含まれます。Oracle Solaris の役割に基づくアクセス制御 (RBAC) 機能を使用して、役割を作成する方法およびユーザーに役割を割り当てる方法については、『Oracle Solaris 11.1 の管理: セキュリティーサービス』のパート III「役割、権利プロファイル、特権」を参照してください。

---

1. **root** になるか、同等の役割になります。
2. プロセスの拡張アカウンティングを起動します。

   ```
   # acctadm -e extended -f /var/adm/exacct/proc process
   ```

3. タスクの拡張アカウンティングを起動します。

   ```
   # acctadm -e extended,mstate -f /var/adm/exacct/task task
   ```

4. フローの拡張アカウンティングを起動します。

   ```
   # acctadm -e extended -f /var/adm/exacct/flow flow
   ```

5 ネットワークの拡張アカウンティングを起動します。

```
# acctadm -e extended -f /var/adm/exacct/net net
```

dladm コマンドと flowadm コマンドによって管理されるリンクとフローに対して acctadm を実行します。

参照 詳細は、acctadm(1M) のマニュアルページを参照してください。

拡張アカウンティングステータスを表示する方法

引数なしで acctadm と入力すると、拡張アカウンティング機能の現在のステータスが表示されます。

```
machine% acctadm
                Task accounting: active
           Task accounting file: /var/adm/exacct/task
          Tracked task resources: extended
        Untracked task resources: none
              Process accounting: active
         Process accounting file: /var/adm/exacct/proc
       Tracked process resources: extended
     Untracked process resources: host
                Flow accounting: active
           Flow accounting file: /var/adm/exacct/flow
          Tracked flow resources: extended
        Untracked flow resources: none
```

この例では、システムタスクアカウンティングが拡張モードと mstate モードで動作しています。プロセスアカウンティングとフローアカウンティングは、拡張モードで動作しています。

注 – 拡張アカウンティングの分野では、マイクロステート (mstate) は、プロセス状態の微小な変化を反映した拡張データを意味し、このデータはプロセス使用状況ファイルで利用できます (proc(4) のマニュアルページを参照)。このデータは、プロセスの活動に関して、基本レコードや拡張レコードよりも非常に詳細な情報を提供します。

使用可能なアカウンティングリソースを表示する方法

使用可能なリソースは、システムやプラットフォームによってさまざまです。acctadm コマンドに -r オプションを付けて実行すると、システム上の使用可能なアカウンティングリソースグループを表示できます。

```
machine% acctadm -r
process:
extended pid,uid,gid,cpu,time,command,tty,projid,taskid,ancpid,wait-status,zone,flag,
memory,mstate     displays as one line
basic    pid,uid,gid,cpu,time,command,tty,flag
task:
extended taskid,projid,cpu,time,host,mstate,anctaskid,zone
basic    taskid,projid,cpu,time
flow:
extended
saddr,daddr,sport,dport,proto,dsfield,nbytes,npkts,action,ctime,lseen,projid,uid
basic    saddr,daddr,sport,dport,proto,nbytes,npkts,action
net:
  extended name,devname,edest,vlan_tpid,vlan_tci,sap,cpuid, \
  priority,bwlimit,curtime,ibytes,obytes,ipkts,opks,ierrpkts \
  oerrpkts,saddr,daddr,sport,dport,protocol,dsfield
  basic    name,devname,edest,vlan_tpid,vlan_tci,sap,cpuid, \
  priority,bwlimit,curtime,ibytes,obytes,ipkts,opks,ierrpkts \
  oerrpkts
```

▼ プロセス、タスク、フロー、およびネットワーク管理のアカウンティングを停止する方法

プロセス、タスク、フロー、およびネットワークのアカウンティングを停止するには、それぞれを個別にオフにします。これには、`-x` オプションを指定した `acctadm` コマンドを使用します。

1. **root** になるか、同等の役割になります。
2. プロセスアカウンティングをオフにします。

   ```
   # acctadm -x process
   ```

3. タスクアカウンティングをオフにします。

   ```
   # acctadm -x task
   ```

4. フローアカウンティングをオフにします。

   ```
   # acctadm -x flow
   ```

5. ネットワーク管理アカウンティングをオフにします。

   ```
   # acctadm -x net
   ```

6. タスクアカウンティング、プロセスアカウンティング、フローアカウンティング、およびネットワークアカウンティングがオフになったことを確認します。

   ```
   # acctadm
          Task accounting: inactive
   ```

```
       Task accounting file: none
     Tracked task resources: none
   Untracked task resources: extended
         Process accounting: inactive
    Process accounting file: none
  Tracked process resources: none
Untracked process resources: extended
            Flow accounting: inactive
       Flow accounting file: none
     Tracked flow resources: none
   Untracked flow resources: extended
            Net accounting: inactive
       Net accounting file: none
     Tracked Net resources: none
   Untracked Net resources: extended
```

libexacct に対する Perl インタフェースの使用

exacct オブジェクトの内容を再帰的に出力する方法

exacct オブジェクトの内容を再帰的に出力するには、次のコードを使用します。この機能は、ライブラリによって Sun::Solaris::Exacct::Object::dump() 関数として提供されています。ea_dump_object() という簡易関数でこの機能を利用することもできます。

```
sub dump_object
     {
             my ($obj, $indent) = @_;
             my $istr = ’  ’ x $indent;

             #
             # Retrieve the catalog tag.  Because we are
             # doing this in an array context, the
             # catalog tag will be returned as a (type, catalog, id)
             # triplet, where each member of the triplet will behave as
             # an integer or a string, depending on context.
             # If instead this next line provided a scalar context, e.g.
             #    my $cat  = $obj->catalog()->value();
             # then $cat would be set to the integer value of the
             # catalog tag.
             #
             my @cat = $obj->catalog()->value();

             #
             # If the object is a plain item
             #
             if ($obj->type() == &EO_ITEM) {
                     #
                     # Note: The ’%s’ formats provide s string context, so
```

```
                        # the components of the catalog tag will be displayed
                        # as the symbolic values. If we changed the '%s'
                        # formats to '%d', the numeric value of the components
                        # would be displayed.
                        #
                        printf("%sITEM\n%s  Catalog = %s|%s|%s\n",
                           $istr, $istr, @cat);
                        $indent++;

                        #
                        # Retrieve the value of the item.  If the item contains
                        # in turn a nested exacct object (i.e., an item or
                        # group),then the value method will return a reference
                        # to the appropriate sort of perl object
                        # (Exacct::Object::Item or Exacct::Object::Group).
                        # We could of course figure out that the item contained
                        # a nested item orgroup by examining the catalog tag in
                        # @cat and looking for a type of EXT_EXACCT_OBJECT or
                        # EXT_GROUP.
                        #
                        my $val = $obj->value();
                        if (ref($val)) {
                                # If it is a nested object, recurse to dump it.
                                dump_object($val, $indent);
                        } else {
                                # Otherwise it is just a 'plain' value, so
                                # display it.
                                printf("%s  Value = %s\n", $istr, $val);
                        }

                #
                # Otherwise we know we are dealing with a group.  Groups
                # represent contents as a perl list or array (depending on
                # context), so we can process the contents of the group
                # with a 'foreach' loop, which provides a list context.
                # In a list context the value method returns the content
                # of the group as a perl list, which is the quickest
                # mechanism, but doesn't allow the group to be modified.
                # If we wanted to modify the contents of the group we could
                # do so like this:
                #    my $grp = $obj->value();   # Returns an array reference
                #    $grp->[0] = $newitem;
                # but accessing the group elements this way is much slower.
                #
                } else {
                        printf("%sGROUP\n%s  Catalog = %s|%s|%s\n",
                            $istr, $istr, @cat);
                        $indent++;
                        # 'foreach' provides a list context.
                        foreach my $val ($obj->value()) {
                                dump_object($val, $indent);
                        }
                        printf("%sENDGROUP\n", $istr);
                }
        }
```

新しいグループレコードを作成してファイルに書き込む方法

新しいグループレコードを作成して /tmp/exacct というファイルに書き込むには、次のスクリプトを使用します。

```
#!/usr/bin/perl

use strict;
use warnings;
use Sun::Solaris::Exacct qw(:EXACCT_ALL);
# Prototype list of catalog tags and values.
     my @items = (
             [ &EXT_STRING | &EXC_DEFAULT | &EXD_CREATOR      => "me"       ],
             [ &EXT_UINT32 | &EXC_DEFAULT | &EXD_PROC_PID     => $$         ],
             [ &EXT_UINT32 | &EXC_DEFAULT | &EXD_PROC_UID     => $<         ],
             [ &EXT_UINT32 | &EXC_DEFAULT | &EXD_PROC_GID     => $(         ],
             [ &EXT_STRING | &EXC_DEFAULT | &EXD_PROC_COMMAND => "/bin/rec" ],
     );

     # Create a new group catalog object.
     my $cat = ea_new_catalog(&EXT_GROUP | &EXC_DEFAULT | &EXD_NONE)

     # Create a new Group object and retrieve its data array.
     my $group = ea_new_group($cat);
     my $ary = $group->value();

     # Push the new Items onto the Group array.
     foreach my $v (@items) {
             push(@$ary, ea_new_item(ea_new_catalog($v->[0]), $v->[1]));
     }

     # Open the exacct file, write the record & close.
     my $f = ea_new_file('/tmp/exacct', &O_RDWR | &O_CREAT | &O_TRUNC)
        || die("create /tmp/exacct failed: ", ea_error_str(), "\n");
     $f->write($group);
     $f = undef;
```

exacct ファイルの内容を出力する方法

exacct ファイルの内容を出力するには、次の Perl スクリプトを使用します。

```
#!/usr/bin/perl

     use strict;
     use warnings;
     use Sun::Solaris::Exacct qw(:EXACCT_ALL);

     die("Usage is dumpexacct <exacct file>\n") unless (@ARGV == 1);

     # Open the exacct file and display the header information.
     my $ef = ea_new_file($ARGV[0], &O_RDONLY) || die(error_str());
     printf("Creator:  %s\n", $ef->creator());
```

```
printf("Hostname: %s\n\n", $ef->hostname());

# Dump the file contents
while (my $obj = $ef->get()) {
        ea_dump_object($obj);
}

# Report any errors
if (ea_error() != EXR_OK && ea_error() != EXR_EOF)  {
        printf("\nERROR: %s\n", ea_error_str());
        exit(1);
}
exit(0);
```

Sun::Solaris::Exacct::Object->dump() からの出力例

73 ページの「新しいグループレコードを作成してファイルに書き込む方法」で作成されたファイルに Sun::Solaris::Exacct::Object->dump() を実行した場合の出力例を示します。

```
Creator:  root
Hostname: localhost
GROUP
       Catalog = EXT_GROUP|EXC_DEFAULT|EXD_NONE
       ITEM
         Catalog = EXT_STRING|EXC_DEFAULT|EXD_CREATOR
         Value = me
       ITEM
         Catalog = EXT_UINT32|EXC_DEFAULT|EXD_PROC_PID
         Value = 845523
       ITEM
         Catalog = EXT_UINT32|EXC_DEFAULT|EXD_PROC_UID
         Value = 37845
       ITEM
         Catalog = EXT_UINT32|EXC_DEFAULT|EXD_PROC_GID
         Value = 10
       ITEM
         Catalog = EXT_STRING|EXC_DEFAULT|EXD_PROC_COMMAND
         Value = /bin/rec
ENDGROUP
```

◆ ◆ ◆ 第 6 章

リソース制御 (概要)

第 4 章「拡張アカウンティング (概要)」で説明したようにシステム上の作業負荷のリソース消費を判定したら、リソースの使用方法に制限を設けることができます。制限を設けると、作業負荷によるリソースの過剰消費を防ぐことができます。「リソース制御」機能は、この目的に使用される制約メカニズムです。

この章の内容は次のとおりです。

- 75 ページの「リソース制御の概念」
- 77 ページの「リソース制御と属性の構成」
- 90 ページの「リソース制御の適用」
- 90 ページの「動作中のシステム上のリソース制御値を一時的に更新する」
- 91 ページの「リソース制御で使用するコマンド」

リソース制御を管理する方法については、第 7 章「リソース制御の管理 (タスク)」を参照してください。

リソース制御の概念

Oracle Solaris オペレーティングシステムでは、プロセスごとのリソース制限という概念が、タスクおよびプロジェクトのエンティティーに拡張されています (第 2 章「プロジェクトとタスク (概要)」に記載)。この拡張機能は、リソース制御 (rctls) 機能によって提供されます。また、割り当ては `/etc/system` 調整可能パラメータを通して設定していましたが、これもリソース制御メカニズムを通して自動的に行われるか、手動で構成するようになりました。

リソース制御には、接頭辞 `zone`、`project`、`task`、または `process` が付きます。リソース制御はシステム全体に適用できます。動作中のシステム上のリソース制御値を更新できます。

このリリースで使用できる標準のリソース制御のリストについては、78 ページの「使用可能なリソース制御」を参照してください。ゾーンごとに使用可能なリソース制御については、247 ページの「リソースタイプのプロパティー」を参照してください。

リソース制限とリソース制御

従来から、UNIX システムにはリソース制限機能があります (*rlimit*)。rlimit の機能を使用すると、管理者は、プロセスが消費できるリソースの量に対して 1 つ以上の数値制限を設定できます。この制限には、プロセスごとの CPU 使用時間、プロセスごとのコアファイルサイズ、プロセスごとの最大ヒープサイズが含まれます。「ヒープサイズ」は、プロセスのデータセグメントに割り当てられるスクラッチメモリー領域のサイズです。

リソース制御機能は、リソース制限機能に対する互換性インタフェースを提供します。リソース制限機能を使用する既存のアプリケーションは、変更せずに、引き続き使用できます。また、既存のアプリケーションは、リソース制御機能を利用するように変更されたアプリケーションと同様に監視することができます。

プロセス間通信とリソース制御

プロセスは、複数の種類のプロセス間通信 (IPC) の 1 つを使用して、互いに通信できます。IPC を使用すると、プロセス間で情報の転送や同期化を行うことができます。リソース制御機能により、カーネルの IPC 機能の動作を定義するリソース制御が提供されます。これらのリソース制御は、`/etc/system` の調整可能パラメータを置換します。

デフォルトのリソース制御値の初期化に使用される廃止済みパラメータが、この Oracle Solaris システム上の `/etc/system` ファイルに含まれている場合があります。ただし、古いパラメータはできるだけ使用しないでください。

どの IPC オブジェクトがプロジェクトの使用状況に影響を与えているかを監視するには、`ipcs` コマンドに `-J` オプションを付けて実行します。表示例については、101 ページの「ipcs を使用する方法」を参照してください。`ipcs` コマンドの詳細については、`ipcs(1)` のマニュアルページを参照してください。

Oracle Solaris システムの調整については、『Oracle Solaris 11.1 カーネルのチューンアップ・リファレンスマニュアル』を参照してください。

リソース制御の制約メカニズム

リソース制御機能は、システムリソースに対する制約メカニズムを提供します。これにより、プロセス、タスク、プロジェクト、およびゾーンが、指定したシステムリソースを過剰消費することを防止できます。このメカニズムは、リソースの過剰消費を防ぐことにより、より管理しやすいシステムを実現します。

制約メカニズムは、容量計画を実施するときにも使用できます。制約を設けることにより、アプリケーションへのリソースの提供を必ずしも拒否することなく、アプリケーションが必要とするリソース量に関する情報を取得できます。

プロジェクトの属性メカニズム

また、リソース制御は、リソース管理機能のための簡単な属性メカニズムとしても利用できます。たとえば、公平配分スケジューラ (FSS) のスケジューリングクラスで動作しているプロジェクトで利用できる CPU の配分は、リソース制御 `project.cpu-shares` によって定義されます。プロジェクトはリソース制御によって一定の配分を割り当てられるため、制御の超過につながる各種のアクションは許可されません。そのため、リソース制御 `project.cpu-shares` の現在値は、指定したプロジェクトの属性とみなすことができます。

また、プロジェクト内のプロセスの集合が消費する物理メモリーを規制するには、別の種類のプロジェクト属性が使用されます。これらの属性には、接頭辞 `rcap` が付きます (たとえば、`rcap.max-rss`)。リソース制御と同様に、この種類の属性も `project` データベース中に構成します。リソース制御はカーネルによって同期的に実行されますが、物理メモリーのリソース上限の制限はリソース上限デーモン `rcapd` によってユーザーレベルで非同期的に強制実行されます。`rcapd` については、第 10 章「リソース上限デーモンによる物理メモリーの制御 (概要)」および `rcapd`(1M) のマニュアルページを参照してください。

`project.pool` 属性は、プロジェクトのプールの結合を指定するために使用されます。リソースプールの詳細については、第 12 章「リソースプール (概要)」を参照してください。

リソース制御と属性の構成

リソース制御機能は、`project` データベースを使用して構成されます。第 2 章「プロジェクトとタスク (概要)」を参照してください。リソース制御とその他の属性は、`project` データベースエントリの最後のフィールドで設定します。各リソース制御に対応付けられる値は、括弧で囲まれ、コンマ区切りのプレーンテキストとして示されます。括弧内の値によって「アクション文節」が構成されます。各アクション文節には、値として特権レベル、しきい値、および特定のしきい値に対応付けられたアクションが含まれます。各リソース制御は複数のアクション文節を持つ

ことができ、各アクション文節もコンマで区切られます。次のエントリは、プロジェクトエンティティーにおけるタスクごとの軽量プロセス (LWP) 制限と、プロセスごとの最長 CPU 時間制限を定義します。process.max-cpu-time は、プロセスの実行時間が合計で 1 時間になるとプロセスに SIGTERM を送信し、1 時間 1 分になると SIGKILL を送信します。表 6–3 を参照してください。

```
development:101:Developers:::task.max-lwps=(privileged,10,deny);
  process.max-cpu-time=(basic,3600,signal=TERM),(priv,3660,signal=KILL)
    typed as one line
```

注 – ゾーンが有効になっているシステムの場合、ゾーン規模のリソース制御はゾーン構成で指定されます。その形式は多少異なります。詳細は、241 ページの「ゾーン構成データ」を参照してください。

rctladm コマンドを使用すると、リソース制御機能の実行時に問い合わせや制御機能の変更を「大域有効範囲」で行うことができます。prctl コマンドを使用すると、実行時にリソース制御機能の問い合わせや変更を「局所有効範囲」で行うことができます。

詳細については、85 ページの「リソース制御値に対応付けられた大域アクションと局所アクション」、および rctladm(1M) と prctl(1) のマニュアルページを参照してください。

注 – ゾーンがインストールされているシステムでは、非大域ゾーンで rctladm を使用して設定を変更することはできません。各リソース制御の大域ログ状態を表示する場合に、非大域ゾーンで rctladm を使用します。

使用可能なリソース制御

次の表に、このリリースで使用できる標準のリソース制御を示します。

この表では、各制御によって制約されるリソースについて説明し、project データベースにおけるそのリソースのデフォルトの単位を示します。デフォルトの単位には次の 2 種類があります。

- 数量は制限される量を意味します。
- インデックスは最大有効識別子を意味します。

したがって、project.cpu-shares は、プロジェクトで使用することが許可されている配分を示します。一方、process.max-file-descriptor は、open(2) システムコールによってプロセスに割り当てることができる最大ファイル番号を指定します。

表 6–1 標準のプロジェクト、タスク、およびプロセスのリソース制御

制御名	説明	デフォルトの単位
project.cpu-cap	1 つのプロジェクトで消費可能な CPU リソース量に対する絶対的な制限。project.cpu-cap 設定と同様、100 の値は 1 つの CPU の 100% を意味します。125 の値は 125% になります。CPU キャップの使用時は、100% がシステム上の 1 つの CPU の上限となります。	数量 (CPU の数)
project.cpu-shares	このプロジェクトに対して、公平配分スケジューラ (FSS(7) のマニュアルページを参照) で使用することが許可されている CPU 配分。	数量 (配分)
project.max-crypto-memory	ハードウェアによる暗号化処理の高速化のために libpkcs11 が使用できるカーネルメモリーの合計量。カーネルバッファーおよびセッション関連の構造体の割り当ては、このリソース制御に対してチャージされます。	サイズ (バイト)
project.max-locked-memory	ロックされる物理メモリーの許容合計量。 priv_proc_lock_memory がユーザーに割り当てられている場合、そのユーザーがすべてのメモリーをロックするのを防ぐため、このリソース制御の設定も検討してください。 project.max-device-locked-memory (これは削除されています) が、このリソース制御に置き換えられたことに注意してください。	サイズ (バイト)
project.max-msg-ids	このプロジェクトに許容されるメッセージキュー ID の最大数。	数量 (メッセージキュー ID の数)
project.max-port-ids	イベントポートの許容最大数。	数量 (イベントポート数)

表 6–1 標準のプロジェクト、タスク、およびプロセスのリソース制御 (続き)

制御名	説明	デフォルトの単位
project.max-processes	このプロセスで同時に使用できるプロセステーブルスロットの最大数。 正常なプロセスとゾンビプロセスの両方がプロセステーブルスロットを占有するため、max-processes 制御がプロセステーブルを消費しつくすゾンビプロセスから保護します。ゾンビプロセスには本来 LWP がないため、max-lwps 制御がこのゾンビプロセス発生の可能性から保護することはできません。	数量 (プロセステーブルスロット数)
project.max-sem-ids	このプロジェクトに許容されるセマフォー ID の最大数。	数量 (セマフォー ID の数)
project.max-shm-ids	このプロジェクトに許容される共有メモリー ID の最大数。	数量 (共有メモリー ID の数)
project.max-shm-memory	このプロジェクトに許容される System V 共有メモリーの合計量。	サイズ (バイト)
project.max-lwps	このプロジェクトで同時に使用できる LWP の最大数。	数量 (LWP 数)
project.max-tasks	このプロジェクトに許容されるタスクの最大数	数量 (タスク数)
project.max-contracts	このプロジェクトに許容される契約の最大数	数量 (契約数)
task.max-cpu-time	このタスクのプロセスで使用できる最長 CPU 時間。	時間 (秒)
task.max-lwps	このタスクのプロセスで同時に使用できる LWP の最大数。	数量 (LWP 数)
task.max-processes	このタスクのプロセスで同時に使用できるプロセステーブルスロットの最大数。	数量 (プロセステーブルスロット数)
process.max-cpu-time	このプロセスで使用できる最長 CPU 時間。	時間 (秒)
process.max-file-descriptor	このプロセスで使用できる最大のファイル記述子インデックス。	インデックス (最大ファイル記述子)
process.max-file-size	このプロセスの書き込みに使用できる最大ファイルオフセット。	サイズ (バイト)

表 6–1 標準のプロジェクト、タスク、およびプロセスのリソース制御 (続き)

制御名	説明	デフォルトの単位
process.max-core-size	このプロセスによって作成されるコアファイルの最大サイズ。	サイズ (バイト)
process.max-data-size	このプロセスで使用できるヒープメモリーの最大サイズ。	サイズ (バイト)
process.max-stack-size	このプロセスに使用できる最大スタックメモリーセグメント。	サイズ (バイト)
process.max-address-space	このプロセスで使用できる、セグメントサイズの総計としての最大アドレス空間。	サイズ (バイト)
process.max-port-events	イベントポートあたりに許容されるイベントの最大数。	数量 (イベント数)
process.max-sem-nsems	セマフォーセットあたりに許容されるセマフォーの最大数。	数量 (セットあたりのセマフォー数)
process.max-sem-ops	1 回の semop コールに許容されるセマフォー操作の最大数 (semget() のコール時にリソース制御からコピーされる値)。	数量 (操作の数)
process.max-msg-qbytes	メッセージキュー内のメッセージの最大バイト数 (msgget() のコール時にリソース制御からコピーされる値)。	サイズ (バイト)
process.max-msg-messages	メッセージキュー内のメッセージの最大数 (msgget() のコール時にリソース制御からコピーされる値)。	数量 (メッセージ数)

リソース制御の設定や変更がまったく行われていないシステム上では、リソース制御のデフォルト値を表示できます。そのようなシステムでは、/etc/system や project データベースにデフォルト以外のエントリが含まれていません。値を表示するには、prctl コマンドを使用します。

ゾーン規模のリソース制御

ゾーン規模のリソース制御は、ゾーン内のすべてのプロセスエンティティーによる総リソース消費を制限します。233 ページの「ゾーン規模のリソース制御の設定」および270 ページの「ゾーンの構成方法」で説明されているとおり、ゾーン規模のリソース制御は、グローバルプロパティー名を使用して設定することもできます。

表 6–2 ゾーンのリソース制御

制御名	説明	デフォルトの単位
zone.cpu-cap	1 つの非大域ゾーンで消費可能な CPU リソース量に対する絶対的な制限。 project.cpu-cap 設定と同様、100 の値は 1 つの CPU の 100% を意味します。125 の値は 125% になります。CPU キャップの使用時は、100% がシステム上の 1 つの CPU の上限となります。	数量 (CPU の数)
zone.cpu-shares	このゾーンに対する公平配分スケジューラ (FSS) の CPU 配分	数量 (配分)
zone.max-lofi	ゾーンによって作成できる lofi デバイスの最大数。 この値は、各ゾーンのマイナーノード名前空間の使用法を制限します。	数量 (lofi デバイスの数)
zone.max-locked-memory	ゾーンで使用できるロックされた物理メモリーの合計量。 priv_proc_lock_memory がゾーンに割り当てられている場合、そのゾーンがすべてのメモリーをロックするのを防ぐため、このリソース制御の設定も検討してください。	サイズ (バイト)
zone.max-lwps	このゾーンで同時に使用できる LWP の最大数	数量 (LWP 数)
zone.max-msg-ids	このゾーンに許容されるメッセージキュー ID の最大数	数量 (メッセージキュー ID の数)
zone.max-processes	このゾーンで同時に使用できるプロセステーブルスロットの最大数。 正常なプロセスとゾンビプロセスの両方がプロセステーブルスロットを占有するため、max-processes 制御がプロセステーブルを消費しつくすゾンビから保護します。ゾンビプロセスには本来 LWP がないため、max-lwps 制御がこのゾンビプロセス発生の可能性から保護することはできません。	数量 (プロセステーブルスロット数)

表 6–2 ゾーンのリソース制御 (続き)

制御名	説明	デフォルトの単位
zone.max-sem-ids	このゾーンに許容されるセマフォー ID の最大数	数量 (セマフォー ID の数)
zone.max-shm-ids	このゾーンに許容される共有メモリー ID の最大数	数量 (共有メモリー ID の数)
zone.max-shm-memory	このゾーンに許容される System V 共有メモリーの合計量	サイズ (バイト)
zone.max-swap	このゾーンのユーザープロセスのアドレス空間マッピングと tmpfs マウントで消費できるスワップの合計量。	サイズ (バイト)

ゾーン規模のリソース制御の構成方法については、247 ページの「リソースタイプのプロパティー」および 270 ページの「ゾーンの構成方法」を参照してください。

ゾーン規模のリソース制御を大域ゾーンに適用することも可能です。詳細は、413 ページの「ゾーンがインストールされている Oracle Solaris システムでの公平配分スケジューラの使用」を参照してください。

単位のサポート

リソース制御の種類を示す大域フラグは、すべてのリソース制御に対して定義されます。これらのフラグは、種類に関する基本情報を prctl などのアプリケーションに伝えるために、システムによって使用されます。アプリケーションはこの情報を使用して、次の内容を判定します。

- 各リソース制御に適した単位の文字列
- 倍率値を解釈するときに使用する正しい倍率

次の大域フラグを使用できます。

大域フラグ	リソース制御の種類ごとの文字列	修飾子	倍率
RCTL_GLOBAL_BYTES	バイト	B	1
		KB	2^{10}
		MB	2^{20}
		GB	2^{30}
		TB	2^{40}
		PB	2^{50}
		EB	2^{60}
RCTL_GLOBAL_SECONDS	秒	s	1
		Ks	10^{3}
		Ms	10^{6}
		Gs	10^{9}
		Ts	10^{12}
		Ps	10^{15}
		Es	10^{18}
RCTL_GLOBAL_COUNT	数	なし	1
		K	10^{3}
		M	10^{6}
		G	10^{9}
		T	10^{12}
		P	10^{15}
		E	10^{18}

リソース制御に倍率値を使用できます。次の例は、倍率付きのしきい値を示します。

```
task.max-lwps=(priv,1K,deny)
```

注 – 単位修飾子は、`prctl`、`projadd`、および`projmod`コマンドに使用できます。`project`データベース自体で単位修飾子を使用することはできません。

リソース制御値と特権レベル

リソース制御のしきい値は、局所アクションのトリガーやログ作成などの大域アクションの発生が可能である実行ポイントを設定します。

リソース制御の各しきい値は、特権レベルに関連付ける必要があります。次の 3 種類の特権レベルのいずれかを使用します。

- 基本値 — 呼び出し元プロセスの所有者が変更できます
- 特権値 — 特権を持っている呼び出し元 (root) だけが変更できます
- システム値 — オペレーティングシステムによる処理が実行されている間は、固定されます

リソース制御は、システムまたはリソースの提供者によって定義されるシステム値を 1 つ持つことが保証されます。システム値は、オペレーティングシステムが提供できるリソースの量を意味します。

特権値はいくつでも定義できます。基本値は 1 つだけ許可されます。特権値を指定しないで実行される操作には、デフォルトで、基本レベルの特権が割り当てられます。

リソース制御値の特権レベルは、リソース制御ブロックの特権フィールドで、RCTL_BASIC、RCTL_PRIVILEGED、または RCTL_SYSTEM のように定義します。詳細は、`setrctl(2)` のマニュアルページを参照してください。`prctl` コマンドを使用すると、基本レベルおよび特権レベルに対応付けられている値を変更できます。

リソース制御値に対応付けられた大域アクションと局所アクション

リソース制御値に対応付けられるアクションには 、2 種類あります。 大域アクションと局所アクションです。

リソース制御値に対応付けられた大域アクション

大域アクションは、システム上のすべてのリソース制御のリソース制御値に適用されます。`rctladm` コマンド (`rctladm(1M)` のマニュアルページに記載) を使用すると、次の動作を実行できます。

- アクティブなシステムリソース制御の大域的状態を表示します
- 大域ログ作成アクションを設定します

リソース制御に対応付けられた大域ログ作成アクションは、無効にしたり有効にしたりできます。`syslog` アクションの程度を設定するには、重要度を `syslog=`*level* のように割り当てます。*level* に設定できる値は次のとおりです。

- debug
- info
- notice
- warning
- err
- crit
- alert
- emerg

デフォルトでは、リソース制御の違反は大域ログ作成では記録されません。レベル n/a は、大域アクションを構成できないリソース制御を示します。

リソース制御値に対応付けられた局所アクション

局所アクションは、制御値を超えようとしているプロセスに対して実行されます。リソース制御に設定された各しきい値に対して、1 つ以上のアクションを対応付けることができます。局所アクションには、3 つの種類があります。 none、deny、および signal= です。これら 3 つのアクションは、次のように使用されます。

none　しきい値を超える量のリソース要求に対して、何のアクションも行いません。このアクションは、アプリケーションの進行に影響を与えることなく、リソースの使用状況を監視するのに役立ちます。プロセスがリソース制御のしきい値を超えたときに大域メッセージを表示することもできます。ただし、このアクションによってプロセスが影響を受けることはありません。

deny　しきい値を超える量のリソース要求を拒否できます。たとえば、task.max-lwps リソース制御に deny アクションが指定されている場合、制御値を超えるような新しいプロセスを作成する fork システムコールは失敗します。fork(2) のマニュアルページを参照してください。

signal=　リソース制御値を超えたときに大域シグナルメッセージを送信するアクションを有効にすることができます。プロセスがしきい値を超えると、プロセスにシグナルが送信されます。プロセスがさらにリソースを消費しても、追加のシグナルが送信されることはありません。使用できるシグナルの一覧については、表 6–3 を参照してください。

すべてのリソース制御にすべてのアクションを適用できるわけではありません。たとえば、プロセスは、その所属先のプロジェクトに割り当てられている CPU 配分を超えることはできません。したがって、project.cpu-shares リソース制御に deny アクションを適用することはできません。

実装上の制限により、しきい値に設定できるアクションは、各制御の大域プロパティーによって制限されます。rctladm(1M) のマニュアルページを参照してください。次の表に、使用できるシグナルアクションを示します。シグナルの詳細については、signal(3HEAD) のマニュアルページを参照してください。

表 6–3 リソース制御値に使用できるシグナル

シグナル	説明	注意事項
SIGABRT	プロセスを終了します。	
SIGHUP	ハングアップシグナルを送信します。開いた回線上でキャリアが検出されなくなったときに発生します。シグナルは、端末を制御しているプロセスグループに送信されます。	
SIGTERM	プロセスを終了します。ソフトウェアによって送信される終了シグナルです。	
SIGKILL	プロセスを終了し、プログラムを強制終了します。	
SIGSTOP	プロセスを停止します。ジョブ制御シグナルです。	
SIGXRES	リソース制御の制限超過です。リソース制御機能によって生成されます。	
SIGXFSZ	プロセスを終了します。ファイルサイズの制限超過です。	RCTL_GLOBAL_FILE_SIZE プロパティー (`process.max-file-size`) を持つリソース制御だけで使用可能です。詳細は、`rctlblk_set_value(3C)` のマニュアルページを参照してください。
SIGXCPU	プロセスを終了します。CPU 時間の制限超過です。	RCTL_GLOBAL_CPUTIME プロパティー (`process.max-cpu-time`) を持つリソース制御だけで使用可能です。詳細は、`rctlblk_set_value(3C)` のマニュアルページを参照してください。

リソース制御のフラグとプロパティー

システム上のリソース制御には、それぞれ特定のプロパティーセットが対応付けられています。このプロパティーセットは、一連のフラグとして定義されます。これらのフラグは、そのリソースが制御されているすべてのインスタンスに対応付けられます。大域フラグは変更できませんが、`rctladm` または `getrctl` システムコールを使って取得できます。

ローカルフラグは、特定のプロセスまたはプロセス集合に対するリソース制御の特定のしきい値について、デフォルトの動作と構成を定義します。あるしきい値のローカルフラグが、同じリソース制御で定義されている別のしきい値の動作に影響することはありません。ただし、大域フラグは、特定の制御に対応付けられているすべての値の動作に影響します。ローカルフラグは、対応する大域フラグによる制約の範囲内で、`prctl` コマンドまたは `setrctl` システムコールを使って変更できます。`setrctl(2)` のマニュアルページを参照してください。

ローカルフラグ、大域フラグ、およびそれらの定義の詳細な一覧については、`rctlblk_set_value(3C)` のマニュアルページを参照してください。

特定のリソース制御がしきい値に達したときのシステムの動作を確認するには、`rctladm` を使ってそのリソース制御の大域フラグを表示します。たとえば、`process.max-cpu-time` の値を表示するには、次のように入力します。

```
$ rctladm process.max-cpu-time
    process.max-cpu-time  syslog=off  [ lowerable no-deny cpu-time inf seconds ]
```

大域フラグは、次のことを示します。

`lowerable` この制御の特権値を下げるのに、スーパーユーザー特権を必要としません。

`no-deny` しきい値を超えても、リソースへのアクセスは拒否されません。

`cpu-time` リソースがしきい値に達したとき、SIGXCPU を送信できます。

`seconds` リソース制御の時間。

`no-basic` 特権タイプ `basic` を持つリソース制御値を設定できません。特権付きリソース制御値だけが許可されます。

`no-signal` リソース制御値に対してローカルのシグナルアクションを設定できません。

`no-syslog` このリソース制御に対して大域の `syslog` メッセージアクションを設定できません。

`deny` しきい値を超えたときに、必ずリソースの要求を拒否します。

`count` リソース制御のカウント (整数) 値。

`bytes` リソース制御のサイズの単位。

リソース制御のローカル値とアクションを表示するには、`prctl` コマンドを使用します。

```
$ prctl -n process.max-cpu-time $$
    process 353939: -ksh
    NAME    PRIVILEGE    VALUE    FLAG   ACTION              RECIPIENT
 process.max-cpu-time
```

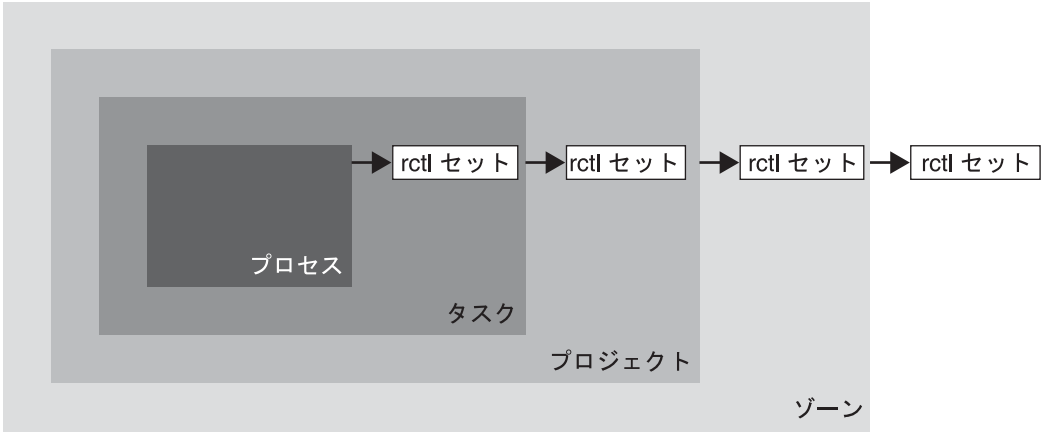
```
        privileged      18.4Es      inf     signal=XCPU                     -
        system          18.4Es      inf     none
```

この例では、2 つのしきい値の両方に max (RCTL_LOCAL_MAXIMAL) フラグが設定されており、リソース制御には inf (RCTL_GLOBAL_INFINITE) フラグが設定されています。inf の値は無限大です。この値は制限を与えません。したがって、構成されているように、両方のしきい値は無限大値を意味し、これらの値を上回ることはありません。

リソース制御の実行

1 つのリソースには、複数のリソース制御を設定できます。リソース制御は、プロセスモデルの包含レベルごとに 1 つずつ設定できます。同じリソース上の異なるコンテナレベルでリソース制御がアクティブな場合、まず、もっとも小さいコンテナの制御が実行されます。したがって、process.max-cpu-time と task.max-cpu-time の両方の制御が同時に検出された場合は、まず process.max-cpu-time に対するアクションが実行されます。

図 6–1 プロセス集合、コンテナの包含関係、およびそのリソース制御セット



リソース制御イベントの大域監視

プロセスのリソース消費が不明な場合がよくあります。リソース消費に関する詳細な情報を入手するには、rctladm コマンドで利用できる大域リソース制御アクションを使用してみてください。rctladm を使用して、リソース制御に syslog アクションを設定します。そのリソース制御が管理するエンティティーでしきい値が検出される

と、構成したログレベルでシステムメッセージが記録されます。詳細は、第 7 章「リソース制御の管理 (タスク)」および rctladm(1M) のマニュアルページを参照してください。

リソース制御の適用

表 6–1 に示されている各リソース制御をプロジェクトに割り当てることができるのは、ログイン時、newtask または su が呼び出されたとき、あるいは、at、batch、cron など、プロジェクトを扱うことができる起動ツールが呼び出されたときです。開始される各コマンドは、呼び出し側のユーザーのデフォルトプロジェクトとは異なるタスクで起動されます。詳細については、login(1)、newtask(1)、at(1)、cron(1M)、および su(1M) のマニュアルページを参照してください。

project データベース内のエントリに対する更新は、/etc/project ファイルまたはネットワークネームサービスのデータベース表現のどちらに対するものであっても、現在アクティブなプロジェクトには適用されません。更新内容は、新しいタスクがログインまたは newtask によってプロジェクトに参加したときに適用されます。

動作中のシステム上のリソース制御値を一時的に更新する

project データベースで変更された値は、プロジェクト内で開始される新しいタスクに対してだけ有効になります。ただし、rctladm および prctl コマンドを使用すると、動作中のシステムのリソース制御を更新できます。

ログステータスの更新

rctladm コマンドは、システム全体で、各リソース制御の大域ログ状態に影響を与えます。このコマンドは、大域的状態を表示し、制御の限度を超えたときに syslog が記録するログのレベルを設定できます。

リソース制御の更新

prctl コマンドを使用すると、プロセスごと、タスクごと、またはプロジェクトごとにリソース制御値とアクションを表示したり、一時的に変更したりできます。プロジェクト ID、タスク ID、またはプロセス ID を入力として指定すると、このコマンドは、制御が定義されているレベルでリソース制御に対して動作します。

変更した値とアクションはすぐに適用されます。ただし、これらの変更が適用されるのは、現在のプロセス、タスク、またはプロジェクトだけです。変更内容は、project データベースには記録されません。システムを再起動すると、変更内容は失われます。リソース制御を永続的に変更するには、project データベースで変更を行う必要があります。

project データベースで変更できるリソース制御設定はすべて、prctl コマンドでも変更できます。基本値と特権値はどちらも、追加、削除が可能です。またそれらのアクションも変更できます。デフォルトで、基本タイプではすべての設定操作が想定されています。root 権限があるプロセスとユーザーは、特権レベルのリソース制御も変更できます。システムリソースの制御は変更できません。

リソース制御で使用するコマンド

リソース制御で使用するコマンドを次の表に示します。

コマンド	説明
ipcs(1)	このコマンドを使用すると、どの IPC オブジェクトがプロジェクトの使用状況に影響を与えているかを監視できます
prctl(1)	このコマンドを使用すると、リソース制御機能の実行時に問い合わせやリソース制御機能の変更をローカルに行うことができます
rctladm(1M)	このコマンドを使用すると、リソース制御機能の実行時に問い合わせやリソース制御機能の変更を大域的に行うことができます

resource_controls(5) のマニュアルページでは、単位や倍率なども含め、プロジェクトデータベース経由で使用可能なリソース制御について説明しています。

◆ ◆ ◆ 第 7 章

7 リソース制御の管理 (タスク)

この章では、リソース制御機能を管理する方法について説明します。

リソース制御機能の概要については、第 6 章「リソース制御 (概要)」を参照してください。

リソース制御の管理 (タスクマップ)

タスク	説明	参照先
リソース制御を設定します。	`/etc/project` ファイルでプロジェクトにリソース制御を設定します。	94 ページの「リソース制御の設定」
アクティブなプロセス、タスク、またはプロジェクトについて、リソース制御値をローカルに取得または変更します。	システム上のアクティブなプロセス、タスク、またはプロジェクトに関連付けられているリソース制御に対し、実行時に問い合わせや変更を行います。	96 ページの「`prctl` コマンドの使用」
稼働中のシステムのリソース制御の大域的状態を表示または更新します。	システム全体の各リソース制御の大域ログ状態を表示します。また、制御値を超えたときに `syslog` で記録するログのレベルを設定します。	100 ページの「`rctladm` の使用」
アクティブなプロセス間通信 (IPC) 機能のステータスを報告します。	アクティブなプロセス間通信 (IPC) 機能について情報を表示します。どの IPC オブジェクトがプロジェクトの使用状況に影響を与えているかを監視します。	101 ページの「`ipcs` の使用」

タスク	説明	参照先
Web サーバーに十分な CPU 容量が割り当てられているかどうかを判定します。	リソース制御に大域アクションを設定します。このアクションを設定すると、リソース制御値を超えたエンティティーについて通知を受け取ることができ、リソース制御値が低すぎるかどうかを判定できます。	102 ページの「Web サーバーに十分な CPU 容量が割り当てられているかどうかを判定する方法」

リソース制御の設定

▼ プロジェクト内の各タスクの最大 LWP 数を設定する方法

この手順では、`x-files` というプロジェクトを `/etc/project` ファイルに追加し、このプロジェクト内に作成されるタスクに適用する LWP の最大数を設定します。

1. **root** になるか、同等の役割になります。
2. **projadd** コマンドに **-K** オプションを付けて実行して、**x-files** というプロジェクトを作成します。このプロジェクト内に作成される各タスクの **LWP** の最大数を **3** に設定します。

   ```
   # projadd -K 'task.max-lwps=(privileged,3,deny)' x-files
   ```

3. **/etc/project** ファイル内のエントリを表示します。それには、次のいずれかの方法を使用します。
   - 次のように入力します。

     ```
     # projects -l
     system
             projid : 0
             comment: ""
             users  : (none)
             groups : (none)
             attribs:
     .
     .
     .
     x-files
             projid : 100
             comment: ""
             users  : (none)
             groups : (none)
             attribs: task.max-lwps=(privileged,3,deny)
     ```

   - 次のように入力します。

```
# cat /etc/project
system:0:System:::
.
.
.
x-files:100::::task.max-lwps=(privileged,3,deny)
```

例 **7–1** セッション例

上記の手順を実行したあと、プロジェクト `x-files` に `newtask` を使用して参加することで新しいタスクを作成した root ユーザーは、そのタスクで実行中は、LWP を 3 つまでしか作成できません。次の注釈付きのセッション例を参照してください。

```
# newtask -p x-files csh

# prctl -n task.max-lwps $$
process: 111107: csh
NAME    PRIVILEGE    VALUE    FLAG   ACTION            RECIPIENT
task.max-lwps
        usage            3
        privileged       3       -   deny                      -
        system       2.15G     max   deny                      -
# id -p
uid=0(root) gid=1(other) projid=100(x-files)

# ps -o project,taskid -p $$
 PROJECT TASKID
 x-files    73

# csh        /* creates second LWP */

# csh        /* creates third LWP */

# csh        /* cannot create more LWPs */
Vfork failed
#
```

▼ プロジェクトに複数の制御を設定する方法

`/etc/project` ファイルには、各プロジェクトごとに複数のリソース制御設定を記述でき、さらに各リソース制御ごとに複数のしきい値を記述できます。しきい値はアクション文節で定義されます。複数の値はコンマで区切られます。

1 **root** になるか、同等の役割になります。

2 **projmod** コマンドに **-s** オプションと **-K** オプションを付けて実行することで、プロジェクト **x-files** にリソース制御を設定します。

```
# projmod -s -K 'task.max-lwps=(basic,10,none),(privileged,500,deny);
process.max-file-descriptor=(basic,128,deny)' x-files      one line in file
```

次の制御が設定されます。

- タスクごとの LWP 最大数について、アクションなしの basic 制御。
- タスクごとの LWP 最大数について、特権レベルの deny 制御。この制御により、94 ページの「プロジェクト内の各タスクの最大 LWP 数を設定する方法」の例のように、最大数を超える数の LWP を作成しようとすると失敗します。
- プロセスごとの最大ファイル記述子は basic レベルに制限されており、最大値を超える open コールはすべて失敗します。

3 次のいずれかの方法で、ファイル内のエントリを表示します。

- 次のように入力します。

```
# projects -l
.
.
.
x-files
        projid : 100
        comment: ""
        users  : (none)
        groups : (none)
        attribs: process.max-file-descriptor=(basic,128,deny)
                 task.max-lwps=(basic,10,none),(privileged,500,deny)      one line in file
```

- 次のように入力します。

```
# cat /etc/project
.
.
.
x-files:100::::process.max-file-descriptor=(basic,128,deny);
task.max-lwps=(basic,10,none),(privileged,500,deny)      one line in file
```

prctl コマンドの使用

prctl コマンドを使用すると、システム上のアクティブなプロセス、タスク、またはプロジェクトに関連付けられているリソース制御に対し、実行時に問い合わせや変更を行うことができます。詳細は、prctl(1) のマニュアルページを参照してください。

▼ prctl コマンドを使ってデフォルトのリソース制御値を表示する方法

この手順は、リソース制御の設定や変更がまったく行われていないシステム上で実行する必要があります。/etc/system ファイルまたは project データベース内には、デフォルト以外のエントリしか記述できません。

- 実行中の現在のシェルなど、任意のプロセスに対して **prctl** コマンドを実行します。

```
# prctl $$
process: 3320: bash
NAME    PRIVILEGE       VALUE    FLAG   ACTION                       RECIPIENT
process.max-port-events
        privileged      65.5K       -   deny                                 -
        system          2.15G     max   deny                                 -
process.max-msg-messages
        privileged      8.19K       -   deny                                 -
        system          4.29G     max   deny                                 -
process.max-msg-qbytes
        privileged      64.0KB      -   deny                                 -
        system          16.0EB    max   deny                                 -
process.max-sem-ops
        privileged        512       -   deny                                 -
        system          2.15G     max   deny                                 -
process.max-sem-nsems
        privileged        512       -   deny                                 -
        system          32.8K     max   deny                                 -
process.max-address-space
        privileged      16.0EB    max   deny                                 -
        system          16.0EB    max   deny                                 -
process.max-file-descriptor
        basic             256       -   deny                              3320
        privileged      65.5K       -   deny                                 -
        system          2.15G     max   deny                                 -
process.max-core-size
        privileged      8.00EB    max   deny                                 -
        system          8.00EB    max   deny                                 -
process.max-stack-size
        basic           10.0MB      -   deny                              3320
        privileged      32.0TB      -   deny                                 -
        system          32.0TB    max   deny                                 -
process.max-data-size
        privileged      16.0EB    max   deny                                 -
        system          16.0EB    max   deny                                 -
process.max-file-size
        privileged      8.00EB    max   deny,signal=XFSZ                     -
        system          8.00EB    max   deny                                 -
process.max-cpu-time
        privileged      18.4Es    inf   signal=XCPU                          -
        system          18.4Es    inf   none                                 -
task.max-cpu-time
        usage               0s
        system          18.4Es    inf   none                                 -
task.max-processes
        usage               2
        system          2.15G     max   deny                                 -
task.max-lwps
        usage               3
        system          2.15G     max   deny                                 -
project.max-contracts
        privileged      10.0K       -   deny                                 -
        system          2.15G     max   deny                                 -
project.max-locked-memory
        usage              0B
        system          16.0EB    max   deny                                 -
project.max-port-ids
```

```
        privileged      8.19K       -   deny                                 -
        system          65.5K     max   deny                                 -
project.max-shm-memory
        privileged       510MB      -   deny                                 -
        system          16.0EB    max   deny                                 -
project.max-shm-ids
        privileged        128       -   deny                                 -
        system          16.8M     max   deny                                 -
project.max-msg-ids
        privileged        128       -   deny                                 -
        system          16.8M     max   deny                                 -
project.max-sem-ids
        privileged        128       -   deny                                 -
        system          16.8M     max   deny                                 -
project.max-crypto-memory
        usage              0B
        privileged       510MB      -   deny                                 -
        system          16.0EB    max   deny                                 -
project.max-tasks
        usage              2
        system          2.15G     max   deny                                 -
project.max-processes
        usage              4
        system          2.15G     max   deny                                 -
project.max-lwps
        usage             11
        system          2.15G     max   deny                                 -
project.cpu-cap
        usage              0
        system          4.29G     inf   deny                                 -
project.cpu-shares
        usage              1
        privileged         1       -   none                                 -
        system          65.5K     max   none                                 -
zone.max-lofi
        usage              0
        system          18.4E     max   deny                                 -
zone.max-swap
        usage            180MB
        system          16.0EB    max   deny                                 -
zone.max-locked-memory
        usage              0B
        system          16.0EB    max   deny                                 -
zone.max-shm-memory
        system          16.0EB    max   deny                                 -
zone.max-shm-ids
        system          16.8M     max   deny                                 -
zone.max-sem-ids
        system          16.8M     max   deny                                 -
zone.max-msg-ids
        system          16.8M     max   deny                                 -
zone.max-processes
        usage             73
        system          2.15G     max   deny                                 -
zone.max-lwps
        usage            384
        system          2.15G     max   deny                                 -
zone.cpu-cap
        usage              0
```

```
        system          4.29G     inf   deny                                 -
zone.cpu-shares
        usage               1
        privileged          1       -   none                                 -
        system          65.5K     max   none
```

▼ prctl コマンドを使って特定のリソース制御の情報を表示する方法

- 実行中の現在のシェルの最大ファイル記述子を表示します。

```
# prctl -n process.max-file-descriptor $$
process: 110453: -sh
NAME    PRIVILEGE       VALUE    FLAG   ACTION       RECIPIENT
process.max-file-descriptor
        basic             256       -   deny            11731
        privileged      65.5K       -   deny                -
        system          2.15G     max   deny
```

▼ prctl を使って値を一時的に変更する方法

次の手順では、prctl コマンドを使用して x-files プロジェクトに新しい特権値を一時的に追加し、プロジェクトあたり 4 つ以上の LWP を使用することを拒否します。その結果は、94 ページの「プロジェクト内の各タスクの最大 LWP 数を設定する方法」の結果と同等になります。

1. **root** になるか、同等の役割になります。
2. **newtask** を使って **x-files** プロジェクトに参加します。

   ```
   # newtask -p x-files
   ```

3. **id** コマンドに **-p** オプションを付けて実行し、正しいプロジェクトに参加できたことを確認します。

   ```
   # id -p
   uid=0(root) gid=1(other) projid=101(x-files)
   ```

4. **project.max-lwps** に新しい特権値を追加して、**LWP** の数を **3** つまでに制限します。

   ```
   # prctl -n project.max-lwps -t privileged -v 3 -e deny -i project x-files
   ```

5. 結果を確認します。

   ```
   # prctl -n project.max-lwps -i project x-files
   process: 111108: csh
   NAME    PRIVILEGE    VALUE    FLAG   ACTION           RECIPIENT
   project.max-lwps
           usage          203
           privileged    1000       -   deny                    -
           system       2.15G     max   deny                    -
   ```

▼ prctl を使ってリソース制御値を下げる方法

1 root になるか、同等の役割になります。

2 prctl コマンドに -r オプションを付けて実行し、process.max-file-descriptor リソース制御の最小値を変更します。

```
# prctl -n process.max-file-descriptor -r -v 128 $$
```

▼ prctl を使ってプロジェクトの制御値を表示、置換、確認する方法

1 root になるか、同等の役割になります。

2 プロジェクト *group.staff* の project.cpu-shares の値を表示します。

```
# prctl -n project.cpu-shares -i project group.staff
project: 2: group.staff
NAME    PRIVILEGE       VALUE    FLAG   ACTION     RECIPIENT
project.cpu-shares
        usage              1
        privileged         1       -   none               -
        system          65.5K     max   none
```

3 project.cpu-shares の現在の値 1 を値 10 で置換します。

```
# prctl -n project.cpu-shares -v 10 -r -i project group.staff
```

4 プロジェクト group.staff の project.cpu-shares の値を表示します。

```
# prctl -n project.cpu-shares -i project group.staff
project: 2: group.staff
NAME    PRIVILEGE       VALUE    FLAG   ACTION     RECIPIENT
project.cpu-shares
        usage              1
        privileged         1       -   none                                    -
        system          65.5K     max   none
```

rctladm の使用

rctladm を使用する方法

rctladm コマンドを使用すると、リソース制御機能の大域的状態を実行時に問い合わせたり変更したりできます。詳細は、rctladm(1M) のマニュアルページを参照してください。

たとえば、rctladm に -e オプションを付けて実行して、リソース制御の大域属性 syslog を有効にすることができます。制御が限界を超えたとき、指定された syslog レベルで通知が記録されます。process.max-file-descriptor の大域属性 syslog を有効にするには、次のように入力します。

```
# rctladm -e syslog process.max-file-descriptor
```

rctladm コマンドを引数なしで使用すると、各リソース制御の大域フラグが大域タイプフラグも含めて表示されます。

```
# rctladm
process.max-port-events     syslog=off  [ deny count ]
process.max-msg-messages    syslog=off  [ deny count ]
process.max-msg-qbytes      syslog=off  [ deny bytes ]
process.max-sem-ops         syslog=off  [ deny count ]
process.max-sem-nsems       syslog=off  [ deny count ]
process.max-address-space   syslog=off  [ lowerable deny no-signal bytes ]
process.max-file-descriptor syslog=off  [ lowerable deny count ]
process.max-core-size       syslog=off  [ lowerable deny no-signal bytes ]
process.max-stack-size      syslog=off  [ lowerable deny no-signal bytes ]
.
.
.
```

ipcs の使用

ipcs を使用する方法

ipcs ユーティリティーを使用すると、アクティブなプロセス間通信 (IPC) 機能について情報を表示できます。詳細は、ipcs(1) のマニュアルページを参照してください。

ipcs に -J オプションを付けて実行すると、どのプロジェクトの制限に対して IPC オブジェクトが割り当てられているかを確認できます。

```
# ipcs -J
    IPC status from <running system> as of Wed Mar 26 18:53:15 PDT 2003
T         ID      KEY        MODE       OWNER    GROUP    PROJECT
Message Queues:
Shared Memory:
m       3600      0       --rw-rw-rw-   uname    staff    x-files
m        201      0       --rw-rw-rw-   uname    staff    x-files
m       1802      0       --rw-rw-rw-   uname    staff    x-files
m        503      0       --rw-rw-rw-   uname    staff    x-files
m        304      0       --rw-rw-rw-   uname    staff    x-files
m        605      0       --rw-rw-rw-   uname    staff    x-files
m          6      0       --rw-rw-rw-   uname    staff    x-files
m        107      0       --rw-rw-rw-   uname    staff    x-files
Semaphores:
s          0      0       --rw-rw-rw-   uname    staff    x-files
```

容量に関する警告

リソース制御に対して大域アクションを設定すると、リソース制御値を超えたエンティティーに関する通知を受け取ることができ、リソース制御値が低すぎるかどうかを判断できます。

たとえば、一般的な作業負荷のための十分な CPU リソースが Web サーバーに割り当てられているかどうかを確認する場合を考えます。この容量は、`sar` データで CPU のアイドル時間と平均負荷率を解析すれば判定できます。また、拡張アカウンティングデータを調べて、Web サーバープロセスで同時に実行しているプロセス数を確認することもできます。

より簡単な方法は、Web サーバーをタスクに配置することです。その上で、`syslog` を使って大域アクションを設定すると、タスクがマシンの性能に適した LWP の計画数を上回ったときに、警告が通知されます。

詳細は、sar(1) のマニュアルページを参照してください。

▼ Web サーバーに十分な CPU 容量が割り当てられているかどうかを判定する方法

1. **`prctl`** コマンドを使用して、**`httpd`** プロセスを含むタスクに **root** が所有する特権レベルのリソース制御を設定します。各タスクの **LWP** の総数を **40** に制限し、すべての局所アクションを無効にします。

   ```
   # prctl -n task.max-lwps -v 40 -t privileged -d all ‘pgrep httpd‘
   ```

2. リソース制御 **`task.max-lwps`** で、システムログの大域アクションを有効にします。

   ```
   # rctladm -e syslog task.max-lwps
   ```

3. 作業負荷がリソース制御を超えるかどうかを監視します。

   作業負荷がリソース制御を超えると、次のような内容が `/var/adm/messages` に記録されます。

   ```
   Jan  8 10:15:15 testmachine unix: [ID 859581 kern.notice]
   NOTICE: privileged rctl task.max-lwps exceeded by task 19
   ```

◆ ◆ ◆ 第 8 章

公平配分スケジューラ (概要)

作業負荷データを解析することによって、特定の作業負荷または作業負荷のグループが CPU リソースを占有しているかどうかを判定できます。作業負荷が CPU 使用量の制限を超えていない場合は、システム上での CPU 時間の割り当て方針を変更することができます。この章で説明する公平配分スケジューリングクラスを使用すると、タイムシェアリング (TS) スケジューリングクラスの優先順位スキームではなく、配分に基づいて CPU 時間を割り当てることができます。

この章の内容は次のとおりです。

- 104 ページの「スケジューラの紹介」
- 104 ページの「CPU 配分の定義」
- 105 ページの「CPU 配分とプロセスの状態」
- 105 ページの「CPU 配分と使用効率」
- 106 ページの「CPU 配分の例」
- 108 ページの「FSS の構成」
- 110 ページの「FSS とプロセッサセット」
- 112 ページの「FSS とほかのスケジューリングクラスの併用」
- 113 ページの「システムのスケジューリングクラスの設定」
- 113 ページの「ゾーンがインストールされているシステムでのスケジューリングクラス」
- 113 ページの「FSS で使用するコマンド」

公平配分スケジューラの使用を開始する方法については、第 9 章「公平配分スケジューラの管理 (タスク)」を参照してください。

スケジューラの紹介

オペレーティングシステムの基本的な仕事は、どのプロセスがシステムリソースへのアクセスを取得できるようにするか調整することです。プロセススケジューラ (別名、ディスパッチャー) は、カーネルの一部であり、プロセスへの CPU の割り当てを制御します。スケジューラには、スケジューリングクラスという概念があります。各スケジューリングクラスでは、クラス内のプロセスのスケジューリングに使用するスケジューリング方針を定義します。Oracle Solaris オペレーティングシステムでのデフォルトのスケジューラである TS スケジューラは、使用可能な CPU へのアクセスをすべてのプロセスに相対的に等しく与えようとします。ただし、特定のプロセスにより多くのリソースを与えたい場合もあります。

公平配分スケジューラ (FSS) では、各作業負荷に対する使用可能な CPU リソースの割り当てを、その作業負荷の重要性に基づいて制御します。この重要性は、各作業負荷に割り当てる CPU リソースの「配分」で表します。

各プロジェクトに CPU 配分を与えて、CPU リソースに対するプロジェクトの使用権を制御します。FSS では、プロジェクトに属するプロセス数ではなく、割り当てられた配分に基づいて、プロジェクト間に CPU リソースが公平に配分されることが保証されています。FSS は、ほかのプロジェクトとの比較に基づいて、CPU リソースを多く使用するプロジェクトの CPU 使用権を減らし、CPU リソースの使用が少ないプロジェクトの CPU 使用権を増やすことで公平さを実現します。

FSS は、カーネルスケジューリングクラスモジュールとクラス固有のバージョンの `dispadmin(1M)` および `priocntl(1)` コマンドから構成されます。FSS が使用するプロジェクト配分は、`project(4)` データベース内の `project.cpu-shares` プロパティーで指定します。

注 – ゾーンがインストールされている Oracle Solaris システムで `project.cpu-shares` リソース制御を使用する場合は、241 ページの「ゾーン構成データ」、376 ページの「非大域ゾーンで使用されるリソース制御」、および413 ページの「ゾーンがインストールされている Oracle Solaris システムでの公平配分スケジューラの使用」を参照してください。

CPU 配分の定義

「配分」という用語は、プロジェクトに割り当てられるシステムの CPU リソースの配分を定義するために使用されます。プロジェクトに割り当てる CPU 配分をほかのプロジェクトよりも多くすると、そのプロジェクトが公平配分スケジューラから受け取る CPU リソースも多くなります。

CPU 配分は、CPU リソースの比率ではありません。配分は、ほかの作業負荷との比較に基づいた作業負荷の相対的な重要性を定義します。プロジェクトに CPU 配分を割り当てる場合に重要なことは、プロジェクトが持つ配分自体ではありません。ほ

かのプロジェクトと比較して、そのプロジェクトが配分をいくつ持っているかを把握することが重要です。また、そのプロジェクトが CPU リソースについて、ほかのいくつのプロジェクトと競合しているかということも考慮に入れる必要があります。

注 – 配分がゼロのプロジェクトに属するプロセスは、常に最下位のシステム優先順位 (0) で実行されます。このようなプロセスが実行されるのは、配分がゼロでないプロジェクトが CPU リソースを使用していないときだけです。

CPU 配分とプロセスの状態

Oracle Solaris システムでは、プロジェクトの作業負荷は、通常複数のプロセスから構成されます。公平配分スケジューラの観点からは、各プロジェクトの作業負荷は、「アイドル」状態か「アクティブ」状態のどちらかです。プロジェクトのどのプロセスも CPU リソースを使用していないとき、プロジェクトはアイドル状態であるといいます。このような場合、プロセスは一般にスリープ (入出力の完了を待機している状態) または停止状態にあります。プロジェクトの 1 つ以上のプロセスが CPU リソースを使用しているとき、プロジェクトはアクティブ状態であるといいます。すべてのアクティブなプロジェクトが持つ配分の合計が、プロジェクトに割り当てられる CPU リソースの配分の計算に使用されます。

アクティブなプロジェクトが増えると、各プロジェクトの CPU 割り当ては減りますが、プロジェクト間の CPU 割り当て比率は変わりません。

CPU 配分と使用効率

配分割り当ては、使用効率とは異なります。CPU リソースの 50% が割り当てられているプロジェクトの CPU 使用率は、平均するとわずか 20% ほどです。その上、配分が CPU 使用量を制限するのは、ほかのプロジェクトと競合するときだけです。プロジェクトに対する割り当てが低い場合でも、そのプロジェクトがシステムで単独に実行されているときは、常に 100% の処理能力を CPU から受け取ります。使用可能な CPU サイクルが浪費されることはありません。つまり、使用可能な CPU サイクルはプロジェクト間に配分されます。

動作中の作業負荷に小さい配分を割り当てると、性能が低下します。ただし、システムが過負荷にならないかぎり、配分割り当て数が原因で作業が完了しないことはありません。

CPU 配分の例

2 つの CPU を搭載したシステムがあり、それらの CPU は CPU にバインドされた 2 つの作業負荷 *A* および *B* を並列に実行しているとします。各作業負荷は別個のプロジェクトとして実行されています。各プロジェクトは、プロジェクト *A* に S_A 配分が割り当てられ、プロジェクト *B* に S_B 配分が割り当てられるように構成されています。

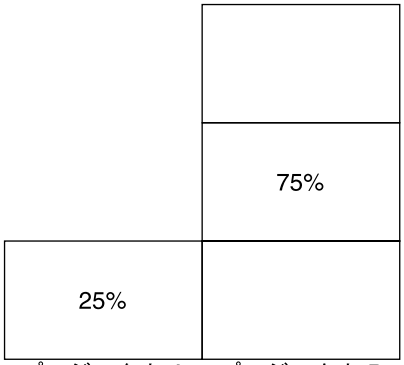
従来の TS スケジューラを使用した場合、システムで実行されている各作業負荷には、平均して同じ量の CPU リソースが与えられます。つまり、各作業負荷にはシステム容量の 50% が割り当てられます。

FSS スケジューラの制御で実行する場合でも、$S_A = S_B$ の配分を割り当てると、各プロジェクトにほぼ等量の CPU リソースが与えられます。これに対して、プロジェクトに異なる配分を与えた場合、CPU リソースの割り当て量は異なります。

次に示す 3 つの例は、さまざまな構成での配分の働きを示しています。これらの例に示されているとおり、配分は、要求が使用可能なリソース量と同じまたはそれを超えている場合にのみ使用量を数学的に正確に表します。

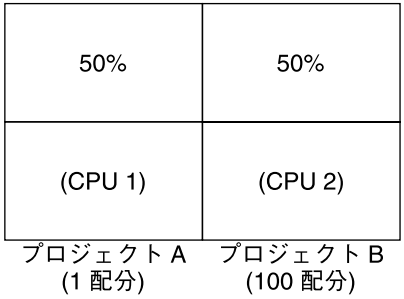
例 1: CPU にバインドされた 2 つのプロセスが各プロジェクトに存在する場合

プロジェクト *A* と *B* がそれぞれ CPU に結合されたプロセスを 2 つ持ち、かつ $S_A = 1$、$S_B = 3$ である場合、配分の合計数は 1 + 3 = 4 になります。この構成で、十分な数の CPU 要求があると、*A* と *B* には、それぞれ CPU リソースの 25%、75% が割り当てられます。



例 2: プロジェクト間に競合がない場合

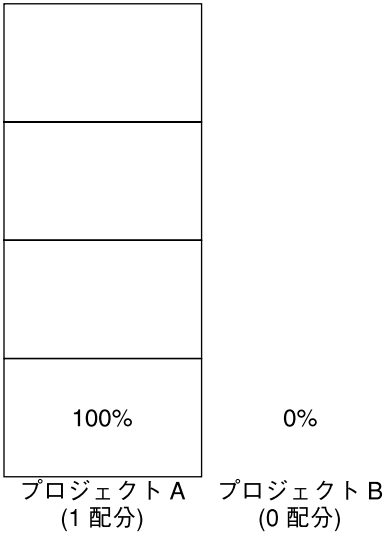
プロジェクト *A* と *B* がそれぞれ CPU に結合されたプロセスを「1 つ」だけ持ち、かつ $S_A = 1$、$S_B = 100$ である場合、配分の合計数は 101 になります。各プロジェクトは、実行中のプロセスを 1 つしか持たないため、CPU を 1 つしか使用できません。この構成では、CPU リソースを得るための競合がプロジェクト間に存在しないので、プロジェクト *A* および *B* には、それぞれ全 CPU リソースの 50% が割り当てられます。この構成の場合、CPU 配分は CPU リソースの割り当てに影響しません。プロジェクトへの割り当ては同じ (50/50) になります。これは、両方のプロジェクトに割り当てられる配分がゼロの場合でも同様です。



例 3: 一方のプロジェクトが実行されない場合

プロジェクト *A* と *B* がそれぞれ CPU に結合されたプロセスを 2 つ持ち、かつ *A* に 1 配分、*B* に 0 配分が与えられている場合、プロジェクト *B* には CPU リソースがまったく割り当てられず、プロジェクト *A* にすべての CPU リソースが割り当てられ

ます。プロジェクト *B* のプロセスは常にシステム優先順位 0 で実行されるため、実行される可能性はまったくありません。これは、プロジェクト *A* のプロセスの方が常に高い優先順位を持っているためです。



FSS の構成

プロジェクトとユーザー

プロジェクトは、FSS スケジューラの作業負荷コンテナです。プロジェクトに割り当てられているユーザーのグループは、個別の管理可能なブロックとして扱われます。個人ユーザー用に独自の配分を持つプロジェクトを作成できます。

ユーザーは、異なる配分が割り当てられているさまざまなプロジェクトのメンバーになることができます。プロセスをあるプロジェクトから別のプロジェクトに移動すると、プロセスに割り当てられる CPU リソース量は変化します。

`project(4)` データベースとネームサービスの詳細については、40 ページの「`project` データベース」を参照してください。

CPU 配分の構成

CPU 配分の構成は `project` データベースのプロパティーとして、ネームサービスによって管理されます。

プロジェクトに関連付けられている最初のタスクまたはプロセスが setproject(3PROJECT) ライブラリ関数を使って作成されると、project データベース内でリソース制御 project.cpu-shares として定義されている CPU 配分がカーネルに渡されます。リソース制御 project.cpu-shares が定義されていないプロジェクトには、1 配分が割り当てられます。

次の例では、/etc/project ファイル内のこのエントリでプロジェクト *x-files* の配分に *5* が設定されています。

```
x-files:100::::project.cpu-shares=(privileged,5,none)
```

プロジェクトに割り当てられている CPU 配分を、プロセスの実行中にデータベースで変更しても、プロジェクトの配分は、その時点では変更されません。変更内容を有効にするには、プロジェクトを再起動する必要があります。

project データベース内のプロジェクトの属性を変更することなくプロジェクトに割り当てられている配分を一時的に変更するには、prctl コマンドを使用します。たとえば、*x-files* プロジェクトに関連付けられているプロセスの実行中に、そのプロジェクトのリソース制御 project.cpu-shares の値を *3* に変更するには、次のように入力します。

```
# prctl -r -n project.cpu-shares -v 3 -i project x-files
```

詳細は、prctl(1) のマニュアルページを参照してください。

-r　　指定されたリソース制御の現在の値を置き換えます。

-n *name*　　リソース制御の名前を指定します。

-v *val*　　リソース制御の値を指定します。

-i *idtype*　　ID タイプを指定します。

x-files　　変更対象を指定します。この例では、プロジェクト *x-files* が変更対象です。

プロジェクト ID 0 のプロジェクト system には、ブート時の初期化スクリプトで起動されるすべてのシステムデーモンが含まれます。system は、無制限の配分を持つプロジェクトとしてみなされます。したがって、プロジェクト system は、ほかのプロジェクトに与えられている配分とは関係なく、常に最初にスケジュールされます。プロジェクト system に無制限の配分を割り当てない場合は、project データベースでこのプロジェクトの配分を変更します。

前述のように、配分がゼロのプロジェクトに属するプロセスには、常にシステム優先順位 0 が与えられます。配分が 1 以上のプロジェクトは、優先順位 1 以上で実行されます。したがって、配分がゼロのプロジェクトは、ゼロ以外の配分を持つプロジェクトが CPU リソースを要求していないときにだけスケジュールされます。

1 つのプロジェクトに割り当てられる配分の最大数は 65535 です。

FSS とプロセッサセット

プロセッサセットに FSS を連携して使用すると、連携させずにプロセッサセットのみで利用できる、各プロセッサセット上で実行するプロジェクト間の CPU リソースの割り当ての制御よりもきめ細かく制御できます。FSS スケジューラは、プロセッサセットを完全に独立したパーティションとして処理します。つまり、各プロセッサセットは、CPU 割り当てについてそれぞれ個別に制御されます。

1 つのプロセッサセットで実行されるプロジェクトの CPU 割り当てが、別のプロセッサセットで実行されるプロジェクトの CPU 配分や動作によって影響を受けることはありません。なぜなら、異なるプロセッサセットで実行されるプロジェクトが同じリソースについて競合することはないからです。競合が発生するのは、プロジェクトが同じプロセッサセット内で実行されている場合だけです。

プロジェクトに割り当てられている配分はシステム全体に適用されます。どのプロセッサセットで実行されようと、プロジェクトの各部分には同じ配分が与えられます。

プロセッサセットが使用されている場合、プロジェクトの CPU 割り当ては、各プロセッサセット内で実行されるアクティブなプロジェクトに対して算出されます。

異なるプロセッサセット内で実行されるプロジェクトのパーティションは、異なる CPU 割り当てを持つことになります。1 つのプロセッサセット内の各プロジェクトパーティションに対する CPU 割り当ては、同じプロセッサセット内で実行されるほかのプロジェクトの割り当てにだけ依存します。

プロセッサセット境界内で実行されるアプリケーションの性能と可用性が、新しいプロセッサセットの導入によって影響を受けることはありません。また、ほかのプロセッサセットで実行されるプロジェクトの配分割り当ての変更によって、アプリケーションが影響を受けることもありません。

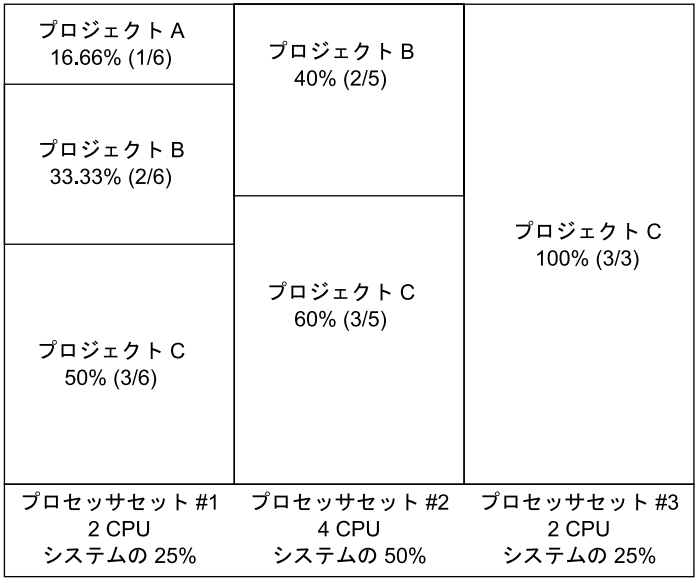
空のプロセッサセット (プロセッサが存在しないセット) や、プロセッサセットにバインドされたプロセスを持たないプロセッサセットは、FSS スケジューラの動作にまったく影響を与えません。

FSS とプロセッサセットの例

8 つの CPU を持つサーバーがプロジェクト *A*、*B*、および *C* 内で CPU にバインドされたアプリケーションをいくつか実行しているものとします。プロジェクト *A* には 1 配分、プロジェクト *B* には 2 配分、プロジェクト *C* には 3 配分がそれぞれ割り当てられています。

プロジェクト *A* は、プロセッサセット 1 だけで実行されています。プロジェクト *B* は、プロセッサセット 1 および 2 で実行されています。プロジェクト *C* は、プロ

セッサセット 1、2、および 3 で実行されています。各プロジェクトには、使用可能なすべての CPU 処理能力を利用するだけの十分な数のプロセスが存在しているものとします。したがって、CPU リソースを得るための競合が各プロセッサセットで常に発生します。



このようなシステムでは、システム全体でのプロジェクトの CPU 割り当ての合計は次のようになります。(pset = プロセッサセット)

プロジェクト	割り当て
プロジェクト A	$4\% = (1/6 \text{ X } 2/8)_{pset1}$
プロジェクト B	$28\% = (2/6 \text{ X } 2/8)_{pset1} + (2/5 * 4/8)_{pset2}$
プロジェクト C	$67\% = (3/6 \text{ X } 2/8)_{pset1} + (3/5 \text{ X } 4/8)_{pset2} + (3/3 \text{ X } 2/8)_{pset3}$

これらの割合は、プロジェクトに与えられている CPU 配分値とは一致しません。ただし、各プロセッサセット内では、プロジェクトごとの CPU 割り当て比率はプロジェクトのそれぞれの配分に比例します。

このシステム上にプロセッサセットが存在しない場合、CPU リソースの配分は、次に示すように、異なったものになります。

プロジェクト	割り当て
プロジェクト A	16.66% = (1/6)
プロジェクト B	33.33% = (2/6)
プロジェクト C	50% = (3/6)

FSS とほかのスケジューリングクラスの併用

デフォルトでは、FSS スケジューリングクラスは、タイムシェアリング (TS)、対話型 (IA)、および固定優先順位 (FX) の各スケジューリングクラスと同じ範囲の優先順位 (0 から 59) を使用します。そのため、これらのスケジューリングクラスのプロセスが同じプロセッサセットを共有しないようにする必要があります。FSS、TS、IA、および FX の各クラスにプロセスが混在すると、予期せぬスケジューリング処理が実行される場合があります。

プロセッサセットを使用する場合は、1 つのシステム内で TS、IA、および FX を FSS と混在させることができます。ただし、各プロセッサセットで実行されるすべてのプロセスは、「1 つの」スケジューリングクラスに所属している必要があります。このようにすると、これらのプロセスが同じ CPU について競合することはありません。プロセッサセットを使用しない場合は、特に FX スケジューラを FSS スケジューリングクラスと併用しないようにしてください。これにより、FX クラスのアプリケーションが高い優先順位を使用して、FSS クラスのアプリケーションの実行を妨げることはありません。

TS クラスと IA クラスのプロセスは、同じプロセッサセット内で、またはプロセッサセットが存在しない同じシステム内で混在させることができます。

Oracle Solaris システムでは、root 権限を持つユーザーに、リアルタイム (RT) スケジューラも提供されています。デフォルトでは、RT スケジューリングクラスは FSS とは異なる範囲のシステム優先順位 (通常は 100 から 159) を使用します。RT と FSS は「互いに素」な範囲 (重複しない範囲) の優先順位を使用しているので、FSS は同じプロセッサセット内の RT スケジューリングクラスと共存できます。ただし、FSS スケジューリングクラスは、RT クラスで実行するプロセスを制御することはできません。

たとえば、4 つのプロセッサから構成されるシステムで、CPU に結合されているシングルスレッドの RT プロセスは 1 つのプロセッサを占有できます。システムが FSS も実行している場合、通常のユーザープロセスは、RT プロセスが使用していない残りの 3 つの CPU について競合します。RT プロセスは CPU を使い続けることはありません。RT プロセスがアイドル状態になったとき、FSS は 4 つのプロセッサをすべて使用します。

次のコマンドを入力して、プロセッサセットが実行しているスケジューリングクラスを特定し、各プロセッサセットが TS、IA、FX、または FSS のプロセスのいずれかを実行するように構成されていることを確認します。

```
$ ps -ef -o pset,class | grep -v CLS | sort | uniq
1 FSS
1 SYS
2 TS
2 RT
3 FX
```

システムのスケジューリングクラスの設定

システムにデフォルトのスケジューリングクラスを設定するには、118 ページの「FSS をデフォルトのスケジューラクラスにする方法」、220 ページの「スケジューリングクラス」、および dispadmin(1M) を参照してください。実行中のプロセスを別のスケジューリングクラスに移動するには、117 ページの「FSS の構成」と priocntl(1) を参照してください。

ゾーンがインストールされているシステムでのスケジューリングクラス

非大域ゾーンでは、システムのデフォルトのスケジューリングクラスが使用されます。システムのデフォルトスケジューリングクラスの設定が更新された場合、非大域ゾーンでは、ブート時またはリブート時にこの新しい設定が取得されます。

この場合に望ましい FSS の使用方法は、dispadmin コマンドを使用して、FSS をシステムのデフォルトのスケジューリングクラスに設定する方法です。このようにすると、すべてのゾーンがシステムの CPU リソースの公平配分を受けることができます。ゾーンが使用されている場合のスケジューリングクラスの詳細については、220 ページの「スケジューリングクラス」を参照してください。

デフォルトのスケジューリングクラスの変更やリブートを行うことなく、実行中のプロセスを別のスケジューリングクラスに移動する方法については、表 25–5 および priocntl(1) のマニュアルページを参照してください。

FSS で使用するコマンド

次の表に示すコマンドにより、公平配分スケジューラに対する主要な管理インタフェースが提供されます。

コマンド	説明
priocntl(1)	指定されたプロセスのスケジューリングパラメータを表示または設定します。実行中のプロセスを別のスケジューリングクラスに移動します。

コマンド	説明
ps(1)	実行中のプロセスに関する情報を一覧表示します。プロセッサセットがどのスケジューリングクラスで実行されているかを示します。
dispadmin(1M)	システム上で利用できるスケジューラをリスト表示します。システムのデフォルトのスケジューラを設定します。FSS スケジューラのタイムクォンタム (time quantum) 値を調べ、調整する場合にも使用されます。
FSS(7)	公平配分スケジューラ (FSS) を記述します。

◆ ◆ ◆ 第 9 章

9 公平配分スケジューラの管理 (タスク)

この章では、公平配分スケジューラ (FSS) の使用方法について説明します。

FSS の概要については、第 8 章「公平配分スケジューラ (概要)」を参照してください。ゾーンが使用されている場合のスケジューリングクラスの詳細については、220 ページの「スケジューリングクラス」を参照してください。

公平配分スケジューラの管理 (タスクマップ)

タスク	説明	参照先
CPU 使用量を監視します。	プロジェクトの CPU 使用量およびプロセッサセット内のプロジェクトの CPU 使用量を監視します。	116 ページの「FSS の監視」
デフォルトのスケジューラクラスを設定します。	FSS などのスケジューラをシステムのデフォルトスケジューラとして設定します。	118 ページの「FSS をデフォルトのスケジューラクラスにする方法」
実行中のプロセスを、あるスケジューリングクラスから FSS クラスなどの別のスケジューリングクラスに移動します。	デフォルトのスケジューリングクラスの変更やリブートを行うことなく、あるスケジューリングクラスから別のスケジューリングクラスにプロセスを手動で移動します。	118 ページの「プロセスを TS クラスから FSS クラスに手動で移動する方法」
すべての実行中のプロセスを、FSS クラスなどの別のスケジューリングクラスに移動します。	デフォルトのスケジューリングクラスの変更やリブートを行うことなく、すべてのスケジューリングクラスから別のスケジューリングクラスにプロセスを手動で移動します。	119 ページの「プロセスをすべてのユーザークラスから FSS クラスに手動で移動する方法」

タスク	説明	参照先
プロジェクトのプロセスを、FSS クラスなどの別のスケジューリングクラスに移動します。	プロジェクトのプロセスを、現在のスケジューリングクラスから別のスケジューリングクラスに手動で移動します。	119 ページの「プロジェクトのプロセスを FSS クラスに手動で移動する方法」
FSS パラメータを調べ、調整します。	スケジューラのタイムクォンタムの値を調整します。「タイムクォンタム」とは、スレッドがプロセッサ上で実行を開始してからそのプロセッサを放棄するまでの時間量のことです。	119 ページの「スケジューラのパラメータを調整する方法」

FSS の監視

prstat コマンド (prstat(1M) のマニュアルページに記載) を使用すると、アクティブなプロジェクトごとの CPU 使用量を監視できます。

タスク用の拡張アカウンティングデータを使用して、長期間使用される CPU リソースの合計量について、プロジェクトごとの統計情報を取得できます。詳細は、第 4 章「拡張アカウンティング (概要)」を参照してください。

▼ システムの CPU 使用量をプロジェクトごとに監視する方法

- システム上で実行されているプロジェクトの **CPU** 使用量を監視するには、**prstat** コマンドに **-J** オプションを付けて実行します。

```
# prstat -J
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
  5107 root     4556K 3268K cpu0    59    0   0:00:00 0.0% prstat/1
  4570 root       83M   47M sleep   59    0   0:00:25 0.0% java/13
  5105 bobbyc   3280K 2364K sleep   59    0   0:00:00 0.0% su/1
  5106 root     3328K 2580K sleep   59    0   0:00:00 0.0% bash/1
     5 root        0K    0K sleep   99  -20   0:00:14 0.0% zpool-rpool/138
   333 daemon   7196K 2896K sleep   59    0   0:00:07 0.0% rcapd/1
    51 netcfg   4436K 3460K sleep   59    0   0:00:01 0.0% netcfgd/5
  2685 root     3328K 2664K sleep   59    0   0:00:00 0.0% bash/1
   101 netadm   4164K 2824K sleep   59    0   0:00:01 0.0% ipmgmtd/6
   139 root     6940K 3016K sleep   59    0   0:00:00 0.0% syseventd/18
  5082 bobbyc   2236K 1700K sleep   59    0   0:00:00 0.0% csh/1
    45 root       15M 7360K sleep   59    0   0:00:01 0.0% dlmgmtd/7
    12 root       23M   22M sleep   59    0   0:00:45 0.0% svc.configd/22
    10 root       15M   13M sleep   59    0   0:00:05 0.0% svc.startd/19
   337 netadm   6768K 5620K sleep   59    0   0:00:01 0.0% nwamd/9
PROJID    NPROC  SWAP   RSS MEMORY      TIME  CPU PROJECT
     1        6   25M   18M   0.9%   0:00:00 0.0% user.root
```

```
     0       73  479M  284M    14%   0:02:31 0.0% system
     3        4   28M   24M   1.1%   0:00:26 0.0% default
    10        2   14M 7288K   0.3%   0:00:00 0.0% group.staff

Total: 85 processes, 553 lwps, load averages: 0.00, 0.00, 0.00
```

▼ プロセッサセット内の CPU 使用量をプロジェクトごとに監視する方法

- 1 つまたは複数のプロセッサセットについて、プロジェクトの CPU 使用量を監視するには、次のように入力します。

  ```
  % prstat -J -C pset-list
  ```

  ここで *pset-list* は、プロセッサセット ID をコンマで区切って並べたリストです。

FSS の構成

Oracle Solaris システムのほかのスケジューリングクラスで使用するコマンドと同じコマンドを、FSS で使用することもできます。スケジューラクラス、スケジューラの調整可能パラメータ、および個々のプロセスのプロパティーを構成できます。

`svcadm restart` を使用すると、スケジューラサービスを再起動することができます。詳細は、`svcadm(1M)` のマニュアルページを参照してください。

システム上のスケジューラクラスのリスト表示

システム上のスケジューラクラスを表示するには、`-l` オプションを指定した `dispadmin` コマンドを使用します。

```
$ dispadmin -l
CONFIGURED CLASSES
==================

SYS     (System Class)
TS      (Time Sharing)
SDC     (System Duty-Cycle Class)
FSS     (Fair Share)
FX      (Fixed Priority)
IA      (Interactive)
```

▼ FSS をデフォルトのスケジューラクラスにする方法

CPU 配分割り当てを有効にするには、FSS をシステムのデフォルトのスケジューラにする必要があります。

`priocntl` と `dispadmin` コマンドを組み合わせて使用することにより、FSS はただちにデフォルトのスケジューラになり、この設定はリブート後も有効です。

1 **root** になるか、同等の役割になります。

2 システムのデフォルトのスケジューラが **FSS** になるように設定します。

```
# dispadmin -d FSS
```

この変更指定は次のリブートで有効になります。リブート後は、システムのすべてのプロセスが FSS スケジューリングクラスで実行されます。

3 リブートを行わずに、この構成をただちに有効にします。

```
# priocntl -s -c FSS -i all
```

▼ プロセスを TS クラスから FSS クラスに手動で移動する方法

デフォルトのスケジューリングクラスを変更した後でリブートしなくても、あるスケジューリングクラスから別のスケジューリングクラスにプロセスを手動で移動できます。次の手順は、TS スケジューリングクラスから FSS スケジューリングクラスにプロセスを手動で移動する方法を示しています。

1 **root** になるか、同等の役割になります。

2 **init** プロセス **(pid 1)** を **FSS** スケジューリングクラスに移動します。

```
# priocntl -s -c FSS -i pid 1
```

3 すべてのプロセスを **TS** スケジューリングクラスから **FSS** スケジューリングクラスに移動します。

```
# priocntl -s -c FSS -i class TS
```

注 – すべてのプロセスは、リブート後には再び TS スケジューリングクラスで実行されます。

▼ プロセスをすべてのユーザークラスから FSS クラスに手動で移動する方法

TS 以外のデフォルトのクラスを使用している場合、たとえば、デフォルトで IA クラスを使用するウィンドウ環境がシステムで実行されている場合があります。デフォルトのスケジューリングクラスを変更した後でリブートしなくても、すべてのプロセスを FSS スケジューリングクラスに手動で移動できます。

1 **root** になるか、同等の役割になります。

2 **init** プロセス **(pid 1)** を **FSS** スケジューリングクラスに移動します。

```
# priocntl -s -c FSS -i pid 1
```

3 すべてのプロセスを現在のスケジューリングクラスから **FSS** スケジューリングクラスに移動します。

```
# priocntl -s -c FSS -i all
```

注 – すべてのプロセスは、リブート後には再びデフォルトのスケジューリングクラスで実行されます。

▼ プロジェクトのプロセスを FSS クラスに手動で移動する方法

プロジェクトのプロセスを、現在のスケジューリングクラスから FSS スケジューリングクラスに手動で移動できます。

1 **root** になるか、同等の役割になります。

2 プロジェクト **ID** *10* で実行するプロセスを **FSS** スケジューリングクラスに移動します。

```
# priocntl -s -c FSS -i projid 10
```

プロジェクトのプロセスは、リブート後には再びデフォルトのスケジューリングクラスで実行されます。

スケジューラのパラメータを調整する方法

`dispadmin` コマンドを使用すると、システムの稼働中にプロセススケジューラパラメータを表示または変更できます。たとえば、`dispadmin` コマンドを使用して、FSS

スケジューラのタイムクォンタム (time quantum) 値を調べ、調整できます。「タイムクォンタム」とは、スレッドがプロセッサ上で実行を開始してからそのプロセッサを放棄するまでの時間量のことです。

システムの稼働中に FSS スケジューラの現在のタイムクォンタムを表示するには、次のように入力します。

```
$ dispadmin -c FSS -g
#
# Fair Share Scheduler Configuration
#
RES=1000
#
# Time Quantum
#
QUANTUM=110
```

`-g` オプションを使用するときに、同時に `-r` オプションも指定すると、タイムクォンタム値の表示に使用する最小単位を指定できます。最小単位を指定しないと、タイムクォンタム値はデフォルトのミリ秒で表示されます。

```
$ dispadmin -c FSS -g -r 100
#
# Fair Share Scheduler Configuration
#
RES=100
#
# Time Quantum
#
QUANTUM=11
```

FSS スケジューリングクラスにスケジューリングパラメータを設定するには、`dispadmin -s` を使用します。*file* 内の値は、`-g` オプションで得られる出力と同じ形式で指定する必要があります。これらの値は、カーネル内の現在の値を上書きします。次の行を入力します。

```
$ dispadmin -c FSS -s file
```

◆ ◆ ◆ 第 10 章

10 リソース上限デーモンによる物理メモリーの制御 (概要)

リソース上限デーモン rcapd を使用すると、リソース上限が定義されたプロジェクト内で動作するプロセスが消費する物理メモリーを規制できます。システムでゾーンを実行している場合は、大域ゾーンから rcapd を使用して、非大域ゾーンでの物理メモリーの消費を規制できます。第 17 章「非大域ゾーンの計画と構成 (タスク)」を参照してください。

この章の内容は次のとおりです。

- 121 ページの「リソース上限デーモンの紹介」
- 122 ページの「リソース上限制御のしくみ」
- 122 ページの「プロジェクトの物理メモリーの使用率を制限する属性」
- 123 ページの「rcapd の構成」
- 127 ページの「rcapstat によるリソース使用効率の監視」
- 129 ページの「rcapd とともに使用されるコマンド」

rcapd ユーティリティーを使用した手順については、第 11 章「リソース上限デーモンの管理 (タスク)」を参照してください。

リソース上限デーモンの紹介

リソース「上限」とは、物理メモリーなどのリソース消費量の上限のことです。物理メモリーの上限はプロジェクトごとに設定します。

リソース上限デーモンとその関連付けられたユーティリティーは、物理メモリーのリソース上限の実行および管理のためのメカニズムを提供します。

リソース制御と同様に、リソース上限を定義するには、project データベース内にあるプロジェクトエントリの属性を使用します。リソース制御はカーネルによって同期的に実行されますが、物理メモリーのリソース上限の制限はリソース上限デーモンによってユーザーレベルで非同期的に実行されます。このリソース上限の制限は非同期的に実行されるため、リソース上限デーモンがサンプリングを行う間隔の分だけ、短時間の遅延が発生します。

rcapd については、rcapd(1M) のマニュアルページを参照してください。プロジェクトと project データベースについては、第 2 章「プロジェクトとタスク (概要)」 および project(4) のマニュアルページを参照してください。リソース制御については、第 6 章「リソース制御 (概要)」を参照してください。

リソース上限制御のしくみ

リソース上限デーモンは、物理メモリー上限が定義されたプロジェクトのリソース使用効率を定期的にサンプリングします。このサンプリング間隔は管理者が指定できます。詳細は、127 ページの「サンプリング間隔の決定」を参照してください。システムの物理メモリー使用効率が上限実行しきい値を超え、ほかの条件にも適合する場合、リソース上限デーモンは、物理メモリー上限が定義されたプロジェクトのリソース消費をその上限以下に減らします。

仮想メモリーシステムは物理メモリーを複数の「ページ」に分割します。「ページ」は、Oracle Solaris メモリー管理サブシステムにおける物理メモリーのベースとなる単位です。データをファイルからメモリーに読み取るとき、仮想メモリーシステムは 1 度に 1 ページずつ読み取ります。この動作のことを、ファイルの「ページイン」と呼びます。リソース消費を減らすとき、リソース上限デーモンは、あまり使用されていないページをスワップデバイス (物理メモリーの外にある領域) に再配置します。この動作のことを「ページアウト」と呼びます。

物理メモリーを管理するために、リソース上限デーモンは、プロジェクトの作業負荷の常駐セットのサイズを、作業セットのサイズに対して調節します。常駐セットとは、物理メモリーに常駐するページのことです。作業セットとは、作業負荷がその処理サイクル中にアクティブに使用するページのことです。作業セットは、プロセスが動作するモードや処理するデータの種類に応じて、そのときどきで変化します。すべての作業負荷がその作業セットの常駐に十分な物理メモリーにアクセスできるのが理想です。しかし、作業セットは、物理メモリーのほか、セカンダリディスク記憶装置にも格納することが可能です。

rcapd は 1 度に 1 つのインスタンスしか実行できません。

プロジェクトの物理メモリーの使用率を制限する属性

物理メモリーリソースの上限をプロジェクトに対して定義するには、project データベースエントリにこの属性を追加して、常駐セットサイズ (RSS) の上限を設定します。

rcap.max-rss　プロジェクト内のプロセスが利用できる物理メモリーの合計 (バイト数)。

たとえば、/etc/project ファイル内の次の行は、10G バイトの RSS 上限を db というプロジェクトに対して設定します。

```
db:100::db,root::rcap.max-rss=10737418240
```

注 – 指定した上限値はシステムによってページのサイズに丸められることがあります。

また、projmod コマンドを使用して、/etc/project ファイル内に rcap.max-rss 属性を設定することもできます。

詳細については、「常駐セットサイズ上限の設定」を参照してください。

rcapd の構成

リソース上限デーモンを構成するには、rcapadm コマンドを使用します。次のような作業が可能です。

- 上限実行しきい値を設定します
- rcapd が実行する動作間隔を設定します
- リソース上限制御を有効または無効にします
- 構成済みのリソース上限デーモンの現在のステータスを表示します

このデーモンを構成するには、root ユーザーであるか、または必要な管理権限を持っている必要があります。

構成の変更を rcapd に適用するには、構成間隔に従うか (126 ページの「rcapd の動作間隔」を参照)、または必要に応じて SIGHUP を送信します (kill(1) のマニュアルページを参照)。

引数なしで使用した場合、rcapadm はリソース上限デーモンの現在のステータスを表示します (構成されている場合のみ)。

次の項では、上限の制限、上限値、および rcapd の動作間隔について説明します。

ゾーン環境がインストールされているシステムでのリソース上限デーモンの使用

ゾーンを構成するときに capped-memory リソースを設定することにより、ゾーンの常駐セットサイズ (RSS) 使用量を制御できます。詳細については、221 ページの「物理メモリーの制御と capped-memory リソース」を参照してください。capped-memory リソースを使用するには、resource-cap パッケージが大域ゾーン内にインストールされている必要があります。大域ゾーンも含むゾーン内で rcapd を実行して、そのゾーン内のプロジェクトにメモリー上限を適用できます。

特定のゾーンで消費可能なメモリー量に、次のリブートまで一時的な上限を設定することができます。135 ページの「ゾーンに一時的なリソース上限を指定する方法」を参照してください。

rcapd をゾーンで使用して、リソース上限が定義されたプロジェクト内で実行されるプロセスによる物理メモリーの消費を規制する場合は、それらのゾーン内でデーモンを構成する必要があります。

別のゾーン内にあるアプリケーションのメモリー上限を選択する場合、通常は、そのアプリケーションが別のゾーン内にあることを考慮する必要はありません。例外は、ゾーン別のサービスの場合です。ゾーン別のサービスは、メモリーを消費します。システムの物理メモリー量およびメモリー上限を決定する際には、このメモリー消費を考慮してください。

メモリー上限実行しきい値

「メモリー上限実行しきい値」とは、上限制限を引き起こす、システム上の物理メモリーの使用効率 (パーセンテージ) のことです。システムの物理メモリー使用率がこのしきい値を超えたとき、メモリー上限が制限されます。この使用率には、アプリケーションやカーネルが使用する物理メモリーも含まれます。この使用効率は、メモリー上限が制限される方法を決定します。

上限が制限されると、プロジェクトの作業負荷からメモリーがページアウトされることがあります。

- メモリーをページアウトすることによって、作業負荷に定義された上限を超えた分のメモリーのサイズを小さくすることができます。
- メモリーをページアウトすることによって、システム上のメモリー上限実行しきい値を超えた物理メモリーの使用率を下げることができます。

作業負荷は、定義された上限までの物理メモリーを使用することが許可されます。システムのメモリー使用率がメモリー上限実行しきい値を下回っている場合、作業負荷は上限より多くのメモリーを使用できます。

上限実行しきい値を設定する方法については、133 ページの「メモリー上限実行しきい値を設定する方法」を参照してください。

上限値の決定

プロジェクトの上限の設定が低すぎると、通常の状態でも、作業負荷が効率的に機能するだけのメモリーを使用できない可能性があります。作業負荷がより多くのメモリーを要求するためページングが発生し、システムの性能に悪影響がでます。

プロジェクトの上限の設定が高すぎると、上限に達する前に、利用可能な物理メモリーを使い果たす可能性があります。この場合、物理メモリーは、rcapd ではなくカーネルによって効率的に管理されます。

プロジェクトの上限を決定するときには、次の要素を考慮します。

入出力システムへの影響

サンプリングした使用率がプロジェクトの上限を超えている場合、リソース上限デーモンはプロジェクトの作業負荷の物理メモリー使用率を減らそうとします。上限が制限されている間は、作業負荷がマッピングしているファイルには、スワップなどのデバイスが使用されます。上限を頻繁に超える作業負荷の場合、その性能は、スワップデバイスの性能に大きく左右されます。このような作業負荷を実行することは、作業負荷の上限と同じサイズの物理メモリーを持つマシン上で作業負荷を実行することと似ています。

CPU 使用率への影響

リソース上限デーモンの CPU 使用率は、このデーモンが上限を制限するプロジェクトの作業負荷内のプロセスの数と、作業負荷のアドレス空間のサイズによって変化します。

リソース上限デーモンの CPU 時間の一部は、作業負荷の使用率のサンプリングに費やされます。作業負荷にプロセスを追加すると、使用率のサンプリングにかかる時間が増えます。

上限値を超えると上限が制限され、リソース上限デーモンの CPU 時間がさらに消費されます。消費される CPU 時間は仮想メモリーの量に比例します。消費される CPU 時間は、作業負荷のアドレス空間の合計サイズの変化によって増減します。この情報は、rcapstat の出力の vm 列に報告されます。詳細は、127 ページの「rcapstat によるリソース使用効率の監視」および rcapstat(1) のマニュアルページを参照してください。

共有メモリーの報告

rcapd デーモンは、ほかのプロセスと共有されているメモリーページ、あるいは、同じプロセス内で複数回マッピングされているメモリーページについて、かなり正確な RSS の見積もりを報告します。異なるプロジェクトのプロセスが同じメモリーを共有している場合、そのメモリーは、そのメモリーを共有しているすべてのプロジェクトの RSS の合計に含められます。

この見積もりは、データベースのように共有メモリーを多用する作業負荷に使用できます。データベースの作業

負荷では、次のようにプロジェクトの通常の使用率をサンプリングすることによって、適切な初期上限値を決定することもできます。prstat コマンドに -J または -Z オプションを付けて実行し、その出力を使用します。詳細は、prstat(1M) のマニュアルページを参照してください。

rcapd の動作間隔

rcapd を定期的に実行するように、rcapd の動作間隔を設定できます。

すべての間隔は秒単位で指定します。次の表で、rcapd の動作とそのデフォルトの間隔値について説明します。

動作	デフォルトの間隔値 (秒)	説明
scan	15	プロジェクトの作業負荷内で動作しているプロセスを走査する間隔の秒数。最小値は 1 秒です。
sample	5	常駐セットサイズのサンプリングから、その後に上限を制限するまでの間の秒数。最小値は 1 秒です。
report	5	ページング統計を更新する間隔の秒数。0 に設定すると、ページング統計は更新されず、rcapstat からの出力も最新の状態を示さなくなります。
config	60	再構成する間隔の秒数。再構成イベントでは、rcapadm は構成ファイルを更新用に読み取って、project データベースに新しいまたは改訂されたプロジェクト上限があるかどうかを走査します。SIGHUP を rcapd に送信すると、再構成が即座に行われます。

間隔を調節する方法については、133 ページの「動作間隔を設定する方法」を参照してください。

rcapd 走査間隔の決定

走査間隔は、rcapd が新しいプロセスを探す頻度を制御します。多くのプロセスが動作しているシステムでは、一覧の走査に時間がかかるため、走査間隔を長くして、消費される CPU 時間の合計を減らしたほうがよい場合もあります。しかし、走査間隔は、あるプロセスの存在が上限が定義されている作業負荷に属するとみなされるまでに最低限必要な時間も意味します。生存期間が短いプロセスを数多く実行する作業負荷の場合、走査間隔が長いと、rcapd はそれらのプロセスが作業負荷に属さないものとみなす可能性があります。

サンプリング間隔の決定

rcapadm で構成したサンプリング間隔は、作業負荷の使用率をサンプリングして上限を超えていた場合に、rcapd が上限を適用するまで待機する最短時間です。サンプリング間隔を短くすると、ほとんどの場合、rcapd が上限を頻繁に制限するため、ページングによる入出力が増えます。しかし、サンプリング間隔を短くすると、特定の作業負荷の物理メモリー使用率が急増した場合に、ほかの作業負荷への影響を抑えることにもなります。サンプリングの合間には、この作業負荷は自由にメモリーを消費でき、上限が定義されているほかの作業負荷のメモリーすらも利用できますが、この合間が狭められることになるのです。

rcapstat に指定したサンプリング間隔が、rcapadm で rcapd に指定したサンプリング間隔よりも短い場合、いくつかの間隔に対する出力がゼロになることがあります。この状況が発生するのは、rcapd が統計を更新する間隔が、rcapadm で指定した間隔よりも長いためです。rcapadm で指定した間隔は、rcapstat で使用されるサンプリング間隔から独立しています。

rcapstat によるリソース使用効率の監視

上限が定義されたプロジェクトのリソース使用効率を監視するには、rcapstat を使用します。rcapstat の報告例については、135 ページの「rcapstat による報告の生成」を参照してください。

報告用のサンプリング間隔、および統計を繰り返す回数を設定できます。

interval サンプリング間隔を指定します (秒)。デフォルトの間隔は 5 秒です。

count 統計を繰り返す回数を指定します。デフォルトでは、終了シグナルを受信するまで、あるいは、rcapd プロセスが終了するまで、rcapstat は統計を報告し続けます。

rcapstat が最初に発行する報告では、ページング統計はデーモンの起動以降の活動を示します。以後の報告では、前回の報告以降の活動を示します。

次の表に、rcapstat の報告の列見出しを定義します。

rcapstat の列見出し	説明
id	上限が定義されたプロジェクトの ID。
プロジェクト	プロジェクト名。
nproc	プロジェクト内のプロセス数。
vm	プロジェクト内のプロセスが使用する仮想メモリーサイズの合計。マップされたファイルおよびデバイスもすべて含みます。単位は、K バイト (K)、M バイト (M)、または G バイト (G) です。
rss	プロジェクト内のプロセスが使用すると推定される常駐セットサイズ (RSS) の合計。単位は、K バイト (K)、M バイト (M)、または G バイト (G) です。共有されるページは考慮されません。
cap	プロジェクトに定義された RSS 上限。メモリー上限を指定する方法については、122 ページの「プロジェクトの物理メモリーの使用率を制限する属性」または rcapd(1M) のマニュアルページを参照してください。
at	前回の rcapstat のサンプリング以降、 rcapd がページアウトしようとしたメモリーの合計。
avgat	前回の rcapstat のサンプリング以降に発生した各サンプリングサイクル中、rcapd がページアウトしようとしたメモリーの平均。rcapd がコレクション RSS をサンプリングする頻度は、rcapadm で設定できます。126 ページの「rcapd の動作間隔」を参照してください。
pg	前回の rcapstat のサンプリング以降、 rcapd が正常にページアウトしたメモリーの合計。
avgpg	前回の rcapstat のサンプリング以降に発生した各サンプリングサイクル中、rcapd が正常にページアウトしたと推定されるメモリーの平均。rcapd がプロセス RSS をサンプリングする頻度は rcapadm で設定できます。126 ページの「rcapd の動作間隔」を参照してください。

rcapd とともに使用されるコマンド

コマンド	説明
rcapstat(1)	上限が定義されたプロジェクトのリソース使用効率を監視します。
rcapadm(1M)	リソース上限デーモンを構成します。構成済みのリソース上限デーモンの現在のステータスを表示します。リソース上限制御を有効または無効にします。一時的なメモリー上限を設定する場合にも使用されます。
rcapd(1M)	リソース上限デーモン。

◆ ◆ ◆ 第 11 章

11 リソース上限デーモンの管理 (タスク)

この章では、リソース上限デーモン rcapd を構成して使用する手順について説明します。

rcapd の概要については、第 10 章「リソース上限デーモンによる物理メモリーの制御 (概要)」を参照してください。

常駐セットサイズ上限の設定

project データベースエントリに rcap.max-rss 属性を追加することによって、プロジェクトに対して物理メモリーリソースの常駐セットサイズ (RSS) の上限を定義します。

▼ プロジェクトに rcap.max-rss 属性を追加する方法

1. **root** になるか、同等の役割になります。
2. この属性を **/etc/project** ファイルに次のように追加します。

   rcap.max-rss=*value*

例 11–1 RSS プロジェクトの上限

/etc/project ファイル内の次の行は、10G バイトの RSS 上限を db というプロジェクトに対して設定します。

```
db:100::db,root::rcap.max-rss=10737418240
```

指定した上限値がシステムによってページのサイズに丸められることがあることに注意してください。

▼ projmod コマンドを使用してプロジェクトに rcap.max-rss 属性を追加する方法

1 root になるか、同等の役割になります。

2 /etc/project ファイルに 10G バイトの rcap.max-rss 属性を設定します。この場合は db という名前のプロジェクトです。

```
# projmod -a -K rcap.max-rss=10GB db
```

/etc/project ファイルには、次の行が含まれるようになります。

```
db:100::db,root::rcap.max-rss=10737418240
```

リソース上限デーモンの構成と使用 (タスクマップ)

タスク	説明	参照先
メモリー上限実行しきい値を設定します。	プロセスが利用できる物理メモリーが少なくなったときに制限する上限を構成します。	133 ページの「メモリー上限実行しきい値を設定する方法」
動作間隔を設定します。	この間隔は、リソース上限デーモンが行う定期的な動作に適用されます。	133 ページの「動作間隔を設定する方法」
リソース上限制御を有効にします。	システムでリソース上限制御を起動します。	134 ページの「リソース上限制御を有効にする方法」
リソース上限制御を無効にします。	システムでリソース上限制御を停止します。	134 ページの「リソース上限制御を無効にする方法」
上限とプロジェクトの情報を報告します。	報告を生成するためのコマンド例を表示します。	135 ページの「上限とプロジェクトの情報の報告」
プロジェクトの常駐セットサイズを監視します。	プロジェクトの常駐セットサイズについて報告を生成します。	136 ページの「プロジェクトの RSS の監視」
プロジェクトの作業セットサイズを決定します。	プロジェクトの作業セットサイズについて報告を生成します。	137 ページの「プロジェクトの作業セットサイズの決定」
メモリー使用効率とメモリー上限を報告します。	各サンプリング間隔に対応する報告の最後に、メモリー使用効率と上限実行しきい値を示す行を出力します。	138 ページの「メモリー使用効率とメモリー上限実行しきい値の報告」

rcapadm によるリソース上限デーモンの管理

このセクションでは、rcapadm を使用してリソース上限デーモンを構成する手順について説明します。詳細は、123 ページの「rcapd の構成」および rcapadm(1M) のマニュアルページを参照してください。rcapadm を使用してゾーンに一時的なリソース上限を指定する方法についても説明します。

引数なしで使用した場合、rcapadm はリソース上限デーモンの現在のステータスを表示します (構成されている場合のみ)。

▼ メモリー上限実行しきい値を設定する方法

上限は、プロセスが利用できる物理メモリーが少なくなるまで制限されないように構成できます。詳細は、124 ページの「メモリー上限実行しきい値」を参照してください。

最小値 (デフォルト) は 0 です。これは、メモリー上限が常に制限されることを意味します。最小値を変更するには、次の手順に従います。

1. **root** になるか、同等の役割になります。
2. **rcapadm** の **-c** オプションを使用することで、メモリー上限を制限するときの物理メモリー使用効率を設定します。

   # rcapadm -c *percent*

   percent は 0 から 100 までの値です。この値を大きくするほど、規制が小さくなります。つまり、上限が定義されたプロジェクトの作業負荷は、システムのメモリー使用効率がこのしきい値を超えない限り、上限を適用されることなく実行できます。

参照 現在の物理メモリーの使用効率と上限実行しきい値を表示する方法については、138 ページの「メモリー使用効率とメモリー上限実行しきい値の報告」を参照してください。

▼ 動作間隔を設定する方法

126 ページの「rcapd の動作間隔」では、rcapd が行う定期的な動作の間隔について説明しています。rcapadm を使用して動作間隔を設定するには、次の手順に従います。

1. **root** になるか、同等の役割になります。
2. **-i** オプションを使用して、動作間隔の値を設定します。

   # rcapadm -i *interval=value,...,interval=value*

注 – すべての動作間隔の値の単位は秒です。

▼ リソース上限制御を有効にする方法

リソース上限制御をシステムで有効にする方法は 3 つあります。リソース上限制御を有効にすると、さらに /etc/rcap.conf ファイルがデフォルト値で設定されます。

1 **root** になるか、同等の役割になります。

2 次のどちらかの方法でリソース上限デーモンを有効にします。

- svcadm コマンドを使って、リソース上限制御を有効にします。

  ```
  # svcadm enable rcap
  ```

- リソース上限デーモンを次のようにして有効にし、ただちに起動し、かつ、システムをブートするたびに起動するようにします。

  ```
  # rcapadm -E
  ```

- リソース上限デーモンをただちには起動せず、ブート時に有効にするには、-n オプションも指定します。

  ```
  # rcapadm -n -E
  ```

▼ リソース上限制御を無効にする方法

リソース上限制御をシステムで無効にする方法は 3 つあります。

1 **root** になるか、同等の役割になります。

2 次のどちらかの方法でリソース上限デーモンを無効にします。

- **svcadm** コマンドを使用して、リソース上限制御をオフにします。

  ```
  # svcadm disable rcap
  ```

- リソース上限デーモンを無効にして、ただちに停止し、かつ、システムをブートしても起動しないようにするには、次のように入力します。

  ```
  # rcapadm -D
  ```

- リソース上限デーモンを停止せずに無効にするには、**-n** オプションも指定します。

  ```
  # rcapadm -n -D
  ```

ヒント – リソース上限デーモンの安全な無効化

rcapadm -D を使用して、rcapd を安全に無効にします。リソース上限デーモンを強制終了すると (kill(1) のマニュアルページを参照)、プロセスが停止状態のままになり、手動で再起動しなければならない場合があります。プロセスの実行を再開するには、prun コマンドを使用します。詳細は、prun(1) のマニュアルページを参照してください。

▼ ゾーンに一時的なリソース上限を指定する方法

この手順は、特定のゾーンで消費可能な最大のメモリー量を割り当てる場合に使用します。この値は、次のリブートまでに限り有効です。持続的な上限を設定するには、zonecfg コマンドを使用します。

1. **root** になるか、同等の役割になります。
2. ゾーン **my-zone** に最大メモリーの値として **512M** バイトを設定します。

```
# rcapadm -z testzone -m 512M
```

rcapstat による報告の生成

リソース上限制御の統計を報告するには、rcapstat を使用します。127 ページの「rcapstat によるリソース使用効率の監視」 コマンドを使って報告を生成する方法については、Monitoring Resource Utilization With rcapstatを参照してください。このセクションでは、報告の列見出しについても説明します。rcapstat(1) のマニュアルページにも、この情報が含まれています。

このセクションでは、さまざまな目的の報告を生成する方法を、例を使用しながら説明します。

上限とプロジェクトの情報の報告

この例では、2 人のユーザーに関連付けられた 2 つのプロジェクトに、上限が定義されています。user1 の上限は 50M バイト、user2 の上限は 10M バイトです。

次のコマンドは、5 つの報告を 5 秒間のサンプリング間隔で生成します。

```
user1machine% rcapstat 5 5
    id project  nproc     vm    rss   cap    at avgat    pg avgpg
112270   user1     24   123M    35M   50M   50M    0K 3312K    0K
 78194   user2      1  2368K  1856K   10M    0K    0K    0K    0K
```

```
    id project  nproc     vm    rss   cap    at avgat    pg avgpg
112270   user1     24   123M    35M   50M    0K    0K    0K    0K
 78194   user2      1  2368K  1856K   10M    0K    0K    0K    0K
    id project  nproc     vm    rss   cap    at avgat    pg avgpg
112270   user1     24   123M    35M   50M    0K    0K    0K    0K
 78194   user2      1  2368K  1928K   10M    0K    0K    0K    0K
    id project  nproc     vm    rss   cap    at avgat    pg avgpg
112270   user1     24   123M    35M   50M    0K    0K    0K    0K
 78194   user2      1  2368K  1928K   10M    0K    0K    0K    0K
    id project  nproc     vm    rss   cap    at avgat    pg avgpg
112270   user1     24   123M    35M   50M    0K    0K    0K    0K
 78194   user2      1  2368K  1928K   10M    0K    0K    0K    0K
```

出力の最初の 3 行は 1 回目の報告です。ここには、2 つのプロジェクトに関する上限とプロジェクトの情報、および rcapd 起動以降のページング統計が記載されています。at と pg の列において、user1 にはゼロより大きな値が入っており、user2 にはゼロが入っています。これは、1 回目の報告の期間中、user1 は上限を超えたが、user2 は超えなかったことを意味します。

2 回目以降の報告では、目立った活動はありません。

プロジェクトの RSS の監視

この例では、プロジェクト user1 を使用しています。このプロジェクトには RSS 上限を超えた RSS が 1 つあります。

次のコマンドは、5 つの報告を 5 秒間のサンプリング間隔で生成します。

```
user1machine% rcapstat 5 5

    id project  nproc    vm   rss   cap    at avgat     pg  avgpg
376565   user1      3 6249M 6144M 6144M  690M  220M  5528K  2764K
376565   user1      3 6249M 6144M 6144M    0M  131M  4912K  1637K
376565   user1      3 6249M 6171M 6144M   27M  147M  6048K  2016K
376565   user1      3 6249M 6146M 6144M 4872M  174M  4368K  1456K
376565   user1      3 6249M 6156M 6144M   12M  161M  3376K  1125K
```

プロジェクト user1 は 3 つのプロセスを持っており、それぞれがアクティブに物理メモリーを使用しています。pg 列の正の値が示しているとおり、rcapd は、このプロジェクトのプロセスの物理メモリーの使用率を上限に合わせて下げようと、メモリーをページアウトし続けています。しかし、rcapd は RSS を上限値未満に保つことに成功していません。これは、rss 値が相応の減少を示していないことでわかります。作業負荷はメモリーをページアウトするとすぐにまたメモリーを使用しており、その結果、RSS の値がまた上がってしまいます。これは、プロジェクトの常駐メモリーがすべてアクティブに使用されており、かつ、作業セットサイズ (*WSS*) が上限よりも大きいことを意味します。そのため、rcapd は、上限に合わせて作業セットの一部をページアウトせざるを得ません。この状況では、次のうちのいずれかが起きるまで、システムのページフォルト率および関連する入出力は高いままです。

- WSS が小さくなる。

- 上限を上げる。
- アプリケーションのメモリーアクセスパターンが変わる。

この状況では、サンプリング間隔を短くすることで、RSS 値と上限値の差を小さくできる可能性があります。rcapd が作業負荷をサンプリングして上限を制限する頻度が上がるからです。

注 – ページフォルトが発生するのは、新しいページを作成する必要があるとき、あるいは、システムがスワップデバイスからページをコピーする必要があるときです。

プロジェクトの作業セットサイズの決定

この例では、前の例と同じプロジェクトを使用します。

前の例では、プロジェクト user1 が自分に許可された上限よりも多くの物理メモリーを使用しているところを示しました。この例では、どれくらいのメモリーをプロジェクトの作業負荷が必要とするのかを示します。

```
user1machine% rcapstat 5 5
    id project  nproc    vm   rss   cap    at avgat     pg  avgpg
376565   user1      3 6249M 6144M 6144M  690M    0K   689M     0K
376565   user1      3 6249M 6144M 6144M    0K    0K     0K     0K
376565   user1      3 6249M 6171M 6144M   27M    0K    27M     0K
376565   user1      3 6249M 6146M 6144M 4872K    0K  4816K     0K
376565   user1      3 6249M 6156M 6144M   12M    0K    12M     0K
376565   user1      3 6249M 6150M 6144M 5848K    0K  5816K     0K
376565   user1      3 6249M 6155M 6144M   11M    0K    11M     0K
376565   user1      3 6249M 6150M   10G   32K    0K    32K     0K
376565   user1      3 6249M 6214M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
```

サイクルの中ほどで、プロジェクト user1 の上限を 6G バイトから 10G バイトに上げました。値を上げることによって上限の制限が止まり、常駐セットサイズが上昇しました。常駐セットサイズを規制するのは、ほかのプロセスと、マシンのメモリー容量だけになりました。rss 列が、プロジェクトの作業セットサイズ (この例では 6247M バイト) を反映して安定する場合があります。この値が、このプロジェクトのプロセスがページフォルトを頻繁に起こさずに動作できる、上限の最小値です。

user1 の上限が 6G バイトであるとき、サンプリング間隔である 5 秒ごとに、rcapd が作業負荷のメモリーの一部をページアウトするにつれて、RSS は減少して入出力は増加します。ページアウトの直後、これらのページを必要とする作業負荷は、(動作し続ける限り) メモリーをページインします。このサイクルは、例の中ほどで上限を10G バイトに上げるまで繰り返されます。その後、RSS は 6.1G バイトで安定しま

す。作業負荷の RSS が上限より低くなったため、これ以後ページングは発生しません。また、ページングに関連する入出力は止まります。このため、観察時にプロジェクトが行なっていた作業の実行には、6.1G バイトが必要であったことがわかります。

vmstat(1M) および iostat(1M) のマニュアルページも参照してください。

メモリー使用効率とメモリー上限実行しきい値の報告

rcapstat の -g オプションを使用すると、次のことを報告できます。

- 現在の物理メモリーの使用効率 (システムにインストールされている物理メモリーに占めるパーセンテージ)
- rcapadm で設定されたシステムメモリー上限実行しきい値

-g オプションを使用すると、各サンプリング間隔に対応する報告の最後に、メモリー使用効率と上限実行しきい値を示す行が出力されます。

```
# rcapstat -g
    id project   nproc    vm   rss   cap    at avgat   pg  avgpg
376565    rcap       0    0K    0K   10G    0K    0K   0K     0K
physical memory utilization: 55%   cap enforcement threshold: 0%
    id project   nproc    vm   rss   cap    at avgat   pg  avgpg
376565    rcap       0    0K    0K   10G    0K    0K   0K     0K
physical memory utilization: 55%   cap enforcement threshold: 0%
```

◆ ◆ ◆ 第 12 章

12 リソースプール (概要)

この章では、次の各技術について説明します。

- リソースプール。マシンリソースの区分に使用されます
- 動的リソースプール (DRP)。設定されているシステムの目標に合うように、各リソースプールのリソース割り当てを動的に調整します

リソースプールと動的リソースプールは、Oracle Solaris サービス管理機能 (SMF) 内のサービスです。これらの各サービスは、別個に有効にできます。

この章の内容は次のとおりです。

- 140 ページの「リソースプールの紹介」
- 141 ページの「動的リソースプールの紹介」
- 141 ページの「リソースプールと動的リソースプールの有効化/無効化について」
- 141 ページの「ゾーンで使用されるリソースプール」
- 142 ページの「リソースプールを使用する場合」
- 143 ページの「リソースプールのフレームワーク」
- 145 ページの「システム上でのプールの実装」
- 146 ページの「`project.pool` 属性」
- 146 ページの「SPARC: 動的再構成の処理とリソースプール」
- 146 ページの「プール構成の作成」
- 147 ページの「動的構成の直接操作」
- 148 ページの「`poold` の概要」
- 148 ページの「動的リソースプールの管理」
- 149 ページの「構成の制約と目標」
- 153 ページの「構成可能な `poold` の機能」
- 156 ページの「動的リソース割り当てのしくみ」
- 159 ページの「`poolstat` によるプール機能とリソース使用効率の監視」
- 161 ページの「リソースプール機能で使用するコマンド」

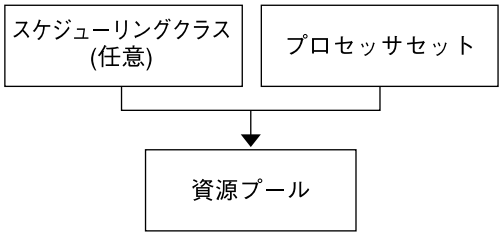
この機能の使用手順については、第 13 章「リソースプールの作成と管理 (タスク)」を参照してください。

リソースプールの紹介

「リソースプール」を使用すると、作業負荷によって特定のリソースが重複して消費されないように、作業負荷を分離することができます。このような方法でリソースを確保すると、さまざまな作業負荷が混在するシステム上で予測どおりの性能を得ることができます。

リソースプールは、プロセッサセット (pset) の構成やスケジューリングクラスの割り当て (任意) に対して一貫した構成メカニズムを提供します。

図 12–1 リソースプールのフレームワーク



プールは、システムで使用可能なさまざまなリソースセットを結合した特定のものと考えることができます。リソースのさまざまな組み合わせを表す各種のプールを作成できます。

```
pool1: pset_default
pool2: pset1
pool3: pset1, pool.scheduler="FSS"
```

リソースプールは、複数のパーティションをグループ化することにより、ラベル付けされている作業負荷とハンドルを対応付けることができます。`/etc/project` ファイル内の各プロジェクトエントリには単一のプールを関連付けることができます。それらのプールを指定するには、`project.pool` 属性を使用します。

プールが有効になっている場合、基本構成は「デフォルトプール」と「デフォルトプロセッサセット」から成ります。ユーザー定義のプールやプロセッサセットを作成し、構成に追加することもできます。CPU は 1 つのプロセッサセットだけに所属できます。ユーザー定義のプールやプロセッサセットは破棄できます。デフォルトプールとデフォルトプロセッサセットは破棄できません。

デフォルトプールの `pool.default` プロパティーは `true` に設定されます。デフォルトプロセッサセットの `pset.default` プロパティーは `true` に設定されます。したがって、デフォルトプールとデフォルトプロセッサセットはどちらも、名前が変更された場合でも識別できます。

ユーザー定義プールメカニズムは主に、5 つ以上の CPU を搭載する大規模なマシンで使用されます。ただし、小規模なマシンでもこの機能を活用することができます。小規模なマシンでは、重要でないリソースパーティションを共有するプールを作成できます。重要なリソースにだけ、専用のプールが使用されます。

動的リソースプールの紹介

動的リソースプールは、システムイベントやアプリケーション負荷の変化に対応して各プールのリソース割り当てを動的に調整するメカニズムを提供します。動的リソースプールによって管理者が決定を行いやすくなり、決定を行う回数も減少します。管理者がシステム性能の目標を指定すると、それを維持するように自動的に調整が行われます。構成に変更を加えると、変更内容がログに記録されます。これらの機能は、主にリソースコントローラ `poold` によって実行されます。動的リソース割り当てが必要なときは、常にこのシステムデーモンをアクティブにしておく必要があります。`poold` は定期的にシステムの負荷を調べ、システムの性能を最適に保つために、リソース消費に関する調整が必要かどうかを判定します。`poold` の構成は `libpool` の構成に保存されています。`poold` の詳細については、`poold(1M)` のマニュアルページを参照してください。

リソースプールと動的リソースプールの有効化/無効化について

リソースプールおよび動的リソースプールを有効化/無効化する方法については、165 ページの「プール機能の有効化と無効化」を参照してください。

ゾーンで使用されるリソースプール

システム上で構成済みのリソースプールにゾーンを関連付ける代わりに、`zonecfg` コマンドを使用して、ゾーンの稼働中に有効になる一時プールを作成することもできます。詳細は、219 ページの「`dedicated-cpu` リソース」を参照してください。

ゾーンが有効になっているシステムの場合、1 つの非大域ゾーンにはリソースプールを 1 つだけ関連付けることができますが、特定のゾーンに割り当てたプールをそのゾーン専用にする必要はありません。また、大域ゾーンから `poolbind` コマンドを使用して、非大域ゾーンの個々のプロセスを別のプールに結合することもできません。非大域ゾーンをプールに関連付ける方法については、269 ページの「ゾーンを構成、検証、および確定する」を参照してください。

プールに対してスケジューリングクラスを設定した場合は、そのプールに非大域ゾーンを関連付けると、そのゾーンではそのスケジューリングクラスがデフォルトで使用されます。

動的リソースプールを使用している場合、実行中の poold インスタンスの有効範囲は大域ゾーンに制限されます。

poolstat ユーティリティーを非大域ゾーンで実行すると、そのゾーンに関連付けられているプールの情報だけが表示されます。非大域ゾーンで引数なしで pooladm コマンドを実行すると、そのゾーンに関連付けられているプールの情報だけが表示されます。

リソースプールのコマンドについては、161 ページの「リソースプール機能で使用するコマンド」を参照してください。

リソースプールを使用する場合

リソースプールは、多くの管理作業に適用できる汎用メカニズムを提供します。

バッチ処理サーバー
: プールの機能を使用して、1 つのサーバーを 2 つのプールに分割します。一方のプールは、ログインセッションとタイムシェアリングユーザーによる対話型作業に使用されます。もう一方のプールは、バッチシステムを介して投入されるジョブに使用されます。

アプリケーションサーバーまたはデータベースサーバー
: アプリケーションの要件に基づいて、対話型アプリケーション用のリソースを区分します。

アプリケーションの段階的な調整
: ユーザーが期待するサービスレベルを設定します。

  最初は、目標とする最終的なサービスの一部だけを実行するマシンを導入することがあります。マシンをオンラインにしたときに、予約方式のリソース管理メカニズムが確立していなければ、ユーザーがサービスに不満を持つ可能性があります。

  たとえば、公平配分スケジューラは CPU の使用効率を最適化します。1 つしかアプリケーションを実行していないマシンの応答時間は速く感じられます。実際には、複数のアプリケーションがロードされると、このような応答時間は得られません。アプリケーションごとに個別のプールを用意することにより、各アプリケーションで使用可能な CPU 数の上限をあらかじめ設定してから、すべてのアプリケーションを運用することができます。

複雑なタイムシェアリングサーバー
: 多数のユーザーをサポートするサーバーを区分します。サーバーの区分によって、ユーザーごとの応答が時間をより確実に予測できる分離メカニズムが提供されます。

ユーザーをグループに分割して個別のプールに結合し、公平配分スケジューラ (FSS) 機能を使用すれば、CPU 割り当てを調整して、優先順位を持つユーザーグループをサポートできます。このような割り当ては、ユーザーの役割や課金などに基づいて行えます。

定期的に変動する作業負荷
: リソースプールを使用して、変動する作業負荷に対応します。

  サイトでの作業負荷の変動が月次、四半期、年次などの周期で予想できる場合があります。サイトでこのような変動が予想できる場合は、`cron` ジョブで `pooladm` を実行して、複数のプール構成を使い分けることができます。(143 ページの「リソースプールのフレームワーク」を参照)。

リアルタイムアプリケーション
: RT スケジューラと専用のプロセッサリソースを使用して、リアルタイムプールを作成します。

システムの使用効率
: システムの目標を設定して適用します。

  自動プールデーモンの機能を使用して、使用可能なリソースを特定してから作業負荷を監視すると、指定した目標がいつ満たされなくなるかを検出できます。可能な場合はデーモンで修正操作を実行したり、状況をログに記録したりできます。

リソースプールのフレームワーク

`/etc/pooladm.conf` 構成ファイルには、静的なプール構成が記述されます。静的構成では、リソースプール機能に関連して管理者がシステムをどのように構成するかを記述できます。別のファイル名を指定することもできます。

サービス管理機能 (SMF) または `pooladm -e` コマンドを使ってリソースプールフレームワークを有効にする場合で、`/etc/pooladm.conf` ファイルが存在するときは、このファイル内の構成がシステムに適用されます。

リソースプールフレームワーク内でのリソースの処置に関する情報は、カーネルで保持されます。これは動的構成と呼ばれ、特定のシステムの、ある時点でのリソースプール機能を表します。動的構成を表示するには、`pooladm` コマンドを使用します。プールやリソースセットについて表示されるプロパティーの順序は、場合によって異なります。動的構成に対する変更は、次の方法で行われます。

- 静的構成ファイルを適用する、間接的な方法
- `poolcfg` コマンドに `-d` オプションを使用する、直接的な方法

場合に応じて起動できるように、複数の静的プール構成ファイルを作成しておくことができます。cron ジョブで pooladm を起動して、複数のプール構成を使い分けることができます。cron ユーティリティーの詳細は、cron(1M) のマニュアルページを参照してください。

デフォルトでは、リソースプールフレームワークは無効になっています。動的構成を作成したり変更したりするには、リソースプールが有効になっている必要があります。リソースプールフレームワークが無効になっている場合でも、poolcfg コマンドまたは libpool コマンドを使って静的構成ファイルを操作することはできます。プール機能が無効になっている場合、静的構成ファイルを作成することはできません。構成ファイルの詳細については、146 ページの「プール構成の作成」を参照してください。

リソースプールおよび poold システムデーモンで使用するコマンドについては、次のマニュアルページに記載されています。

- pooladm(1M)
- poolbind(1M)
- poolcfg(1M)
- poold(1M)
- poolstat(1M)
- libpool(3LIB)

/etc/pooladm.conf の内容

次の要素は、動的構成も含め、すべてのリソースプール構成に使用できます。

system　システムの全体的な動作に影響を与えるプロパティー

プール　リソースプールの定義

pset　プロセッサセットの定義

cpu　プロセッサの定義

これらの要素に含まれているプロパティーを操作することで、リソースプールフレームワークの状態と動作を変更できます。たとえば、プールプロパティー pool.importance は、プールの相対的な重要性を示します。このプロパティーは、リソースの競合が発生した場合の解決に使用されます。詳細は、libpool(3LIB) を参照してください。

プールのプロパティー

プール機能では、プール、リソース、またはコンポーネントに設定される、名前と型の指定されたプロパティーがサポートされています。管理者は、プールのさまざ

まな要素に対して、追加のプロパティーを設定することもできます。プロジェクト属性に似たプロパティー名前空間が使用されます。

たとえば、次のコメントは、特定の Datatree データベースに pset が関連付けられていることを示します。

```
Datatree,pset.dbname=warehouse
```

プロパティーの型の詳細については、152 ページの「poold のプロパティー」を参照してください。

注 – いくつかの特殊プロパティーが内部使用のために予約されています。これらを設定したり削除したりすることはできません。詳細は、libpool(3LIB) のマニュアルページを参照してください。

システム上でのプールの実装

ユーザー定義のプールをシステム上に実装するには、次のどちらかの方法を使用します。

- Oracle Solaris ソフトウェアがブートすると、init スクリプトは /etc/pooladm.conf ファイルが存在するかどうかを検査します。このファイルが検出され、プールが有効化されると、pooladm が呼び出され、この構成をアクティブなプール構成にします。システムは、/etc/pooladm.conf で指定されている編成に従って、動的な構成を作成し、マシンのリソースは指定どおりに区分されます。
- Oracle Solaris システムが起動しているときは、pooladm コマンドを使用して、プール構成が存在しない場合はプール構成を起動したり、プール構成を変更したりできます。デフォルトでは、pooladm コマンドは /etc/pooladm.conf の内容を使用します。ただし、別のディレクトリとファイル名を指定し、そのファイルを使用してプール構成を変更することもできます。

リソースプールを有効化または無効化する方法については、165 ページの「プール機能の有効化と無効化」を参照してください。ユーザー定義のプールやリソースが使用されている間は、プール機能を無効にすることはできません。

リソースプールを構成するには、root 権限を持っているか、または必要な権利プロファイルを持っている必要があります。

poold リソースコントローラは、動的リソースプール機能とともに起動されます。

project.pool 属性

/etc/project ファイル内のプロジェクトエントリに project.pool 属性を追加すると、そのエントリに単一のプールを関連付けることができます。プロジェクトで開始される新しい作業は、適切なプールに結合されます。詳細は、第 2 章「プロジェクトとタスク (概要)」を参照してください。

たとえば、projmod コマンドを使用して、/etc/project ファイル内のプロジェクト *sales* に project.pool 属性を設定できます。

```
# projmod -a -K project.pool=mypool sales
```

SPARC: 動的再構成の処理とリソースプール

動的再構成 (DR) を使用すると、システムの実行中にハードウェアを再構成できます。DR 操作により、特定の種類のリソースが増加または減少することもあれば、影響を受けないこともあります。DR は使用可能なリソース量に影響を与えることがあるので、プール機能を DR 操作に含めておく必要があります。DR 処理が開始されると、プールのフレームワークは構成の妥当性を検証します。

現在のプール構成が無効にならないかぎり、DR 処理は、独自の構成ファイルが更新されるまで実行を続けます。無効な構成とは、使用可能なリソースでサポートできない構成のことです。

DR 処理によってプール構成が無効になった場合、操作は失敗し、メッセージログにメッセージが書き込まれます。構成処理を強制的に最後まで実行するには、DR の強制オプションを使用します。強制オプションで処理を続行すると、プール構成は、新しいリソース構成に合うように変更されます。DR 処理と強制オプションについては、使用している Sun ハードウェアの動的再構成に関するユーザーガイドを参照してください。

動的リソースプールを使用している場合、poold デーモンがアクティブになっている間に、その制御からパーティションが除外されることがあります。詳細は、158 ページの「リソース不足の特定」を参照してください。

プール構成の作成

構成ファイルには、システム上で作成されるプールに関する記述が含まれます。構成ファイルには、操作可能な構成要素が記述されています。

- システム
- pool

- pset
- cpu

操作可能な構成要素については、poolcfg(1M) を参照してください。

プールが有効になっている場合、構造化された /etc/pooladm.conf ファイルを次の 2 つの方法で作成できます。

- pooladm コマンドに -s オプションを付けて実行して、現在のシステム上のリソースを検出し、その結果を構成ファイルに記録します。

  この方法をお勧めします。プール機能で操作できるシステム上のアクティブなリソースとコンポーネントがすべて記録されます。リソースには、既存のプロセッサセットの構成が含まれます。最後に、プロセッサセットの名前を変更したり、必要に応じてプールを作成したりして、構成を変更できます。
- poolcfg コマンドに -c オプションと discover サブコマンドまたは create system *name* サブコマンドを付けて実行して、新しいプール構成を作成します。

  これらのオプションは、以前の各リリースとの下位互換性を保つために残されています。

/etc/pooladm.conf ファイルを変更するには、poolcfg または libpool を使用します。このファイルを直接編集しないでください。

動的構成の直接操作

poolcfg コマンドに -d オプションを付けて実行すると、動的構成内の CPU リソースタイプを直接操作できます。リソースを転送するには、次の 2 つの方法があります。

- 識別された利用可能なリソースすべてをセット間で転送するように要求します。
- 特定の ID を持つリソースだけをターゲットセットに転送します。リソース構成が変更されたときやシステムのリブート後は、リソースに関連付けられているシステム ID が変わることがあります。

具体例は、178 ページの「リソースの転送」を参照してください。

DRP が使用中の場合は、リソース転送が poold からアクションをトリガーする場合があることに注意してください。詳細は、148 ページの「poold の概要」を参照してください。

poold の概要

プールのリソースコントローラ poold は、システムターゲットと観察可能な統計情報を使用して、管理者によって指定されたシステム性能の目標を維持します。動的リソース割り当てが必要なときは、常にこのシステムデーモンをアクティブにしておく必要があります。

poold リソースコントローラは、使用可能なリソースを特定してから作業負荷を監視して、システム使用率に関する目標がいつ満たされなくなるかを検出できます。poold は、これらの目標の観点から別の構成を検討し、改善操作を実行します。可能な場合は、目標を満たすようにリソースを再構成します。この操作が不可能な場合は、ユーザーによって指定された目標が達成不可能になったことをログに記録します。再構成を行なった後、デーモンは作業負荷目標の監視を再開します。

poold では決定の履歴が保持され、確認に使用されます。決定履歴を使用すると、それまでに行なった再構成のうち、改善効果を示さなかったものを削除できます。

作業負荷の目標が変更された場合や、システムで使用可能なリソースが変更された場合は、再構成が非同期に実行されることもあることに注意してください。

動的リソースプールの管理

DRP サービスは、サービス識別子 svc:/system/pools/dynamic 下のサービス管理機能 (SMF) によって管理されます。

有効化、無効化、再起動の要求など、このサービスに関する管理作業は、svcadm コマンドを使用して実行できます。サービスのステータスは、svcs コマンドを使用して照会できます。詳細については、svcs(1) および svcadm(1M) のマニュアルページを参照してください。

DRP の制御方法として望ましいのは SMF インタフェースですが、下位互換性を維持するために、次の方法も使用できます。

- 動的リソース割り当てが不要になった場合は、SIGQUIT シグナルまたは SIGTERM シグナルを使って poold を停止できます。これらのシグナルはどちらも、poold を正常に終了させます。
- poold では、リソースやプールの構成の変更が自動的に検出されます。ただし、SIGHUP シグナルを使用して、再構成を強制的に実行することもできます。

構成の制約と目標

poold は、管理者の指示に基づいて再構成を行います。これらの指示は、一連の制約および目標として指定します。poold はこれらの指定に基づき、可能性のあるさまざまな構成を、既存の構成に対する相対値として決定します。次に poold は、現在の構成のリソース割り当てを変更して、候補となる新しい構成を生成します。

構成の制約

制約は、構成に加えることのできる変更を一部除外することで、作成可能な構成の範囲に影響を与えます。libpool 構成で次の制約を指定できます。

- CPU 割り当ての最小値と最大値
- セットから移動できない固定コンポーネント
- プールの重要要素

プールプロパティーの詳細については、libpool(3LIB) のマニュアルページと144 ページの「プールのプロパティー」を参照してください。

使用方法については、174 ページの「構成の制約を設定する方法」を参照してください。

pset.min プロパティーと pset.max プロパティーの制約

これら 2 つのプロパティーは、プロセッサセットに割り当てることのできるプロセッサの最小数と最大数を制限します。これらのプロパティーの詳細については、表 12–1 を参照してください。

同じ Oracle Solaris インスタンスのリソースパーティション内では、これらの制約の範囲内で、あるパーティションから別のパーティションにリソースを割り当てることができます。リソースセットに関連付けられているプールに結合することで、リソースにアクセスできるようになります。この結合はログイン時に実行されるか、または、PRIV_SYS_RES_CONFIG 特権を持っている管理者が手動で行います。

cpu.pinned プロパティーの制約

cpu-pinned プロパティーは、DRP で特定の CPU を現在のプロセッサセットから移動してはならないことを示します。この libpool プロパティーを設定すると、プロセッサセット内で実行されている特定のアプリケーションでのキャッシュ使用効率を最大限に高めることができます。

このプロパティーの詳細については、表 12–1 を参照してください。

pool.importance プロパティーの制約

pool.importance プロパティーは、管理者が定義した、プールの相対的な重要度を示します。

構成の目標

目標は制約と同様の方法で指定されます。目標の全一覧については、表 12–1 を参照してください。

目標には 2 つのカテゴリがあります。

作業負荷に依存する目標　作業負荷に依存する目標とは、システムで実行される作業負荷の性質によって変化する目標です。たとえば、utilization 目標などがあります。リソースセットの使用効率の数値は、そのセットでアクティブになっている作業負荷の性質によって変化します。

作業負荷に依存しない目標　作業負荷に依存しない目標とは、システムで実行される作業負荷の性質によって変化しない目標です。たとえば、CPU の locality 目標などがあります。リソースセットの近傍性の評価値は、そのセットでアクティブになっている作業負荷の性質によって変化することはありません。

次の 3 種類の目標を定義できます。

名前	有効な要素	演算子	値
wt-load	system	なし	なし
locality	pset	なし	loose | tight | none
utilization	pset	<> ~	0–100%

目標は、libpool 構成内のプロパティー文字列に格納されます。プロパティー名は、次のとおりです。

- system.poold.objectives
- pset.poold.objectives

目標の構文は次のとおりです。

- objectives = objective [; objective]*
- objective = [n:] keyword [op] [value]

どの目標にも、オプションで重要性を表す接頭辞を付けることができます。重要性の値は目標に乗算され、この目標が目標関数の評価に与える影響を高めます。指定できる範囲は、0 から INT64_MAX (9223372036854775807) までです。指定されていない場合、重要性のデフォルト値は 1 です。

一部の要素タイプでは、複数の種類の目標がサポートされています。pset などはその一例です。このような要素には、複数の種類の目標を指定できます。また、1 つの pset 要素に複数の使用効率目標を指定することもできます。

使用例については、175 ページの「構成の目標を定義する方法」を参照してください。

wt-load 目標

wt-load 目標では、リソースの使用効率に合わせてリソースを割り当てるような構成が有利に導かれます。この目標が有効になっていると、より多くのリソースを使用するリソースセットには、より多くのリソースが与えられることになります。wt-load は「重み付けされた負荷 (weighted load)」を意味します。

最小値と最大値のプロパティーを使って満足のいく制約を設定したあとで、これらの制約の範囲内でデーモンが自由にリソースを操作できるようにする場合に、この目標を使用してください。

locality 目標

locality 目標は、近傍性グループ (lgroup) データによって測定される近傍性が、選択された構成に対して与える影響を変化させます。近傍性は応答時間と定義することもできます。lgroup は、CPU リソースとメモリーリソースを表します。lgroup は、Oracle Solaris システムでリソースどうしの距離を調べるために使用されます。測定単位は時間です。近傍性グループによる抽象化については、『Programming Interfaces Guide』の「Locality Groups Overview」を参照してください。

この目標には、次の 3 つの値のいずれかを指定できます。

tight　この値を設定すると、リソースの近傍性を最大にするような構成が有利に導かれます。

loose　この値を設定すると、リソースの近傍性を最小にするような構成が有利に導かれます。

none　この値を設定すると、どのような構成が有利になるかは、リソースの近傍性に依存しません。これは locality 目標のデフォルト値です。

一般に、locality 目標は tight に設定することをお勧めします。ただし、メモリー帯域幅を最大にする場合や、リソースセットに対する DR 操作の影響を最小にする場合は、この目標を loose に設定するか、デフォルト値の none にしておいてください。

utilization 目標

utilization 目標では、指定された使用効率目標を満たしていないパーティションにリソースを割り当てるような構成が有利に導かれます。

この目標は、演算子と値で指定されます。演算子は次のとおりです。

< 「小なり」演算子は、指定された値が最大のターゲット値であることを示します。

> 「大なり」演算子は、指定された値が最小のターゲット値であることを示します。

~ 「ほぼ等しい」演算子は、指定された値がターゲット値であり、いくらかの変動が許容されることを示します。

pset には、演算子の種類ごとに使用効率目標を 1 つ設定できます。

- ~ 演算子を設定した場合、< 演算子や > 演算子は設定できません。
- < 演算子や > 演算子を設定した場合、~ 演算子は設定できません。< 演算子の設定と > 演算子の設定が互いに矛盾してはならないことに注意してください。

< 演算子と > 演算子の両方を使用すると、範囲を指定できます。これらの値は、重複しないように検証されます。

構成目標の例

次の例では、pset に対する次のような目標が poold によって評価されます。

- utilization を 30% から 80% の範囲に保つ必要がある。
- プロセッサセットの locality を最大にする必要がある。
- 各目標の重要性はデフォルト値の 1 とする。

例 12–1 poold の目標の例

```
pset.poold.objectives "utilization > 30; utilization < 80; locality tight"
```

その他の使用例については、175 ページの「構成の目標を定義する方法」を参照してください。

poold のプロパティー

プロパティーには 4 つのカテゴリがあります。

- 構成
- 制約
- 目標

- 目標パラメータ

表 12–1　定義済みのプロパティー名

プロパティー名	タイプ	カテゴリ	説明
system.poold.log-level	string	構成	ロギングレベル
system.poold.log-location	string	構成	ログの場所
system.poold.monitor-interval	uint64	構成	監視のサンプリング間隔
system.poold.history-file	string	構成	決定履歴の場所
pset.max	uint64	制約	このプロセッサセットに割り当てられる CPU の最大数
pset.min	uint64	制約	このプロセッサセットに割り当てられる CPU の最小数
cpu.pinned	bool	制約	CPU がこのプロセッサセットに固定されているかどうか
system.poold.objectives	string	目標	poold の目標式の構文に従った書式付き文字列
pset.poold.objectives	string	目標	poold の目標式の構文に従った書式付き文字列
pool.importance	int64	目標パラメータ	ユーザーが割り当てた重要性

構成可能な poold の機能

デーモンの動作の次のような部分は構成可能です。

- 監視の間隔
- ロギングレベル
- ログの場所

これらのオプションは、プール構成で指定します。コマンド行から poold を起動する方法でも、ログレベルを制御できます。

poold の監視間隔

プロパティー名 system.poold.monitor-interval を使用して、値をミリ秒単位で指定します。

poold のログ情報

ログを通じて 3 つのカテゴリの情報が提供されます。これらのカテゴリは、ログで次のように識別されています。

- 構成
- 監視
- 最適化

プロパティー名 system.poold.log-level を使用して、ログパラメータを指定します。このプロパティーが指定されていない場合、ログレベルのデフォルト値は NOTICE です。パラメータのレベルは階層的になっています。ログレベルとして DEBUG を設定すると、poold は、すべての定義済みメッセージをログに記録します。INFO レベルでは、ほとんどの管理者にとって適度な情報が得られます。

コマンド行で poold コマンドに -l オプションとパラメータを付けて実行すると、生成するログ情報のレベルを指定できます。

次のパラメータを使用できます。

- ALERT
- CRIT
- ERR
- WARNING
- NOTICE
- INFO
- DEBUG

これらのパラメータレベルは、syslog に使用される同様のレベルと直接対応づけられます。syslog の使用方法の詳細については、156 ページの「ログの場所」を参照してください。

poold のログ機能を構成する方法の詳細については、177 ページの「poold のログレベルを設定する方法」を参照してください。

情報ログ機能の構成

生成されるメッセージの種類は次のとおりです。

ALERT　　libpool 構成へのアクセスに関連する問題など、libpool 機能の基本的な予期せぬ障害。これによってデーモンは終了します。今すぐ管理者が対応する必要があります。

CRIT　予期せぬ障害に起因する問題。これによってデーモンは終了します。今すぐ管理者が対応する必要があります。

ERR　処理を制御するユーザー指定のパラメータに関連する問題。リソースセットの使用効率目標が衝突していて、解決不能な場合などです。管理者が介入して目標を修正する必要があります。poold は、衝突している目標を無視することで、改善操作を試みます。ただし、エラーによっては、デーモンが終了することもあります。

WARNING　構成パラメータの設定に関連する警告。構成パラメータの設定が技術的には正しくても、特定の実行環境には適さない場合などです。たとえば、すべての CPU リソースを固定にすると、poold はプロセッサセット間で CPU リソースを移動することができなくなります。

DEBUG　構成処理をデバッグするときに必要となる詳細情報が含まれたメッセージ。通常は、管理者がこの情報を使用することはありません。

情報ログ機能の監視

生成されるメッセージの種類は次のとおりです。

CRIT　予期せぬ監視障害に起因する問題。これによってデーモンは終了します。今すぐ管理者が対応する必要があります。

ERR　予期せぬ監視エラーに起因する問題。管理者が介入して修正する必要がある場合もあります。

NOTICE　リソース制御の領域移行に関するメッセージ。

INFO　リソース使用効率の統計情報に関するメッセージ。

DEBUG　監視処理をデバッグするときに必要となる詳細情報が含まれたメッセージ。通常は、管理者がこの情報を使用することはありません。

情報ログ機能の最適化

生成されるメッセージの種類は次のとおりです。

WARNING　これらのメッセージは、最適な決定を行う際の問題に関連して表示されることがあります。たとえば、最小値と最大値または固定コンポーネントの数によって、厳しすぎる制約をリソースセットに与えている場合などです。

これらのメッセージは、最適な再割り当てを行う際の、予期せぬ制限に起因する問題について表示されることがあります。たとえば、リソースを消費するプロセスがバインドされているプロセッサセットから最後のプロセッサを削除する場合などです。

NOTICE　これらのメッセージは、使用可能な構成や、決定履歴の上書きに起因して実装されない構成について表示されることがあります。

INFO　これらのメッセージは、考慮される代替構成について表示されることがあります。

DEBUG　最適化処理をデバッグするときに必要となる詳細情報が含まれたメッセージ。通常は、管理者がこの情報を使用することはありません。

ログの場所

system.poold.log-location プロパティーを使用して、poold のログの出力先を指定します。poold の出力先として SYSLOG の場所を指定できます (syslog(3C) のマニュアルページを参照)。

このプロパティーが指定されていない場合、poold ログのデフォルトの出力先は /var/log/pool/poold です。

コマンド行から poold を呼び出す場合、このプロパティーは使用しません。ログエントリは、呼び出しを行なった端末の stderr に出力されます。

logadm によるログ管理

poold がアクティブになっている場合、logadm.conf ファイル内に、デフォルトファイル /var/log/pool/poold を管理するためのエントリが含まれています。このエントリは次のとおりです。

```
/var/log/pool/poold -N -s 512k
```

logadm(1M) および logadm.conf(4) のマニュアルページを参照してください。

動的リソース割り当てのしくみ

このセクションでは、リソースを動的に割り当てるために poold で使用される手順と要因について説明します。

使用可能なリソースについて

poold プロセスの有効範囲内で使用できるすべてのリソースが、使用可能なリソースと見なされます。1 つの Oracle Solaris インスタンスが最大の制御範囲になります。

ゾーンが有効になっているシステムの場合、実行中の poold インスタンスの有効範囲は、大域ゾーンに制限されます。

使用可能なリソースの特定

リソースプールには、アプリケーションで消費できるすべてのシステムリソースが含まれます。

実行中の Oracle Solaris インスタンスごとに、1 つのパーティションには、CPU など 1 種類のリソースを割り当てる必要があります。1 種類のリソースを 1 つまたはそれ以上のパーティションに割り当ててもかまいません。各パーティションには、一意のリソースセットが含まれます。

たとえば、4 つの CPU と 2 つのプロセッサセットを持つマシンは、次のように設定されます。

pset 0: 0 1

pset 1: 2 3

ここで、コロンのあとの 0、1、2、3 という数字は、CPU の ID を表しています。これら 2 つのプロセッサセットで、4 つの CPU すべてが使用されていることに注目してください。

このマシンで次のような設定は不可能です。

pset 0: 0 1

pset 1: 1 2 3

CPU 1 を同時に割り当てることができる pset は 1 つだけなので、このような設定はできません。

リソースが属しているパーティション以外のパーティションからは、そのリソースにアクセスすることはできません。

使用可能なリソースを発見するために、poold はアクティブなプール構成を調べてパーティションを見つけます。制御対象となっているリソースの種類ごとに、すべてのパーティションに含まれているすべてのリソースが集計され、使用可能なリソースの合計量が求められます。

poold では、このリソース量を基にして処理が行われます。ただし、この基本量に制約が適用され、poold で割り当てを行う必要がある際の柔軟性が制限されることもあります。使用可能な制約については、149 ページの「構成の制約」を参照してください。

リソース不足の特定

poold の制御範囲とは、効率のよい区分と管理を主に poold が担当している、使用可能なリソースセットのことです。ただし、この制御範囲にあるリソースを操作できるメカニズムがほかにもあり、それが構成に影響を与えることもあります。poold がアクティブになっている間に特定のパーティションが制御範囲外になった場合、poold は使用可能なリソースを適切に操作することによってその制御を取り戻そうとします。poold デーモンは、有効範囲内に追加のリソースを見つけられない場合、リソース不足に関する情報をログに記録します。

リソース使用効率の判定

poold は通常、その制御範囲にあるリソースの使用率を監視することに時間の大部分を費やします。この監視は、作業負荷に依存する目標が満たされているかどうかを確認するために実行されます。

たとえば、プロセッサセットの場合、セット内のすべてのプロセッサについてすべての測定が実行されます。リソース使用効率は、サンプリング間隔に対して、リソースが使用された時間の割合を示します。リソース使用効率はパーセンテージで表され、0 - 100 の値です。

制御違反の特定

149 ページの「構成の制約と目標」で説明した指示は、システムが目標を満たさないことを検出するために使用されます。これらの目標は、作業負荷に直接関連しています。

ユーザーが構成した目標を満たしていないパーティションは、制御違反です。制御違反には、同期と非同期の 2 種類があります。

- 同期目標違反は、デーモンによって作業負荷の監視中に検出されます。
- 非同期目標違反は、デーモンの監視動作とは無関係に発生します。

非同期の目標違反は、次のようなイベントによって引き起こされます。

- 制御範囲に対してリソースが追加または削除された。
- 制御範囲が再構成された。
- poold リソースコントローラが再起動された。

作業負荷に関連しない目標が目標関数の評価に与える影響は、評価のたびに一定であると見なされます。作業負荷に関連しない目標が再評価されるのは、いずれかの非同期違反によって再評価処理が引き起こされたときだけです。

適切な改善操作の決定

リソースコントローラは、リソースを消費するプロセスでリソースが不足していると判定した場合、まずそのリソースを増やして性能を改善しようとします。

制御範囲について構成で指定された目標を満たすように、別の構成が検討され評価されます。

この処理では、リソースの移動結果を監視し、各リソースパーティションの応答性を評価しながら、徐々に細かい調整が行われます。決定履歴を参照して、それまでに行なった再構成のうちで改善効果を示さなかったものが削除されます。履歴データの関連度をより詳しく評価するために、プロセスの名前や数量といったほかの情報も使用されます。

修正操作を実行できない場合、デーモンは状況をログに記録します。詳細は、154 ページの「poold のログ情報」を参照してください。

poolstat によるプール機能とリソース使用効率の監視

システムのプールが有効になっている場合にリソースの使用効率を監視するには、poolstat ユーティリティーを使用します。このユーティリティーは、システム上でアクティブになっているすべてのプールを調べ、選択された出力モードに基づいて統計情報を報告します。poolstat の統計を使用すると、どのリソースパーティションの使用効率が高いかを判定できます。これらの統計情報を解析して、システムで多くのリソースが要求されたときにリソースの再割り当てを行う方法を決定できます。

poolstat ユーティリティーのオプションを使用すると、特定のプールを調べたり、リソースセット固有の統計情報を報告したりできます。

システムにゾーンが実装されている場合は、非大域ゾーンで poolstat を使用すると、そのゾーンのプールに関連付けられているリソースに関する情報が表示されます。

poolstat ユーティリティーの詳細については、poolstat(1M) のマニュアルページを参照してください。poolstat のタスクと使用方法については、182 ページの「poolstat を使ってプールに関連付けられているリソースについて統計情報を報告する」を参照してください。

poolstat の出力

デフォルトの出力形式では、poolstat は、見出し行を出力したあと、プールごとに 1 行ずつ表示されます。各プールの行は、プール ID とプール名で始まります。それに続く列は、プールに接続されているプロセッサセットに関する統計データです。複数のプールに接続されているリソースセットは、各プールで 1 回ずつ表示されるので、複数回表示されます。

列見出しは次のとおりです。

id　プール ID。

pool　プール名。

rid　リソースセット ID。

rset　リソースセット名。

type　リソースセットの種類。

min　リソースセットの最小サイズ。

max　リソースセットの最大サイズ。

size　リソースセットの現在のサイズ。

used　リソースセットのうちで現在使用されている部分のサイズ。

この値は、リソースセットの使用効率にリソースセットのサイズを乗算して計算されます。前回のサンプリング間隔でリソースセットが再構成されている場合、この値は報告されないことがあります。報告されなかった値はハイフン (-) で示されます。

load　リソースセットに加えられている負荷の絶対値。

このプロパティーの詳細については、libpool(3LIB) のマニュアルページを参照してください。

poolstat の出力では、次のことを指定できます。

- 列の順序
- 表示する見出し

poolstat の動作間隔の調整

poolstat で実行する操作をカスタマイズできます。報告用のサンプリング間隔、および統計を繰り返す回数を設定できます。

interval poolstat が実行する定期的な処理の間隔を調整します。すべての間隔は秒単位で指定します。

count 統計を繰り返す回数を指定します。デフォルトでは、poolstat は 1 回だけ統計情報を報告します。

interval と *count* を指定しなかった場合、統計は 1 回だけ報告されます。*interval* を指定し、*count* を指定しなかった場合、統計報告が無限に繰り返されます。

リソースプール機能で使用するコマンド

次の表に記載されているコマンドは、プール機能を管理するための主要なインタフェースとなります。ゾーンが有効になっているシステムでこれらのコマンドを使用する方法については、141 ページの「ゾーンで使用されるリソースプール」を参照してください。

マニュアルページ	説明
pooladm(1M)	システムのプール機能を有効または無効にします。特定の構成をアクティブにします。または、現在の構成を削除して、関連付けられているリソースをデフォルトのステータスに戻します。オプションを指定しないで実行すると、pooladm は、現在の動的プール構成を表示します。
poolbind(1M)	プロジェクト、タスク、およびプロセスを手動でリソースプールに結合できます。
poolcfg(1M)	プールやセットに対する構成操作を実行します。このツールを使って作成された構成は、pooladm によってターゲットホスト上でインスタンス化されます。 poolcfg に -c オプションと info サブコマンド引数を付けて実行すると、/etc/pooladm.conf に保存されている静的構成の情報が表示されます。このコマンドにファイル名の引数を追加すると、そのファイルに保存されている静的構成の情報が表示されます。たとえば、poolcfg -c info /tmp/newconfig では、/tmp/newconfig というファイルに保存されている静的構成の情報が表示されます。
poold(1M)	プールシステムデーモン。このデーモンは、システムターゲットと観察可能な統計情報を使用して、管理者によって指定されたシステム性能の目標を維持します。目標が満たされていないときに修正操作を実行できない場合、poold は状況をログに記録します。

マニュアルページ	説明
poolstat(1M)	プールに関連付けられているリソースについて統計情報を表示します。システム管理者にとって性能解析が簡単になり、リソースを区分または再区分するタスクに役立つ情報が得られます。特定のプールを調べたり、リソースセット固有の統計情報を報告したりするためのオプションも用意されています。

ライブラリの API は、libpool で提供されます (libpool(3LIB) のマニュアルページを参照)。プログラムからプール構成を操作するには、このライブラリを使用します。

◆ ◆ ◆ 第 13 章

13 リソースプールの作成と管理 (タスク)

この章では、システムのリソースプールを設定し管理する方法について説明します。

リソースプールの概要については、第 12 章「リソースプール (概要)」を参照してください。

リソースプールの管理 (タスクマップ)

タスク	説明	参照先
リソースプールを有効または無効にします。	システムのリソースプールをアクティブまたは無効にします。	165 ページの「プール機能の有効化と無効化」
動的リソースプールを有効または無効にします。	システムの動的リソースプール機能をアクティブまたは無効にします。	165 ページの「プール機能の有効化と無効化」
静的なリソースプール構成を作成します。	現在の動的構成と一致する静的構成ファイルを作成します。詳細は、143 ページの「リソースプールのフレームワーク」を参照してください。	169 ページの「静的構成を作成する方法」
リソースプール構成を変更します。	追加のプールを作成するなどして、システムのプール構成を変更します。	170 ページの「構成の変更方法」

タスク	説明	参照先
リソースプールをスケジューリングクラスに対応付けます。	プールをスケジューリングクラスに対応付けることで、プールに結合されているすべてのプロセスが、指定されたスケジューラを使用できるようにします。	172 ページの「プールをスケジューリングクラスに対応付ける方法」
構成の制約を設定し、構成の目標を定義します。	poold に対して、修正操作を実行するときに考慮する目標を指定します。構成の目標の詳細については、148 ページの「poold の概要」を参照してください。	174 ページの「構成の制約を設定する方法」および175 ページの「構成の目標を定義する方法」
ログのレベルを設定します。	poold で生成するログ情報のレベルを指定します。	177 ページの「poold のログレベルを設定する方法」
poolcfg コマンドでテキストファイルを使用します。	poolcfg コマンドにテキストファイルから入力します。	177 ページの「poolcfg でコマンドファイルを使用する方法」
カーネルでリソースを転送します。	カーネルでリソースを転送します。たとえば、特定の ID を持つリソースをターゲットセットに転送します。	178 ページの「リソースの転送」
プール構成を起動します。	デフォルト構成ファイル内の構成を起動します。	178 ページの「プール構成を起動する方法」
プール構成を確定する前に、プール構成を検証します。	検証が実行されるとどうなるかをテストするために、プール構成を検証します。	179 ページの「構成を確定する前に構成を検証する方法」
システムからプール構成を削除します。	プロセッサセットなど、対応付けられているすべてのリソースがデフォルトのステータスに戻ります。	179 ページの「プール構成を削除する方法」
プロセスをプールに結合します。	システムで実行中のプロセスをリソースプールに手動で対応付けます。	180 ページの「プロセスをプールに結合する方法」
タスクやプロジェクトをプールに結合します。	タスクやプロジェクトをリソースプールに対応付けます。	180 ページの「タスクまたはプロジェクトをプールに結合する方法」
新しいプロセスをリソースプールに結合します。	プロジェクト内の新しいプロセスを特定のプールに自動的に結合するには、project データベース内の各エントリに属性を追加します。	181 ページの「プロジェクトの project.pool 属性を設定する方法」

タスク	説明	参照先
project 属性を使ってプロセスを別のプールに結合します。	新たに開始されるプロセスについて、プールとの結合を変更します。	181 ページの「project 属性を使ってプロセスを別のプールに結合する方法」
poolstat ユーティリティーを使って報告を生成します。	指定した間隔で複数の報告を生成します。	182 ページの「特定の間隔で複数の報告を生成する」
リソースセットの統計情報を報告します。	poolstat ユーティリティーを使って pset リソースセットの統計情報を報告します。	182 ページの「リソースセットの統計情報を報告する」

プール機能の有効化と無効化

svcadm コマンド (svcadm(1M) のマニュアルページに記載) を使用して、システム上のリソースプールサービスおよび動的リソースプールサービスを有効または無効に設定できます。

pooladmコマンド (pooladm(1M) のマニュアルページに記載) を使用すると、次のタスクも実行できます。

- プール機能を有効にして、プールを操作できるようにします
- プール機能を無効にして、プールを操作できないようにします

注– システムをアップグレードする際に、リソースプールフレームワークが有効で、/etc/pooladm.conf ファイルが存在する場合、プールサービスが有効になり、このファイル内の構成がシステムに適用されます。

▼ svcadm を使用してリソースプールサービスを有効にする方法

1 **root** になるか、同等の役割になります。

2 リソースプールサービスを有効にします。

```
# svcadm enable system/pools:default
```

▼ svcadm を使用してリソースプールサービスを無効にする方法

1. root になるか、同等の役割になります。
2. リソースプールサービスを無効にします。

```
# svcadm disable system/pools:default
```

▼ svcadm を使用して動的リソースプールサービスを有効にする方法

1. root になるか、同等の役割になります。
2. 動的リソースプールサービスを有効にします。

```
# svcadm enable system/pools/dynamic:default
```

例 13–1 リソースプールサービスに対する動的リソースプールサービスの依存

この例では、DRP を実行する場合に、最初にリソースプールを有効にする必要があることを示します。

リソースプールと動的リソースプールの間には、依存関係があります。DRP は、リソースプールに依存するサービスになっています。DRP の有効化/無効化は、リソースプールとは関係なく実行できます。

次の例では、リソースプールと動的リソースプールの両方が現在無効に設定されています。

```
# svcs "*pool*"
STATE          STIME    FMRI
disabled       2011     svc:/system/pools:default
disabled       2011     svc:/system/pools/dynamic:default
```

動的リソースプールを有効にします。

```
# svcadm enable svc:/system/pools/dynamic:default
# svcs -a | grep pool
STATE          STIME    FMRI
disabled       2011     svc:/system/pools:default
offline        2011     svc:/system/pools/dynamic:default
```

DRP サービスはまだオフラインです。

svcs コマンドの -x オプションを使って、DRP サービスがオフラインになっている原因を特定します。

```
# svcs -x "*pool*"
svc:/system/pools:default (resource pools framework)
 State: disabled since Sat Feb 12 02:36:15 2011
Reason: Disabled by an administrator.
   See: http://support.oracle.com/msg/SMF-8000-05
   See: libpool(3LIB)
   See: pooladm(1M)
   See: poolbind(1M)
   See: poolcfg(1M)
   See: poolstat(1M)
Impact: This service is not running.

svc:/system/pools/dynamic:default (dynamic resource pools)
 State: disabled since Sat Feb 12 02:36:16 2011
Reason: Disabled by an administrator.
   See: http://support.oracle.com/msg/SMF-8000-05
   See: poold(1M)
Impact: This service is not running.
```

リソースプールサービスを有効にして、DRP サービスを実行可能にします。

```
# svcadm enable svc:/system/pools:default
```

`svcs "*pool*"` コマンドを使用すると、システムによって次の情報が表示されます。

```
# svcs "*pool*"
STATE          STIME    FMRI
online         2011     svc:/system/pools/dynamic:default
online         2011     svc:/system/pools:default
```

例 13–2 リソースプールサービスが無効な場合の動的リソースプールへの影響

両方のサービスがオンラインで、リソースプールサービスを無効にする場合は、次のコマンドを実行します。

```
# svcadm disable svc:/system/pools:default
```

`svcs "*pool*"` コマンドを使用すると、システムによって次の情報が表示されます。

```
# svcs "*pool*"
STATE          STIME    FMRI
disabled       2011     svc:/system/pools:default
online         2011     svc:/system/pools/dynamic:default
```

ただし、リソースプールサービスが無効になるため、結果として DRP サービスが `offline` になります。

```
# svcs "*pool*"
STATE          STIME    FMRI
disabled       2011     svc:/system/pools:default
offline        2011     svc:/system/pools/dynamic:default
```

DRP サービスがオフラインになっている原因を特定します。

```
# svcs -x "*pool*"
svc:/system/pools:default (resource pools framework)
 State: disabled since Sat Feb 12 02:36:15 2011
Reason: Disabled by an administrator.
   See: http://support.oracle.com/msg/SMF-8000-05
   See: libpool(3LIB)
   See: pooladm(1M)
   See: poolbind(1M)
   See: poolcfg(1M)
   See: poolstat(1M)
Impact: 1 dependent service is not running.  (Use -v for list.)

svc:/system/pools/dynamic:default (dynamic resource pools)
 State: offline since Sat Feb 12 02:36:15 2011
Reason: Service svc:/system/pools:default is disabled.
   See: http://support.oracle.com/msg/SMF-8000-GE
   See: poold(1M)
   See: /var/svc/log/system-pools-dynamic:default.log
Impact: This service is not running.
```

DRP が機能するためには、リソースプールを起動する必要があります。たとえば、`pooladm` コマンドと `-e` オプションを使ってリソースプールを起動できます。

```
# pooladm -e
```

その後、`svcs "*pool*"` コマンドを実行すると、次のように表示されます。

```
# svcs "*pool*"
STATE          STIME    FMRI
online         2011     svc:/system/pools:default
online         2011     svc:/system/pools/dynamic:default
```

▼ `svcadm` を使用して動的リソースプールサービスを無効にする方法

1. **root** になるか、同等の役割になります。
2. 動的リソースプールサービスを無効にします。

   ```
   # svcadm disable system/pools/dynamic:default
   ```

▼ `pooladm` を使ってリソースプールを有効にする方法

1. **root** になるか、同等の役割になります。
2. プール機能を有効にします。

   ```
   # pooladm -e
   ```

▼ pooladm を使ってリソースプールを無効にする方法

1. **root** になるか、同等の役割になります。
2. プール機能を無効にします。

```
# pooladm -d
```

プールの構成

▼ 静的構成を作成する方法

`-s` オプションを付けて `/usr/sbin/pooladm` を実行して、現在の動的構成と一致する静的構成ファイルを作成します。これにより、リブート後にも変更内容が保持されます。別のファイル名を指定した場合を除き、デフォルトの場所 `/etc/pooladm.conf` が使用されます。

`pooladm` コマンドに `-c` オプションを付けて実行して、構成を確定します。次に、`pooladm` コマンドに `-s` オプションを付けて実行して、動的構成の状態と一致するように静的構成ファイルを更新します。

注 – 動的構成と一致する新しい構成を作成するには、以前の機能である `poolcfg -c discover` を使用するよりも、より新しい機能である `pooladm -s` を使用することをお勧めします。

始める前に　使用しているシステムでプールを有効にします。

1. **root** になるか、同等の役割になります。
2. 現在の動的構成と一致するように静的構成ファイルを更新します。

```
# pooladm -s
```

3. 構成ファイルの内容を読みやすい形式で表示します。

構成には、システムによって作成されたデフォルトの要素が含まれています。

```
# poolcfg -c info
system tester
        string  system.comment
        int     system.version 1
        boolean system.bind-default true
        int     system.poold.pid 177916
```

```
        pool pool_default
                int     pool.sys_id 0
                boolean pool.active true
                boolean pool.default true
                int     pool.importance 1
                string  pool.comment
                pset    pset_default

        pset pset_default
                int     pset.sys_id -1
                boolean pset.default true
                uint    pset.min 1
                uint    pset.max 65536
                string  pset.units population
                uint    pset.load 10
                uint    pset.size 4
                string  pset.comment
                boolean testnullchanged true

                cpu
                        int     cpu.sys_id 3
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 2
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 1
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 0
                        string  cpu.comment
                        string  cpu.status on-line
```

4 **/etc/pooladm.conf** にある構成を確定します。

```
# pooladm -c
```

5 (オプション) **/tmp/backup** という静的構成ファイルに動的構成をコピーするには、次のように入力します。

```
# pooladm -s /tmp/backup
```

▼ 構成の変更方法

構成を拡張するために、pset_batch というプロセッサセットと pool_batch というプールを作成します。次に、このプールとプロセッサセットを結合によって対応付けます。

サブコマンドの引数に空白が含まれている場合は、引用符で囲むことを忘れないでください。

1 **root** になるか、同等の役割になります。

2 **pset_batch** というプロセッサセットを作成します。

```
# poolcfg -c 'create pset pset_batch (uint pset.min = 2; uint pset.max = 10)'
```

3 **pool_batch** というプールを作成します。

```
# poolcfg -c 'create pool pool_batch'
```

4 このプールとプロセッサセットを結合によって対応付けます。

```
# poolcfg -c 'associate pool pool_batch (pset pset_batch)'
```

5 対応付けた後の構成を表示します。

```
# poolcfg -c info
system tester
        string  system.comment kernel state
        int     system.version 1
        boolean system.bind-default true
        int     system.poold.pid 177916

        pool pool_default
                int     pool.sys_id 0
                boolean pool.active true
                boolean pool.default true
                int     pool.importance 1
                string  pool.comment
                pset    pset_default

        pset pset_default
                int     pset.sys_id -1
                boolean pset.default true
                uint    pset.min 1
                uint    pset.max 65536
                string  pset.units population
                uint    pset.load 10
                uint    pset.size 4
                string  pset.comment
                boolean testnullchanged true

                cpu
                        int     cpu.sys_id 3
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 2
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 1
```

```
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 0
                        string  cpu.comment
                        string  cpu.status on-line

        pool pool_batch
                boolean pool.default false
                boolean pool.active true
                int pool.importance 1
                string pool.comment
                pset pset_batch

        pset pset_batch
                int pset.sys_id -2
                string pset.units population
                boolean pset.default true
                uint pset.max 10
                uint pset.min 2
                string pset.comment
                boolean pset.escapable false
                uint pset.load 0
                uint pset.size 0

                cpu
                        int     cpu.sys_id 5
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 4
                        string  cpu.comment
                        string  cpu.status on-line
```

6 **/etc/pooladm.conf** にある構成を確定します。

```
# pooladm -c
```

7 **(オプション) /tmp/backup** という静的構成ファイルに動的構成をコピーするには、次のように入力します。

```
# pooladm -s /tmp/backup
```

▼ プールをスケジューリングクラスに対応付ける方法

プールをスケジューリングクラスに対応付けることで、プールに結合されているすべてのプロセスがこのスケジューラを使用できるようになります。このためには、`pool.scheduler` プロパティーにスケジューラの名前を設定します。次の例では、プール `pool_batch` を公平配分スケジューラ (FSS) に対応付けます。

1 **root** になるか、同等の役割になります。

2 **pool_batch** プールを変更して、**FSS** に対応付けます。

```
# poolcfg -c 'modify pool pool_batch (string pool.scheduler="FSS")'
```

3 対応付けた後の構成を表示します。

```
# poolcfg -c info
system tester
        string  system.comment
        int     system.version 1
        boolean system.bind-default true
        int     system.poold.pid 177916

        pool pool_default
                int     pool.sys_id 0
                boolean pool.active true
                boolean pool.default true
                int     pool.importance 1
                string  pool.comment
                pset    pset_default

        pset pset_default
                int     pset.sys_id -1
                boolean pset.default true
                uint    pset.min 1
                uint    pset.max 65536
                string  pset.units population
                uint    pset.load 10
                uint    pset.size 4
                string  pset.comment
                boolean testnullchanged true

                cpu
                        int     cpu.sys_id 3
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 2
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 1
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 0
                        string  cpu.comment
                        string  cpu.status on-line

        pool pool_batch
                boolean pool.default false
                boolean pool.active true
                int pool.importance 1
                string pool.comment
                string pool.scheduler FSS
                pset batch
```

```
        pset pset_batch
                int pset.sys_id -2
                string pset.units population
                boolean pset.default true
                uint pset.max 10
                uint pset.min 2
                string pset.comment
                boolean pset.escapable false
                uint pset.load 0
                uint pset.size 0

                cpu
                        int     cpu.sys_id 5
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 4
                        string  cpu.comment
                        string  cpu.status on-line
```

4 **/etc/pooladm.conf** にある構成を確定します。

```
# pooladm -c
```

5 **(オプション) /tmp/backup** という静的構成ファイルに動的構成をコピーするには、次のように入力します。

```
# pooladm -s /tmp/backup
```

▼ 構成の制約を設定する方法

制約は、構成に加えることのできる変更を一部除外することで、作成可能な構成の範囲に影響を与えます。ここでは、cpu.pinned プロパティーを設定する手続きを示します。

次の例では、cpuid は整数です。

1 **root** になるか、同等の役割になります。

2 静的構成または動的構成内の **cpu.pinned** プロパティーを変更します。

- ブート時の (静的) 構成を変更します。

  ```
  # poolcfg -c 'modify cpu <cpuid> (boolean cpu.pinned = true)'
  ```

- ブート時の構成を変更せずに、実行中の (動的) 構成を変更します。

  ```
  # poolcfg -dc 'modify cpu <cpuid> (boolean cpu.pinned = true)'
  ```

▼ 構成の目標を定義する方法

poold に対して、修正操作を実行するときに考慮する目標を指定できます。

次の手順では、wt-load 目標を設定して、poold がリソースの使用効率に合わせてリソースを割り当てるようにします。この構成目標を達成しやすくするために、locality 目標は無効にします。

1. **root** になるか、同等の役割になります。
2. システム **tester** を変更して、**wt-load** 目標を優先するようにします。

```
# poolcfg -c 'modify system tester (string system.poold.objectives="wt-load")'
```

3. デフォルトのプロセッサセットの **locality** 目標を無効にします。

```
# poolcfg -c 'modify pset pset_default (string pset.poold.objectives="locality none")'     one line
```

4. **pset_batch** プロセッサセットの **locality** 目標を無効にします。

```
# poolcfg -c 'modify pset pset_batch (string pset.poold.objectives="locality none")'     one line
```

5. 対応付けた後の構成を表示します。

```
# poolcfg -c info
system tester
        string  system.comment
        int     system.version 1
        boolean system.bind-default true
        int     system.poold.pid 177916
        string  system.poold.objectives wt-load

        pool pool_default
                int     pool.sys_id 0
                boolean pool.active true
                boolean pool.default true
                int     pool.importance 1
                string  pool.comment
                pset    pset_default

        pset pset_default
                int     pset.sys_id -1
                boolean pset.default true
                uint    pset.min 1
                uint    pset.max 65536
                string  pset.units population
                uint    pset.load 10
                uint    pset.size 4
                string  pset.comment
                boolean testnullchanged true
                string  pset.poold.objectives locality none

                cpu
                        int     cpu.sys_id 3
                        string  cpu.comment
                        string  cpu.status on-line
```

```
                cpu
                        int     cpu.sys_id 2
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 1
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 0
                        string  cpu.comment
                        string  cpu.status on-line

        pool pool_batch
                boolean pool.default false
                boolean pool.active true
                int pool.importance 1
                string pool.comment
                string pool.scheduler FSS
                pset batch

        pset pset_batch
                int pset.sys_id -2
                string pset.units population
                boolean pset.default true
                uint pset.max 10
                uint pset.min 2
                string pset.comment
                boolean pset.escapable false
                uint pset.load 0
                uint pset.size 0
                string  pset.poold.objectives locality none

                cpu
                        int     cpu.sys_id 5
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 4
                        string  cpu.comment
                        string  cpu.status on-line
```

6 **/etc/pooladm.conf** にある構成を確定します。

```
# pooladm -c
```

7 (オプション) **/tmp/backup** という静的構成ファイルに動的構成をコピーするには、次のように入力します。

```
# pooladm -s /tmp/backup
```

▼ poold のログレベルを設定する方法

poold が生成するログ情報のレベルを指定するには、poold 構成の system.poold.log-level プロパティーを設定します。poold の構成は libpool の構成に保存されています。詳細については、154 ページの「poold のログ情報」および poolcfg(1M) と libpool(3LIB) のマニュアルページを参照してください。

コマンド行で poold コマンドを使用する方法でも、poold で生成するログ情報のレベルを指定できます。

1. root になるか、同等の役割になります。
2. **poold** コマンドに **-l** オプションとパラメータ (**INFO** など) を付けて実行することで、ログのレベルを設定します。

```
# /usr/lib/pool/poold -l INFO
```

使用可能なパラメータについては、154 ページの「poold のログ情報」を参照してください。デフォルトのログレベルは NOTICE です。

▼ poolcfg でコマンドファイルを使用する方法

poolcfg コマンドに -f オプションを付けて使用すると、poolcfg コマンドの -c オプションに指定する引数をテキストファイルから入力できます。この方法は、一連の操作を実行する場合に使用します。複数のコマンドを処理した場合でも、それらのコマンドがすべて正常に終了するまで、構成は更新されません。特に大規模な構成や複雑な構成の場合は、この手法を使用した方が、個々のサブコマンドを起動するよりも便利です。

コマンドファイルでは、# という文字はコメント記号として機能し、その行の残り部分がコメントと見なされます。

1. 入力ファイル **poolcmds.txt** を作成します。

```
$ cat > poolcmds.txt
create system tester
create pset pset_batch (uint pset.min = 2; uint pset.max = 10)
create pool pool_batch
associate pool pool_batch (pset pset_batch)
```

2. root になるか、同等の役割になります。
3. コマンドを実行します。

```
# /usr/sbin/poolcfg -f poolcmds.txt
```

リソースの転送

-d オプションを付けた poolcfg に -c オプションの transfer サブコマンド引数を付けて実行すると、カーネルでリソースを転送できます。-d オプションは、コマンドにファイルから入力するのではなく、直接カーネル上で実行することを示します。

次の手順では、2 つの CPU をプロセッサセット pset1 からプロセッサセット pset2 にカーネルで移動します。

▼ プロセッサセット間で CPU を移動する方法

1 **root** になるか、同等の役割になります。

2 **2 つの CPU を pset1 から pset2 に移動します。**

from 文節と to 文節は、どの順序で使用してもかまいません。to 文節と from 文節は、1 つのコマンドにそれぞれ 1 つだけ使用できます。

```
# poolcfg -dc 'transfer 2 from pset pset1 to pset2'
```

例 13–3 プロセッサセット間で CPU を移動する別の方法

あるリソースタイプの特定のリソースの ID を指定して転送する場合は、別の構文が用意されています。たとえば、次のコマンドは、ID が 0 と 2 の 2 つの CPU を pset_large プロセッサセットに割り当てます。

```
# poolcfg -dc 'transfer to pset pset_large (cpu 0; cpu 2)'
```

参考 トラブルシューティング

要求を満たすための十分なリソースがない場合や、指定された ID が見つからない場合は、転送は失敗し、エラーメッセージが表示されます。

プール構成の起動と削除

pooladm コマンドを使用すると、特定のプール構成をアクティブにしたり、現在アクティブになっているプール構成を削除したりできます。このコマンドの詳細については、pooladm(1M) のマニュアルページを参照してください。

▼ プール構成を起動する方法

デフォルト構成ファイル /etc/pooladm.conf に保存されている構成を起動するには、pooladm に -c オプション (構成の確定) を付けて実行します。

1. **root** になるか、同等の役割になります。
2. **/etc/pooladm.conf** にある構成を確定します。

   ```
   # pooladm -c
   ```

3. (オプション) たとえば **/tmp/backup** という静的構成ファイルに動的構成をコピーするには、次のように入力します。

   ```
   # pooladm -s /tmp/backup
   ```

▼ 構成を確定する前に構成を検証する方法

`-n` オプションと `-c` オプションをともに使用すると、検証が実行されるとどうなるかをテストできます。構成が実際に確定されることはありません。

次のコマンドは、`/home/admin/newconfig` に保存されている構成を検証します。検出されたエラー条件が表示されますが、構成自体は変更されません。

1. **root** になるか、同等の役割になります。
2. 構成を確定する前に構成を検証します。

   ```
   # pooladm -n -c /home/admin/newconfig
   ```

▼ プール構成を削除する方法

現在アクティブになっている構成を削除して、プロセッサセットなどの関連付けられているすべてのリソースをデフォルトのステータスに戻すには、`-x` オプション (構成の削除) を使用します。

1. **root** になるか、同等の役割になります。
2. 現在アクティブになっている構成を削除します。

   ```
   # pooladm -x
   ```

   `-x` オプションを付けて `pooladm` を実行すると、ユーザーが定義したすべての要素が動的構成から削除されます。すべてのリソースがデフォルトの状態に戻り、プールとの結合もすべてデフォルトプールとの結合で置換されます。

参考 プロセッサセット内におけるスケジューリングクラスの混在

TS クラスのプロセスと IA クラスのプロセスを同一プロセッサセット内で混在させても問題はありません。1 つのプロセッサセット内でその他のスケジューリングクラスを混在させると、予期できない結果が生じる可能性があります。`pooladm -x` を使用した結果、1 つのプロセッサセット内にスケジューリングクラスが混在している場合

は、priocntl コマンドを使用して、実行中のプロセスを別のスケジューリングクラスに移動してください。118 ページの「プロセスを TS クラスから FSS クラスに手動で移動する方法」を参照してください。priocntl(1) のマニュアルページも参照してください。

プール属性の設定とプールへの結合

リソースプールをプロジェクトに関連付けるために、project.pool 属性を設定できます。

実行中のプロセスをプールに結合するには、次の 2 つの方法を使用できます。

- poolbind コマンド (poolbind(1M) のマニュアルページに記載) を使用して、特定のプロセスを指定されたリソースプールに結合します。
- project データベース内の project.pool 属性を使用して、新しいログインセッションや newtask コマンドで起動されるタスクを結合するプールを指定します。newtask(1)、projmod(1M)、および project(4) のマニュアルページを参照してください。

▼ プロセスをプールに結合する方法

次の手順では、poolbind コマンドに -p オプションを付けて実行して、プロセス (この例では、現在のシェル) を ohare というプールに手動で結合します。

1. **root** になるか、同等の役割になります。
2. プロセスをプールに手動で結合します。

```
# poolbind -p ohare $$
```

3. **poolbind** に **-q** オプションを付けて実行することで、プロセスとプールの結合を確認します。

```
$ poolbind -q $$
155509 ohare
```

プロセス ID とプールへの結合が表示されます。

▼ タスクまたはプロジェクトをプールに結合する方法

タスクまたはプロジェクトをプールに結合するには、poolbind コマンドに -i オプションを指定します。次の例では、airmiles プロジェクト内のすべてのプロセスを laguardia プールに結合します。

1 **root** になるか、同等の役割になります。

2 **airmiles** プロジェクト内のすべてのプロセスを **laguardia** プールに結合します。

```
# poolbind -i project -p laguardia airmiles
```

▼ プロジェクトの project.pool 属性を設定する方法

プロジェクトのプロセスをリソースプールに結合するために、project.pool 属性を設定できます。

1 **root** になるか、同等の役割になります。

2 **project** データベース内の各エントリに **project.pool** 属性を追加します。

```
# projmod -a -K project.pool=poolname  project
```

▼ project 属性を使ってプロセスを別のプールに結合する方法

studio と backstage という 2 つのプールを持つ構成が存在するものとします。/etc/project ファイルの内容は、次のとおりです。

```
user.paul:1024::::project.pool=studio
user.george:1024::::project.pool=studio
user.ringo:1024::::project.pool=backstage
passes:1027::paul::project.pool=backstage
```

この構成の場合、ユーザー paul によって起動されるプロセスは、デフォルトで studio プールに結合されます。

ユーザー paul は、起動するプロセスのプール結合を変更できます。paul は、newtask を使用して (この場合は passes プロジェクト内で起動することで)、作業を backstage プールに結合することもできます。

1 **passes** プロジェクトでプロセスを起動します。

```
$ newtask -l -p passes
```

2 **poolbind** コマンドに **-q** オプションを付けて実行し、プロセスとプールの結合を確認します。また、二重ドル記号 **($$)** を使用して親シェルのプロセス番号をコマンドに渡します。

```
$ poolbind -q $$
6384  pool backstage
```

プロセス ID とプールへの結合が表示されます。

poolstat を使ってプールに関連付けられているリソースについて統計情報を報告する

poolstat コマンドを使用すると、プールに関連付けられているリソースについて統計情報を表示できます。詳細は、159 ページの「poolstat によるプール機能とリソース使用効率の監視」および poolstat(1M) のマニュアルページを参照してください。

このセクションでは、さまざまな目的の報告を生成する方法を、例を使用しながら説明します。

poolstat のデフォルトの出力を表示する

引数なしで poolstat と入力すると、見出し行に続いて、1 行に 1 つずつプールが表示されます。情報の行には、プール ID、プール名、およびプールに接続されているプロセッサセットに関するリソース統計が表示されます。

```
machine% poolstat
                           pset
       id pool           size used load
        0 pool_default      4  3.6  6.2
        1 pool_sales        4  3.3  8.4
```

特定の間隔で複数の報告を生成する

次のコマンドは、3 つの報告を 5 秒間のサンプリング間隔で生成します。

```
machine% poolstat 5 3
                            pset
 id pool                  size used load
 46 pool_sales               2  1.2  8.3
  0 pool_default             2  0.4  5.2
                            pset
 id pool                  size used load
 46 pool_sales               2  1.4  8.4
  0 pool_default             2  1.9  2.0
                            pset
 id pool                  size used load
 46 pool_sales               2  1.1  8.0
  0 pool_default             2  0.3  5.0
```

リソースセットの統計情報を報告する

次の例では、poolstat コマンドに -r オプションを付けて実行し、プロセッサセットのリソースセットの統計情報を報告します。リソースセット pset_default は複数のプールに接続されているので、このプロセッサセットは各プールで 1 回ずつ表示されます。

```
machine% poolstat -r pset
       id pool          type rid rset          min  max size used load
        0 pool_default  pset  -1 pset_default    1  65K    2  1.2  8.3
        6 pool_sales    pset   1 pset_sales      1  65K    2  1.2  8.3
        2 pool_other    pset  -1 pset_default    1  10K    2  0.4  5.2
```

◆ ◆ ◆ 第 14 章

リソース管理の構成例

この章では、リソース管理のフレームワークについて考察し、仮想的なサーバー統合プロジェクトについて説明します。

この章の内容は次のとおりです。

- 185 ページの「統合前の構成」
- 186 ページの「統合後の構成」
- 186 ページの「構成の作成」
- 188 ページの「構成の表示」

統合前の構成

この例では、5 つのアプリケーションを 1 つのシステムに統合します。対象となるアプリケーションは、それぞれリソース要件、ユーザー数、およびアーキテクチャーが異なります。現在、各アプリケーションは、それぞれの要件を満たす専用サーバーに置かれています。次の表にアプリケーションとその特性を示します。

アプリケーション	特性
アプリケーションサーバー	2 CPU を超えるとスケーラビリティーが低くなります
アプリケーションサーバー用のデータベースインスタンス	負荷の高いトランザクション処理
テストおよび開発環境用のアプリケーションサーバー	GUI に基づいたコードテスト
トランザクション処理サーバー	応答時間を保証します
スタンドアロンのデータベースインスタンス	大量のトランザクションを処理し、複数のタイムゾーンに対してサービスを提供します

統合後の構成

次の構成は、リソースプールと動的リソースプールの機能を有効にしている 1 つのシステムにアプリケーションを統合するために使用します。

- アプリケーションサーバーは、2 つの CPU から構成されるプロセッサセットを持ちます。
- アプリケーションサーバーのデータベースインスタンスとスタンドアロンのデータベースインスタンスは、4 つ以上の CPU から構成される 1 つのプロセッサセットに統合されます。スタンドアロンのデータベースインスタンスはそのリソースの 75% が保証されます。
- テストおよび開発用のアプリケーションサーバーには IA スケジューリングクラスを適用して、UI の応答性を保証します。メモリーを制約して、不正なコードによる影響を低減します。
- トランザクション処理サーバーには 2 つ以上の CPU から構成される専用のプロセッサセットを割り当てて、応答時間を短縮します。

この構成の対象には、各リソースセットのプロセッササイクルを消費している実行中の既知のアプリケーションすべてが含まれます。したがって、プロセッサリソースを必要としているセットにプロセッサリソースをセット間で転送できるように、次のような制約を設定します。

- `wt-load` 目標を設定して、使用効率の高いリソースセットに、より多くのリソースを割り当てるようにします。
- `locality` 目標を `tight` に設定して、プロセッサの近傍性が最大になるようにします。

また、どのリソースセットについても使用効率が 80% を超えないようにする制約も適用します。この制約により、アプリケーションは必要なリソースに確実にアクセスできます。さらに、トランザクション処理のプロセッサセットについては、使用効率を 80% 以下に保つという目標の重要性を、ほかの目標の 2 倍にします。この重要性は構成で定義します。

構成の作成

`/etc/project` データベースファイルを編集します。エントリを追加して必要なリソース制御を実装し、ユーザーをリソースプールにマップしたら、ファイルを表示します。

```
# cat /etc/project
.
.
.
user.app_server:2001:Production Application Server:::project.pool=appserver_pool
```

```
user.app_db:2002:App Server DB:::project.pool=db_pool;project.cpu-shares=(privileged,1,deny)
development:2003:Test and development::staff:project.pool=dev_pool;
process.max-address-space=(privileged,536870912,deny)     keep with previous line
user.tp_engine:2004:Transaction Engine:::project.pool=tp_pool
user.geo_db:2005:EDI DB:::project.pool=db_pool;project.cpu-shares=(privileged,3,deny)
.
.
.
```

注 – 開発チームはタスクを開発プロジェクトで実行する必要があります。これは、このプロジェクトへのアクセスをユーザーのグループ ID (GID) で制限しているためです。

pool.host という名前で入力ファイルを作成し、必要なリソースプールの構成に使用します。次に、ファイルを表示します。

```
# cat pool.host
create system host
create pset dev_pset (uint pset.min = 0; uint pset.max = 2)
create pset tp_pset (uint pset.min = 2; uint pset.max=8)
create pset db_pset (uint pset.min = 4; uint pset.max = 6)
create pset app_pset (uint pset.min = 1; uint pset.max = 2)
create pool dev_pool (string pool.scheduler="IA")
create pool appserver_pool (string pool.scheduler="TS")
create pool db_pool (string pool.scheduler="FSS")
create pool tp_pool (string pool.scheduler="TS")
associate pool dev_pool (pset dev_pset)
associate pool appserver_pool (pset app_pset)
associate pool db_pool (pset db_pset)
associate pool tp_pool (pset tp_pset)
modify system tester (string system.poold.objectives="wt-load")
modify pset dev_pset (string pset.poold.objectives="locality tight; utilization < 80")
modify pset tp_pset (string pset.poold.objectives="locality tight; 2: utilization < 80")
modify pset db_pset (string pset.poold.objectives="locality tight;utilization < 80")
modify pset app_pset (string pset.poold.objectives="locality tight; utilization < 80")
```

pool.host 入力ファイルを使って構成を更新します。

```
# poolcfg -f pool.host
```

構成をアクティブにします。

```
# pooladm -c
```

システム上でフレームワークが有効になっています。

DRP を有効にします。

```
# svcadm enable pools/dynamic:default
```

構成の表示

フレームワークの構成には、システムによって作成されたデフォルトの要素も含まれています。この構成を表示するには、次のように入力します。

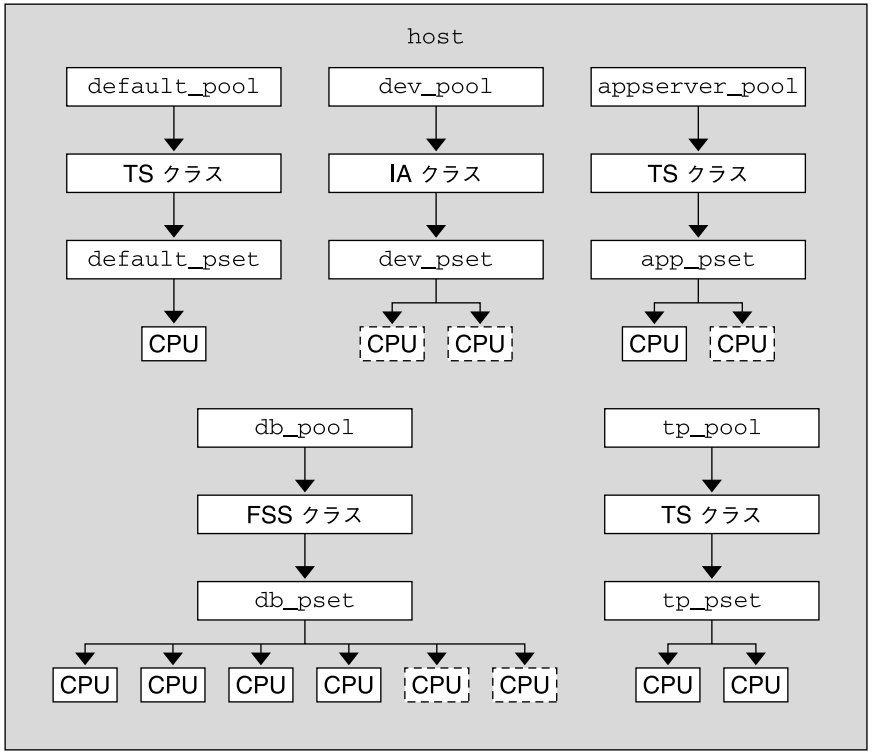
```
# pooladm
system host
        string  system.comment
        int     system.version 1
        boolean system.bind-default true
        int     system.poold.pid 177916
        string  system.poold.objectives wt-load

        pool dev_pool
                int     pool.sys_id 125
                boolean pool.default false
                boolean pool.active true
                int     pool.importance 1
                string  pool.comment
                string  pool.scheduler IA
                pset    dev_pset

        pool appserver_pool
                int     pool.sys_id 124
                boolean pool.default false
                boolean pool.active true
                int     pool.importance 1
                string  pool.comment
                string  pool.scheduler TS
                pset    app_pset

        pool db_pool
                int     pool.sys_id 123
                boolean pool.default false
                boolean pool.active true
                int     pool.importance 1
                string  pool.comment
                string  pool.scheduler FSS
                pset    db_pset

        pool tp_pool
                int     pool.sys_id 122
                boolean pool.default false
                boolean pool.active true
                int     pool.importance 1
                string  pool.comment
                string  pool.scheduler TS
                pset    tp_pset

        pool pool_default
                int     pool.sys_id 0
                boolean pool.default true
                boolean pool.active true
                int     pool.importance 1
                string  pool.comment
                string  pool.scheduler TS
                pset    pset_default
```

```
pset dev_pset
        int     pset.sys_id 4
        string  pset.units population
        boolean pset.default false
        uint    pset.min 0
        uint    pset.max 2
        string  pset.comment
        boolean pset.escapable false
        uint    pset.load 0
        uint    pset.size 0
        string  pset.poold.objectives locality tight; utilization < 80

pset tp_pset
        int     pset.sys_id 3
        string  pset.units population
        boolean pset.default false
        uint    pset.min 2
        uint    pset.max 8
        string  pset.comment
        boolean pset.escapable false
        uint    pset.load 0
        uint    pset.size 0
        string  pset.poold.objectives locality tight; 2: utilization < 80

        cpu
                int     cpu.sys_id 1
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 2
                string  cpu.comment
                string  cpu.status on-line

pset db_pset
        int     pset.sys_id 2
        string  pset.units population
        boolean pset.default false
        uint    pset.min 4
        uint    pset.max 6
        string  pset.comment
        boolean pset.escapable false
        uint    pset.load 0
        uint    pset.size 0
        string  pset.poold.objectives locality tight; utilization < 80

        cpu
                int     cpu.sys_id 3
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 4
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 5
```

```
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 6
                        string  cpu.comment
                        string  cpu.status on-line
        pset app_pset
                int     pset.sys_id 1
                string  pset.units population
                boolean pset.default false
                uint    pset.min 1
                uint    pset.max 2
                string  pset.comment
                boolean pset.escapable false
                uint    pset.load 0
                uint    pset.size 0
                string  pset.poold.objectives locality tight; utilization < 80
                cpu
                        int     cpu.sys_id 7
                        string  cpu.comment
                        string  cpu.status on-line

        pset pset_default
                int     pset.sys_id -1
                string  pset.units population
                boolean pset.default true
                uint    pset.min 1
                uint    pset.max 4294967295
                string  pset.comment
                boolean pset.escapable false
                uint    pset.load 0
                uint    pset.size 0

                cpu
                        int     cpu.sys_id 0
                        string  cpu.comment
                        string  cpu.status on-line
```

フレームワークのグラフィック表示が続きます。

図 14–1 サーバー統合の構成



注 – 上記の図のプール `db_pool` では、スタンドアロンのデータベースインスタンスに CPU リソースの 75% が保証されています。

パート II

Oracle Solaris ゾーン

このパートでは、Oracle Solaris ゾーンソフトウェア区分技術について説明します。この技術を使用すると、オペレーティングシステムサービスを仮想化し、アプリケーションを実行するための隔離された環境を作成できます。このように隔離することで、あるゾーンで実行中のプロセスが、ほかのゾーンで実行中のプロセスを監視したり、それらに影響を与えたりすることを防ぐことができます。

◆ ◆ ◆ 第 15 章

Oracle Solaris ゾーンの紹介

Oracle Solaris オペレーティングシステムの Oracle Solaris ゾーン機能では、システム上にアプリケーションを実行するための隔離された環境が提供されます。

この章では、ゾーンの概要について説明します。

また、この章では、次の一般的なゾーンについての各トピックについても説明しています。

- 196 ページの「ゾーンの概要」
- 197 ページの「このリリースの Oracle Solaris ゾーンについて」
- 200 ページの「ブランドゾーンについて」
- 202 ページの「ゾーンを使用する場合」
- 204 ページの「ゾーンのしくみ」
- 212 ページの「非大域ゾーンによって提供される機能」
- 213 ページの「システムのゾーンの設定 (タスクマップ)」

システムにゾーンを作成する準備が完了している場合は、第 16 章「非大域ゾーンの構成 (概要)」に進んでください。

注 – Oracle Solaris 10 ゾーンの詳細は、パート III「Oracle Solaris 10 ゾーン」を参照してください。

Oracle Solaris Trusted Extensions システム上でゾーンを使用する方法については、『Trusted Extensions 構成と管理』の第 13 章「Trusted Extensions でのゾーンの管理」を参照してください。

ゾーンの概要

Oracle Solaris ゾーン区分技術は、オペレーティングシステムサービスを仮想化して、アプリケーションを実行するための隔離された安全な環境を提供するために使用します。非大域ゾーンは、ゾーンと呼ばれ、Oracle Solaris オペレーティングシステムの 1 つのインスタンス内で作成される仮想化されたオペレーティングシステム環境です。オペレーティングシステムのインスタンスは大域ゾーンと呼ばれます。

仮想化の目標は、個々のデータセンターコンポーネントの管理からリソースのプールの管理に移行することです。サーバーを正しく仮想化すると、サーバー使用効率が向上し、サーバー資産をより効率的に使用できるようになります。個々のシステムの隔離を維持するサーバー統合プロジェクトで成功を収めるためにも、サーバーの仮想化は重要です。

仮想化は、複数のホストおよびサービスを単一のマシン上に統合する必要がある場合に推進されます。仮想化すると、ハードウェア、インフラストラクチャー、および管理を共有することにより、コストを削減できます。次のような利点があります。

- ハードウェアの使用効率の向上
- より柔軟なリソース割り当て
- 電力要件の削減
- 管理コストの低下
- 保有コストの低減
- システムのアプリケーション間の管理およびリソースの境界

ゾーンを作成すると、そのアプリケーション実行環境で実行されるプロセスは、システムのほかの部分から隔離されます。この分離を行うことで、1 つのゾーン内で稼働しているプロセスがほかのゾーンで稼働しているプロセスを監視したりそれらのプロセスに影響を及ぼしたりすることが防止されます。root 資格で実行されているプロセスであっても、ほかのゾーンの活動を表示したり影響を与えたりすることはできません。Oracle Solaris ゾーンを使用すれば、ハードウェアリソースを同時に共有しながら、1 サーバーあたりに 1 つのアプリケーションという配備モデルを維持できます。

また、ゾーンにより、アプリケーションを配備するマシンの物理的属性からアプリケーションを分離する抽象層も提供されます。このような属性の例として、物理デバイスパスがあります。

ゾーンは、Oracle Solaris 10 またはその後の Oracle Solaris リリースを稼働しているマシン上で使用できます。システムに作成できるゾーン数の上限は 8192 です。1 つのシステムで効率的にホストできるゾーン数は、すべてのゾーンで実行されるアプリケーションソフトウェアに必要な総リソース量、およびそのシステムのサイズによって決まります。

これらの概念については、第 17 章「非大域ゾーンの計画と構成 (タスク)」を参照してください。

このリリースの Oracle Solaris ゾーンについて

このセクションでは、Oracle Solaris ゾーンの新機能と変更点 (Oracle Solaris 10 リリース以降に行われた改良を含む) の概要について説明します。

このリリースのデフォルトの非大域ゾーンは solaris です。これについては、このガイドおよび solaris(5) のマニュアルページで説明されています。

単一の大域ゾーン内で実行されている非大域ゾーンは、Oracle Solaris 11.1 リリースのサポート対象プラットフォームとして定義されているすべてのアーキテクチャーでサポートされます。

Oracle Solaris リリースとマシンアーキテクチャーを確認するには、次のように入力します。

```
#uname -r -m
```

solaris ゾーンは、ブランドゾーンフレームワーク (brands(5) マニュアルページに記載) を使用して、大域ゾーンにインストールされているソフトウェアと同じソフトウェアがインストールされたゾーンを実行します。solaris 非大域ゾーンを使用している場合、システムソフトウェアは大域ゾーンとの間で同期が常に取れている必要があります。ゾーン内のシステムソフトウェアパッケージは、イメージパッケージングシステム (IPS) を使用して管理されます。IPS は Oracle Solaris 11 リリース上のパッケージングシステムで、solaris ゾーンはこのモデルを使用します。

Oracle Solaris 11 Express リリース上に作成されるデフォルトの ipkg ゾーンは、solaris ゾーンにマップされます。200 ページの「ipkg ゾーンの solaris ゾーンへの変換について」を参照してください。

自動インストール (AI) のマニフェスト内に指定された各非大域ゾーンは、クライアントのインストールの一部としてインストールおよび構成されます。非大域ゾーンは、大域ゾーンのインストール後の初回リブート時にインストールされ、構成されます。システムの最初のブート時に、ゾーンの自己アセンブリ SMF サービスの svc:/system/zones-install:default は、大域ゾーンの AI マニフェスト内で定義された各非大域ゾーンを構成およびインストールします。詳細については、『Oracle Solaris 11.1 システムのインストール』を参照してください。また、インストール済みの Oracle Solaris システムでゾーンを手動で構成およびインストールすることも可能です。

パッケージ更新の場合は、--proxy オプションを使用してイメージに永続的なプロキシを設定するようにしてください。永続的なイメージプロキシ構成を使用しない場合は、http_proxy および https_proxy 環境変数を設定できます。

ゾーンは、順次ではなく並列的に更新されるように構成できます。並列更新により、システム上のすべてのゾーンを更新するのに必要な時間が大幅に短縮されます。

ゾーンは、デフォルトでは排他的 IP タイプで作成されます。ネットワーク接続構成が指定されない場合、VNIC は anet リソースによってゾーン構成内に自動的に含まれます。詳細は、224 ページの「ゾーンネットワークインタフェース」を参照してください。

共有ストレージ上のゾーンには、zonecfg rootzpool リソースがあります。ゾーンは専用の zpool にカプセル化されます。共有ストレージ上のゾーンは、ゾーン用の共有ストレージリソースにアクセスし、それらを管理します。

zonecfg anet リソースでは、IP over InfiniBand (IPoIB) データリンクを指定するために使用される 2 つの新しいプロパティーが使用可能です。IPoIB は、solaris と solaris10 の両方のブランドゾーンでサポートされています。

RDS (Reliable Datagram Sockets) IPC プロトコルは、排他的 IP 非大域ゾーンと共有 IP 非大域ゾーンの両方でサポートされています。

fsstat ユーティリティーは、ゾーンをサポートするように拡張されました。fsstat ユーティリティーは、ゾーン別および集計の統計情報を提供します。

solaris ゾーンを NFS サーバーにすることができます (361 ページの「ゾーンでの NFS サーバーの実行」に記載)。

試行 (「ドライラン」とも呼ばれる) の zoneadm attach -n は、zonecfg の検査を提供しますが、パッケージ内容の検査は実行しません。

ファイルを引数に取る zoneadm のすべてのオプションには、絶対パスが必要です。

Oracle Solaris 10 ゾーンは、Oracle Solaris 11 上に Oracle Solaris 10 環境を提供します。Oracle Solaris 11 システム上の solaris10 ゾーンに Oracle Solaris 10 システムまたはゾーンを移行できます。

zonep2vchk ツールは、Oracle Solaris 11 システムまたは Oracle Solaris 10 システムの Oracle Solaris 11 リリースを実行しているシステム上のゾーンへの移行に影響を与える可能性のある、ネットワークの問題を含む問題を特定します。zonep2vchk ツールは、移行が始まる前にソースシステム上で実行されます。このツールは、ターゲットシステム上で使用するための zonecfg スクリプトも出力します。このスクリプトは、ソースシステムの構成と一致するゾーンを作成します。詳細は、第 22 章「ゾーンの移行と zonep2vchk ツールについて」を参照してください。

Oracle Solaris 10 リリース上の solaris ゾーンと native ゾーンとの間の次の相違に注意してください。

- Oracle Solaris 11 システムでは、Oracle Solaris 10 システム上のデフォルトである native ブランドではなく、solaris ブランドが作成されます。
- solaris ゾーンは、完全ルートタイプのみです。

  Oracle Solaris 10 上で使用できるネイティブゾーンの疎ルートタイプは、SVR4 パッケージ管理システムを使用し、IPS はこのフレームワークを使用しません。疎ルートタイプと同様の読み取り専用ルートゾーン構成を使用できます。

- このリリースのゾーンには、これらの領域での Oracle Solaris 10 リリースとは異なる、次のソフトウェア管理関連の機能があります。
  - IPS パッケージと SVR4 パッケージ。
  - インストール、切り離し、接続、および Physical-To-Virtual 機能。
  - 非大域ゾーンルートが ZFS データセットである。

    大域ゾーンにインストールされたパッケージは、現在および将来のすべてのゾーンにインストールされなくなります。一般的に、IPS パッケージと SVR4 パッケージの両方において、大域ゾーンのパッケージ内容は、各ゾーンのパッケージ内容に影響しません。
- 非大域ゾーンはブート環境を使用します。ゾーンは、ZFS ブート環境 (BE) を管理するためのユーザーインタフェースコマンドである `beadm` と統合されています。

  `beadm` コマンドは、大域ゾーン内と同様に、`pkg update` 用にゾーンの内部でサポートされます。`beadm` コマンドは、ゾーンに関連付けられている、アクティブでないゾーン BE を削除できます。`beadm(1M)` のマニュアルページを参照してください。
- 有効になっているすべての IPS パッケージリポジトリが、ゾーンのインストール中にアクセス可能である必要があります。詳細は、300 ページの「構成済みのゾーンをインストールする方法」を参照してください。
- ゾーンソフトウェアは、最初は最小限に抑えられています。ゾーンが必要とするその他のパッケージは、追加する必要があります。詳細は、solaris publisher (`http://pkg.oracle.com/solaris/release/`) を参照してください。

ゾーンでは、次のような Oracle Solaris 11.1 の各製品と機能を使用できます。

- Oracle Solaris ZFS 暗号化
- ネットワーク仮想化および QoS
- CIFS と NFS

非大域ゾーン内では、次の機能を構成できません。

- 共有 IP ゾーンでの DHCP アドレスの割り当て
- `ndmpd`
- SMB サーバー
- SSL プロキシサーバー
- `zpool` コマンドを使用した ZFS プールの管理

読み取り専用の solaris 非大域ゾーン

不変ゾーンは、読み取り専用ルートを持つゾーンです。読み取り専用ゾーンは、file-mac-profile プロパティーを設定することによって構成できます。複数の構成が利用できます。読み取り専用ゾーンルートは、安全な実行時の境界を拡張します。

zonecfg add dataset を使用して追加のデータセットが指定されたゾーンは、それらのデータセットに対する完全な制御権を引き続き持ちます。ファイルシステムが読み取り専用に設定されていない限り、zonecfg add fs を使用して追加のファイルシステムが指定されたゾーンは、それらのファイルシステムに対する完全な制御権を引き続き持ちます。

詳細は、第 27 章「不変ゾーンの構成と管理」を参照してください。

ipkg ゾーンの solaris ゾーンへの変換について

Oracle Solaris 11 Express リリースのお客様をサポートするために、ipkg ゾーンとして構成されたゾーンは solaris ゾーンに変換され、Oracle Solaris 11.1 に対する pkg update または zoneadm attach の実行時に solaris として報告されます。ipkg の名前がゾーンの構成時に使用された場合、その名前が solaris の名前にマップされます。Oracle Solaris 11 Express ホストからエクスポートされた zonecfg ファイルのインポートがサポートされます。

zonecfg info や zoneadm list -v などのコマンドの出力には、Oracle Solaris 11.1 システム上のデフォルトのゾーンに対して solaris というブランドが表示されます。

ブランドゾーンについて

システム上の非大域ゾーンは、デフォルトでは大域ゾーンと同じオペレーティングシステムソフトウェアを実行します。Oracle Solaris オペレーティングシステムのブランドゾーン (BrandZ) 機能は、Oracle Solaris ゾーンの単純な拡張です。BrandZ フレームワークは、大域ゾーンのオペレーティング環境とは異なるオペレーティング環境を含む非大域ブランドゾーンを作成するために使用します。ブランドゾーンは、Oracle Solaris オペレーティングシステムでアプリケーションを実行するために使用します。BrandZ フレームワークは、さまざまな方法で Oracle Solaris ゾーンインフラストラクチャーを拡張します。これらの拡張には、ゾーン内で異なるオペレーティングシステム環境を実行するための機能を提供するような複雑なものもあれば、新しい機能を提供するために基本のゾーンコマンドを拡張するような簡単なものもあります。たとえば、Oracle Solaris 10 ゾーンは、Oracle Solaris 10 オペレーティングシステムをエミュレートできるブランド非大域ゾーンです。大域ゾーンと同じオペレーティングシステムを共有するデフォルトのゾーンの場合も、ブランドを使用して構成します。

ブランドは、ゾーンにインストールできるオペレーティング環境を定義し、ゾーンにインストールされたソフトウェアが正しく機能するようにゾーン内でのシステムの動作方法を決定します。また、ゾーンのブランドにより、アプリケーションの起動時に正しいアプリケーションタイプが識別されます。すべてのブランドゾーンの管理は、標準のゾーン構造に対する拡張を通して実行されます。管理手順のほとんどはすべてのゾーンで同一です。

定義済みのファイルシステムや特権などの、デフォルトで構成に含まれているリソースについては、ブランドに関するドキュメントで説明されています。

BrandZ はゾーンのツールを次のように拡張します。

- `zonecfg` コマンドを使用して、ゾーンの構成時にゾーンのブランドタイプを設定します。
- `zoneadm` コマンドを使用して、ゾーンのブランドタイプの報告とゾーンの管理を行います。

ラベルが有効にされている Oracle Solaris Trusted Extensions システムにはブランドゾーンを構成およびインストールできますが、ブートされるブランドが認定されたシステム構成上のラベル付きブランドでない限り、このシステム構成上でブランドゾーンをブートすることはできません。

構成済み状態にあるゾーンのブランドは変更することができます。ブランドゾーンのインストールが完了したあとは、そのブランドの変更や削除を行うことはできません。

注意 – 既存の Oracle Solaris 10 システムを Oracle Solaris 11 リリースが稼働するシステム上の `solaris10` ブランドゾーンに移行することを計画している場合は、最初に既存のゾーンをすべてターゲットシステムに移行する必要があります。ゾーンは入れ子にならないため、システムの移行処理は既存のゾーンをすべて使用不可にします。詳細は、パート III「Oracle Solaris 10 ゾーン」を参照してください。

ブランドゾーンで実行中のプロセス

ブランドゾーンでは、ブランドゾーンで実行中のプロセスだけに適用される一連の介入ポイントがカーネル内に用意されます。

- これらのポイントは、`syscall` パス、プロセスローディングパス、スレッド作成パスなどのパス内に見つかります。
- これらの各ポイントで、ブランドは Oracle Solaris の標準的な動作を補完したり置き換えたりすることができます。

ブランドは `librtld_db` のプラグインライブラリを提供することもできます。デバッガ (`mdb(1)` に記載) や DTrace (`dtrace(1M)` に記載) といった Oracle Solaris のツール

は、このプラグインライブラリを使用することによって、ブランドゾーン内で実行中のプロセスのシンボル情報にアクセスできます。

ゾーンでは、静的にリンクされたバイナリはサポートされません。

このリリースで使用できる非大域ゾーン

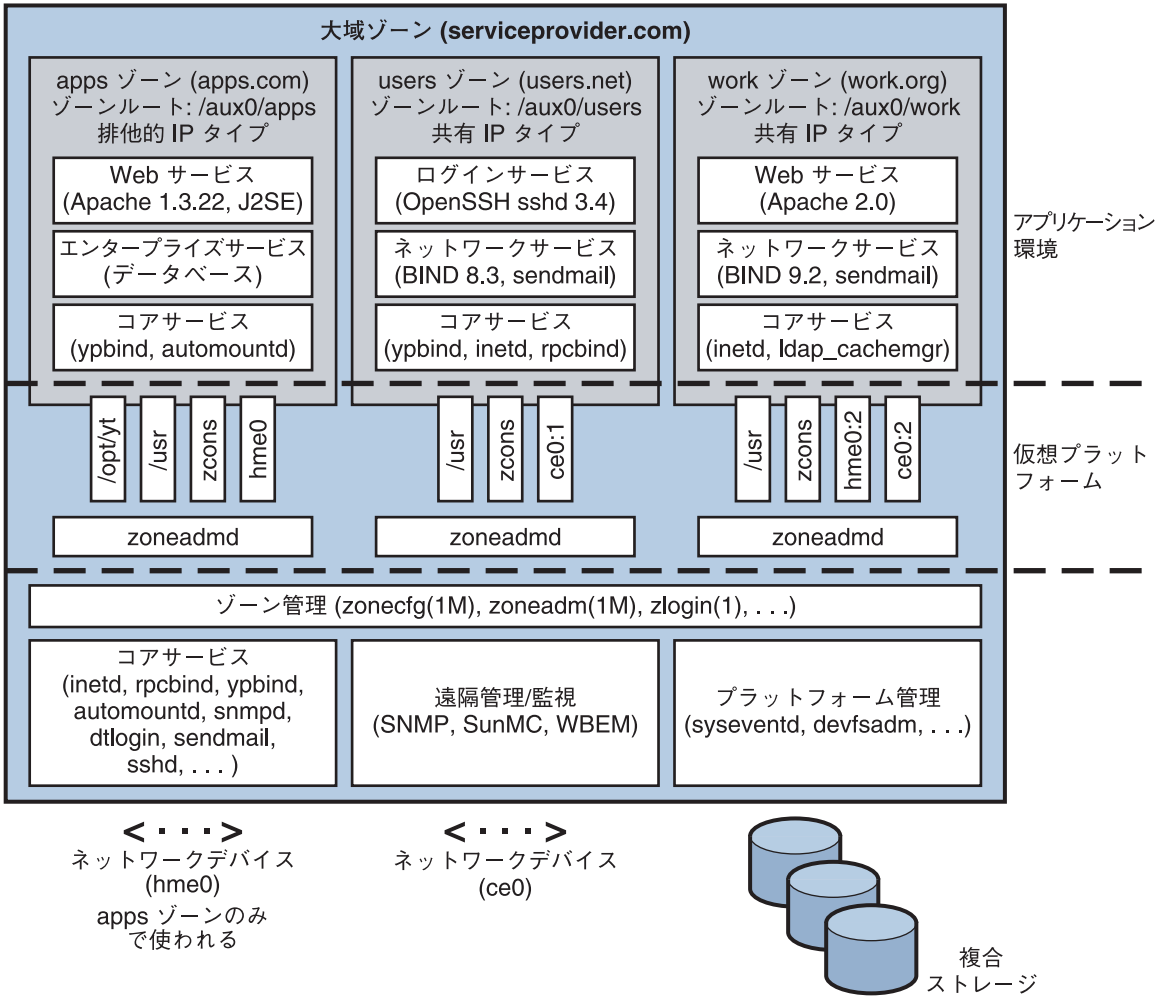
このリリースには、デフォルトの Oracle Solaris ゾーンのほかに、Oracle Solaris 10 ゾーン (`solaris10` ブランドゾーン) 製品が含まれています。詳細は、パート III「Oracle Solaris 10 ゾーン」を参照してください。

ゾーンを使用する場合

ゾーンは、多くのアプリケーションを 1 台のサーバー上で統合できる環境で使用するともっとも高い効果を発揮します。多数のマシンの管理は複雑でコストがかかるため、より大規模で拡張性の高いサーバーにアプリケーションを統合することが望まれます。

次の図は、3 つのゾーンから成るシステムを示しています。この統合された環境の例では、`apps`、`users`、および `work` の各ゾーンは、ほかのゾーンの作業負荷とは無関係に作業負荷を実行しています。この例は、同じアプリケーションのさまざまなバージョンをそれぞれ異なるゾーンで実行でき、悪影響を引き起こすことなく、統合の要件を満たすことができることを示しています。ゾーンごとにカスタマイズされたサービスを提供できます。

図 15–1 ゾーンによるサーバー統合の例



ゾーンを使用すると、システムのリソースをより効率的に利用できます。リソースの動的再割り当てにより、使用されていないリソースを必要に応じてほかのゾーンに移動できます。障害およびセキュリティーの隔離により、動作状態の悪いアプリケーションのために使用効率の低い専用のシステムを用意する必要がなくなります。ゾーンを使用すると、このようなアプリケーションをほかのアプリケーションと統合できます。

ゾーンを使用すると、総合的なシステムセキュリティーを維持しながら、管理機能の一部を委譲できます。

ゾーンのしくみ

非大域ゾーンは、1 つの箱と考えることができます。この箱の中では、システムのほかの部分と相互に作用することなく、1 つ以上のアプリケーションを実行できます。ゾーンは、ソフトウェアで定義される柔軟な境界を使用して、ソフトウェアアプリケーションやサービスを隔離します。これにより、Oracle Solaris オペレーティングシステムの同じ 1 つのインスタンス内で実行される複数のアプリケーションを互いに独立して管理できます。したがって、同じアプリケーションのさまざまなバージョンをそれぞれ異なるゾーンで実行でき、構成の要件を満たすことができます。

ゾーンに割り当てられたプロセスは、同じゾーンに割り当てられたほかのプロセスを操作、監視したり、これらのプロセスと直接通信したりできます。システムのほかのゾーンに割り当てられたプロセスや、ゾーンに割り当てられていないプロセスに対しては、このような機能は実行できません。異なるゾーンに割り当てられたプロセスどうしでは、ネットワーク API を介した通信のみ可能です。

ゾーンが独自の排他的 IP インスタンスを持っているのか、IP 層の構成と状態を大域ゾーンと共有しているのかに応じて、IP ネットワーク接続を 2 通りの方法で構成できます。排他的 IP がデフォルトのタイプです。ゾーンの IP タイプの詳細については、224 ページの「ゾーンネットワークインタフェース」を参照してください。構成については、270 ページの「ゾーンの構成方法」を参照してください。

Oracle Solaris システムごとに 1 つの大域ゾーンがあります。大域ゾーンは 2 つの機能を持っています。大域ゾーンは、システムのデフォルトのゾーンであり、システム全体の管理に使用されるゾーンでもあります。非大域ゾーン (単に「ゾーン」と呼ばれる) が、大域管理者またはゾーンのセキュリティープロファイルを持つユーザーによって作成されていない場合、すべてのプロセスが大域ゾーンで実行されます。

非大域ゾーンの構成、インストール、管理、およびアンインストールは、大域ゾーンからのみ行うことができます。システムハードウェアからブートできるのは、大域ゾーンだけです。物理デバイス、共有 IP ゾーンでのルーティング、動的再構成 (DR) といったシステム基盤の管理は、大域ゾーンでのみ行うことができます。大域ゾーンで実行されるプロセスは、適切な権限が付与されていれば、ほかのゾーンに関連付けられているオブジェクトにもアクセスできます。

非大域ゾーンの特権付きプロセスには許可されていない操作を、大域ゾーンの特権のないプロセスが実行できることもあります。たとえば、大域ゾーンのユーザーは、システムのすべてのプロセスに関する情報を表示できます。この機能がサイトで問題になる場合は、大域ゾーンへのアクセスを制限します。

大域ゾーンも含め、各ゾーンにはゾーン名が割り当てられます。大域ゾーンの名前は常に `global` となります。各ゾーンには、一意の数値 ID も与えられます。これは、ゾーンのブート時にシステムによって割り当てられます。大域ゾーンには、常に ID `0` が割り当てられます。ゾーンの名前と数値 ID については、237 ページの「`zonecfg` コマンドの使用」を参照してください。

各ゾーンには、ノード名も割り当てられます。これは、ゾーン名とは完全に独立した名前です。ノード名は、ゾーンの管理者によって割り当てられます。詳細は、360 ページの「非大域ゾーンのノード名」を参照してください。

各ゾーンには、ルートディレクトリのパスが設定されます。これは、大域ゾーンのルートディレクトリに対する相対パスです。詳細は、237 ページの「zonecfg コマンドの使用」を参照してください。

デフォルトでは、非大域ゾーンのスケジューリングクラスは、システムのスケジューリングクラスと同じに設定されます。ゾーンのスケジューリングクラスを設定する方法については、220 ページの「スケジューリングクラス」を参照してください。

ゾーンの機能別のサマリー

次の表に、大域ゾーンと非大域ゾーンの特性をまとめます。

ゾーンの種類	特性
広域	■ システムによって ID 0 が割り当てられます ■ システムでブートでき、実行される Oracle Solaris カーネルの単一のインスタンスを提供します ■ Oracle Solaris システムソフトウェアパッケージの完全なインストールが含まれています ■ 追加のソフトウェアパッケージや、パッケージを通してインストールされない追加のソフトウェア、ディレクトリ、ファイル、その他のデータが含まれている場合もあります ■ 大域ゾーンにインストールされているすべてのソフトウェアコンポーネントに関する情報を格納した、一貫性のある完全な製品データベースを提供します ■ 大域ゾーンのホスト名やファイルシステムテーブルなど、大域ゾーンのみに固有の構成情報を保持します ■ すべてのデバイスとすべてのファイルシステムが認識される、唯一のゾーンです ■ 非大域ゾーンの存在と構成が認識される、唯一のゾーンです ■ 非大域ゾーンの構成、インストール、管理、またはアンインストールを行うことができる、唯一のゾーンです

ゾーンの種類	特性
非大域	■ ゾーンのブート時にシステムによってゾーン ID が割り当てられます ■ 大域ゾーンからブートされる Oracle Solaris カーネルの下で処理を共有します ■ 完全な Oracle Solaris オペレーティングシステムソフトウェアパッケージのインストール済みのサブセットが含まれています ■ 追加のインストール済みソフトウェアパッケージを含めることができます ■ 追加のソフトウェア、ディレクトリ、ファイル、およびパッケージによってインストールされない、非大域ゾーンで作成されたその他のデータを含めることができます ■ ゾーンにインストールされているすべてのソフトウェアコンポーネントに関する情報を格納した、一貫性のある完全な製品データベースを保持します ■ ほかのゾーンの存在を認識できません ■ 自身を含め、ゾーンのインストール、管理、アンインストールを行うことはできません ■ 非大域ゾーンのホスト名やファイルシステムテーブルなど、その非大域ゾーンのみに固有の構成情報を保持します ■ 独自のタイムゾーン設定を持つことができます

非大域ゾーンの管理のしくみ

大域管理者は、スーパーユーザー特権または同等の管理権限を持ちます。大域ゾーンにログインすると、大域管理者はシステム全体を監視したり制御したりできます。

非大域ゾーンは「ゾーン管理者」が管理できます。大域管理者は、219 ページの「admin リソース」で説明されているように、ゾーン管理者に必要な承認を割り当てます。ゾーン管理者の特権は、特定の非大域ゾーンのみに制限されます。

非大域ゾーンの作成のしくみ

非大域ゾーンの構成とインストールは、自動インストール (AI) のクライアントインストールの一環として指定できます。詳細については、『Oracle Solaris 11.1 システムのインストール』を参照してください。

Oracle Solaris システム上にゾーンを作成する場合、大域管理者は zonecfg コマンドを使用して、ゾーンの仮想プラットフォームとアプリケーション環境に対して各種パラメータを指定することによってゾーンを構成します。次に、大域管理者がゾーン

をインストールします。大域管理者は、ゾーン管理コマンド zoneadm を使用して、ゾーンに対応するファイルシステム階層にソフトウェアをパッケージレベルでインストールします。ゾーンをブートするには、zoneadm コマンドを使用します。次に、大域管理者または承認されたユーザーは、zlogin コマンドを使用してインストールされたゾーンにログインできます。役割によるアクセス制御 (RBAC) を使用中の場合、ゾーン管理者は solaris.zone.manage/*zonename* の承認を持っている必要があります。

ゾーンの構成については、第 16 章「非大域ゾーンの構成 (概要)」を参照してください。ゾーンのインストールについては、第 18 章「非大域ゾーンのインストール、停止処理、停止、アンインストール、クローニングについて (概要)」を参照してください。ゾーンへのログインについては、第 20 章「非大域ゾーンへのログイン (概要)」を参照してください。

非大域ゾーンの状態モデル

非大域ゾーンの状態は、次の 7 つのいずれかになります。

構成済み
: ゾーンの構成は完了し、安定した記憶領域に確定されています。ただし、ゾーンのアプリケーション環境の要素のうち、最初のブート後に指定する必要のあるものは、まだ含まれていません。

不完全
: インストール処理やアンインストール処理の途中は、zoneadm によってターゲットゾーンの状態が「不完全」に設定されます。処理が正常に完了すると、適切な状態に設定されます。

  損傷を受けたインストール済みゾーンは、zoneadm の mark サブコマンドを使用して、不完全のマークを付けることができます。不完全な状態のゾーンは、zoneadm list - iv の出力に表示されます。

使用不可
: ゾーンがインストールされているが、ゾーンを検証、準備、ブート、接続、または移動できないことを示します。ゾーンは次の時点で使用不可状態になります。

  - ゾーンのストレージが使用できず、svc:/system/zones:default が起動したとき (システムのブート中など)
  - ゾーンのストレージが使用できないとき
  - アーカイブの抽出に成功したあとでアーカイブベースのインストールが失敗したとき

- ゾーンのソフトウェアと大域ゾーンのソフトウェアに互換性がないとき (不適切な -F (強制的) 接続のあとなど)

インストール済み	ゾーンの構成はシステム上でインスタンス化されています。zoneadm コマンドは、指定した Oracle Solaris システム上で、この構成を正常に使用できることを確認するために使用します。パッケージはゾーンのルートパスにインストールされます。この状態では、ゾーンに関連付けられた仮想プラットフォームはありません。
準備完了	ゾーンの仮想プラットフォームが確立されています。カーネルにより zsched プロセスが作成され、ネットワークインタフェースが設定されてゾーンで使用可能になり、ファイルシステムがマウントされ、デバイスの構成が完了しています。システムにより、一意のゾーン ID が割り当てられます。この状態では、ゾーンに関連付けられたプロセスは起動されていません。
稼働	ゾーンのアプリケーション環境に関連付けられたユーザープロセスが稼働状態です。アプリケーション環境に関連付けられた最初のユーザープロセス (init) が作成されるとすぐに、ゾーンの状態は「稼働」になります。
停止処理中および停止	これらは、ゾーンの停止処理の間に見られる遷移状態です。ただし、なんらかの理由でゾーンを停止処理できない場合は、ゾーンがどちらかの状態で停止します。

zoneadm コマンドを使用してこれらの状態間の遷移を開始する方法は、第 19 章「非大域ゾーンのインストール、ブート、停止処理、停止、アンインストール、およびクローニング (タスク)」および zoneadm(1M) のマニュアルページに記載されています。

表 15–1 ゾーンの状態に影響を与えるコマンド

ゾーンの現在の状態	適用できるコマンド

構成済み	`zonecfg -z` *zonename* `verify`
	`zonecfg -z` *zonename* `commit`
	`zonecfg -z` *zonename* `delete`
	`zoneadm -z` *zonename* `attach`
	`zoneadm -z` *zonename* `verify`
	`zoneadm -z` *zonename* `install`
	`zoneadm -z` *zonename* `clone`
	`zoneadm -z` *zonename* `mark` *incomplete*
	`zoneadm -z` *zonename* `mark` *unavailable*
	`zonecfg` を使用して、構成済みまたはインストール済みの状態にあるゾーンの名前を変更することもできます。

不完全	`zoneadm -z` *zonename* `uninstall`
使用不可	`zoneadm -z` *zonename* `uninstall` は、指定されたシステムからゾーンをアンインストールします。 `zoneadm -z` *zonename* `attach` `zonecfg -z` *zonename* は、`zonepath` やその他のインストール済み状態のときに変更できるプロパティーまたはリソースを変更するために使用できます。

インストール済み	`zoneadm -z` *zonename* `ready` (オプション)
	`zoneadm -z` *zonename* `boot`
	`zoneadm -z` *zonename* `uninstall` は、指定されたゾーンの構成をシステムからアンインストールします。
	`zoneadm -z` *zonename* `move` *path*
	`zoneadm -z` *zonename* `detach`
	`zonecfg -z` *zonename* を使用すると、プロパティーの `attr`、`bootargs`、`capped-memory`、`dataset`、`capped-cpu`、`dedicated-cpu`、`device`、`fs`、`ip-type`、`limitpriv`、`net`、`rctl`、または `scheduling-class` を追加または削除することができます。インストール済み状態のゾーンの名前を変更することもできます。
	`zoneadm -z` *zonename* `mark` *incomplete*
	`zoneadm -z` *zonename* `mark` *unavailable*

表 15–1 ゾーンの状態に影響を与えるコマンド (続き)

ゾーンの現在の状態	適用できるコマンド
準備完了	`zoneadm -z` *zonename* `boot` `zoneadm halt` とシステムリブートを実行すると、準備完了状態のゾーンがインストール済み状態に戻ります。 `zonecfg -z` *zonename* を使用すると、プロパティーの `attr`、`bootargs`、 `capped-memory`、 `dataset`、 `capped-cpu`、`dedicated-cpu`、 `device`、 `fs`、 `ip-type`、 `limitpriv`、 `net`、 `rctl`、または `scheduling-class` を追加または削除することができます。

稼働	`zlogin` *options* `zonename`
	
	`zoneadm -z` *zonename* `reboot`
	
	`zoneadm -z`*zonename* `halt` を実行すると、準備完了状態のゾーンがインストール済み状態に戻ります。
	
	`zoneadm halt` とシステムのリブートを実行すると、稼働状態のゾーンがインストール済み状態に戻ります。
	
	`zoneadm - z shutdown` は、ゾーンを完全に停止処理します。
	
	`zonecfg -z` *zonename* を使用すると、プロパティーの `attr`、`bootargs`、 `capped-memory`、 `dataset`、 `capped-cpu`、`dedicated-cpu`、 `device`、 `fs`、 `ip-type`、 `limitpriv`、 `anet`、 `net`、`rctl`、 または `scheduling-class` を追加または削除することができます。`zonepath` リソースは変更できません。

注 – `zonecfg` 経由で変更されたパラメータは、稼働中のゾーンには影響しません。変更を適用するには、ゾーンをリブートする必要があります。

非大域ゾーンの特性

ゾーンを使用すると、必要に応じてほぼどのような単位にも細かく隔離できます。専用の CPU、物理デバイス、物理メモリーの一部分などをゾーンに割り当てる必要はありません。このようなリソースは、1 つのドメインまたはシステムで実行される複数のゾーンに渡って多重化するか、オペレーティングシステムに用意されているリソース管理機能を使ってゾーンごとに割り当てることができます。

ゾーンごとにカスタマイズされたサービスを提供できます。基本的なプロセス隔離を強化するために、同じゾーン内のプロセスのみ互いに認識したりシグナルを送信したりできます。ゾーン間で基本的な通信を行うには、各ゾーンに IP 接続機能を持たせます。あるゾーンで実行中のアプリケーションが、別のゾーンのネットワークトラフィックを監視することはできません。それぞれのパケットストリームが同じ物理インタフェースを通過する場合でも、この隔離は維持されます。

各ゾーンには、ファイルシステム階層の一部分が割り当てられます。各ゾーンは、ファイルシステム階層で割り当てられた部分ツリーに限定されます。したがって、特定のゾーンで実行されている作業負荷は、別のゾーンで実行されているほかの作業負荷のディスク上のデータにアクセスすることはできません。

ネームサービスで使用されるファイルは、ゾーン独自のルートファイルシステムのビュー内に置かれます。したがって、異なるゾーンのネームサービスは互いに隔離され、サービスごとに異なる構成を使用できます。

非大域ゾーンでのリソース管理機能の使用

リソース管理機能を使用する場合は、リソース管理制御の境界とゾーンの境界をそろえる必要があります。このように境界をそろえることで、名前空間のアクセス、セキュリティー隔離、およびリソースの使用状況をすべて制御できる、より完成された仮想マシンのモデルを作成できます。

ゾーンで各種のリソース管理機能を使用するための特殊要件については、このドキュメントでこれらの機能に関連する各章を参照してください。

ゾーン関連の SMF サービス

大域ゾーン内のゾーン関連の SMF サービスには、次のようなものがあります。

`svc:/system/zones:default`
: `autoboot=true` である各ゾーンを起動します。

`svc:/system/zones-install:default`
: 必要に応じて、初回ブート時にゾーンのインストールを行います。

`svc:/application/pkg/zones-proxyd:default`
: パッケージシステムがゾーンに対してシステムリポジトリへのアクセスを提供するために使用します。

`svc:/application/pkg/system-repository:default`
: ゾーンのインストールやその他の `pkg` 操作中に使用される `pkg` データとメタデータをキャッシュするキャッシュプロキシサーバー。`pkg(1)` および `pkg(5)` のマニュアルページを参照してください。

`svc:/system/zones-monitoring:default`
: `zonestatd` を制御します。

`svc:/application/pkg/zones-proxy-client:default` ゾーンプロキシクライアント SMF サービスは、非大域ゾーンでのみ実行されます。このサービスは、パッケージシステムがゾーンに対してシステムリポジトリへのアクセスを提供するために使用します。

非大域ゾーンの監視

現在実行中のゾーンの CPU、メモリー、およびリソース制御の使用効率について報告する場合は、397 ページの「非大域ゾーンでの zonestat ユーティリティーの使用」を参照してください。zonestat ユーティリティーは、排他的 IP ゾーン内のネットワーク帯域幅の使用効率についても報告を生成します。排他的 IP ゾーンには、独自の IP に関連付けられた状態と 1 つ以上の専用のデータリンクが保持されます。

fsstat ユーティリティーを使用して、非大域ゾーンのファイル操作の統計情報を報告できます。fsstat(1M) のマニュアルページおよび360 ページの「fsstat ユーティリティーを使用した非大域ゾーンの監視」を参照してください。

非大域ゾーンによって提供される機能

非大域ゾーンは、次のような機能を提供します。

セキュリティー

大域ゾーン以外のゾーンにプロセスを配置したあとは、そのプロセス自体やそのプロセスの子がゾーンを変更することはできません。

ネットワークサービスをゾーンで実行できます。ネットワークサービスをゾーンで実行すると、セキュリティー違反が発生した場合の損害を抑えることができます。ゾーン内で実行されているソフトウェアのセキュリティー欠陥を侵入者が悪用できた場合でも、そのゾーン内で可能な一連の操作しか実行できません。ゾーン内で使用できる特権は、システム全体で使用できる特権の一部のみです。

隔離

複数のアプリケーションが異なる信頼ドメインで動作する場合や、大域リソースへの排他的アクセスを必要とする場合、または、大域の構成を使用すると問題を示すような場合でも、ゾーンを使用することでこれらのアプリケーションを同じマシン上に配備できます。アプリケーションが互いのネットワークトラフィック、ファイルシステムデータ、プロセスの活動などを監視したり妨害したりすることもできなくなります。

ネットワーク隔離

デフォルトでは、ゾーンは排他的 IP タイプとして構成されます。ゾーンは IP 層で大域ゾーンから分離され、かつ相互に分離されます。この分離は、運用上とセキュリティー上の両方の理由で有用です。ゾーンは、専用の LAN または VLAN を使用して異なるサブネット上で通信する必要があるアプリケーションを統合するために使用できます。ゾーンごとに独自の IP 層のセキュリティー規則を定義することもできます。

仮想化	ゾーンによって提供される仮想環境では、物理デバイスやシステムのプライマリ IP アドレスとホスト名などの詳細をアプリケーションから隠すことができます。同じアプリケーション環境を、物理的に異なるマシンで維持管理することもできます。仮想化された環境では、各ゾーンを個別に管理できます。非大域ゾーンでゾーン管理者によって行われる操作は、システムのほかの部分に影響を与えません。
隔離単位	ゾーンを使用すると、ほぼどのような単位にも細かく隔離できます。詳細は、210 ページの「非大域ゾーンの特性」を参照してください。
環境	セキュリティーや隔離の目標を達成するために必要な場合を除き、アプリケーションの実行される環境がゾーンによって変更されることはありません。ゾーンを使用するために、新しい API や ABI にアプリケーションを移植する必要はありません。代わりに、ゾーンでは Oracle Solaris の標準インタフェースとアプリケーション環境が提供されます。ただし、いくつかの制限があります。これらの制限は主に、特権付き操作を実行しようとするアプリケーションに影響を与えます。 大域ゾーンで実行されるアプリケーションは、追加のゾーンが構成されたかどうかにかかわらず、変更なしで実行できます。

システムのゾーンの設定 (タスクマップ)

次の表に、システム上ではじめてゾーンを設定する際に必要となるタスクの基本概要を示します。

タスク	説明	参照先
ゾーンで実行するアプリケーションを特定します。	システムで実行されるアプリケーションを見直します。 ■ ビジネス目標にとってどのアプリケーションが重要かを判定します。 ■ 実行するアプリケーションのシステム要件を評価します。	必要に応じて、ビジネス目標とシステムのドキュメントを参照してください。

タスク	説明	参照先
構成するゾーンの数を決定します。	次の内容を評価します。 ■ ゾーンで実行する予定のアプリケーションの性能要件。 ■ インストールされるゾーンごとの 1G バイトのディスク容量の可用性。必要な容量は、そのゾーン内にインストールされるソフトウェアに依存し、それに応じて調整する必要があります。ZFS 圧縮を使用すると、必要なディスク容量が低減します。 非大域ゾーンのインストール、およびその後のパッケージのインストールと更新の間に、ある程度の一時領域が必要になることに注意してください。1G バイトのディスク容量要件は、このことを考慮に入れています。	265 ページの「現在のシステム設定の評価」を参照してください。
ゾーンでマシンリソースを区分するために、リソースプールまたは割り当てられる CPU を使用するかどうかを決定します。	システム上でリソース管理機能も使用する場合は、リソース管理の境界とゾーンの境界をそろえます。ゾーンを構成する前にリソースプールを構成します。 `zonecfg` のプロパティーを使用すると、ゾーン規模のリソース制御とプール機能をゾーンにすばやく追加できます。	270 ページの「ゾーンの構成方法」および第 13 章「リソースプールの作成と管理 (タスク)」を参照してください。

タスク	説明	参照先
事前構成タスクを行います。	ゾーン名とゾーンパスを決定します。 ゾーンに対する追加の要件 (共有ストレージ上でゾーンをホストするかどうかなど) を決定します。 デフォルトでは、非大域ゾーンは anet リソースを使用して排他的 IP タイプとして作成されます。anet リソースは、非大域ゾーンに対して自動の仮想 NIC (VNIC) を作成します。別の選択肢として、net リソースを使用して、ゾーンを共有 IP ゾーンまたは排他的 IP ゾーンとして構成することもできます。各ゾーンに必要なファイルシステムとデバイスを決定します。ゾーンのスケジューリングクラスを決定します。標準のデフォルト特権セットでは十分でない場合は、ゾーンのプロセスを制限するための特権セットを決定します。zonecfg の設定の中には自動的に特権を追加するものがあります。たとえば、ip-type=exclusive はネットワークスタックの構成および管理に必要な複数の特権を自動的に追加します。	ゾーンの名前とパス、IP タイプ、IP アドレス、ファイルシステム、デバイス、スケジューリングクラス、および特権については、第 16 章「非大域ゾーンの構成 (概要)」および 265 ページの「現在のシステム設定の評価」を参照してください。非大域ゾーンでのデフォルトの特権および構成可能な特権のリストについては、378 ページの「非大域ゾーン内の特権」を参照してください。IP 機能については、369 ページの「共有 IP 非大域ゾーンにおけるネットワーク」および371 ページの「排他的 IP 非大域ゾーンにおけるネットワーク」を参照してください。
構成を作成します。	非大域ゾーンの構成を行います。	269 ページの「ゾーンを構成、検証、および確定する」および zonecfg(1M) のマニュアルページを参照してください。

タスク	説明	参照先
大域管理者または適切な承認を持つユーザーとして、構成されたゾーンを検証およびインストールします。	ゾーンにログインする前に、ゾーンの確認とインストールを行う必要があります。 ゾーンの初期内部構成は、インストール時に作成されて構成されます。	第 18 章「非大域ゾーンのインストール、停止処理、停止、アンインストール、クローニングについて (概要)」、第 19 章「非大域ゾーンのインストール、ブート、停止処理、停止、アンインストール、およびクローニング (タスク)」、`sysconfig`(1M)、および『Oracle Solaris 11.1 システムのインストール』の第 6 章「Oracle Solaris インスタンスの構成解除または再構成」を参照してください。
大域管理者または適切な承認が付与されたユーザーとして、非大域ゾーンをブートします。	それぞれのゾーンをブートして稼働状態にします。	第 18 章「非大域ゾーンのインストール、停止処理、停止、アンインストール、クローニングについて (概要)」および第 19 章「非大域ゾーンのインストール、ブート、停止処理、停止、アンインストール、およびクローニング (タスク)」を参照してください。
この新しいゾーンを本稼働用に準備します。	ユーザーアカウントの作成、ソフトウェアの追加、およびゾーン構成のカスタマイズを行います。	新たにインストールしたマシンを設定するためのドキュメントを参照してください。ゾーンがインストールされているシステムに関連する特殊な考慮事項については、このガイドを参照してください。

◆ ◆ ◆ 第 16 章

16 非大域ゾーンの構成 (概要)

この章では、非大域ゾーンの構成の概要について説明します。

この章の内容は次のとおりです。

- 217 ページの「ゾーンのリソースについて」
- 218 ページの「インストール前の構成処理」
- 218 ページの「ゾーンのコンポーネント」
- 237 ページの「zonecfg コマンドの使用」
- 238 ページの「zonecfg のモード」
- 241 ページの「ゾーン構成データ」
- 259 ページの「Tecla コマンド行編集ライブラリ」

ゾーンの構成について学んだあとで、第 17 章「非大域ゾーンの計画と構成 (タスク)」に進み、システムにインストールする非大域ゾーンを構成します。

ゾーンのリソースについて

ゾーン内で制御できるリソースには、次のものがあります。

- リソースプールまたは割り当てられる CPU。マシンリソースの区分に使用されます。
- リソース制御機能。システムリソースに対する制約メカニズムを提供します。
- スケジューリングクラス。使用可能な CPU リソースのゾーン間での割り当てを、それらの重要性に基づいて制御できます。この重要性は、各ゾーンに割り当てる CPU リソースの「配分」で表します。

ゾーン管理での権利プロファイルと役割の使用

プロファイルと役割については、『Oracle Solaris 11 セキュリティーガイドライン』の「Oracle Solaris セキュリティーのテクノロジー」を参照してください。

インストール前の構成処理

システムに非大域ゾーンをインストールして使用する前に、そのゾーンを構成する必要があります。

`zonecfg` コマンドを使用すると、構成を作成したり、指定されたリソースやプロパティーが仮定のシステム上で有効かどうかを判定したりできます。特定の構成について `zonecfg` で実行される検査では、次のことが確認されます。

- ゾーンパスが指定されていること。
- 各リソースの必須プロパティーがすべて指定されていること。
- 構成に矛盾がないこと。たとえば、`anet` リソースが存在する場合、ゾーンは排他的 IP タイプであり、共有 IP ゾーンにはできません。また、別名が付いたデータセットでデバイスに関する潜在的競合が存在する場合、`zonecfg` コマンドは警告を出力します。

`zonecfg` コマンドの詳細は、`zonecfg(1M)` のマニュアルページを参照してください。

ゾーンのコンポーネント

このセクションでは、構成できる必須および省略可能なゾーンコンポーネントについて説明します。ゾーン名とゾーンのパスだけが必要です。詳細は、241 ページの「ゾーン構成データ」を参照してください。

ゾーンの名前とパス

ゾーンの名前とパスを選択する必要があります。ゾーンは、ZFS データセット上に存在する必要があります。ZFS データセットは、ゾーンのインストール時または接続時に自動的に作成されます。ZFS データセットを作成できない場合、ゾーンのインストールや接続は行われません。ゾーンのパスの親ディレクトリは、データセットでもなければならないことに注意してください。

ゾーンの自動ブート

`autoboot` プロパティーの設定は、大域ゾーンのブート時にゾーンが自動的にブートされるかどうかを決定します。ゾーンサービス `svc:/system/zones:default` も有効になっている必要があります。

読み取り専用ルートゾーンの file-mac-profile プロパティー

solaris ゾーンでは、file-mac-profile は読み取り専用ルートでゾーンを構成するために使用します。

詳細は、第 27 章「不変ゾーンの構成と管理」を参照してください。

admin リソース

admin 設定を使用すると、ゾーン管理の承認を設定できます。承認を定義するための推奨の方法は、zonecfg コマンドによる方法です。

user　ユーザー名を指定します。

auths　ユーザー名に対して承認を指定します。

solaris.zone.login	RBAC を使用中の場合、対話型ログインには、solaris.zone.login/*zonename* の承認が必要です。ゾーン内では、パスワード認証が実行されます。
solaris.zone.manage	RBAC を使用中の場合、非対話型ログイン、またはパスワード認証の省略のために solaris.zone.manage/ *zonename* の承認が必要です。
solaris.zone.clonefrom	RBAC を使用中の場合、別のゾーンのコピーを作成するサブコマンドは solaris.zone.clonefrom/ *source_zone* の承認を必要とします。

dedicated-cpu リソース

dedicated-cpu リソースは、非大域ゾーンの実行中にシステムのプロセッサの一部をそのゾーン専用に割り当てることを指定します。ゾーンのブート時に、ゾーンの実行中に使用される一時プールが動的に作成されます。

zonecfg で指定すると、移行時にプールの設定が伝達されます。

dedicated-cpu リソースは、ncpus の制限を設定し、必要に応じて importance も設定します。

ncpus
: CPU の数を指定するか、CPU の数の範囲を 2–4 などと指定します。リソースプールの動的な動作を得るために範囲を指定する場合は、次の手順も実行してください。

  - importance プロパティーを設定します。
  - poold サービスを有効にします。手順については、166 ページの「svcadm を使用して動的リソースプールサービスを有効にする方法」を参照してください。

importance
: 動的な動作を得るために CPU 範囲を使用する場合は、importance プロパティーも設定してください。importance は「省略可能な」プロパティーであり、プールの相対的な重要性を定義します。このプロパティーが必要となるのは、ncpus に範囲を指定した場合で、poold によって管理される動的リソースプールを使用しているときだけです。poold が実行されていない場合、importance は無視されます。poold が実行されている場合、importance が設定されていないと、importance はデフォルト値の 1 になります。詳細は、150 ページの「pool.importance プロパティーの制約」を参照してください。

注 – capped-cpu リソースと dedicated-cpu リソースには互換性がありません。cpu-shares リソース制御と dedicated-cpu リソースには互換性がありません。

---

capped-cpu リソース

capped-cpu リソースは、1 つのプロジェクトまたは 1 つのゾーンで消費可能な CPU リソース量に対して絶対的できめの細かい制限を設けます。プロセッサセットと組み合わせて使用すると、CPU キャップはセット内の CPU 使用率を制限します。capped-cpu リソースには、小数点第 2 位までの正の小数である 1 つの ncpus プロパティーがあります。このプロパティーは、CPU のユニット数に対応しています。このリソースには範囲を指定できません。このリソースには小数を指定できます。ncpus を指定する場合、1 の値は 1 つの CPU の 100% を意味します。1.25 の値は 125% を意味します。100% がシステム上の 1 つの CPU の上限となります。

注 – capped-cpu リソースと dedicated-cpu リソースには互換性がありません。

スケジューリングクラス

公平配分スケジューラ (FSS) を使用すると、使用可能な CPU リソースのゾーン間での割り当てを、ゾーンの重要性に基づいて制御できます。この重要性は、各ゾーンに割り当てる CPU リソースの「配分」で表します。CPU リソースのゾーン間での割

り当てを管理するために FSS を使用していない場合でも、ゾーン内のプロジェクトに配分を設定するために FSS を使用するよう、ゾーンのスケジューリングクラスを設定することができます。

cpu-shares プロパティーを明示的に設定すると、公平配分スケジューラ (FSS) はそのゾーンのスケジューリングクラスとして使用されます。ただし、この場合に望ましい FSS の使用方法は、dispadmin コマンドを使用して、FSS をシステムのデフォルトのスケジューリングクラスに設定する方法です。このようにすると、すべてのゾーンがシステムの CPU リソースの公平配分を受けることができます。ゾーンに対して cpu-shares が設定されていない場合、そのゾーンはシステムのデフォルトのスケジューリングクラスを使用します。ゾーンのスケジューリングクラスは、次の処理によって設定されます。

- zonecfg の scheduling-class プロパティーを使ってゾーンのスケジューリングクラスを設定できます。
- リソースプール機能を使ってゾーンのスケジューリングクラスを設定できます。ゾーンがプールに関連付けられている場合、そのプールの pool.scheduler プロパティーに有効なスケジューリングクラスが設定されていれば、ゾーンで実行されるプロセスは、デフォルトでそのスケジューリングクラスで実行されます。140 ページの「リソースプールの紹介」 および 172 ページの「プールをスケジューリングクラスに対応付ける方法」 を参照してください。
- cpu-shares リソース制御が設定されている場合で、別の処理を通して FSS がゾーンのスケジューリングクラスとして設定されていないときは、ゾーンのブート時に zoneadmd によってスケジューリングクラスが FSS に設定されます。
- ほかの処理を通してスケジューリングクラスが設定されていない場合、ゾーンはシステムのデフォルトのスケジューリングクラスを継承します。

priocntl (priocntl(1) のマニュアルページに記載) を使用すると、デフォルトのスケジューリングクラスの変更やリブートを行うことなく、実行中のプロセスを別のスケジューリングクラスに移動できます。

物理メモリーの制御と capped-memory リソース

capped-memory リソースは、physical、swap、および locked メモリーの制限を設定します。各制限はオプションですが、少なくとも 1 つは設定する必要があります。capped-memory リソースを使用するには、resource-cap パッケージが大域ゾーン内にインストールされている必要があります。

- 大域ゾーンから rcapd を使用してゾーンのメモリー上限を設定する場合は、このリソースの値を決定します。capped-memory リソースの physical プロパティーは、ゾーンの max-rss 値として rcapd で使用されます。
- capped-memory リソースの swap プロパティーは、zone.max-swap リソース制御を設定するための望ましい方法です。

- capped-memory リソースの locked プロパティーは、zone.max-locked-memory リソース制御を設定するための望ましい方法です。

注 – 通常はアプリケーションが多量のメモリーをロックすることはありませんが、ゾーンのアプリケーションによってメモリーがロックされることがわかっている場合は、ロックされるメモリーを設定するとよいでしょう。ゾーンの信頼が問題になる場合は、ロックされるメモリーの上限を、システムの物理メモリーの 10 パーセントまたはゾーンの物理メモリー上限の 10 パーセントに設定することもできます。

詳細は、第 10 章「リソース上限デーモンによる物理メモリーの制御 (概要)」、第 11 章「リソース上限デーモンの管理 (タスク)」、および 270 ページの「ゾーンの構成方法」を参照してください。ゾーンに一時的なリソース上限を設定する方法については、135 ページの「ゾーンに一時的なリソース上限を指定する方法」を参照してください。

rootzpool リソース

zonecfg ユーティリティーに含まれるオプションの rootzpool リソースは、ゾーンのインストール専用の ZFS zpool を作成するために使用します。ゾーンのルート ZFS zpool は、1 つ以上の URI (Universal Resource Identifier) によって定義された共有ストレージデバイスでホストできます。必須の storage プロパティーは、ゾーンのルート zfs ファイルシステムを格納するストレージオブジェクトの URI を識別します。特定のゾーンに対して定義できる rootzpool は 1 つだけです。このストレージは、ゾーンがブートされたときに、そのゾーンのために自動的に構成されます。

対応する zpool は、ゾーンのインストールまたはゾーンの接続操作中に自動的に作成またはインポートされます。ゾーンのアンインストールまたは切り離しを行うと、次のアクションが実行されます。

- ストレージリソースが自動的に構成解除されます。
- 対応する zpool が自動的にエクスポートまたは破棄されます。

事前作成済みの zpool をゾーンのインストールに再利用するには、zpool をシステムからエクスポートする必要があります。

ゾーンのフレームワークでは、次の URI タイプがサポートされています。

- dev

  ローカルデバイスパスの URI

  書式:

```
dev:local-path-under-/dev
dev://absolute-path-with-dev
dev:absolute-path-with-dev
```

例:

```
dev:dsk/c7t0d0s0
dev:///dev/dsk/c7t0d0s0
dev:/dev/dsk/c7t0d0s0
```

- `lu` (論理ユニット)

  ファイバチャネル (FC) および Serial Attached SCSI (SAS)

  書式:

  ```
  lu:luname.naa.ID
  lu:initiator.naa.ID,target.naa.ID,luname.naa.ID
  ```

  例:

```
lu:luname.naa.5000c5000288fa25
lu:initiator.naa.2100001d38089fb0,target.naa.2100001d38089fb0,luname.naa.5000c5000288fa25
```

- `iscsi`

  iSCSI の URI

  書式:

  ```
  iscsi:///luname.naaID
  iscsi://host[:port]/luname.naa.ID
  ```

  例:

  ```
  iscsi:///luname.naa.600144f03d70c80000004ea57da10001
  iscsi://[::1]/luname.naa.600144f03d70c80000004ea57da10001
  iscsi://127.0.0.1/luname.naa.600144f03d70c80000004ea57da10001
  iscsi://127.0.0.1:3620/luname.naa.600144f03d70c80000004ea57da10001
  iscsi://hostname:3620/luname.naa.600144f03d70c80000004ea57da10001
  ```

ストレージの URI に基づいて共有オブジェクトを管理するには、`suriadm` ツールを使用します。ID、NAA (Name Address Authority)、および既存のストレージオブジェクトの URI の取得については、`suriadm(1M)` および `suri(5)` のマニュアルページを参照してください。

システムは、関連するゾーンに対して新規作成またはインポートされた `rootzpool` に名前を割り当てます。割り当てられる名前は、*zonename*`_rpool` という形式になります。

`storage` プロパティーは、`rootzpool` リソーススコープ内から次のコマンドを使用して管理されます。

- `add storage` *URI string*
- `remove storage` *URI string*

zpool リソースの自動追加

zpool を非大域ゾーンに委任するには、zonecfg ユーティリティーでオプションの zpool リソースを構成します。zpool は、ゾーンがブートされたときに、そのゾーンのために自動的に構成されます。

対応する zpool は、ゾーンのインストールまたはゾーンの接続操作中に自動的に作成またはインポートされます。

ゾーンのアンインストールまたは切り離しを行うと、次のアクションが実行されます。

- ストレージリソースが自動的に構成解除されます。
- 対応する zpool が自動的にエクスポートまたは破棄されます。

必須の storage プロパティーは、このリソースに関連付けられたストレージオブジェクトの URI を識別します。

storage プロパティーは、zpool リソーススコープ内の次の設定を使用して管理されます。

- add storage *URI string*
- remove storage *URI string*

zpool リソースには name プロパティーが必須です。このプロパティーは、ゾーンに委任される zpool の名前で使用されます。ZFS ファイルシステムの name コンポーネントにスラッシュ (/) を含めることはできません。

新規作成またはインポートされた zpool に割り当てられる名前は、*zonename_name* という形式になります。これは、非大域ゾーン内で表示可能な zpool 名です。

注 – ストレージオブジェクトに既存のパーティション、zpool、または UFS ファイルシステムが含まれている場合は、ゾーンのインストールが失敗することがあります。詳細は、300 ページの「構成済みのゾーンをインストールする方法」の手順 4 を参照してください。

ゾーンネットワークインタフェース

ネットワーク接続を提供するために zonecfg ユーティリティーによって構成されるゾーンネットワークインタフェースは、ゾーンのブート時に自動的に設定されてゾーン内に配置されます。

インターネットプロトコル (IP) 層は、ネットワークのパケットの受信と配信を行います。この層には、IP ルーティング、アドレス解決プロトコル (ARP)、IP セキュリティーアーキテクチャー (IPsec)、および IP フィルタが含まれます。

非大域ゾーンに使用できる IP タイプには、共有 IP と排他的 IP の 2 種類があります。排他的 IP がデフォルトの IP タイプです。共有 IP ゾーンは、大域ゾーンとネットワークインタフェースを共有します。大域ゾーン内の構成は、共有 IP ゾーンを使用するために `ipadm` ユーティリティーによって実行する必要があります。排他的 IP ゾーンには専用のネットワークインタフェースが必要です。`anet` リソースを使用して排他的 IP ゾーンを構成すると、専用の VNIC が自動的に作成され、そのゾーンに割り当てられます。自動化された `anet` リソースを使用することによって、大域ゾーン内にデータリンクを作成および構成し、非大域ゾーンにデータリンクを割り当てる必要がなくなります。次のことを行うには、`anet` リソースを使用します。

- 大域ゾーンの管理者が、非大域ゾーンに割り当てられたデータリンク用の特定の名前を選択できるようにする
- 複数のゾーンで同じ名前のデータリンクを使用できるようにする

下位互換性を維持するために、事前構成済みのデータリンクを非大域ゾーンに割り当てることができます。

各タイプの IP 機能については、369 ページの「共有 IP 非大域ゾーンにおけるネットワーク」および 371 ページの「排他的 IP 非大域ゾーンにおけるネットワーク」を参照してください。

注 – リンク保護 (『Oracle Solaris Administration: Network Interfaces and Network Virtualization』の第 20 章「Using Link Protection in Virtualized Environments」に記載) が、ゾーンを実行するシステム上で使用できます。この機能は、大域ゾーン内で構成されます。

データリンクについて

データリンクとは、OSI プロトコルスタックのレイヤー 2 インタフェースのことです。このインタフェースは、システム内で STREAMS DLPI (v2) インタフェースとして表されます。このようなインタフェースは、TCP/IP などのプロトコルスタックの下で接続できます。データリンクは、「物理インタフェース」 (たとえば、ネットワークインタフェースカード (NIC)) とも呼ばれます。データリンクは、`zonecfg`(1M) を使用して構成される `physical` プロパティーです。`physical` プロパティーは、VNIC (『Oracle Solaris Administration: Network Interfaces and Network Virtualization』のパート III「Network Virtualization and Resource Management」に記載) にすることも可能です。

データリンクの例としては、物理インタフェース (e1000g0 や bge1 など)、NIC (bge3 など)、アグリゲーション (aggr1、aggr2 など)、VLAN タグ付きインタフェース (e1000g123000 と bge234003 (それぞれ e1000g0 上の VLAN 123、および bge3 上の VLAN 234) など) があります。

IP over Infiniband (IPoIB) の使用については、247 ページの「リソースタイプのプロパティー」の anet の説明を参照してください。

共有 IP 非大域ゾーン

共有 IP ゾーンは、大域ゾーンからの既存の IP インタフェースを使用します。ゾーンには、1 つ以上の専用の IP アドレスが存在する必要があります。共有 IP ゾーンは、IP 層の構成と状態を大域ゾーンと共有します。次の両方の条件が満たされる場合、ゾーンは共有 IP インスタンスを使用すべきです。

- 大域ゾーンと非大域ゾーンが同じサブネット上に存在するかどうかに関係なく、非大域ゾーンが大域ゾーンによって使用されるデータリンクと同じデータリンクを使用する計画である。
- 排他的 IP ゾーンによって提供されるその他の機能は必要でない。

共有 IP ゾーンには、zonecfg コマンドの net リソースを使用して、1 つ以上の IP アドレスを割り当てます。大域ゾーンでデータリンク名も構成する必要があります。

zonecfg net リソースで、address と physical のプロパティーを設定する必要があります。defrouter プロパティーはオプションです。

大域ゾーン内で共有 IP タイプのネットワーク接続構成を使用するには、自動ネットワーク構成ではなく ipadm を使用する必要があります。ネットワーク接続構成を ipadm によって行うかどうかを決定するには、次のコマンドを実行します。表示される応答は、DefaultFixed でなければなりません。

```
# svcprop -p netcfg/active_ncp svc:/network/physical:default
DefaultFixed
```

共有 IP ゾーンに割り当てられる IP アドレスは、論理ネットワークインタフェースに関連付けられます。

大域ゾーンから ipadm コマンドを使用すると、稼働中のゾーンの論理インタフェースの割り当てまたは削除を行うことができます。

インタフェースを追加するには、次のコマンドを使用します。

```
global# ipadm set-addrprop -p zone=my-zone net0/addr1
```

インタフェースを削除するには、次のコマンドのいずれかを使用します。

```
global# ipadm set-addrprop -p zone=global net0/addr
```

または

```
global# ipadm reset-addrprop -p zone net0/addr1
```

詳細は、370 ページの「共有 IP ネットワークインタフェース」を参照してください。

排他的 IP 非大域ゾーン

排他的 IP は、非大域ゾーンのデフォルトのネットワーク接続構成です。

排他的 IP ゾーンには、独自の IP に関連付けられた状態と 1 つ以上の専用のデータリンクが保持されます。

排他的 IP ゾーン内では、次の各機能を使用できます。

- DHCPv4 および IPv6 ステートレスアドレスの自動構成
- IP フィルタ。ネットワークアドレス変換 (NAT) 機能も含む
- IP ネットワークマルチパス (IPMP)
- IP ルーティング
- TCP/UDP/SCTP および IP/ARP レベルのチューニング可能パラメータを設定するための `ipadm`
- IP セキュリティー (IPsec) と Internet Key Exchange (IKE)。これは、IPsec セキュリティーアソシエーション用の認証済み鍵材料のプロビジョニングを自動化する

排他的 IP ゾーンを構成する方法には、次の 2 つがあります。

- `zonecfg` ユーティリティーの `anet` リソースを使用して、ゾーンのブート時にゾーンに対して一時的な VNIC を自動的に作成し、またゾーンの停止時にそれを削除します。
- 大域ゾーン内にデータリンクを事前構成し、排他的 IP ゾーンにそれを割り当てます。これは、`zonecfg` ユーティリティーの `net` リソースを使用して行います。データリンクは、`net` リソースの `physical` プロパティーを使用して指定します。`physical` プロパティーは、VNIC (『Oracle Solaris Administration: Network Interfaces and Network Virtualization』のパート III「Network Virtualization and Resource Management」に記載) にすることも可能です。`net` リソースの `address` プロパティーは設定されません。

排他的 IP ゾーンは、デフォルトでは関連付けられたインタフェース上のすべての IP アドレスを構成および使用できます。必要に応じて、`allowed-address` プロパティーを使用して IP アドレスのコンマ区切りリストを指定することもできます。排他的 IP ゾーンは、`allowed-address` リスト内にない IP アドレスを使用することはできません。また、ゾーンのブート時に、`allowed-address` リスト内のすべてのアドレ

スが排他的 IP ゾーンに対して自動で持続的に構成されます。このインタフェース構成が不要な場合、 configure-allowed-address プロパティーを false に設定する必要があります。デフォルト値は true です。

データリンクを割り当てると snoop コマンドが使用可能になります。

dladm コマンドを show-linkprop サブコマンドとともに使用して、実行中の排他的 IP ゾーンに対するデータリンクの割り当てを表示できます。dladm コマンドを set-linkprop サブコマンドとともに使用すると、実行中のゾーンに対して追加のデータリンクを割り当てることができます。使用例については、410 ページの「排他的 IP 非大域ゾーンでのデータリンクの管理」を参照してください。

独自のデータリンクセットが割り当てられた、実行中の排他的 IP ゾーン内では、ipadm コマンドを使用して IP を構成できます。このコマンドには、論理インタフェースの追加または削除を行う機能が備わっています。ゾーンの IP 構成は、大域ゾーン内の場合と同様に sysconfig インタフェース (sysconfig(1M) のマニュアルページに記載) を使用して設定できます。

排他的 IP ゾーンの IP 構成は、大域ゾーンから zlogin コマンドを使用することによってのみ表示できます。

```
global# zlogin zone1 ipadm show-addr
ADDROBJ           TYPE     STATE        ADDR
lo0/v4            static   ok           127.0.0.1/8
nge0/_b           dhcp     ok           10.134.62.47/24
lo0/v6            static   ok           ::1/128
nge0/_a           addrconf ok           fe80::2e0:81ff:fe5d:c630/10
```

非大域ゾーンでの Reliable Datagram Sockets のサポート

RDS (Reliable Datagram Sockets) IPC プロトコルは、排他的 IP 非大域ゾーンと共有 IP 非大域ゾーンの両方でサポートされています。RDSv3 ドライバは、SMF サービス rds として有効になっています。デフォルトでは、このサービスはインストール後に無効になります。このサービスは、適切な承認を付与されたゾーン管理者が特定の非大域ゾーン内で有効にすることができます。zlogin のあとで、rds をそれが実行される各ゾーン内で有効にすることができます。

例 16–1 非大域ゾーン内で rds サービスを有効にする方法

1. 排他的 IP ゾーンまたは共有 IP ゾーン内で RDSv3 サービスを有効にするには、zlogin して svcadm enable コマンドを実行します。

   ```
   # svcadm enable rds
   ```

2. rds が有効になっていることを確認します。

   ```
   # svcs rds
       STATE          STIME    FMRI
       online         22:50:53 svc:/system/rds:default
   ```

詳細は、svcadm(1M) のマニュアルページを参照してください。

共有 IP 非大域ゾーンと排他的 IP 非大域ゾーンのセキュリティーの相違

共有 IP ゾーン内のアプリケーションは、スーパーユーザーも含め、zonecfg ユーティリティーを介してゾーンに割り当てられた IP アドレス以外をソース IP アドレスとしてパケットを送信することはできません。このタイプのゾーンには、任意のデータリンク (レイヤー 2) パケットを送受信するアクセス権はありません。

一方、排他的 IP ゾーンの場合は、zonecfg によって指定されたデータリンク全体がゾーンに対して許可されます。結果として、排他的 IP ゾーン内では、スーパーユーザーまたは必要な権利プロファイルを持つユーザーは、大域ゾーン内と同様に、それらのデータリンク上でなりすましのパケットを送信できます。IP アドレスのなりすましは、allowed-address プロパティーを設定することによって使用不可にできます。anet リソースの場合、link-protection プロパティーを設定することによって、mac-nospoof や dhcp-nospoof などの追加の保護を有効にできます。

共有 IP 非大域ゾーンと排他的 IP 非大域ゾーンの同時使用

共有 IP ゾーンは常に IP 層を大域ゾーンと共有し、排他的 IP ゾーンは常に独自の IP 層インスタンスを持っています。共有 IP ゾーンと排他的 IP ゾーンの両方を同じマシンで使用することができます。

ゾーンでマウントされるファイルシステム

各ゾーンには、デフォルトでそのゾーンに委任された ZFS データセットが保持されています。このデフォルトの委任されたデータセットは、デフォルトの大域ゾーンのデータセットレイアウトのデータセットレイアウトを模倣します。 .../rpool/ROOT というデータセットは、ブート環境を含んでいます。このデータセットは、直接操作しないでください。存在する必要がある rpool データセットは、デフォルトで .../rpool にマウントされます。.../rpool/export、および .../rpool/export/home データセットは /export と /export/home にマウントされます。これらの非大域ゾーンのデータセットは、対応する大域ゾーンのデータセットと使用方法が同じで、同様に管理できます。ゾーン管理者は、.../rpool, .../rpool/export および .../rpool/export/home のデータセット内に追加のデータセットを作成できます。

zfs(1M) のマニュアルページで説明されている zfs コマンドを使用して、ゾーンの rpool/ROOT ファイルシステムから始まる階層内でファイルシステムの作成、削除、または名前変更を行うべきではありません。zfs コマンドは、canmount、mountpoint、sharesmb、zoned、com.oracle.*:*、com.sun:*、および org.opensolaris.*.* 以外のプロパティーを設定する場合に使用できます。

通常、ゾーンでマウントされるファイルシステムには、次のものが含まれます。

- 仮想プラットフォームの初期化時にマウントされる一連のファイルシステム
- アプリケーション環境自体の内部からマウントされる一連のファイルシステム

これらのセットには、たとえば次のようなファイルシステムが含まれます。

- none または legacy 以外の mountpoint があり、また canmount プロパティーに yes の値が保持された ZFS ファイルシステム。
- ゾーンの /etc/vfstab ファイルで指定されたファイルシステム。
- AutoFS によるマウントおよび AutoFS によってトリガーされるマウント。autofs プロパティーは、sharectl (sharectl(1M) に記載) を使用して設定します。
- ゾーン管理者が明示的に実行するマウント

  また、稼働中のゾーン内でのファイルシステムのマウント権限は、zonecfg fs-allowed プロパティーで定義します。このプロパティーは、zonecfg add fs リソースまたは add dataset リソースを使用してゾーンにマウントされたファイルシステムには適用されません。デフォルトでは、ゾーンのデフォルトの委任されたデータセットである hsfs ファイルシステム内でのファイルシステム、および NFS などのネットワークファイルシステムのマウントだけが、ゾーン内で許可されます。

注意 – アプリケーション環境内部から実行される、デフォルト以外のマウントには、いくつかの制限事項があります。これらの制限事項は、ほかのゾーンに悪影響を与えないようにするために、ゾーン管理者がシステムのほかの部分に対するサービスを拒否できないようにします。

一部のファイルシステムについては、ゾーン内部からマウントする場合にセキュリティー制限があります。ほかのファイルシステムは、ゾーン内でマウントされたときに特有の動作を行います。詳細は、361 ページの「ファイルシステムと非大域ゾーン」を参照してください。

データセットの詳細は、datasets(5) のマニュアルページを参照してください。BE の詳細は、『Oracle Solaris 11.1 ブート環境の作成と管理』を参照してください。

ファイルシステムのマウントと更新

pkg(5) のマニュアルページで説明されているように、ゾーンのシステムイメージの一部であるファイル、シンボリックリンク、またはディレクトリを非表示にしてファイルシステムをマウントする方法はサポートされていません。/usr/local に内容を提供するパッケージがインストールされていない場合は、/usr/local にファイルシステムをマウントすることが許容されます。しかし、いずれかのパッケージ

(SVR4 パッケージを含む) が /usr/local で始まるパスにファイル、ディレクトリ、またはシンボリックリンクを提供する場合は、/usr/local へのファイルシステムのマウントはサポートされません。/mnt へのファイルシステムの一時的なマウントはサポートされています。

ゾーン内でファイルシステムがマウントされる順序のため、/export がゾーンの rpool/export データセットまたは別の委任されたデータセットに由来する場合は、fs リソースが /export/filesys にファイルシステムをマウントすることはできません。

ゾーン内のホスト ID

非大域ゾーンでは、大域ゾーンの hostid とは異なる hostid プロパティーを設定できます。これはたとえば、別のシステム上のゾーンに移行されたマシンなどの場合に行います。このゾーン内に現在あるアプリケーションは、元の hostid に依存している場合があります。詳細は、241 ページの「リソースタイプとプロパティー」を参照してください。

非大域ゾーンの /dev ファイルシステム

zonecfg コマンドは、規則照合方式を使って、特定のゾーンにどのデバイスを配置するかを指定します。いずれかのルールに一致するデバイスは、ゾーンの /dev ファイルシステムに追加されます。詳細は、270 ページの「ゾーンの構成方法」を参照してください。

非大域ゾーンのリムーバブル lofi デバイス

リムーバブルループバックファイル (lofi) デバイスは、CD-ROM デバイスのように機能し、非大域ゾーンに構成できます。デバイスがマップするファイルを変更したり、複数の lofi デバイスを作成して、同じファイルを読み取り専用モードで使用したりできます。このタイプの lofi デバイスを作成するには、lofiadm コマンドを -r オプションとともに使用します。作成時にファイル名は必要ありません。リムーバブル lofi デバイスのライフサイクル中は、ファイルを空のデバイスに関連付けたり、空でないデバイスからファイルの関連付けを解除したりできます。1 つのファイルを同時に複数のリムーバブル lofi デバイスに安全に関連付けることができます。リムーバブル lofi デバイスは読み取り専用です。通常の読み取り/書き込み lofi デバイスまたはリムーバブル lofi デバイスにマップされたファイルを再マッピングすることはできません。潜在的な lofi デバイスの数は、大域ゾーンで zonecfg(1M) を使用して設定できる zone.max-lofi リソース制御によって制限されます。

作成後のリムーバブル lofi デバイスは読み取り専用です。リムーバブル lofi デバイスに対して書き込み操作が行われると、lofi ドライバはエラーを返します。

lofiadm コマンドは、リムーバブル lofi デバイスを一覧表示する場合にも使用します。

例 16–2 関連するファイルを含むリムーバブル lofi デバイスを作成する

```
# lofiadm -r /path/to/file
/dev/lofi/1
```

例 16–3 空のリムーバブル lofi デバイスを作成する

```
# lofiadm -r
/dev/lofi/2
```

例 16–4 リムーバブル lofi デバイスにファイルを挿入する

```
# lofiadm -r /path/to/file /dev/lofi/1
/dev/lofi/1
```

詳細は、lofiadm(1M)、zonecfg(1M)、および lofi(7D) のマニュアルページを参照してください。81 ページの「ゾーン規模のリソース制御」も参照してください。

非大域ゾーン内でのディスク形式のサポート

ディスクのパーティション分割と uscsi コマンドの使用は、zonecfg ツールを使用すると有効になります。例については、247 ページの「リソースタイプのプロパティー」の device を参照してください。uscsi コマンドの詳細は、uscsi(7I) を参照してください。

- 委任は、solaris ゾーンでサポートされるだけです。
- ディスクは、-D オプションを指定した prtconf コマンドを使用して表示される sd ターゲットを使用する必要があります。prtconf(1M)を参照してください。

構成可能な特権

ゾーンのブート時に、*safe* 特権のデフォルトセットが構成に含められます。これらの特権は、ゾーン内の特権プロセスがシステムのほかの非大域ゾーン内のプロセスや大域ゾーン内のプロセスに影響を及ぼすことを防ぐため、安全と見なされます。zonecfg コマンドを使用して、次の操作を実行できます。

- デフォルトの特権セットに追加します。ただし、この種の変更を行うと、あるゾーン内のプロセスがグローバルリソースを制御できるようになって、ほかのゾーン内のプロセスに影響する場合があります。

- デフォルトの特権セットから削除します。ただし、この種の変更を行うと、実行に必要な特権がないため一部のプロセスが正しく動作しなくなる場合があります。

注 – わずかですが、この時点でゾーンのデフォルト特権セットから削除できない特権があります。同じように、特権セットに追加できない特権もあります。

詳細は、378 ページの「非大域ゾーン内の特権」、270 ページの「ゾーンの構成方法」、および privileges(5) のマニュアルページを参照してください。

リソースプールの関連付け

第 13 章「リソースプールの作成と管理 (タスク)」の説明に従ってシステムでリソースプールを構成した場合は、ゾーンを構成するときに pool プロパティーを使用して、リソースプールの 1 つにゾーンを関連付けることができます。

リソースプールが構成されていない場合でも、非大域ゾーンの実行中にシステムのプロセッサの一部をそのゾーン専用に割り当てるよう、dedicated-cpu リソースを使用して指定できます。ゾーンの実行中に使用される一時プールが動的に作成されます。zonecfg によって指定すると、移行時にプールの設定が伝達されます。

注 – pool プロパティーによって設定される持続的プールを使用するゾーン構成と、dedicated-cpu リソースによって構成される一時プールには、互換性がありません。これら 2 つのプロパティーは、どちらか 1 つしか設定できません。

ゾーン規模のリソース制御の設定

大域管理者または適切な承認を持つユーザーは、ゾーン規模の特権付きリソース制御をゾーンに対して設定できます。ゾーン規模のリソース制御は、ゾーン内のすべてのプロセスエンティティーによる総リソース消費を制限します。

これらの制限は、大域ゾーンと非大域ゾーンのどちらに対しても、zonecfg コマンドを使用して指定します。270 ページの「ゾーンの構成方法」を参照してください。

ゾーン規模のリソース制御を設定するためのより簡単な推奨される方法は、cpu-cap などの rctl リソースではなく、capped-cpu などのプロパティー名またはリソースを使用することです。

zone.cpu-cap リソース制御は、1 つのゾーンで消費可能な CPU リソースの量に対する絶対的な制限を設定します。設定と同様、100 の値は 1 つの CPU の 100% を意味します。125 の値は 125% になります。CPU キャップの使用時は、100% がシステム上の 1 つの CPU の上限となります。

注 – capped-cpu リソースを設定する場合は、単位に小数を使用できます。この値は zone.cpu-cap リソース制御と相互に関連していますが、設定値はその 100 分の 1 になります。設定値 1 はリソース制御の設定値 100 に等しくなります。

---

zone.cpu-shares リソース制御は、公平配分スケジューラ (FSS) の CPU 配分の制限をゾーンに対して設定します。CPU 配分は、まずゾーンに対して割り当てられたあとで、project.cpu-shares エントリの指定に従って、ゾーン内のプロジェクトに分配されます。詳細は、413 ページの「ゾーンがインストールされている Oracle Solaris システムでの公平配分スケジューラの使用」を参照してください。この制御のグローバルプロパティー名は cpu-shares です。

zone.max-locked-memory リソース制御は、1 つのゾーンで使用できる、ロックされた物理メモリーの量に制限を設定します。ゾーン内のプロジェクト間でのロックされたメモリーリソースの割り当ては、project.max-locked-memory リソース制御を使用して制御できます。詳細は、表 6–1 を参照してください。

zone.max-lofi リソース制御は、1 つのゾーンで作成可能な lofi デバイスの数を制限します。

1 つのゾーンの LWP が多くなりすぎると、ほかのゾーンに影響を与えることがあります。zone.max-lwps リソース制御は、これを防ぐことで、リソースの隔離を向上させます。ゾーン内のプロジェクト間での LWP リソースの割り当ては、project.max-lwps リソース制御を使用して制御できます。詳細は、表 6–1 を参照してください。この制御のグローバルプロパティー名は max-lwps です。

zone.max-processes リソース制御は、1 つのゾーン内のあまりに多くの LWP が他のゾーンに影響を与えることを防ぐことによって、リソースの隔離性を高めます。ゾーン内の各プロジェクトに対するプロセステーブルスロットリソースの割り当ては、project.max-processes リソース制御 (78 ページの「使用可能なリソース制御」に記載) を使用して設定できます。この制御のグローバルプロパティー名は max-processes です。zone.max-processes リソース制御は、zone.max-lwps リソース制御を含むこともできます。zone.max-processes が設定されていて zone.max-lwps が設定されていない場合、zone.max-lwps はゾーンのブート時に暗黙的に zone.max-processes の値の 10 倍に設定されます。正常なプロセスとゾンビプロセスの両方がプロセステーブルスロットを占有するため、max-processes 制御がプロセステーブルを消費しつくすゾンビプロセスから保護します。ゾンビプロセスには本来 LWP がないため、max-lwps でこの可能性から保護することはできません。

zone.max-msg-ids、zone.max-sem-ids、zone.max-shm-ids、および zone.max-shm-memory の各リソース制御は、ゾーン内のすべてのプロセスで使用される System V リソースを制限します。ゾーン内のプロジェクト間での System V リソースの割り当ては、これらのリソース制御の project バージョンを使用して制御できます。これらの制御のグローバルプロパティー名は、max-msg-ids、max-sem-ids、max-shm-ids、および max-shm-memory です。

zone.max-swap リソース制御は、ゾーン内のユーザープロセスのアドレス空間マッピングと tmpfs マウントで消費されるスワップを制限します。prstat -Z の出力は「スワップ」列を表示します。報告されるスワップは、ゾーンのプロセスと tmpfs マウントで消費されるスワップの合計量です。この値により、各ゾーンで予約されているスワップを監視しやすくなり、適切な zone.max-swap 設定を選択することができます。

表 16–1 ゾーン規模のリソース制御

制御名	グローバルプロパティー名	説明	デフォルトの単位	使用される値
zone.cpu-cap		このゾーンに対する CPU リソース量の絶対的な制限。	数量 (CPU の数)、パーセントで表されます。 注 – capped-cpu リソースとして設定する場合は、単位に小数を使用できます。	
zone.cpu-shares	cpu-shares	このゾーンに対する公平配分スケジューラ (FSS) の CPU 配分	数量 (配分)	

表 16–1 ゾーン規模のリソース制御 (続き)

制御名	グローバルプロパティー名	説明	デフォルトの単位	使用される値
zone.max-locked-memory		ゾーンで使用できるロックされた物理メモリーの合計量。 priv_proc_lock_memory がゾーンに割り当てられている場合、そのゾーンがすべてのメモリーをロックするのを防ぐため、このリソース制御の設定も検討してください。	サイズ (バイト)	capped-memory の locked プロパティー
zone.max-lofi	max-lofi	1 つのゾーンで作成可能な lofi デバイスの数に対する制限	数量 (lofi デバイスの数)	
zone.max-lwps	max-lwps	このゾーンで同時に使用できる LWP の最大数	数量 (LWP 数)	
zone.max-msg-ids	max-msg-ids	このゾーンに許容されるメッセージキュー ID の最大数	数量 (メッセージキュー ID の数)	
zone.max-processes	max-processes	このゾーンで同時に使用できるプロセステーブルスロットの最大数	数量 (プロセステーブルスロット数)	
zone.max-sem-ids	max-sem-ids	このゾーンに許容されるセマフォー ID の最大数	数量 (セマフォー ID の数)	
zone.max-shm-ids	max-shm-ids	このゾーンに許容される共有メモリー ID の最大数	数量 (共有メモリー ID の数)	
zone.max-shm-memory	max-shm-memory	このゾーンに許容される System V 共有メモリーの合計量	サイズ (バイト)	

表 16–1 ゾーン規模のリソース制御 (続き)

制御名	グローバルプロパティー名	説明	デフォルトの単位	使用される値
zone.max-swap		このゾーンのユーザープロセスのアドレス空間マッピングと tmpfs マウントで消費できるスワップの合計量。	サイズ (バイト)	capped-memory の swap プロパティー

prctl コマンドを使用すると、実行中のプロセスに対してこれらの制限を指定できます。例については、413 ページの「prctl コマンドを使用して大域ゾーンの FSS 配分を設定する方法」を参照してください。prctl コマンドで指定された制限には持続性がありません。システムがリブートされると、制限は無効になります。

ゾーンのコメントの追加

attr リソースの型を使ってゾーンのコメントを追加できます。詳細は、270 ページの「ゾーンの構成方法」を参照してください。

zonecfg コマンドの使用

zonecfg コマンド (zonecfg(1M) のマニュアルページに記載) は、非大域ゾーンを構成するために使用します。

zonecfg コマンドは、大域ゾーンのリソース管理設定を持続的に指定するためにも使用できます。たとえば、このコマンドを使用して、dedicated-cpu リソースを使って専用の CPU を使用するように大域ゾーンを構成できます。

zonecfg コマンドは、対話型モード、コマンド行モード、またはコマンドファイルモードで使用できます。このコマンドを使用して、次の操作を実行できます。

- ゾーン構成を作成または削除 (破棄) します
- 特定の構成にリソースを追加します
- 構成に追加したリソースのプロパティーを設定します
- 特定の構成からリソースを削除します
- 構成の照会または確認を行います
- 構成を確定します
- 前の構成に戻します
- ゾーンの名前を変更します
- zonecfg のセッションを終了します

zonecfg のプロンプトは次のような形式です。

zonecfg:*zonename*>

ファイルシステムなど、特定のリソースタイプの構成を行うときは、そのリソースタイプもプロンプトに表示されます。

zonecfg:*zonename*:fs>

この章で説明する zonecfg のさまざまなコンポーネントの使用方法を示す手順など、詳細については、第 17 章「非大域ゾーンの計画と構成 (タスク)」で説明しています。

zonecfg のモード

このユーザーインタフェースでは「有効範囲」という概念が使用されます。有効範囲は、「大域」または「リソース固有」のどちらかです。デフォルトの有効範囲は大域です。

大域有効範囲で add サブコマンドまたは select サブコマンドを使用すると、特定のリソースが選択されます。すると、有効範囲がそのリソースタイプに変わります。

- add サブコマンドの場合、end、cancel のいずれかのサブコマンドを使用すると、リソースの指定が完了します。
- select サブコマンドの場合、end、cancel のいずれかのサブコマンドを使用すると、リソースの変更が完了します。

すると、有効範囲が大域に戻ります。

add、remove、set などのように、有効範囲によって異なる意味を持つサブコマンドもあります。

zonecfg の対話型モード

対話型モードでは、次のサブコマンドがサポートされます。サブコマンドで使用する意味とオプションの詳細については、zonecfg(1M) のマニュアルページを参照してください。破壊的な操作や作業内容の消失を伴うようなサブコマンドの場合、処理を実行する前にユーザーの確認が求められます。-F (強制) オプションを使用すると、この確認手順を省略できます。

help　一般ヘルプまたは特定のリソースに関するヘルプを表示します。

```
zonecfg:my-zone:capped-cpu> help
```

create　指定された新しいゾーンに使用するメモリー内構成の構成を開始します。次のような目的に使用されます。

- Oracle Solaris のデフォルト設定を新しい構成に適用します。この方法がデフォルトです。

- -t *template* オプションを使用して、指定したテンプレートと同一の構成を作成します。ゾーン名がテンプレート名から新しいゾーン名に変更されます。
- -F オプションを使用して、既存の構成を上書きします。
- -b オプションを使用して、なにも設定されていない空の構成を作成します。

`export` 標準出力または指定された出力ファイルに、コマンドファイルに使用できる形式で構成を出力します。

`add` 大域有効範囲では、指定されたリソースタイプを構成に追加します。

リソース固有の有効範囲では、指定された名前と値を持つプロパティーを追加します。

詳細は、270 ページの「ゾーンの構成方法」および zonecfg(1M) のマニュアルページを参照してください。

`set` 指定されたプロパティー名を、指定されたプロパティー値に設定します。zonepath などの大域的なプロパティーと、リソース固有のプロパティーがあることに注意してください。このコマンドは、大域有効範囲とリソース固有の有効範囲の両方で使用できます。

`select` 大域有効範囲でのみ使用できます。指定されたタイプのリソースのうち、指定されたプロパティー名とプロパティー値の対の条件に一致するものを、変更対象として選択します。有効範囲がそのリソースタイプに変わります。リソースが一意に識別されるように、プロパティーの名前と値の対を十分な数だけ指定する必要があります。

`clear` 省略可能な設定の値をクリアーします。必須の設定はクリアーできません。ただし、必須の設定のいくつかは、新しい値を割り当てることによって変更できます。

`remove` 大域有効範囲では、指定されたリソースタイプを削除します。リソースタイプが一意に識別されるように、プロパティーの名前と値の対を十分な数だけ指定する必要があります。プロパティーの名前と値の対をまったく指定しないと、すべてのインスタンスが削除されます。該当するものが複数ある場合は、-F オプションを使用していない限り、確認を求めるメッセージが表示されます。

リソース固有の有効範囲では、指定された名前と値を持つプロパティーを現在のリソースから削除します。

`end` リソース固有の有効範囲でのみ使用できます。リソースの指定を終了します。

次に、zonecfg コマンドは、現在のリソースが正しく指定されているかどうかを確認します。

- リソースが正しく指定されている場合は、そのリソースがメモリー内に保持される構成に追加され、有効範囲が大域に戻ります。
- 指定が不完全な場合は、必要な作業を示すエラーメッセージが表示されます。

`cancel` リソース固有の有効範囲でのみ使用できます。リソースの指定を終了し、有効範囲を大域に戻します。リソースの指定が不完全な場合、そのリソースは保持されません。

`delete` 指定された構成を破棄します。メモリーと安定した記憶領域の両方から構成を削除します。`delete` に `-F` (強制) オプションを使用する必要があります。

注意 – この操作は即時に実行されます。確定手順は行われず、削除されたゾーンを元に戻すことはできません。

`info` 現在の構成または大域のリソースプロパティー `zonepath`、`autoboot`、および `pool` に関する情報を表示します。リソースタイプが指定されている場合は、そのタイプのリソースについてのみ情報を表示します。リソース固有の有効範囲では、このサブコマンドは、追加または変更しようとしているリソースにのみ適用されます。

`verify` 現在の構成が正しいかどうかを確認します。各リソースに必須プロパティーがすべて指定されていることを確認します。`rootzpool` リソースグループとそのプロパティーの構文を確認します。URI によって指定されているストレージのアクセス可能性は確認されません。

`commit` 現在の構成をメモリーから安定した記憶領域に確定します。メモリー内の構成を確定するまでは、`revert` サブコマンドで変更内容を削除できます。`zoneadm` で構成を使用するには、その構成を確定する必要があります。`zonecfg` セッションを完了するときに、この操作の実行が自動的に試みられます。正しい構成のみ確定できるので、確定操作では自動的に確認も行われます。

`revert` 構成を最後に確定されたときの状態に戻します。

`exit` `zonecfg` のセッションを終了します。`exit` に `-F` (強制) オプションを使用できます。

必要な場合は、`commit` 操作が自動的に試行されます。EOF 文字を使ってセッションを終了することもできることに注意してください。

zonecfg のコマンドファイルモード

コマンドファイルモードでは、ファイルから入力されます。このファイルを生成するには、238 ページの「zonecfg の対話型モード」で説明されている export サブコマンドを使用します。構成を標準出力に出力するか、-f オプションで指定した出力ファイルに出力することができます。

ゾーン構成データ

ゾーン構成データは、「リソース」と「プロパティー」という 2 種類のエンティティーから成ります。各リソースは、タイプのほかにも 1 つ以上のプロパティーを持つことがあります。プロパティーは名前と値から成ります。どのようなプロパティーセットを持つかは、リソースタイプによって異なります。

必須プロパティーは、zonename と zonepath だけです。

リソースタイプとプロパティー

リソースとプロパティーのタイプの説明は、次のとおりです。

zonename

ゾーンの名前。ゾーン名には次のような規則が適用されます。

- 各ゾーンの名前は一意でなければならない。
- ゾーン名では大文字と小文字が区別される。
- ゾーン名は英数字で始まる必要がある。

  名前には、英数字、下線 (_)、ハイフン (-)、およびピリオド (.) を使用できます。
- 名前の長さは 63 文字以内でなければならない。
- global という名前と SYS で始まるすべての名前は、予約されているので使用できません。

zonepath

zonepath プロパティーは、ゾーンがインストールされるパスを指定します。各ゾーンには、ルートディレクトリのパスが設定されます。これは、大域ゾーンのルートディレクトリに対する相対パスです。インストール時には、大域ゾーンのディレクトリの可視性が制限されている必要があります。ゾーンのパスの所有者は root で、モードは 700 であることが必要です。ゾーンのパスが存在しない場合、インストール時に自動的に作成されます。アクセス権が正しくない場合、自動的に修正されます。

非大域ゾーンのルートパスは 1 つ下のレベルになります。ゾーンのルートディレクトリの所有権とアクセス権は、大域ゾーンのルートディレクトリ (/) と同じになります。ゾーンのディレクトリの所有者は root で、モードは 755 であることが必要です。この階層構造により、大域ゾーンのユーザーでも権限を持っていない場合は、非大域ゾーンのファイルシステムと行き来できなくなります。

ゾーンは、ZFS データセット上に存在する必要があります。ZFS データセットは、ゾーンのインストール時または接続時に自動的に作成されます。ZFS データセットを作成できない場合、ゾーンのインストールや接続は行われません。

パス	説明
/zones/my-zone	zonecfg zonepath
/zones/my-zone/root	ゾーンのルート

詳細は、367 ページの「ファイルシステムの行き来」を参照してください。

注 – zoneadm の move サブコマンドで新しいフルパス zonepath を指定することにより、ゾーンを同じシステム上の別の場所に移動できます。手順については、312 ページの「非大域ゾーンの移動」を参照してください。

---

autoboot

このプロパティーを true に設定すると、大域ゾーンのブート時にこのゾーンが自動的にブートされます。これは、デフォルトでは false に設定されます。ゾーンサービス svc:/system/zones:default が無効になっている場合、このプロパティーの設定にもかかわらず、ゾーンは自動的にはブートしません。svcadm(1M) のマニュアルページに記載されているように、svcadm コマンドを使用してゾーンサービスを有効にできます。

```
global# svcadm enable zones
```

pkg update を実行中のこの設定については、348 ページの「ゾーンのパッケージの概要」を参照してください。

bootargs

このプロパティーは、ゾーンのブート引数を設定するために使用します。reboot、zoneadm boot、または zoneadm reboot コマ

ンドで無効にされた場合を除き、このブート引数が適用されます。294 ページの「ゾーンのブート引数」を参照してください。

`limitpriv`

このプロパティーは、デフォルト以外の特権マスクを指定するために使用します。378 ページの「非大域ゾーン内の特権」を参照してください。

特権を追加するには、特権名だけを指定するか、特権名の前に `priv_` 付けて指定します。特権を除外するには、名前の前にダッシュ (`-`) または感嘆符 (`!`) を付けます。複数の特権は、コンマで区切り、引用符 (“) で囲みます。

`priv_str_to_set(3C)`で説明されているように、特殊な特権セット `none`、`all`、および `basic` は、それぞれの通常の定義に展開されます。ゾーン構成は大域ゾーンで行われるため、特殊な特権セット `zone` は使用できません。特定の特権を追加または削除してデフォルトの特権セットを変更するのが一般的な使用方法であるため、特殊なセットである `default` はデフォルトの特権セットにマップされます。`limitpriv` プロパティーの先頭に `default` がある場合、デフォルトセットに展開されます。

次のエントリは、`dtrace_proc` 特権と `dtrace_user` 特権だけを必要とする DTrace プログラムをゾーンで使用できるようにします。

```
global# zonecfg -z userzone
zonecfg:userzone> set limitpriv="default,dtrace_proc,dtrace_user"
```

ゾーンの特権セットに不許可の特権が含まれる場合、必須の特権が欠落している場合、または未知の特権が含まれる場合、ゾーンの検証、準備、またはブートの試行は失敗し、エラーメッセージが表示されます。

`scheduling-class`

このプロパティーは、ゾーンのスケジューリングクラスを設定します。詳細とヒントについては、220 ページの「スケジューリングクラス」を参照してください。

`ip-type`

このプロパティーは、すべての非大域ゾーンのために設定する必要があります。227 ページの「排他的 IP 非大域ゾーン」、226 ページの「共有 IP 非大域ゾーン」、および 270 ページの「ゾーンの構成方法」を参照してください。

`dedicated-cpu`

このリソースは、ゾーンの実行中にシステムのプロセッサの一部をそのゾーン専用に割り当てます。`dedicated-cpu` リソース

	は、ncpus の制限を設定し、必要に応じて importance も設定します。詳細は、219 ページの「dedicated-cpu リソース」を参照してください。
capped-cpu	このリソースは、ゾーンの実行中にゾーンで消費可能な CPU リソースの量に対する制限を設定します。この capped-cpu リソースは、ncpus に制限を設けます。詳細は、220 ページの「capped-cpu リソース」を参照してください。
capped-memory	このリソースは、ゾーンのメモリー上限を設定する際に使用される各プロパティーをグループ化します。capped-memory リソースは、physical、swap、および locked メモリーの制限を設定します。これらのプロパティーの少なくとも 1 つは指定する必要があります。capped-memory リソースを使用するには、service/resource-cap パッケージが大域ゾーン内にインストールされている必要があります。
anet	anet リソースは、排他的 IP ゾーンのブート時にゾーンに対して一時的な VNIC インタフェースを自動的に作成し、またゾーンの停止時にそれを削除します。
net	net リソースは、非大域ゾーンに大域ゾーン内にある既存のネットワークインタフェースを割り当てます。ネットワークインタフェースリソースは、インタフェースの名前です。各ゾーンでは、インストール済み状態から準備完了状態に移行するときに設定される、複数のネットワークインタフェースを保持できます。
dataset	データセットは、ファイルシステム、ボリューム、またはスナップショットの総称です。ZFS データセットリソースを追加すると、ストレージ管理を非大域ゾーンに委任できるようになります。委任されたデータセットがファイルシステムの場合は、ゾーン管理者はそのデータセット内のファイルシステムの作成と破棄、およびデータセットのプロパティーの変更を行うことができます。ゾーン管理者は、スナップショット、子のファイルシステムとボリューム、その子孫のクローンを作成できます。委任されたデータセットがボリュームの場合は、ゾーン管理者はプロパティーの設定とスナップショットの作成を行うことができます。ゾーン管理者は、ゾーンに追加されていないデータセットを操作したり、ゾーンに割り当てられているデータセットに設定されている最上位レベルの割り当て制限を超過したりすることはできません。データセットが非大域ゾーンに委任されると、zoned プロパティーが自動的に設定されます。ゾーン管理者がマウントポイントを受け入れられな

い値に設定しなければならない可能性があるため、ゾーンファイルシステムを大域ゾーンにマウントすることはできません。

次の方法で、ZFS データセットをゾーンに追加できます。

- `lofs` マウントされたファイルシステムとして (大域ゾーンとの領域共有のみが目的の場合)
- 委任されたデータセットとして

『Oracle Solaris 11.1 の管理: ZFS ファイルシステム』の第 9 章「Oracle Solaris ZFS の高度なトピック」、361 ページの「ファイルシステムと非大域ゾーン」、および `datasets(5)` のマニュアルページを参照してください。

データセットの問題については、第 28 章「Oracle Solaris ゾーンで発生するさまざまな問題のトラブルシューティング」も参照してください。

`fs`

各ゾーンでは、インストール済み状態から準備完了状態に移行するときにマウントする各種のファイルシステムを指定できます。ファイルシステムリソースは、ファイルシステムのマウントポイントのパスを指定します。ゾーンでファイルシステムを使用する方法の詳細については、361 ページの「ファイルシステムと非大域ゾーン」を参照してください。

注 – 非大域ゾーン内の UFS ファイルシステムを `fs` リソースによって使用するには、インストールの後か、または AI マニフェストスクリプトを使用して、`system/file-system/ufs` パッケージをゾーンにインストールする必要があります。

`fs` リソースによって追加された UFS ファイルシステムの割り当て制限情報を取得するために、`quota` コマンド (`quota(1M)` を参照) は使用できません。

`fs-allowed`

このプロパティーを設定すると、ゾーン管理者によって作成されたか、または NFS を使用してインポートされた、該当のタイプの任意のファイルシステムをマウントし、またそのファイルシステムを管理するための機能がゾーン管理者に提供されます。また、稼働中のゾーン内でのファイルシステムのマウント権限は、`fs-allowed` プロパティーで制限します。デフォルトでは、`hsfs` ファイルシステムおよび NFS などのネットワークファイルシステムのマウントだけがゾーン内で許可されます。

このプロパティーは、ゾーンに委任されたブロックデバイスまたは ZVOL デバイスでも使用できます。

`fs-allowed` プロパティーは、ゾーン内からマウントできる、追加のファイルシステムのコンマ区切りリスト (たとえば、`ufs,pcfs`) を受け入れます。

```
zonecfg:my-zone> set fs-allowed=ufs,pcfs
```

このプロパティーは、`add fs` プロパティーまたは `add dataset` プロパティーを使用して大域ゾーンによって管理されるゾーンのマウントに影響を与えません。

セキュリティーに関する考慮事項については、361 ページの「ファイルシステムと非大域ゾーン」および373 ページの「非大域ゾーンでのデバイスの使用」を参照してください。

`device`

デバイスリソースは、デバイス照合の指定子です。各ゾーンでは、インストール済み状態から準備完了状態に移行するときに構成するべきデバイスを指定できます。

注 – 非大域ゾーン内の UFS ファイルシステムを `device` リソースによって使用するには、インストールの後か、または AI マニフェストスクリプトを使用して、`system/file-system/ufs` パッケージをゾーンにインストールする必要があります。

`pool`

このプロパティーは、システム上のリソースプールをゾーンに関連付けるために使用します。1 つのプール内のリソースを複数のゾーンが共有してもかまいません。219 ページの「`dedicated-cpu` リソース」も参照してください。

`rctl`

`rctl` リソースは、ゾーン規模のリソース制御に使用されます。リソース制御は、ゾーンがインストール済み状態から準備完了状態に移行するときに有効になります。

詳細は、233 ページの「ゾーン規模のリソース制御の設定」を参照してください。

注 – `rctl` リソースの代わりに *zonefig* の `set global_property_name` サブコマンドを使用してゾーン規模のリソース制御を構成するには、270 ページの「ゾーンの構成方法」を参照してください。

---

attr
: この汎用属性は、ユーザーコメントとして使用したり、ほかのサブシステムで使用したりできます。attr の name プロパティーは、英数字で始まる必要があります。name プロパティーには、英数字、ハイフン (-)、およびピリオド (.) を使用できます。zone. で始まる属性名はシステム用に予約されています。

リソースタイプのプロパティー

リソースには、構成可能なプロパティーもあります。リソースタイプとそれに関連付けられるプロパティーは次のとおりです。

admin
: ユーザー名と特定のゾーンに対するそのユーザーの承認を定義します。

```
zonecfg:my-zone> add admin
zonecfg:my-zone:admin> set user=zadmin
zonecfg:my-zone:admin> set auths=login,manage
zonecfg:my-zone:admin> end
```

auths プロパティーには、次の値を使用できます。

- login (solaris.zone.login)
- manage (solaris.zone.manage)
- clone (solaris.zone.clonefrom)

これらの auths はゾーンの作成を可能にするわけではありません。この機能は、ゾーンのセキュリティープロファイル内に含まれています。

rootzpool
: storage

ゾーンのインストール専用の ZFS zpool を提供するためのストレージオブジェクトの URI を識別します。URI と storage に指定可能な値については、222 ページの「rootzpool リソース」を参照してください。ゾーンのインストール時に、zpool が自動的に作成されるか、事前に作成された zpool がインポートされます。*my-zone*_rpool という名前が割り当てられます。

```
zonecfg:my-zone> add rootzpool
zonecfg:my-zone:rootzpool> add storage dev:dsk/c4t1d0
zonecfg:my-zone:rootzpool> end
```

ミラー化構成を作成する場合は、storage プロパティーを追加できます。

```
add storage dev:dsk/c4t1d0
add storage dev:dsk/c4t3d0
```

1 つのゾーンに構成できる rootzpool リソースは 1 つだけです。

zpool　　storage、name

zpool をゾーンに委任するためのストレージオブジェクトの URI を 1 つ以上定義します。URI と storage プロパティーに指定可能な値については、222 ページの「rootzpool リソース」を参照してください。name プロパティーに指定可能な値は、zpool(1M) のマニュアルページで定義されています。

この例では、zpool ストレージリソースがゾーンに委任されています。zpool は自動的に作成されるか、または以前に作成された zpool がインストール中にインポートされます。zpool の名前は my-zone_pool1 です。

```
zonecfg:my-zone> add zpool
zonecfg:my-zone:zpool> set name=pool1
zonecfg:my-zone:zpool> add storage dev:dsk/c4t2d0
zonecfg:my-zone:zpool> add storage dev:dsk/c4t4d0
zonecfg:my-zone:zpool> end
```

ゾーン構成には、1 つ以上の zpool リソースを含めることができます。

dedicated-cpu　　ncpus、importance

CPU の数を指定し、必要に応じてプールの相対的な重要性も指定します。次の例では、ゾーン my-zone で使用する CPU の範囲を指定します。importance も設定します。

```
zonecfg:my-zone> add dedicated-cpu
zonecfg:my-zone:dedicated-cpu> set ncpus=1-3
zonecfg:my-zone:dedicated-cpu> set importance=2
zonecfg:my-zone:dedicated-cpu> end
```

capped-cpu　　ncpus

CPU の数を指定します。次の例では、ゾーン my-zone の CPU 数のキャップを 3.5 に指定します。

```
zonecfg:my-zone> add capped-cpu
zonecfg:my-zone:capped-cpu> set ncpus=3.5
zonecfg:my-zone:capped-cpu> end
```

capped-memory　　physical、swap、locked

ゾーン my-zone のメモリー制限を指定します。各制限はオプションですが、少なくとも 1 つは設定する必要があります。

```
zonecfg:my-zone> add capped-memory
zonecfg:my-zone:capped-memory> set physical=50m
zonecfg:my-zone:capped-memory> set swap=100m
zonecfg:my-zone:capped-memory> set locked=30m
zonecfg:my-zone:capped-memory> end
```

capped-memory リソースを使用するには、resource-cap パッケージが大域ゾーン内にインストールされている必要があります。

fs

dir、special、raw、type、options

fs リソースのパラメータは、ファイルシステムをマウントする方法と場所を決定する値を指定します。fs のパラメータは次のように定義されています。

dir　ファイルシステムのマウントポイントを指定します

special　大域ゾーンからマウントするブロック型特殊デバイスの名前またはディレクトリを指定します

raw　ファイルシステム (ZFS には適用されない) をマウントする前に、fsck の実行対象の raw デバイスを指定します。

type　ファイルシステムのタイプを指定します

options　mount コマンドで使用されるオプションに似たマウントオプションを指定します

次の例の各行では、大域ゾーン内の pool1/fs1 という名前のデータセットを、構成されるゾーン内で /shared/fs1 としてマウントすることを指定しています。使用するファイルシステムのタイプは ZFS です。

```
zonecfg:my-zone> add fs
zonecfg:my-zone:fs> set dir=/shared/fs1
zonecfg:my-zone:fs> set special=pool1/fs1
zonecfg:my-zone:fs> set type=zfs
zonecfg:my-zone:fs> end
```

パラメータの詳細については、361 ページの「-o nosuid オプション」、364 ページの「セキュリティーの制限およびファイルシステムの動作」、および fsck(1M) と mount(1M) のマニュアルページを参照してください。また、セクション 1M のマニュアルページには、特定のファイルシステムに固有のマウントオプションに関するものがあります。このようなマニュアルページの名前は、mount_*filesystem* という形式です。

注 – このリソースを使用して追加された UFS ファイルシステムの割り当て制限情報を取得するために、quota コマンド (quota(1M) を参照) は使用できません。

datasetname、alias	name
	
	次の例では、データセット *sales* を非大域ゾーンでマウントして可視にし、大域ゾーンでは不可視にするように指定します。
	
	`zonecfg:my-zone> add dataset`
	`zonecfg:my-zone> set name=tank/sales`
	`zonecfg:my-zone> end`
	
	委任されたデータセットは、次の例に示されているようにデフォルト以外の別名を持つことができます。データセットの別名には、スラッシュ (/) を含めることができないことに注意してください。
	
	`zonecfg:my-zone> add dataset`
	`zonecfg:my-zone:dataset> set name=tank/sales`
	`zonecfg:my-zone:dataset> set alias=data`
	`zonecfg:my-zone:dataset> end`
	
	デフォルトの別名に戻すには、clear alias を使用します。
	
	`zonecfg:my-zone> clear alias`

anet	linkname、 lower-link、 allowed-address、 auto-mac-address、 configure-allowed-address、 defrouter、 linkmode (IPoIB)、 mac-address (IPoIB 以外)、 mac-slot (IPoIB 以外)、 mac-prefix (IPoIB 以外)、 mtu、 maxbw、 pkey (IPoIB)、 priority、 vlan-id (IPoIB 以外)、 rxfanout、 rxrings、 txrings、 link-protection、 allowed-dhcp-cids zonecfg では、IPoIB データリンクに次の anet プロパティーを設定しないでください。 ■ mac-address ■ mac-prefix ■ mac-slot ■ vlan-id zonecfg では、IPoIB 以外のデータリンクに次の anet プロパティーを設定しないでください。 ■ linkmode

- pkey

anet リソースは、ゾーンのブート時に自動 VNIC インタフェースまたは IPoIB インタフェースを作成し、ゾーンの停止時にその VNIC または IPoIB インタフェースを削除します。リソースプロパティーは、zonecfg コマンドを使用して管理されます。使用できるプロパティーのすべてのテキストについては、zonecfg(1M) のマニュアルページを参照してください。

lower-link	作成されるリンクの基礎となるリンクを指定します。auto に設定すると、zoneadmd デーモンは、ゾーンがブートするたびに VNIC が作成されるリンクを自動的に選択します。 ブート中に VNIC を自動的に作成するためのデータリンクを選択すると、すべての IPoIB リンクがスキップされます。
linkname	自動作成される VNIC インタフェースまたは IPoIB インタフェースの名前を指定します。
mac-address (**IPoIB** には非適用)	指定された値またはキーワードに基づいて VNIC の MAC アドレスを設定します。値がキーワードでない場合は、ユニキャスト MAC アドレスとして解釈されます。サポートされるキーワードについては、zonecfg(1M) のマニュアルページを参照してください。ランダムな MAC アドレスを選択すると、生成されたアドレスはゾーンの再ブートおよ

	びゾーンの切り離しと接続の操作後にも保持されます。
pkey (**IPoIB** のみ)	IPoIB データリンクインタフェースを作成するために使用するパーティションキーを設定します。このプロパティーは必須です。指定した pkey は、0x 接頭辞があるかどうかに関係なく、常に 16 進数として扱われます。
linkmode (**IPoIB** のみ)	データリンクインタフェースの linkmode を設定します。デフォルト値は cm です。有効な値は次のとおりです。 cm (デフォルト) 接続モード。このモードでは、65520 バイトのデフォルトの MTU を使用し、65535 バイトの最大 MTU をサポートします。

ud	低信頼データグラムモード。リモートノードに対して接続モードが使用できない場合は、代わりに低信頼データグラムモードが自動的に使用されます。このモードでは、2044 のデフォルトの MTU を使用し、4092 バイトの最大 MTU をサポートします。

`allowed-address`	排他的 IP ゾーン用の IP アドレスを構成し、また排他的 IP ゾーンで使用できる、構成可能な IP アドレスセットの制限も行います。複数のアドレスを指定するには、IP アドレスのコンマ区切りリストを使用します。
`defrouter`	`defrouter` プロパティーを使用すると、非大域ゾーンと大域ゾーンが別個のネットワーク上に存在する場合にデフォルト経路を設定できます。 `defrouter` プロパティーが設定されたゾーンは、大域ゾーン用に構成されていないサブネット上に存在する必要があります。

`zonecfg` コマンドで `SYSdefault` テンプレートを使用してゾーンを作成したときに、ほかの IP リソースが設定されていない場合は、次のプロパティーを持つ `anet` リソースがゾーン構成に自動的に組み込まれます。`linkname` は、物理 Ethernet リンクに対して自動的に作成され、`net`*N* という形式の最初に使用できる名前 (`net0`) に設定されます。デフォルト値を変更するには、`zonecfg` コマンドを使用します。

デフォルトでは、物理 Ethernet リンク (たとえば、`nxge0`) に対して自動 VNIC が作成され、この VNIC にファクトリ MAC アドレスが割り当てられます。オプションの `lower-link` プロパティーは、自動 VNIC が作成される、基礎となるリンク `nxge0` に設定されます。`zonecfg` コマンドを使用して、リンク名、ベースとなる物理リンク、MAC アドレス、帯域幅制限などの VNIC プロパティー、およびその他の VNIC プロパティーを指定できます。`ip-type=exclusive` も指定する必要があります。

```
zonecfg:my-zone> set ip-type=exclusive
zonecfg:my-zone> add anet
zonecfg:my-zone:anet> set linkname=net0
zonecfg:my-zone:anet> set lower-link=auto
zonecfg:my-zone:anet> set mac-address=random
```

```
zonecfg:my-zone:anet> set link-protection=mac-nospoof
zonecfg:my-zone:anet> end
```

次の例は、物理リンク net5 上の IPoIB データリンクインタフェースを使用し、IB のパーティションキーを 0xffff にして構成されたゾーンを示しています。

```
zonecfg:my-zone> set ip-type=exclusive
zonecfg:my-zone:anet> add anet
zonecfg:my-zone:anet> set linkname=ib0
zonecfg:my-zone:anet> set lower-link=net5
zonecfg:my-zone:anet> set pkey=0xffff
zonecfg:my-zone:anet> end
```

プロパティーの詳細は、zonecfg(1M) のマニュアルページを参照してください。リンクの各プロパティーの詳細は、dladm(1M) のマニュアルページを参照してください。

net	address、allowed-addressphysical 、defrouter

注 – 共有 IP ゾーンの場合は、IP アドレスと物理デバイスの両方を指定する必要があります。必要に応じて、デフォルトのルーターを設定できます。

排他的 IP ゾーンの場合は、物理インタフェースだけを指定する必要があります。

- allowed-address プロパティーは、排他的 IP ゾーンで使用できる、構成可能な IP アドレスのセットを制限します。
- defrouter プロパティーを使用すると、非大域ゾーンと大域ゾーンが別個のネットワーク上に存在する場合にデフォルト経路を設定できます。
- defrouter プロパティーが設定されたゾーンは、大域ゾーン用に構成されていないサブネット上に存在する必要があります。
- デフォルトのルーターを持つゾーンからのトラフィックは、宛先のゾーンに戻る前にそのルーターに送信されます。

共有 IP ゾーンがそれぞれ異なるサブネット上にある場合は、大域ゾーンでデータリンクを構成しないでください。

共有 IP ゾーンの場合の次の例では、物理インタフェースの nge0 の IP アドレスが 192.168.0.1 のゾーンに追加されています。システム上のネットワークインタフェースをリスト表示するには、次のように入力します。

```
global# ipadm show-if -po ifname,class,active,persistent
lo0:loopback:yes:46--
nge0:ip:yes:----
```

出力の各行には、ループバックの行を除いて、ネットワークインタフェースの名前があります。説明に loopback が含まれている行は、カードには当てはまりません。46 の persistent フラグは、そのインタフェースが大域ゾーン内で持続的に構成されていることを示します。yes という有効な値は、そのインタフェースが現在構成されていることを示しています。また class の ip の値は、nge0 がループバックインタフェースではないことを示しています。ゾーンのデフォルト経路は、10.0.0.1 に設定されています。defrouter プロパティーの設定はオプションです。ip-type=shared は必須です。

```
zonecfg:my-zone> set ip-type=shared
zonecfg:my-zone> add net
zonecfg:my-zone:net> set physical=nge0
zonecfg:my-zone:net> set address=192.168.0.1
zonecfg:my-zone:net> set defrouter=10.0.0.1
zonecfg:my-zone:net> end
```

排他的 IP ゾーンの場合の次の例では、bge1 上の VLAN の bge32001 リンクが物理インタフェースに使用されています。使用可能なデータリンクを調べるには、dladm show-link を使用してください。allowed-address プロパティーは、ゾーンが使用できる IP アドレスを抑制します。defrouter プロパティーは、デフォルト経路を設定するために使用します。ip-type=exclusive も指定する必要があります。

```
zonecfg:my-zone> set ip-type=exclusive
zonecfg:my-zone> add net
zonecfg:myzone:net> set allowed-address=10.1.1.32/24
zonecfg:my-zone:net> set physical=bge32001
zonecfg:myzone:net> set defrouter=10.1.1.1
zonecfg:my-zone:net> end
```

add net 手順では、物理デバイスタイプだけを指定します。physical プロパティーは、VNIC (『Oracle Solaris Administration: Network Interfaces and Network Virtualization』のパート III「Network Virtualization and Resource Management」に記載) にすることも可能です。

注 – Oracle Solaris オペレーティングシステムは、すべての Ethernet タイプインタフェースをサポートします。また、それらのデータリンクは dladm コマンドを使用して管理できます。

device

match、allow-partition、allow-raw-io

マッチングするデバイス名は、マッチングするパターンまたは絶対パスにできます。allow-partition と allow-raw-io の両方を true または false に設定できます。デフォルトは false です。allow-partition はパーティション分割を可能にします。allow-raw-io は uscsi を可能にします。これらのリソースの詳細は、zonecfg(1M) を参照してください。

次の例には、ディスクデバイスに対する uscsi 操作がゾーンの構成に含まれています。

```
zonecfg:my-zone> add device
zonecfg:my-zone:device> set match=/dev/*dsk/cXtYdZ*
zonecfg:my-zone:device> set allow-raw-io=true
zonecfg:my-zone:device> end
```

Veritas Volume Manager デバイスは、add device を使用して非大域ゾーンに委任されます。

注意 – デバイスを追加する場合、制限およびセキュリティーの懸念事項については、373 ページの「非大域ゾーンでのデバイスの使用」、375 ページの「非大域ゾーンでのアプリケーションの実行」、および378 ページの「非大域ゾーン内の特権」を参照してください。

rctl

name、value

使用可能なゾーン規模のリソース制御は、次のとおりです。

- zone.cpu-cap
- zone.cpu-shares (推奨: cpu-shares)
- zone.max-locked-memory
- zone.max-lofi
- zone.max-lwps (推奨: max-lwps)
- zone.max-msg-ids (推奨: max-msg-ids)
- zone.max-processes(推奨: max-processes
- zone.max-sem-ids (推奨: max-sem-ids)

- zone.max-shm-ids (推奨: max-shm-ids)
- zone.max-shm-memory (推奨: max-shm-memory)
- zone.max-swap

ゾーン規模のリソース制御を設定する場合に望ましい、より簡単な方法は、rctl リソースの代わりにプロパティー名を使用する方法です。詳細は、270 ページの「ゾーンの構成方法」を参照してください。add rctl を使ってゾーン内のゾーン規模のリソース制御エントリを構成する場合、その形式は project データベース内のリソース制御エントリの形式とは異なります。ゾーン構成では、rctl リソースタイプは、名前と値の対 3 つから成ります。これらの名前は、priv、limit、および action です。これらの名前には、単純な値がそれぞれ設定されます。

```
zonecfg:my-zone> add rctl
zonecfg:my-zone:rctl> set name=zone.cpu-shares
zonecfg:my-zone:rctl> add value (priv=privileged,limit=10,action=none)
zonecfg:my-zone:rctl> end

zonecfg:my-zone> add rctl
zonecfg:my-zone:rctl> set name=zone.max-lwps
zonecfg:my-zone:rctl> add value (priv=privileged,limit=100,action=deny)
zonecfg:my-zone:rctl> end
```

リソース制御と属性の概要については、第 6 章「リソース制御 (概要)」および376 ページの「非大域ゾーンで使用されるリソース制御」を参照してください。

attr　　name、type、value

次の例では、ゾーンに関するコメントを追加します。

```
zonecfg:my-zone> add attr
zonecfg:my-zone:attr> set name=comment
zonecfg:my-zone:attr> set type=string
zonecfg:my-zone:attr> set value="Production zone"
zonecfg:my-zone:attr> end
```

export サブコマンドを使用すると、ゾーン構成を標準出力に出力できます。構成は、コマンドファイルに使用できる形式で保存されます。

ゾーンの構成例

ゾーンの構成例として、次のようなものがあります。

- rootzpool リソースを含む共有ストレージ上のゾーン。オペレーティングシステムソフトウェアは専用の zpool に格納され、オペレーティングシステム以外のすべてのソフトウェアおよびデータはゾーンルートのデータセットの下位にある

データセットに格納されます。オペレーティングシステムのスナップショットやクローンを作成するシステム操作によって、オペレーティングシステム以外のソフトウェアおよびデータのスナップショットやクローンも作成されます。

- `rootzpool` リソースと 1 つ以上の `zpool` リソースを含む共有ストレージ上のゾーン。オペレーティングシステムは専用の `rootzpool` に格納され、オペレーティングシステム以外のすべてのソフトウェアおよびデータはほかの zpool 内に格納されます。オペレーティングシステムのスナップショットやクローンを作成するシステム操作には、オペレーティングシステム以外のソフトウェアおよびデータは含まれません。
- デフォルトのシステム固有の `zonepath` を使用したゾーン。このゾーンでは、`zonepath` にオペレーティングシステムが格納されます。オペレーティングシステムのスナップショットやクローンを作成するシステム操作に、オペレーティングシステム以外のソフトウェアおよびデータを含めることができます。

Tecla コマンド行編集ライブラリ

付属の Tecla コマンド行編集ライブラリは、`zonecfg` コマンドで使用できます。このライブラリにより、コマンド行の履歴メカニズムと編集サポートが提供されます。

詳細については、`tecla(5)` のマニュアルページを参照してください。

◆ ◆ ◆ 第 17 章

17 非大域ゾーンの計画と構成 (タスク)

この章では、システムにゾーンを構成する前に実行する必要のある操作について説明します。また、ゾーンの構成方法、ゾーン構成の変更方法、およびシステムからゾーン構成を削除する方法についても説明します。

ゾーン構成処理の概要については、第 16 章「非大域ゾーンの構成 (概要)」を参照してください。

`solaris10` ブランドゾーンの構成については、パート III「Oracle Solaris 10 ゾーン」を参照してください。

非大域ゾーンの計画と構成 (タスクマップ)

ゾーンを使用できるようにシステムを設定する前に、まず、情報を収集してゾーンの構成方法を決定する必要があります。次のタスクマップに、ゾーンの計画および構成方法の概要を示します。

タスク	説明	参照先
ゾーンの全体的な計画を立てます。	■ システムで稼働しているアプリケーションを評価し、ゾーン内で実行するアプリケーションを決定します。 ■ ゾーン内で固有のファイルを保持するディスク領域の可用性を評価します。 ■ リソース管理機能も使用している場合は、リソース管理境界に合わせてゾーンを配列する方法を決定します。 ■ リソースプールを使用する場合は、必要に応じてプールを構成します。 ■ ゾーンを共有ストレージでホストするかどうかを決定します。	使用状況の履歴を参照してください。265 ページの「必要なディスク容量」および141 ページの「ゾーンで使用されるリソースプール」も参照してください。
ゾーンの名前を決定します。	命名規則に基づいてゾーンの名前を決定します。	241 ページの「ゾーン構成データ」および266 ページの「ゾーンのホスト名」を参照してください。
ゾーンパス (必須) を決定します。	各ゾーンには、ルートディレクトリのパスが設定されます。これは、大域ゾーンのルートディレクトリに対する相対パスです。	241 ページの「ゾーン構成データ」を参照してください。
リソースプールを構成しない場合は、CPU 制限の必要性を評価します。`zonecfg` で指定すると、移行時にプールの設定が伝達されます。	アプリケーションの要件を確認します。	219 ページの「`dedicated-cpu` リソース」を参照してください。

タスク	説明	参照先
大域ゾーンから rcapd を使用してゾーンのメモリー上限を設定する場合は、メモリー割り当ての必要性を評価します。	アプリケーションの要件を確認します。	第 10 章「リソース上限デーモンによる物理メモリーの制御(概要)」、第 11 章「リソース上限デーモンの管理(タスク)」、および 221 ページの「物理メモリーの制御と capped-memory リソース」を参照してください。
FSS をシステムのデフォルトのスケジューラにします。	各ゾーンに CPU 配分を与えて、CPU リソースに対するゾーンの使用権を制御します。FSS では、割り当てられた配分に基づいて、プロジェクト間に CPU リソースが公平に配分されることが保証されています。	第 8 章「公平配分スケジューラ(概要)」および 220 ページの「スケジューリングクラス」を参照してください。

タスク	説明	参照先
排他的 IP がゾーンのデフォルトのタイプです。	anet リソースで構成された排他的 IP ゾーンの場合、システムはゾーンがブートするたびに VNIC を自動的に作成します。net リソースで構成された排他的 IP ゾーンの場合、ゾーンに割り当てるデータリンクを決定します。ゾーンには、1 つ以上のネットワークインタフェースへの排他的アクセスが必要です。インタフェースは、VNIC、bge1 などの個別の LAN、または bge2000 などの個別の VLAN にできます。『Oracle Solaris Administration: Network Interfaces and Network Virtualization』のパート III「Network Virtualization and Resource Management」を参照してください。 共有 IP ゾーンを構成する場合は、ゾーン用の IP アドレスを入手するか、または構成します。構成に基づき、ネットワークアクセスを行う非大域ゾーンごとに 1 つ以上の IP アドレスを取得する必要があります。	266 ページの「ゾーンのホスト名およびネットワーク要件の決定」、270 ページの「ゾーンの構成方法」、および『Oracle Solaris 11.1 ネットワークの構成と管理』を参照してください。
ゾーン内にマウントするファイルシステムを決定します。	アプリケーションの要件を確認します。	詳細は、229 ページの「ゾーンでマウントされるファイルシステム」を参照してください。
ゾーンで使用可能にするべきネットワークインタフェースを決定します。	アプリケーションの要件を確認します。	詳細は、370 ページの「共有 IP ネットワークインタフェース」を参照してください。
非大域ゾーンのデフォルトの特権セットを変更する必要があるかどうかを決定します。	特権セットを確認します。 デフォルトの特権、追加および削除が可能な特権、および現時点では使用できない特権があります。	378 ページの「非大域ゾーン内の特権」を参照してください。

タスク	説明	参照先
各ゾーンで構成するべきデバイスを決定します。	アプリケーションの要件を確認します。	使用するアプリケーションのドキュメントを参照してください。
ゾーンを構成します。	zonecfg を使用してゾーンの構成を作成します。	269 ページの「ゾーンを構成、検証、および確定する」を参照してください。
構成したゾーンを検証および確定します。	指定されたリソースおよびプロパティーが仮想サーバー上で有効かどうかを判定します。	269 ページの「ゾーンを構成、検証、および確定する」を参照してください。

現在のシステム設定の評価

ゾーンは、Oracle Solaris 10 またはそれ以降のリリースが稼働する任意のマシンで使用できます。次に、ゾーンの使用に関連したマシンの主な考慮事項を示します。

- 各ゾーン内部で稼働するアプリケーションの性能要件。
- 各ゾーン内部で固有のファイルを保持するディスク容量がどれだけ利用可能か。

必要なディスク容量

ゾーンが消費可能なディスク容量に関する制限はありません。大域管理者または適切な承認を持つユーザーが、容量の制限を担当します。大域管理者は、非大域ゾーンのルートファイルシステムを保持するのに十分なローカルストレージまたは共有ストレージがあることを確認する必要があります。小規模な単一プロセッサシステムでも、同時に稼働する多数のゾーンをサポートできます。

非大域ゾーンにインストールされるパッケージの特性は、ゾーンの容量要件に影響を与えます。また、パッケージの数も 1 つの要因です。

ディスク要件は、大域ゾーンに現在インストールされているパッケージとインストール済みのソフトウェアによって使用されるディスク容量によって決定されます。

1 つのゾーンでは、ゾーンあたりに最低 150M バイトの空きディスク容量を必要とします。ただし、大域ゾーンが標準のすべての Oracle Solaris パッケージでインストールされている場合、必要な空きディスク容量は通常 500M バイトから 1G バイトです。さらに多くのソフトウェアが追加される場合、この数値は増える可能性があります。

ゾーンごとに 40M バイトの追加 RAM が推奨されていますが、十分なスワップ空間のあるマシンでは、これは必須ではありません。

ゾーンサイズの制限

ZFS データセットによって支援される zonepath を持つゾーンでは、ZFS データセットの割り当て制限を使用してゾーンサイズを制限できます。zonepath データセットにアクセスできる管理者は、データセットの quota および reservation プロパティーを変更して、各ゾーンが消費できる最大のディスク容量を制御できます。これらのプロパティーについては、zfs(1M) のマニュアルページで説明されています。

また、管理者は、ZFS ボリュームを固定サイズで作成して、そのボリュームのデータセット内にゾーンをインストールすることもできます。このボリュームは、その内部にインストールされたゾーンのサイズを制限します。

ゾーンのホスト名およびネットワーク要件の決定

ゾーンのホスト名を決定する必要があります。

排他的 IP ゾーンの内部で、大域ゾーンに対して行なったようにアドレスを構成します。

ネットワーク接続を持つ共有 IP ゾーンに対して、次のいずれかを実行する必要があります。

- ゾーンに IPv4 アドレスを割り当てます
- ゾーンに IPv6 アドレスを構成して割り当てます

排他的 IP タイプと共有 IP タイプの詳細は、224 ページの「ゾーンネットワークインタフェース」を参照してください

ゾーンのホスト名

NIS または DNS ネームサービス、あるいは LDAP ディレクトリサービスを使用している場合は、サーバー上に存在するデータベース (*hosts.byname* など) にホスト情報が格納されます。

ネームサービス用にローカルファイルを使用する場合は、/etc/inet/hosts ファイル内で hosts データベースが保持されます。ゾーンネットワークインタフェースのホスト名の解決は、/etc/inet/hosts 内のローカル hosts データベースで行われます。あるいは、共有 IP ゾーンの場合、ゾーンの構成時に IP アドレス自体を直接指定することで、ホスト名の解決を不要にできます。詳細は、hosts(4) および nodename(4) のマニュアルページを参照してください。『Oracle Solaris 11.1 ネットワークの構成と管理』の第 7 章「IPv4 リファレンス」も参照してください。

共有 IP ゾーンのネットワークアドレス

ネットワーク接続を必要とする各共有 IP ゾーンには、1 つ以上の一意の IP アドレスが与えられます。IPv4 と IPv6 の両方のアドレスがサポートされます。

IPv4 のゾーンネットワークアドレス

IPv4 を使用している場合、アドレスを取得してゾーンに割り当てます。

IP アドレスとともに接頭辞の長さも指定できます。この接頭辞の書式はアドレス/接頭辞の長さ です。たとえば、`192.168.1.1/24` の場合、使用するアドレスは `192.168.1.1` で、使用するネットマスクは `255.255.255.0`、または最初の 24 ビットがビット 1 であるマスクです。

共有 IP ゾーンの場合は、ゾーンの構成時に IP アドレス自体を直接指定することで、ホスト名の解決を不要にできます。

詳細は、`hosts(4)`、`netmasks(4)`、および `nodename(4)` を参照してください。

IPv6 のゾーンネットワークアドレス

IPv6 を使用している場合、アドレスを手動で構成する必要があります。通常、次の 2 種類のアドレスを最小限構成する必要があります。

リンクローカルアドレス
: リンクローカルアドレスの書式は、`fe80::`*<64 ビットインタフェース ID>*`/10` です。`/10` は、接頭辞の長さが 10 ビットであることを示します。

サブネット上で構成された大域接頭辞から作成されたアドレス
: 大域ユニキャストアドレスは、管理者がサブネットごとに構成した 64 ビット接頭辞および 64 ビットのインタフェース ID に基づきます。接頭辞は、IPv6 を使用するよう構成されている同一サブネット上の任意のシステムで、`ipadm show-addr` コマンドを実行して取得できます。

  通常、64 ビットのインタフェース ID は、システムの MAC アドレスから取得されます。次の方法で、ゾーン用の一意の代替アドレスを大域ゾーンの IPv4 アドレスから取得できます。

  <16 ビットのゼロ>:<IPv4 アドレスの上位 16 ビット>:<IPv4 アドレスの下位 16 ビット>:<ゾーンで一意の番号>

  たとえば、大域ゾーンの IPv4 アドレスが 192.168.200.10 である場合、ゾーン固有の番号 1 を使用する非大域ゾーン用の適正なリンクローカルアドレスは、`fe80::c0a8:c80a:1/10` になります。そのサブネットで使用中の大域接頭辞が `2001:0db8:aabb:ccdd/64` である場合、同じ非大域ゾーン用の一意の大域ユニキャストアドレスは `2001:0db8:aabb:ccdd::c0a8:c80a:1/64` です。IPv6 アドレスを構成する際、接頭辞の長さを指定する必要があることに注意してください。

リンクローカルおよび大域ユニキャストアドレスの詳細は、ipadm(1M) および inet6(7P) のマニュアルページを参照してください。

排他的 IP ゾーンのネットワークアドレス

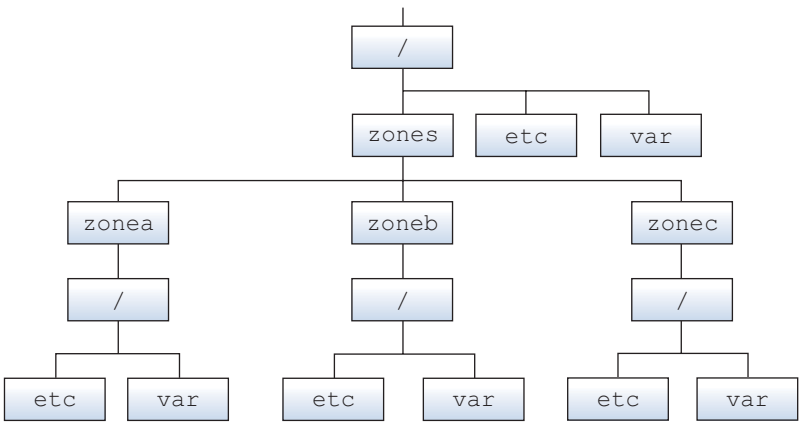
排他的 IP ゾーンの内部で、大域ゾーンと同様の方法でアドレスを構成します。DHCP および IPv6 ステートレスアドレスの自動構成を使用してアドレスを構成することもできます。

ファイルシステムの構成

仮想プラットフォームを設定する際、実行するマウントを多数指定できます。ループバック仮想ファイルシステム (LOFS) を使用してファイルシステムをゾーンにループバックマウントする場合、nodevices オプションを指定してファイルシステムをマウントする必要があります。nodevices オプションの詳細は、361 ページの「ファイルシステムと非大域ゾーン」を参照してください。

LOFS を使用すると、代替パス名を使用してファイルにアクセスできるように、新しい仮想ファイルシステムを作成できます。非大域ゾーンでは、ループバックマウントにより、ファイルシステム階層がゾーンのルート下に複製されているように見えます。ゾーン内では、ゾーンのルートから始まるパス名を使ってすべてのファイルにアクセスできるようになります。LOFS マウントでは、ファイルシステムの名前空間が維持されます。

図 17–1 ループバックマウントされたファイルシステム



詳細は、lofs(7S) のマニュアルページを参照してください。

非大域ゾーン構成の作成、改訂、および削除 (タスクマップ)

タスク	説明	参照先
非大域ゾーンを構成します。	zonecfg コマンドを使用してゾーンの作成、構成の検証および確定を行います。スクリプトを使用して、システム上の複数のゾーンを構成およびブートすることもできます。 非大域ゾーンの構成を表示するには、zonecfg コマンドを使用できます。	269 ページの「ゾーンを構成、検証、および確定する」、276 ページの「複数のゾーンを構成するスクリプト」
ゾーン構成を変更します。	これらの手順を使用して、ゾーンの構成内のリソースタイプを変更するか、ゾーンの名前などのプロパティータイプを変更するか、またはゾーンに専用のデバイスを追加します。	281 ページの「zonecfg コマンドを使用してゾーン構成を変更する」
ゾーン構成を元に戻すか、ゾーン構成を削除します。	zonecfg コマンドを使用して、ゾーン構成に対して行なったリソース設定を取り消すか、ゾーン構成を削除します。	284 ページの「zonecfg コマンドを使用してゾーン構成を元に戻す、または削除する」
ゾーン構成を削除します。	zonecfg コマンドと delete サブコマンドを使用して、システムからゾーン構成を削除します。	286 ページの「ゾーン構成を削除する方法」

ゾーンを構成、検証、および確定する

次のアクションを実行するには、zonecfg コマンド (zonecfg(1M) のマニュアルページに記載) を使用します。

- ゾーン構成を作成します
- 必要な情報がすべて存在することを確認します
- 非大域ゾーン構成を確定します

zonecfg コマンドは、大域ゾーンのリソース管理設定を永続的に指定する場合にも使用できます。

zonecfg ユーティリティーを使用してゾーンを構成する際、revert サブコマンドを使用して、リソースの設定を元に戻すことができます。284 ページの「ゾーン構成を元に戻す方法」を参照してください。

システムに複数のゾーンを構成するスクリプトについては、276 ページの「複数のゾーンを構成するスクリプト」を参照してください。

非大域ゾーンの構成を表示する方法については、280 ページの「非大域ゾーンの構成を表示する方法」を参照してください。

▼ ゾーンの構成方法

非大域ゾーンの作成に必須の要素は、`zonename` および `zonepath` プロパティーだけです。そのほかのリソースおよびプロパティーはオプションです。省略可能なリソースには、`dedicated-cpu` リソースと `capped-cpu` リソースのどちらを使用するかを決めるなど、選択肢の中から選ぶ必要があるものもあります。使用可能な `zonecfg` のプロパティーとリソースについては、241 ページの「ゾーン構成データ」を参照してください。

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を持つユーザーである必要があります。

1. **root** になるか、同等の役割になります。

2. 選択したゾーン名を使用して、ゾーン構成を設定します。

   この手順例では、`my-zone` という名前を使用します。

   ```
   global# zonecfg -z my-zone
   ```

   このゾーンの初回構成時には、次のシステムメッセージが表示されます。

   ```
   my-zone: No such zone configured
   Use 'create' to begin configuring a new zone.
   ```

3. 新しいゾーン構成を作成します。

   この手順では、デフォルト設定を使用します。

   ```
   zonecfg:my-zone> create
   create: Using system default template 'SYSdefault'
   ```

4. この手順でゾーンのパス **/zones/my-zone** を設定します。

   ```
   zonecfg:my-zone> set zonepath=/zones/my-zone
   ```

   ゾーンは、ZFS データセット上に存在する必要があります。ZFS データセットは、ゾーンのインストール時または接続時に自動的に作成されます。ZFS データセットを作成できない場合、ゾーンのインストールや接続は行われません。ゾーンのパスの親ディレクトリが存在する場合、その親ディレクトリはマウントされるデータセットのマウントポイントである必要があります。

5 **autoboot** 値を設定します。

`true` に設定すると、大域ゾーンのブート時にこのゾーンが自動的にブートします。デフォルト値は `false` です。ゾーンを自動的にブートするには、ゾーンサービス `svc:/system/zones:default` も有効になっている必要があります。このサービスは、デフォルトで有効になっています。

```
zonecfg:my-zone> set autoboot=true
```

6 ゾーンの持続的なブート引数を設定します。

```
zonecfg:my-zone> set bootargs="-m verbose"
```

7 **1** つの **CPU** をこのゾーン専用に割り当てます。

```
zonecfg:my-zone> add dedicated-cpu
```

a. **CPU** の数を設定します。

```
zonecfg:my-zone:dedicated-cpu> set ncpus=1-2
```

b. **(**オプション**)** 重要性を設定します。

```
zonecfg:my-zone:dedicated-cpu> set importance=10
```

デフォルト値は `1` です。

c. 指定を終了します。

```
zonecfg:my-zone:dedicated-cpu> end
```

8 権限のデフォルトセットを修正します。

```
zonecfg:my-zone> set limitpriv="default,sys_time"
```

この行は、システムクロックを設定する機能をデフォルトの特権セットに追加します。

9 スケジューリングクラスを **FSS** に設定します。

```
zonecfg:my-zone> set scheduling-class=FSS
```

10 メモリー上限を追加します。

```
zonecfg:my-zone> add capped-memory
```

a. メモリー上限を設定します。

```
zonecfg:my-zone:capped-memory> set physical=1g
```

b. スワップメモリーの上限を設定します。

```
zonecfg:my-zone:capped-memory> set swap=2g
```

c. ロックされたメモリーの上限を設定します。

```
zonecfg:my-zone:capped-memory> set locked=500m
```

d. メモリー上限の指定を終了します。

```
zonecfg:my-zone:capped-memory> end
```

注 – capped-memory リソースを使用するには、resource-cap パッケージが大域ゾーン内にインストールされている必要があります。

---

11 ファイルシステムを追加します。

```
zonecfg:my-zone> add fs
```

a. ファイルシステムのマウントポイント (この手順では **/usr/local**) を設定します。

```
zonecfg:my-zone:fs> set dir=/usr/local
```

b. 大域ゾーン内の **/opt/local** を、構成中のゾーン内で **/usr/local** としてマウントすることを指定します。

```
zonecfg:my-zone:fs> set special=/opt/local
```

非大域ゾーン内では、/usr/local ファイルシステムは読み取りおよび書き込みが可能です。

c. ファイルシステムのタイプ (この手順では **lofs**) を指定します。

```
zonecfg:my-zone:fs> set type=lofs
```

このタイプは、カーネルとそのファイルシステムとの相互動作の方法を示します。

d. ファイルシステムの指定を終了します。

```
zonecfg:my-zone:fs> end
```

この手順を複数回実行することで、複数のファイルシステムを追加できます。

12 必要に応じて **hostid** を設定します。

```
zonecfg:my-zone> set hostid=80f0c086
```

13 ストレージプール *tank* の *sales* という **ZFS** データセットを追加します

```
zonecfg:my-zone> add dataset
```

a. **ZFS** データセット *sales* のパスを指定します。

```
zonecfg:my-zone> set name=tank/sales
```

b. データセットの指定を終了します。

```
zonecfg:my-zone> end
```

ゾーン管理者は、データセット内のスナップショット、ファイルシステム、ボリュームを作成および破棄できます。ゾーン管理者は、データセットのプロパティーを変更したり、圧縮や暗号化を制御したりできます。

14 自動 **VNIC** を持つ排他的 **IP** ゾーンを作成します。

```
zonecfg:my-zone> set ip-type=exclusive

zonecfg:my-zone> add anet
```

a. 作成されるリンクの基礎となるリンクとして **auto** を指定します。

```
zonecfg:my-zone:anet> set lower-link=auto
```

zoneadmd デーモンは、ゾーンがブートするたびに、VNIC が作成されるリンクを自動的に選択します。データリンクを選択するときは、IPoIB リンクはスキップされます。

b. 指定を終了します。

```
zonecfg:my-zone:anet> end
```

15 デバイスを追加します。

```
zonecfg:my-zone> add device
```

a. デバイスの一致 **(**この手順では **/dev/sound/*)** を設定します。

```
zonecfg:my-zone:device> set match=/dev/sound/*
```

b. デバイスの指定を終了します。

```
zonecfg:my-zone:device> end
```

この手順を複数回実行することで、複数のデバイスを追加できます。

16 **IB** 診断ツール以外の **OFUV (Open Fabrics User Verbs)** のコンポーネントに対する **OFUV** デバイスを追加します。

```
zonecfg:my-zone> add device
```

a. デバイスの一致 **(**この手順では **infiniband/ofs/*)** を設定します。

```
zonecfg:my-zone:device> set match=infiniband/ofs/*
```

b. デバイスの指定を終了します。

```
zonecfg:my-zone:device> end
```

IB 診断ツールは、非大域ゾーンではサポートされていません。追加されたデバイスは、OFUV のコンポーネント (Verb や rdma_cm など) とともに使用できます。

この手順を複数回実行することで、複数のデバイスを追加できます。

17 **IB** 診断ツール以外の **OFUV** のコンポーネントに対する **OFUV** デバイスを追加します。

```
zonecfg:my-zone> add device
```

a. デバイスの一致 **(**この手順では **infiniband/hca/*)** を設定します。

```
zonecfg:my-zone:device> set match=infiniband/hca/*
```

b. デバイスの指定を終了します。

```
zonecfg:my-zone:device> end
```

IB 診断ツールは、非大域ゾーンではサポートされていません。追加されたデバイスは、OFUV のコンポーネント (Verb や rdma_cm など) とともに使用できます。

この手順を複数回実行することで、複数のデバイスを追加できます。

18 **format** コマンドによるディスクのラベル付けを可能にするには、ディスクおよび **LUN** 全体をゾーンに委任し、また **allow-partition** プロパティーを設定している必要があります。

```
zonecfg:my-zone> add device
```

a. この手順で、デバイスの一致 **/dev/*dsk/c2t40d3*** を設定します。

```
zonecfg:my-zone:device> set match=/dev/*dsk/c2t40d3*
```

b. **allow-partition** を **true** になるように設定します。

```
zonecfg:my-zone:device> set allow-partition=true
```

c. デバイスの指定を終了します。

```
zonecfg:my-zone:device> end
```

この手順を複数回実行することで、複数のデバイスを追加できます。

19 ディスクに対して **uscsi** 操作を可能にするには、**allow-raw-io** プロパティーを設定している必要があります。

```
zonecfg:my-zone> add device
```

a. この手順で、デバイスの一致 **/dev/*dsk/c2t40d3*** を設定します。

```
zonecfg:my-zone:device> set match=/dev/*dsk/c2t40d3*
```

b. **allow-raw-io** を **true** になるように設定します。

```
zonecfg:my-zone:device> set allow-raw-io=true
```

c. デバイスの指定を終了します。

```
zonecfg:my-zone:device> end
```

注意 – ゾーンでディスクに対する uscsi 操作の実行を可能にすると、ゾーンはディスクと同じバスに接続されている、すべての他のデバイスにもアクセスできます。したがって、この機能を有効にするとセキュリティーリスクが生じ、同じバス上のリソースを使用する大域ゾーンまたはほかのゾーンに対する攻撃が可能になります。uscsi(7I) を参照してください。

この手順を複数回実行することで、複数のデバイスを追加できます。

20 プロパティー名を使用して、ゾーン規模のリソース制御を追加します。

```
zonecfg:my-zone> set max-sem-ids=10485200
```

この手順を複数回実行することで、複数のリソース制御を追加できます。

21 リソースタイプ **attr** を使用してコメントを追加します。

```
zonecfg:my-zone> add attr
```

a. 名前を **comment** に設定します。

```
zonecfg:my-zone:attr> set name=comment
```

b. タイプを **string** に設定します。

```
zonecfg:my-zone:attr> set type=string
```

c. 値をゾーンの内容を示すコメントに設定します。

```
zonecfg:my-zone:attr> set value="This is my work zone."
```

d. リソースタイプ **attr** の指定を終了します。

```
zonecfg:my-zone:attr> end
```

22 ゾーンの構成を検証します。

```
zonecfg:my-zone> verify
```

23 ゾーンの構成を確定します。

```
zonecfg:my-zone> commit
```

24 **zonecfg** コマンドを終了します。

```
zonecfg:my-zone> exit
```

プロンプトで `commit` コマンドを明示的に入力しなくても、`exit` を入力するか EOF が発生すると、`commit` の実行が自動的に試みられます。

参考 コマンド行での複数のサブコマンドの使用

ヒント – `zonecfg` コマンドは、複数のサブコマンドもサポートします。次に示すように、同じシェル呼び出しで引用符で囲み、セミコロンで区切ります。

```
global# zonecfg -z my-zone "create ; set zonepath=/zones/my-zone"
```

共有 IP ゾーンの場合、`zonecfg net` リソースで静的アドレスを割り当てできるだけです。このアドレスは、コマンド行で指定することはできません。

次に進む手順

確定済みのゾーン構成をインストールする方法については、299 ページの「ゾーンのインストールとブート」を参照してください。

複数のゾーンを構成するスクリプト

このスクリプトを使用して、システムで複数のゾーンを構成およびブートできます。作成されるゾーンは、anet リソースを使用したデフォルトの排他的 IP ゾーンです。

スクリプトを実行する前に、次のように SCI Tool を実行して構成プロファイルを作成します。

```
global# sysconfig create-profile -o sc_config.xml
```

スクリプトには、次のパラメータを指定します。

- 作成するゾーンの数
- 接頭辞 *zonename*
- 基本ディレクトリとして使用するディレクトリ
- 新しく作成される構成プロファイルのフルパス名

大域ゾーン内の root 権限を持つ大域管理者か、またはこのスクリプトを実行するための正しい権利プロファイルを持つユーザーである必要があります。

```
#!/bin/ksh
#
# Copyright 2006-2012 Oracle Corporation. All rights reserved.
# Use is subject to license terms.
#

#
# This script serves as an example of how to instantiate several zones
# with no administrative interaction.  Run the script with no arguments to
# get a usage message.  The general flow of the script is:
#
#   1) Parse and check command line arguments
#   2) Configure all zones that are not yet configured
#   3) Install the first zone, if needed
#   4) Create the remaining zones as clones of the first zone
#
# Upon successful completion, the requested number of zones will be
# been installed and booted.
#

export PATH=/usr/bin:/usr/sbin

me=$(basename $0)
function fail_usage {
    print -u2 "Usage:
```

```
    $me <#-of-zones> <zonename-prefix> <basedir> <sysconfig.xml>

Generate sysconfig.xml with:
    sysconfig create-profile -o sysconfig.xml

When running sysconfig, choose \"Automatically\" or \"None\" for network
configuration. The value entered for \"Computer Name\" will ignored:
each zone's nodename will be set to match the zone name."

    exit 2
}

function log {
    print "$(date +%T) $@"
}

function error {
    print -u2 "$me: ERROR: $@"
}

function get_zone_state {
    zoneadm -z "$1" list -p 2>/dev/null | cut -d: -f3
}

#
# Parse and check arguments
#
(( $# != 4 )) && fail_usage

# If $1 is not a number nzones will be set to 0.
integer nzones=$1
if (( nzones < 1 )); then
    error "Invalid number of zones \"$1\""
    fail_usage
fi
# Be sure that zonename prefix is an allowable zone name and not too long.
prefix=$2
if [[ $prefix != @([a-zA-Z0-9])*([-_.a-zA-Z0-9]) || ${#prefix} > 62 ]]; then
    error "Invalid zonename prefix"
    fail_usage
fi
# Be sure that basedir is an absolute path.  zoneadm will create the directory
# if needed.
dir=$3
if [[ $dir != /* ]]; then
    error "Invalid basedir"
    fail_usage
fi
# Be sure the sysconfig profile is readable and ends in .xml
sysconfig=$4
if [[ ! -f $sysconfig || ! -r $sysconfig || $sysconfig != *.xml ]]; then
    error "sysconfig profile missing, unreadable, or not *.xml"
    fail_usage
fi

#
# Create a temporary directory for all temp files
#
export TMPDIR=$(mktemp -d /tmp/$me.XXXXXX)
```

```
if [[ -z $TMPDIR ]]; then
    error "Could not create temporary directory"
    exit 1
fi
trap 'rm -rf $TMPDIR' EXIT

#
# Configure all of the zones
#
for (( i=1; i <= nzones; i++ )); do
    zone=$prefix$i
    state=$(get_zone_state $zone)
    if [[ -n $state ]]; then
        log "Skipping configuration of $zone: already $state"
        continue
    fi

    log "Configuring $zone"
    zonecfg -z "$zone" "create; set zonepath=$dir/$zone"
    if (( $? != 0 )); then
        error "Configuration of $zone failed"
        exit 1
    fi
done

#
# Install the first zone, then boot it for long enough for SMF to be
# initialized.  This will make it so that the first boot of all the clones
# goes much more quickly.
#
zone=${prefix}1
state=$(get_zone_state $zone)
if [[ $state == configured ]]; then
    log "Installing $zone"

    # Customize the nodename in the sysconfig profile
    z_sysconfig=$TMPDIR/$zone.xml
    search="<propval type=\"astring\" name=\"nodename\" value=\".*\"/>"
    replace="<propval type=\"astring\" name=\"nodename\" value=\"$zone\"/>"
    sed "s|$search|$replace|" $sysconfig > $z_sysconfig

    zoneadm -z $zone install -c $z_sysconfig
    if (( $? != 0 )); then
        error "Installation of $zone failed."
        rm -f $z_sysconfig
        exit 1
    fi
    rm -f $z_sysconfig
elif [[ $state != installed ]]; then
    error "Zone $zone is currently in the $state state."
    error "It must be in the installed state to be cloned."
    exit 1
fi
# Boot the zone no further than single-user.  All we really want is for
# svc:/system/manifest-import:default to complete.
log "Booting $zone for SMF manifest import"
zoneadm -z $zone boot -s
if (( $? != 0 )); then
    error "Failed to boot zone $zone"
```

```
    exit 1
fi
# This zlogin will return when manifest-import completes
log "Waiting for SMF manifest import in $zone to complete"
state=
while [[ $state != online ]]; do
    printf "."
    sleep 1
    state=$(zlogin $zone svcs -Ho state \
        svc:/system/manifest-import:default 2>/dev/null)
done
printf "\n"
log "Halting $zone"
zoneadm -z $zone halt
if (( $? != 0 )); then
    error "failed to halt $zone"
    exit 1
fi
firstzone=$zone

#
# Clone and boot the remaining zones
#
for (( i=2; i <= $nzones; i++ )); do
    zone=$prefix$i

    # Be sure that it needs to be installed
    state=$(get_zone_state $zone)
    if [[ $state != configured ]]; then
        log "Skipping installation of $zone: current state is $state."
        continue
    fi

    log "Cloning $zone from $firstzone"

    # Customize the nodename in the sysconfig profile
    z_sysconfig=$TMPDIR/$zone.xml
    search='<propval type="astring" name="nodename" value=".*"/>'
    replace='<propval type="astring" name="nodename" value="'$zone'"/>'
    sed "s|$search|$replace|" $sysconfig > $z_sysconfig

    # Clone the zone
    zoneadm -z $zone clone -c $z_sysconfig $firstzone
    if (( $? != 0 )); then
        error "Clone of $firstzone to $zone failed"
        rm -f $z_sysconfig
        exit 1
    fi
    rm -f $z_sysconfig

    # Boot the zone
    log "Booting $zone"
    zoneadm -z $zone boot
    if (( $? != 0 )); then
        error "Boot of $zone failed"
        exit 1
    fi
done
```

```
#
# Boot the first zone now that clones are done
#
log "Booting $firstzone"
zoneadm -z $firstzone boot
if (( $? != 0 )); then
    error "Boot of $firstzone failed"
    exit 1
fi

log "Completed in $SECONDS seconds"
exit 0
```

スクリプトの出力:

```
$ ./buildzones
Usage:
        buildzones <#-of-zones> <zonename-prefix> <basedir> <sysconfig.xml>

Generate sysconfig.xml with:
        sysconfig create-profile -o sysconfig.xml

When running sysconfig, choose "Automatically" or "None" for network
configuration. The value entered for "Computer Name" will be ignored:
each zone's nodename will be set to match the zone name.

# ~user/scripts/buildzones 3 bz /tank/bz /var/tmp/sysconfig.xml
12:54:04 Configuring bz1
12:54:05 Configuring bz2
12:54:05 Configuring bz3
12:54:05 Installing bz1
A ZFS file system has been created for this zone.
Progress being logged to /var/log/zones/zoneadm.20110816T195407Z.bz1.install
       Image: Preparing at /tank/bz/bz1/root.

 Install Log: /system/volatile/install.24416/install_log
AI Manifest: /usr/share/auto_install/manifest/zone_default.xml
  SC Profile: /tmp/buildzones.F4ay4T/bz1.xml
    Zonename: bz1
Installation: Starting ....
```

▼ 非大域ゾーンの構成を表示する方法

大域ゾーン内の大域管理者か、またはこの手順を実行するための正しい権利プロファイルを持つユーザーである必要があります。

1. **root** になるか、同等の役割になります。
2. ゾーンの構成を表示します。

```
global# zonecfg -z zonename info
```

zonecfg コマンドを使用してゾーン構成を変更する

zonecfg コマンドを使用して、次の操作を実行することもできます。

- ゾーン構成内のリソースタイプを変更します
- ゾーン構成内のプロパティーの値をクリアーします
- 複製したデバイスをゾーンに追加します
- ゾーンの特権セットを変更します
- ストレージを追加および削除します

▼ ゾーン構成内のリソースタイプを変更する方法

リソースタイプを選択して、そのリソースの仕様を変更できます。

大域ゾーン内の大域管理者か、またはこの手順を実行するための正しい権利プロファイルを持つユーザーである必要があります。

1. **root** になるか、同等の役割になります。
2. 変更するゾーン (この手順では **my-zone**) を選択します。

   ```
   global# zonecfg -z my-zone
   ```
3. 変更するリソースタイプ (リソース制御など) を選択します。

   ```
   zonecfg:my-zone> select rctl name=zone.cpu-shares
   ```
4. 現在の値を削除します。

   ```
   zonecfg:my-zone:rctl> remove value (priv=privileged,limit=20,action=none)
   ```
5. 新しい値を追加します。

   ```
   zonecfg:my-zone:rctl> add value (priv=privileged,limit=10,action=none)
   ```
6. 改定された **rctl** の指定を終了します。

   ```
   zonecfg:my-zone:rctl> end
   ```
7. ゾーンの構成を確定します。

   ```
   zonecfg:my-zone> commit
   ```
8. **zonecfg** コマンドを終了します。

   ```
   zonecfg:my-zone> exit
   ```

   プロンプトで commit コマンドを明示的に入力しなくても、exit を入力するか EOF が発生すると、commit の実行が自動的に試みられます。

   zonecfg で行なった確定済みの変更は、ゾーンの次回ブート時に適用されます。

▼ ゾーン構成内のプロパティーをクリアーする方法

スタンドアロンのプロパティーをリセットするには、この手順を使用します。

1. **root** になるか、同等の役割になります。
2. 変更するゾーン (この手順では **my-zone**) を選択します。

   ```
   global# zonecfg -z my-zone
   ```

3. 変更対象のプロパティー (この手順では既存のプールの関連付け) をクリアーします。

   ```
   zonecfg:my-zone> clear pool
   ```

4. ゾーンの構成を確定します。

   ```
   zonecfg:my-zone> commit
   ```

5. **zonecfg** コマンドを終了します。

   ```
   zonecfg:my-zone> exit
   ```

   プロンプトで `commit` コマンドを明示的に入力しなくても、`exit` を入力するか EOF が発生すると、`commit` の実行が自動的に試みられます。

   `zonecfg` で行なった確定済みの変更は、ゾーンの次回ブート時に適用されます。

▼ ゾーンの名前を変更する方法

この手順を使用すると、構成済み状態またはインストール済み状態にあるゾーンの名前を変更できます。

`zonename` は既存の zpool 名の一部であるため、`rootzpool` または `zpool` リソースを含むインストール済み状態のゾーンの名前は変更できません。これらのゾーンの名前を変更するには、この手順の最後の「共有ストレージ上のゾーンの名前変更」を参照してください。

大域ゾーン内の大域管理者か、またはこの手順を実行するための正しい権利プロファイルを持つユーザーである必要があります。

1. **root** になるか、同等の役割になります。
2. 名前を変更するゾーン (この手順では **my-zone**) を選択します。

   ```
   global# zonecfg -z my-zone
   ```

3. ゾーンの名前を **newzone** などに変更します。

   ```
   zonecfg:my-zone> set zonename=newzone
   ```

4 変更を確定します。

```
zonecfg:newzone> commit
```

5 **zonecfg** コマンドを終了します。

```
zonecfg:newzone> exit
```

zonecfg で行なった確定済みの変更は、ゾーンの次回ブート時に適用されます。

参考 共有ストレージ上のゾーンの名前変更

zonename は既存の zpool 名の一部であるため、rootzpool または zpool リソースを含むインストール済み状態のゾーンの名前は変更できません。インストール済みでオンラインの zpool を含む共有ストレージ上のゾーンの名前を変更するには、次の手順を実行します。この手順では、my-zone ゾーンの名前を変更します。

- ゾーン (この手順では my-zone) を切り離します。

  ```
  # zoneadm -z my-zone detach
  ```
- zonecfg コマンドを使用してゾーンの名前を変更します。

  ```
  # zonecfg -z my-zone ; "set zonename=newname ; set zonepath=/store/newname"
  ```
- zoneadm attach を使用してゾーンを再接続します。

  ```
  # zoneadm -z newname attach
  ```

▼ 専用のデバイスをゾーンに追加する方法

次に、非大域ゾーン構成内に走査デバイスを配置する手順を示します。

大域ゾーン内の大域管理者か、またはこの手順を実行するための適切な承認を持つユーザーである必要があります。

1 **root** になるか、同等の役割になります。

2 デバイスを追加します。

```
zonecfg:my-zone> add device
```

3 デバイスの一致 (この手順では **/dev/scsi/scanner/c3t4***) を設定します。

```
zonecfg:my-zone:device> set match=/dev/scsi/scanner/c3t4*
```

4 デバイスの指定を終了します。

```
zonecfg:my-zone:device> end
```

5 **zonecfg** コマンドを終了します。

```
zonecfg:my-zone> exit
```

▼ 大域ゾーンの zone.cpu-shares を設定する方法

大域ゾーンの配分を永続的に設定する場合に、ここで説明する手順を使用します。

大域ゾーン内の大域管理者か、またはこの手順を実行するための正しい権利プロファイルを持つ大域ゾーン内のユーザーである必要があります。

1. **root** になるか、同等の役割になります。
2. **zonecfg** コマンドを使用します。

   ```
   # zonecfg -z global
   ```

3. 大域ゾーンの配分を **5** に設定します。

   ```
   zonecfg:global> set cpu-shares=5
   ```

4. **zonecfg** を終了します。

   ```
   zonecfg:global> exit
   ```

zonecfg コマンドを使用してゾーン構成を元に戻す、または削除する

zonecfg(1M) のマニュアルページの記述に従って、zonecfg コマンドを使用し、ゾーンの構成を元に戻すか、またはゾーン構成を削除します。

▼ ゾーン構成を元に戻す方法

zonecfg ユーティリティーによるゾーンの構成中にゾーン構成に対して行なったリソース設定を取り消すには、revert サブコマンドを使用します。

大域ゾーン内の大域管理者か、またはこの手順を実行するためのゾーンのセキュリティー権利プロファイルを持つ大域ゾーン内のユーザーである必要があります。

1. **root** になるか、同等の役割になります。
2. **tmp-zone** という名前のゾーンを構成中に、**info** と入力して構成を表示します。

   ```
   zonecfg:tmp-zone> info
   ```

   構成の net リソースセグメントが、次のように表示されます。

   ```
   .
   .
   .
   fs:
   ```

```
        dir: /tmp
        special: swap
        type: tmpfs
net:
        address: 192.168.0.1
        physical: eri0
device
        match: /dev/pts/*
.
.
.
```

3 ネットアドレスを削除します。

```
zonecfg:tmp-zone> remove net address=192.168.0.1
```

4 **net** エントリが削除されたことを確認します。

```
zonecfg:tmp-zone> info

.
.
.
fs:
        dir: /tmp
        special: swap
        type: tmpfs
device
        match: /dev/pts/*
.
.
.
```

5 **revert** と入力します。

```
zonecfg:tmp-zone> revert
```

6 次の質問に **yes** で応答します。

```
Are you sure you want to revert (y/[n])? y
```

7 ネットアドレスが再び存在することを確認します。

```
zonecfg:tmp-zone> info

.
.
.
fs:
        dir: /tmp
        special: swap
        type: tmpfs
net:
        address: 192.168.0.1
        physical: eri0
device
        match: /dev/pts/*
.
```

.
.

▼ ゾーン構成を削除する方法

zonecfg と delete サブコマンドを使用して、システムからゾーン構成を削除します。

大域管理者か、またはこの手順を実行するためのセキュリティー権利プロファイルを持つ大域ゾーン内のユーザーである必要があります。

1 **root** になるか、同等の役割になります。

2 次の **2** つの方法のいずれかを使用して、ゾーン **a-zone** のゾーン構成を削除します。

- -F オプションを使用して操作を強制実行します。

```
global# zonecfg -z a-zone delete -F
```

- システムプロンプトに yes で応答し、対話的にゾーンを削除します。

```
global# zonecfg -z a-zone delete
Are you sure you want to delete zone a-zone (y/[n])? y
```

◆ ◆ ◆ 第 18 章

18 非大域ゾーンのインストール、停止処理、停止、アンインストール、クローニングについて (概要)

この章では、Oracle Solaris システムへのゾーンのインストールについて説明します。また、仮想プラットフォームおよびアプリケーション環境を管理する 2 つのプロセス zoneadmd および zsched についても説明します。ゾーンの停止、リブート、クローニング、およびアンインストールに関する情報も提供します。

この章の内容は次のとおりです。

- 287 ページの「ゾーンのインストールと管理の概要」
- 288 ページの「ゾーンの構築」
- 292 ページの「zoneadmd デーモン」
- 293 ページの「zsched ゾーンスケジューラ」
- 293 ページの「ゾーンアプリケーション環境」
- 293 ページの「ゾーンの停止処理、停止、リブート、およびアンインストールについて」
- 296 ページの「非大域ゾーンのクローニングについて」

非大域ゾーンのクローン、非大域ゾーンのインストールとブート、および非大域ゾーンの停止やアンインストールの手順については、第 19 章「非大域ゾーンのインストール、ブート、停止処理、停止、アンインストール、およびクローニング (タスク)」を参照してください。

solaris10 ブランドゾーンのインストールの詳細は、第 33 章「solaris10 ブランドゾーンのインストール」を参照してください。

ゾーンのインストールと管理の概要

zoneadm コマンド (zoneadm(1M) のマニュアルページに記載) は、非大域ゾーンをインストールおよび管理するための主要なツールです。zoneadm コマンドを使用する操作は、大域ゾーンから実行する必要があります。役割に基づくアクセス制御 (RBAC) を使用中の場合、別のゾーンのコピーを作成するサブコマンドには、solaris.zone.clonefrom/*source_zone* の承認が必要です。

zoneadm コマンドを使用すると、次のタスクを実行できます。

- ゾーンを検証します
- ゾーンをインストールします
- ゾーンを接続します
- インストールされているゾーンの状態を「不完全」に変更します。
- ゾーンをブートします。これは、通常の Oracle Solaris システムのブートに似ています。
- 稼働中のゾーンに関する情報を表示します
- ゾーンを停止処理します
- ゾーンを停止します
- ゾーンをリブートします
- ゾーンをアンインストールします
- 同じシステム上で、ゾーンを別の場所へ再配置します
- 同一システムの既存ゾーンの構成に基づいて、新しいゾーンをプロビジョニングします
- ゾーンを移行します。zonecfg コマンドとともに使用します

ゾーンのインストールおよび検証手順については、第 19 章「非大域ゾーンのインストール、ブート、停止処理、停止、アンインストール、およびクローニング (タスク)」および zoneadm(1M) のマニュアルページを参照してください。zoneadm list コマンドでサポートされるオプションについては、zoneadm(1M) のマニュアルページも参照してください。ゾーンの構成手順については、第 17 章「非大域ゾーンの計画と構成 (タスク)」および zonecfg(1M) のマニュアルページを参照してください。ゾーンの状態については、207 ページの「非大域ゾーンの状態モデル」に記載されています。

ゾーンの Oracle Solaris 監査レコードの生成を計画している場合は、非大域ゾーンをインストールする前に 383 ページの「ゾーン内での Oracle Solaris 監査の使用」を参照してください。

ゾーンの構築

このセクションの内容は、既存のゾーンのクローニングにではなく、初期のゾーン構築に適用されます。

ゾーンは、zoneadm install - m コマンドに渡されるマニフェストによって指定されたパッケージを使用してインストールされます。マニフェストが指定されない場合、デフォルトのマニフェストは pkg:/group/system/solaris-small-server を使用します。新しいゾーンは、デフォルトの solaris 構成とログ (SMF リポジトリの

/etc、/var) を保持し、これらは、zoneadm install -s に渡されたプロファイル、およびすべての zonecfg add net エントリに指定されたネットワーク接続情報によって変更されるだけです。

システムリポジトリ、ゾーンの構成された発行元、および大域ゾーンとの同期が維持されたパッケージについては、第 24 章「ゾーンがインストールされている Oracle Solaris 11.1 システムでの自動インストールおよびパッケージ」を参照してください。

ゾーンのルートファイルシステムに必要とされるファイルは、システムによりゾーンのルートパス内にインストールされます。

正常にインストールされたゾーンは、ブートと初期ログインの準備が整っています。

ゾーンのインストール時に、次の場所からデータが参照またはコピーされます。

- インストールされていないパッケージ
- CD および DVD 内のデータ
- ネットワークインストールイメージ

また、大域ゾーンに存在可能な、次のタイプの情報は、インストール中のゾーンにはコピーされません。

- /etc/passwd ファイル内の新規または変更されたユーザー
- /etc/group ファイル内の新規または変更されたグループ
- DHCP アドレスの割り当てなどのネットワークサービスの構成
- sendmail などのネットワークサービスのカスタマイズ
- ネームサービスなどのネットワークサービスの構成
- 新規または変更された crontab、プリンタ、およびメールファイル
- システムログ、メッセージ、およびアカウンティングファイル

Oracle Solaris 監査を使用する場合、ファイルへの変更が必要になる場合があります。詳細は、383 ページの「ゾーン内での Oracle Solaris 監査の使用」を参照してください。

ゾーンの状態がインストール済みから準備完了に移行する際、構成ファイルで指定されたリソースセットが追加されます。システムにより、一意のゾーン ID が割り当てられます。ファイルシステムがマウントされ、ネットワークインタフェースが設定され、デバイスが構成されます。準備完了状態に移行すると、仮想プラットフォームでユーザープロセスを開始する準備が整います。準備完了状態では、仮想プラットフォームを管理するため、zsched および zoneadmd プロセスが開始されます。

- sched に類似したシステムスケジューリングプロセスである zsched が、ゾーンに関連付けられたカーネルリソースの追跡に使用されます。
- zoneadmd は、ゾーン管理デーモンです。

準備完了状態のゾーンには、稼働中のユーザープロセスは存在しません。準備完了状態のゾーンと稼働中のゾーンの主な違いは、稼働中のゾーンでは 1 つ以上のプロセスが稼働している点です。詳細は、init(1M) のマニュアルページを参照してください。

ゾーンのインストール方法

solaris ブランドのインストーラは、次の方法を使用してゾーンのインストールをサポートします。

- デフォルトリポジトリの solaris publisher (http://pkg.oracle.com/solaris/release/)。
- Oracle Solaris リリースが稼働しているインストール済みシステムまたは solaris 非大域ゾーンのイメージ。

  システムイメージは、ZFS 送信ストリームにできます。サポートされているイメージには、ほかに cpio(1) アーカイブや pax(1) xustar アーカイブなどがあります。cpio アーカイブは gzip または bzip2 ユーティリティーで圧縮できます。また、システムのルートツリーの最上位パス、または既存のゾーンパスも使用できます。

  ゾーンをシステムまたは非大域ゾーンのイメージからインストールするには、-a または -d オプションのどちらかが必要です。必要に応じて、パッケージの更新が行われます。-a オプションも -d オプションも使用しない場合、ゾーンはソフトウェアリポジトリからインストールされます。
- ゾーンの BE (zoneadm install -z *zbe* を使用)。必要に応じて、パッケージの更新が行われます。

次の表に、インストーラオプションを示します。例のコマンド行については、300 ページの「構成済みのゾーンをインストールする方法」を参照してください。

オプション	説明
-m *manifest*	AI マニフェストは、ゾーンのインストール方法を定義する XML ファイルです。ファイルの引数は絶対パスで指定する必要があります。

オプション	説明
-c *profile* | *dir*	構成時に適用する、プロファイルまたはプロファイルのディレクトリを指定します。ファイルの引数は絶対パスで指定する必要があります。プロファイルが適用されると、構成手順が非対話的に実行されます。プロファイルを指定しないと、システムの構成に対話型システム構成ツールが使用されます。すべてのプロファイルは、.xml ファイル拡張子を持つ必要があります。-c にディレクトリオプションを指定すると、そのディレクトリ内のすべてが有効で、正しく形成された構成プロファイルである必要があります。
-a *archive*	非大域ゾーンをインストールするために使用されるアーカイブへのパス。アーカイブは gzip または bzip を使用して圧縮できます。-d オプションと -a オプションは両立しません。 -a *archive* オプションを使用すると、必要に応じてパッケージの更新が行われます。必要な場合は、zoneadm attach サブコマンドを使用して、ゾーンをその元のホストに再接続できます。
-d *path*	インストール済みシステムまたは非大域ゾーンのルートディレクトリのパス。必要に応じて、パッケージの更新が行われます。*path* にハイフン (-) を指定すると、zonepath にシステムイメージがすでに配置されているとみなされます。-d オプションと -a オプションは両立しません。
-p	ゾーンのインストール後にシステム ID を保持します。-p オプションと -u オプションは両立しません。
-s	サイレントインストールします。-s オプションと -v オプションは両立しません。
-u	インストール後にゾーンを構成解除し、ゾーンブート時に新しい構成を求めるプロンプトを表示します。-p オプションと -u オプションは両立しません。
-U	必要に応じて、すべてのパッケージを最新のバージョンに更新して、大域ゾーンにインストールされているパッケージとの互換性を持たせます。

オプション	説明
-v	インストールプロセスの詳細情報を出力します。-s オプションと -v オプションは両立しません。
-x	force-zpool-import を -x オプションとともに使用すると、使用中と見られる zpool が強制的にインポートされます。 ストレージオブジェクトに既存のパーティション、zpool、または UFS ファイルシステムが含まれる場合は、install が失敗し、エラーメッセージが表示されます。zoneadm install に対して -x オプションを使用すると、インストールが続行され、既存のデータが上書きされます。このオプションは、zpool create -f コマンドに似ています。 force-zpool-create-all を -x オプションとともに使用すると、すべての zpool リソースが強制的に作成されます。このオプションを特定の zpool または zpool のセットに制限するには、force-zpool-create=*zpoolname* を使用します。使用方法については、zoneadm(1M) のマニュアルページを参照してください。

zoneadmd デーモン

ゾーン管理デーモン zoneadmd は、ゾーンの仮想プラットフォーム管理用の主要なプロセスです。このデーモンは、ゾーンのブートおよび停止処理の管理も担当します。システム上のアクティブな (準備完了、稼働中、または停止処理中の) ゾーンごとに、1 つの zoneadmd プロセスが存在します。

zoneadmd デーモンは、ゾーン構成での指定に従ってゾーンを設定します。このプロセスには、次の処理が含まれます。

- ゾーン ID を割り当てて、zsched システムプロセスを開始する
- ゾーン規模のリソース制御を設定する
- ゾーン構成の指定に従ってゾーンのデバイスを準備する
- ネットワークインタフェースを設定する
- ループバックおよび従来のファイルシステムをマウントする
- ゾーンコンソールデバイスをインスタンス化および初期化する

zoneadmd デーモンが実行中でない場合、zoneadm によりこのデーモンが自動的に起動されます。このため、何らかの理由でこのデーモンが動作していない場合、zoneadm を呼び出してそのゾーンを管理すると zoneadmd を再起動します。

zoneadmd デーモンのマニュアルページは、zoneadmd(1M) です。

zsched ゾーンスケジューラ

アクティブなゾーンとは、準備完了状態、稼働状態、または停止処理状態のゾーンのことです。すべてのアクティブなゾーンには、カーネルプロセス zsched が関連付けられています。ゾーンのために処理を実行するカーネルスレッドは、zsched により所有されています。zsched プロセスにより、ゾーンサブシステムがカーネルスレッドをゾーンごとに追跡することが可能になります。

ゾーンアプリケーション環境

ゾーンアプリケーション環境の作成には、zoneadm コマンドが使用されます。

ゾーンの内部構成は、sysconfig インタフェースを使用して指定します。内部構成では、使用するネームサービス、デフォルトのロケールおよびタイムゾーン、ゾーンのルートパスワード、およびほかのアプリケーション環境特性を指定します。sysconfig インタフェースについては、『Oracle Solaris 11.1 システムのインストール』の第 6 章「Oracle Solaris インスタンスの構成解除または再構成」および sysconfig(1M) のマニュアルページに記載されています。ゾーンのデフォルトロケールおよびタイムゾーンは、大域設定には関係なく構成できます。

ゾーンの停止処理、停止、リブート、およびアンインストールについて

このセクションでは、ゾーンの停止、リブート、アンインストール、およびクローニングの概要について説明します。

ゾーンの停止処理

zoneadm shutdown c コマンドは、完全にゾーンを停止処理するために使用します。この操作は、ゾーン内で /usr/sbin/init 0 を実行するのと同等です。-r オプションも指定すると、ゾーンはリブートされます。サポートされるブートオプションについては、294 ページの「ゾーンのブート引数」を参照してください。

svc:/system/zones サービスは、大域ゾーンの停止処理時に、zoneadm shutdown を使用してゾーンを完全に停止処理します。

shutdown サブコマンドはゾーンが正常に停止処理されるまで待機します。この操作が妥当な時間内で完了しない場合、 zoneadm halt を使用してゾーンを強制的に停止できます。307 ページの「ゾーンの停止方法」を参照してください。

ゾーンを停止する

zoneadm halt コマンドを使用して、ゾーン内で実行されているすべてのプロセスを終了し、仮想プラットフォームを削除します。これにより、ゾーンはインストール済みの状態に戻されます。すべてのプロセスが終了し、デバイスが構成解除され、ネットワークインタフェースが破棄され、ファイルシステムがアンマウントされ、カーネルデータ構造が破棄されます。

halt コマンドにより、ゾーン内部の停止処理スクリプトが実行されることはありません。ゾーンを停止処理するには、293 ページの「ゾーンの停止処理」を参照してください。別の方法として、ゾーンにログインして停止処理を実行することもできます。330 ページの「zlogin を使用してゾーンを停止処理する方法」を参照してください。

停止操作に失敗する場合は、424 ページの「ゾーンが停止しない」を参照してください。

ゾーンをリブートする

zoneadm reboot コマンドを使用してゾーンをリブートします。ゾーンは停止し、その後再ブートします。ゾーンのリブート時に、ゾーン ID が変更されます。

ゾーンのブート引数

ゾーンでは、次のブート引数を zoneadm boot および reboot コマンドに使用できます。

- -i *altinit*
- -m *smf_options*
- -s

次の定義が適用されます。

-i *altinit*
: 最初のプロセスとなる代替実行可能ファイルを選択します。*altinit* は実行可能ファイルへの有効なパスでなければなりません。デフォルトの最初のプロセスについては、init(1M) のマニュアルページに記載されています。

-m *smf_options*
: SMF のブート動作を制御します。復元オプションとメッセージオプションという、2 種類のオプションがあります。メッセージオプションは、ブート中に表示されるメッセージの種類と数を決定します。サービスオプションは、システムのブートに使用されるサービスを決定します。

  復元オプションは次のとおりです。

debug　標準のサービス別出力と、ログに記録されるすべての svc.startd メッセージを出力します。

milestone=*milestone*　指定されたマイルストーンで定義されているサブグラフにブートします。有効なマイルストーンは、none、 single-user、 multi-user、 multi-user-server、 および all です。

メッセージオプションは次のとおりです。

quiet　標準のサービス別出力と、管理者の介入を必要とするエラーメッセージを出力します。

verbose　標準のサービス別出力と、詳細情報を提供するメッセージを出力します。

-s　マイルストーン svc:/milestone/single-user:default に対してのみブートします。このマイルストーンは、init のレベル s と同等です。

使用例については、304 ページの「ゾーンのブート方法」および 305 ページの「ゾーンをシングルユーザーモードでブートする方法」を参照してください。

Oracle Solaris サービス管理機能 (SMF) と init については、『Oracle Solaris の管理: 一般的なタスク』の第 6 章「サービスの管理 (概要)」、svc.startd(1M) および init(1M) を参照してください。

ゾーンの autoboot 設定

大域ゾーンのブート時にゾーンを自動的にブートするには、ゾーンの構成内の autoboot リソースプロパティーを true に設定します。デフォルトの設定は false です。

ゾーンを自動的にブートするには、ゾーンサービス svc:/system/zones:default も有効になっている必要があります。このサービスは、デフォルトで有効になっています。

pkg update を実行中の autoboot の設定については、348 ページの「ゾーンのパッケージの概要」を参照してください。

ゾーンのアンインストール

ゾーンのルートファイルシステム内のすべてのファイルをアンインストールする場合に、zoneadm uninstall コマンドを使用します。-F (force) オプションを合わせて指定しない限り、処理を続行する前に、コマンドプロンプトにより実行の確認が求められます。実行した操作を元に戻すことはできないため、uninstall コマンドは慎重に使用してください。

非大域ゾーンのクローニングについて

クローンを使用すると、システムの既存の構成済みおよびインストール済みゾーンをコピーして、新しいゾーンを同一のシステム上に迅速にプロビジョニングできます。少なくとも、複数のゾーンで同一であってはならないコンポーネントに対しては、プロパティーとリソースをリセットする必要があります。したがって、zonepath は常に変更する必要があります。さらに、共有 IP ゾーンの場合は、各 net リソースの IP アドレスが異なっている必要があります。排他的 IP ゾーンの場合は、各 net リソースの physical プロパティーが異なっている必要があります。通常、アプリケーション固有の構成はクローンで再構成する必要があります。たとえば、ゾーン内にデータベースインスタンスがあり、そのゾーンをクローニングする場合は、クローン内のデータベースインスタンスを、別のインスタンスとして自身を認識するように再構成しなければならない可能性があります。

- ゾーンのクローニングは、ゾーンのインストールほど時間がかかりません。
- 新規ゾーンには、パッケージの追加やファイルの変更など、ソースゾーンをカスタマイズする過程で加えられた変更がすべて含まれます。

ソースの zonepath とターゲットの zonepath が両方とも ZFS 上にあり、同じプールに含まれる場合、zoneadm clone コマンドは自動的に ZFS を使用してゾーンをクローンします。ZFS クローンを使用する場合、データが変更されるまでデータは実際にはコピーされません。したがって、最初のクローンにかかる時間はごくわずかです。zoneadm コマンドは、ソース zonepath の ZFS スナップショットを取得して、ターゲット zonepath を設定します。ZFS クローンの名前には、宛先ゾーンの zonepath が使用されます。

注 – ZFS クローンを使用してソースを複製できる場合でも、ZFS クローンは行わず ZFS の zonepath をコピーするように指定することができます。

詳細は、310 ページの「同一システム上での非大域ゾーンのクローニング」を参照してください。

◆ ◆ ◆ 第 19 章

19 非大域ゾーンのインストール、ブート、停止処理、停止、アンインストール、およびクローニング (タスク)

この章では、非大域ゾーンのインストールおよびブート方法について説明します。クローニングを使って同一のシステムにゾーンをインストールする方法についても説明します。ゾーンの停止、リブート、アンインストールなどのインストールに関連するほかのタスクについて説明します。既存の非大域ゾーンを同じマシンの新しい場所に移動します。また、既存の非大域ゾーンを同じマシン上の新しい場所に移動し、システムからゾーンを完全に削除する手順も含まれています。

ゾーンのインストールおよび関連する操作に関する一般的な情報については、第 18 章「非大域ゾーンのインストール、停止処理、停止、アンインストール、クローニングについて (概要)」を参照してください。

`solaris10` ブランドゾーンのインストールとクローンの詳細は、第 33 章「`solaris10` ブランドゾーンのインストール」を参照してください。

ゾーンのインストール(タスクマップ)

タスク	説明	参照先
(任意) ゾーンをインストールする前に、構成済みのゾーンを検証します。	ゾーンがインストール要件を満たしていることを確認します。この手順を省略した場合、ゾーンのインストール時に検証が自動的に実行されます。	299 ページの「(オプション) インストール前に構成済みのゾーンを検証する方法」
構成済みのゾーンをインストールします。	構成済みの状態にあるゾーンをインストールします。	300 ページの「構成済みのゾーンをインストールする方法」
ゾーンの汎用一意識別子 (UUID) を取得します。	ゾーンのインストール時に割り当てられるこの個別の識別子は、ゾーンを識別するための代替手段になります。	302 ページの「インストールされた非大域ゾーンの UUID を取得する方法」
(任意) インストール済みのゾーンを準備完了状態に移行します。	ゾーンをすぐにブートして使用する場合、この手順は省略できます。	303 ページの「(オプション) インストール済みのゾーンを準備完了状態に移行する方法」
ゾーンをブートします。	ゾーンをブートすると、ゾーンが稼働状態になります。ゾーンは、準備完了状態またはインストール済み状態からブートできます。	304 ページの「ゾーンのブート方法」
ゾーンをシングルユーザーモードでブートします。	マイルストーン `svc:/milestone/single-user:default` に対してのみブートします。このマイルストーンは、`init` のレベル `s` と同等です。`init(1M)` および `svc.startd(1M)` のマニュアルページを参照してください。	305 ページの「ゾーンをシングルユーザーモードでブートする方法」

ゾーンのインストールとブート

zoneadm(1M) のマニュアルページで説明されている zoneadm コマンドを使用して、非大域ゾーンのインストールタスクを実行します。ゾーンのインストールを実行するには、大域管理者または適切な承認を持つユーザーである必要があります。この章に示す例では、269 ページの「ゾーンを構成、検証、および確定する」で使用したゾーン名およびゾーンパスを使用します。

▼ (オプション) インストール前に構成済みのゾーンを検証する方法

ゾーンをインストールする前に検証できます。実行する検査の 1 つは、十分なディスクサイズがあるかどうかの検査です。この手順を省略した場合、ゾーンのインストール時に検証が自動的に実行されます。

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を持つユーザーである必要があります。

1 **root** になるか、同等の役割になります。

2 **-z** オプションをゾーン名および **verify** サブコマンドとともに使用して、**my-zone** という名前の構成済みゾーンを検証します。

```
global# zoneadm -z my-zone verify
```

ゾーンパスの検証に関する次のメッセージが表示されます。

```
WARNING: /zones/my-zone does not exist, so it could not be verified.
    When 'zoneadm install' is run, 'install' will try to create
    /zones/my-zone, and 'verify' will be tried again,
    but the 'verify' may fail if:
    the parent directory of /zones/my-zone is group- or other-writable
    or

    /zones/my-zone overlaps with any other installed zones
    or
    /zones/my-zone is not a mountpoint for a zfs file system.
```

ただし、エラーメッセージが表示され、ゾーンの検証に失敗した場合は、メッセージに従って修正を行い、コマンドを再度実行してください。

エラーメッセージが表示されない場合は、ゾーンをインストールできます。

参考 共有ストレージ上のゾーンの検証

共有ストレージに構成されているゾーンでは、ゾーンが構成済み状態の場合は、zonecfg verify によって、構成されているどの zpool リソースもシステム上でまだオンラインになっていないことが検証されます。

共有ストレージに構成されているゾーンでは、ゾーンがインストール済み状態の場合は、`zoneadm verify` コマンドによって、`zpool` および `rootzpool` リソースとして構成されたすべての zpool がシステム上でオンラインになっていることが検証されます。リソースが使用できない場合は、`verify` が失敗し、失敗した zpool に関する情報が表示されます。

▼ 構成済みのゾーンをインストールする方法

この手順を使用して、構成済みの非大域ゾーンをインストールします。インストールオプションについては、290 ページの「ゾーンのインストール方法」を参照してください。

ゾーンは、独自の ZFS データセット上に存在する必要があります。ZFS のみがサポートされます。`zoneadm install` コマンドは、ゾーンのインストール時に `zonepath` に対して ZFS ファイルシステム (データセット) を自動的に作成します。ZFS データセットを作成できない場合、ゾーンのインストールは行われません。

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を持つユーザーである必要があります。

1. **root** になるか、同等の役割になります。
2. **`zoneadm`** コマンドを **`install`** サブコマンドとともに使用し、**`zonepath`** **ZFS** に自動的に **ZFS** データセットを作成して、構成済みのゾーン **`my-zone`** をインストールします。ゾーンのパスの親ディレクトリもデータセットである必要があり、データセットでない場合はファイルシステムの作成が失敗することに注意してください。

   - ゾーンをインストールします。

     ```
     global# zoneadm -z my-zone install
     ```

   - リポジトリからゾーンをインストールします。

     ```
     global# zoneadm -z my-zone install -m manifest -c [ profile | dir ]
     ```

   - 次のコマンドを実行して、イメージからゾーンをインストールします。

     ```
     global# zoneadm -z my-zone install -a archive -s -u
     ```

   - 次のコマンドを実行して、ディレクトリからゾーンをインストールします。

     ```
     global# zoneadm -z my-zone install -d path -p -v
     ```

   このゾーンに対して ZFS ファイルシステムが作成されたことが表示されます。

   ゾーンのルートファイルシステムに必要なファイルおよびディレクトリがゾーンのルートパスにインストールされる際、さまざまなメッセージが表示されます。

3 (オプション)エラーメッセージが表示され、ゾーンのインストールに失敗した場合は、次のように入力してゾーンの状態を取得します。

```
global# zoneadm list -v
# zoneadm list -cvd
  ID NAME             STATUS      PATH                       BRAND    IP
   0 global           running     /                          solaris  shared
   - my-zone          configured  /zones/my-zone             solaris  excl
```

- 状態が構成済みであると表示された場合は、メッセージに示された修正を行い、zoneadm install コマンドを再度実行します。
- 状態が不完全であると表示された場合は、最初に次のコマンドを実行します。

  ```
  global# zoneadm -z my-zone uninstall
  ```

  メッセージに示された修正を行い、zoneadm install コマンドを再度実行します。

4 (オプション)ストレージオブジェクトに既存のパーティション、**zpool**、または **UFS** ファイルシステムが含まれる場合は、**install** が失敗し、エラーメッセージが表示されます。

force サブコマンドを使用する前に、ソースゾーンをアンインストール済みの状態にする必要があります。

```
zoneadm -z my-zone uninstall
```

その後、zoneadm install に対して -x オプションを使用して、インストールを続行し、既存のデータを上書きします。

```
-x force-zpool-import
-x force-zpool-create=zpoolname
-x force-zpool-create=zpoolname1,zpoolname2,zpoolname3
-x force-zpool-create-all
```

このオプションは、zpool create -f コマンドに似ています。

-x force-zpool-create=*zpoolname* は、1 回または複数回使用できます。

5 インストールが完了したら、**list** サブコマンドに **-i** オプションおよび **-v** オプションを指定してインストール済みのゾーンを一覧表示し、ステータスを確認します。

```
global# zoneadm list -iv
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris    shared
 -  my-zone  installed    /zones/my-zone                 solaris    excl
```

注意事項 ゾーンのインストールが中断または失敗した場合は、ゾーンの状態は不完全なままになります。uninstall -F を使用して、ゾーンを構成済みの状態にリセットします。

次の手順　このゾーンは、デフォルトでは、『Oracle Solaris の管理: 一般的なタスク』の第 7 章「サービスの管理 (手順)」で説明されている最小ネットワーク構成でインストールされています。ゾーンへのログイン時に、オープンなネットワーク構成に切り替えることも、個別のサービスを有効または無効に設定することもできます。詳細は、331 ページの「サービスの有効化」を参照してください。

▼ インストールされた非大域ゾーンの UUID を取得する方法

ゾーンのインストール時に、汎用一意識別子 (UUID) がゾーンに割り当てられます。UUID は、`zoneadm` に `list` サブコマンドと `-c -p` オプションを使用することで取得できます。UUID は、5 番目に表示されるフィールドです。

- インストールされたゾーンの **UUID** を表示します。

```
global# zoneadm list -cp
```

次のような情報が表示されます。

```
0:global:running:/::solaris:shared:-:none

6:my-zone:running:/zones/my-zone:61901255-35cf-40d6-d501-f37dc84eb504:solaris:excl:-:
```

例 19–1　コマンド内でゾーンの UUID を使用する方法

```
global# zoneadm -z my-zone -u 61901255-35cf-40d6-d501-f37dc84eb504:solaris:excl list -v
```

`-u` *uuid-match* と `-z` *zonename* の両方が存在する場合、最初に UUID に基づいてマッチングが行われます。指定した UUID のゾーンが見つかった場合はそのゾーンが使用され、`-z` パラメータは無視されます。指定した UUID のゾーンが見つからなかった場合、システムはゾーン名で検索を実行します。

参考　UUID について

ゾーンをアンインストールすることも、同名のゾーンを内容を変えて再インストールすることもできます。ゾーンの内容を変更せずにゾーンの名前を変更することも可能です。こうした理由から、UUID はゾーン名よりも信頼性の高いハンドルです。

参照　詳細は、`zoneadm(1M)` および `libuuid(3LIB)` を参照してください。

▼ インストールした非大域ゾーンに不完全のマークを付ける方法

システムに加えられた管理上の変更のためにゾーンが使用不可になるか、矛盾が生じた場合、インストールしたゾーンの状態を不完全に変更できます。

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を持つユーザーである必要があります。

1. **root** になるか、同等の役割になります。
2. ゾーン **testzone** に不完全のマークを付けます。

   ```
   global# zoneadm -z testzone mark incomplete
   ```

3. **list** サブコマンドに **-i** オプションと **-v** オプションを使って、ステータスを確認します。

   ```
   global# zoneadm list -iv
   ```

   次のような情報が表示されます。

   ```
   ID  NAME     STATUS       PATH                           BRAND      IP
    0  global   running      /                              solaris    shared
    -  my-zone  installed    /zones/my-zone                 solaris    excl
    -  testzone incomplete   /zones/testzone                solaris    excl
   ```

参考 ゾーンへの不完全のマーク付け

-R *root* オプションを、 zoneadm の mark サブコマンドや list サブコマンドとともに使用して、代替ブート環境を指定できます。詳細は、zoneadm(1M) のマニュアルページを参照してください。

注 – ゾーンへの不完全のマーク付けは、取り消すことができません。不完全のマークが付けられたゾーンに実行可能なのは、ゾーンをアンインストールして、構成済みの状態に戻す操作だけです。309 ページの「ゾーンをアンインストールする方法」を参照してください。

▼ (オプション)インストール済みのゾーンを準備完了状態に移行する方法

準備完了状態に移行すると、仮想プラットフォームでユーザープロセスを開始する準備が整います。準備完了状態のゾーンには、内部で実行中のユーザープロセスは存在しません。

ゾーンをすぐにブートして使用する場合、この手順は省略できます。ゾーンのブート時に、準備完了状態への移行が自動的に行われます。

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を持つユーザーである必要があります。

1 **root** になるか、同等の役割になります。

2 **zoneadm** コマンドを **-z** オプション、ゾーン名 (**my-zone**)、および **ready** サブコマンドとともに使用することで、そのゾーンを準備完了状態に移行します。

```
global# zoneadm -z my-zone ready
```

3 プロンプトで、**zoneadm list** コマンドに **-v** オプションを指定して、ステータスを確認します。

```
global# zoneadm list -v
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris    shared
 1  my-zone  ready        /zones/my-zone                 solaris    excl
```

システムにより一意のゾーン ID 1 が割り当てられていることに注目してください。

▼ ゾーンのブート方法

ゾーンをブートすると、ゾーンが稼働状態になります。ゾーンは、準備完了状態またはインストール済み状態からブートできます。ブートしたインストール済み状態のゾーンは、準備完了状態から稼働状態に透過的に移行します。稼働状態のゾーンに対してはゾーンへのログインが可能です。

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を持つユーザーである必要があります。

1 **root** になるか、同等の役割になります。

2 **zoneadm** コマンドを **-z** オプション、ゾーン名 (**my-zone**)、および **boot** サブコマンドとともに使用することで、ゾーンをブートします。

```
global# zoneadm -z my-zone boot
```

3 ブートが完了したら、**list** サブコマンドに **-v** オプションを指定してステータスを確認します。

```
global# zoneadm list -v
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris    shared
 1  my-zone  running      /zones/my-zone                 solaris    excl
```

例 19–2 ゾーンのブート引数を指定する

-m verbose オプションを使用してゾーンをブートします。

```
global# zoneadm -z my-zone boot -- -m verbose
```

-m verbose ブートオプションを使用してゾーンをリブートします。

```
global# zoneadm -z my-zone reboot -- -m verbose
```

ゾーン管理者が -m verbose オプションを使用してゾーン *my-zone* をリブートします。

```
my-zone# reboot -- -m verbose
```

▼ ゾーンをシングルユーザーモードでブートする方法

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を持つユーザーである必要があります。

1. **root** になるか、同等の役割になります。
2. ゾーンをシングルユーザーモードでブートします。

```
global# zoneadm -z my-zone boot -- -s
```

次に進む手順

ゾーンにログインして初期内部構成を実行する方法については、第 20 章「非大域ゾーンへのログイン (概要)」および第 21 章「非大域ゾーンへのログイン (タスク)」を参照してください。

非大域ゾーンの停止処理、停止、リブート、アンインストール、クローニング、および削除 (タスクマップ)

タスク	説明	参照先
ゾーンを停止処理します。	shutdown 手順を使用して、停止処理スクリプトを実行することによって、ゾーンを完全に停止処理します。また、zlogin の方法もサポートされています。詳細は、330 ページの「zlogin を使用してゾーンを停止処理する方法」を参照してください。	307 ページの「ゾーンの停止方法」
ゾーンを停止します。	停止手順を実行して、ゾーンのアプリケーション環境と仮想プラットフォームの両方を削除します。この手順により、ゾーンが準備完了状態からインストール済み状態に戻されます。ゾーンの完全な停止処理を行う方法については、330 ページの「zlogin を使用してゾーンを停止処理する方法」を参照してください。	307 ページの「ゾーンの停止方法」
ゾーンをリブートします。	リブートの手順を実行すると、ゾーンが停止してから再びブートします。	308 ページの「ゾーンをリブートする方法」
ゾーンをアンインストールします。	この手順は、ゾーンのルートファイルシステム内のすべてのファイルを削除します。「この手順は、十分注意して実行する必要があります。」実行した操作を元に戻すことはできません。	309 ページの「ゾーンをアンインストールする方法」
同一システムの既存ゾーンの構成に基づいて、新しい非大域ゾーンをプロビジョニングします。	ゾーンのクローニングは、ゾーンのインストールより高速な代替手段です。ただし、インストールの前にはやはり新規ゾーンを構成してください。	310 ページの「同一システム上での非大域ゾーンのクローニング」
システムから非大域ゾーンを削除します。	この手順を実行すると、システムからゾーンが完全に削除されます。	313 ページの「システムから非大域ゾーンを削除する」

ゾーンの停止処理、停止、リブート、およびアンインストール

▼ ゾーンを停止処理する方法

停止処理手順は、ゾーンを完全に停止処理します。

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を持つユーザーである必要があります。

1 **root** になるか、同等の役割になります。

2 システムで稼働中のゾーンの一覧を表示します。

```
global# zoneadm list -v
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris    shared
 1  my-zone  running      /zones/my-zone                 solaris    excl
```

3 **zoneadm** コマンドを **-z** オプション、ゾーン名 (**my-zone** など)、および **shutdown** サブコマンドとともに使用することで、指定されたゾーンを停止処理します。

```
global# zoneadm -z my-zone shutdown
```

4 また、**-r** オプションを指定してゾーンをリブートします。

```
global# zoneadm -z my-zone shutdown -r boot_options
```

例 19–2 を参照してください。

5 システム上で稼働するゾーンをリスト表示して、ゾーンが停止処理されたことを確認します。

```
global# zoneadm list -v
```

▼ ゾーンの停止方法

この停止手順を使用して、ゾーンのアプリケーション環境と仮想プラットフォームの両方を削除します。ゾーンの完全な停止処理を行う方法については、330 ページの「zlogin を使用してゾーンを停止処理する方法」を参照してください。

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を持つユーザーである必要があります。

1 **root** になるか、同等の役割になります。

2 システムで稼働中のゾーンの一覧を表示します。

```
global# zoneadm list -v
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris    shared
 1  my-zone  running      /zones/my-zone                 solaris    excl
```

3 **zoneadm** コマンドを **-z** オプション、ゾーン名 (**my-zone** など)、および **halt** サブコマンドとともに使用することで、指定されたゾーンを停止します。

```
global# zoneadm -z my-zone halt
```

4 システム内のゾーンの一覧を再度表示して、**my-zone** が停止していることを確認します。

```
global# zoneadm list -iv
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris    shared
 -  my-zone  installed    /zones/my-zone                 solaris    excl
```

5 ゾーンを再び起動する場合は、次のコマンドをブートします。

```
global# zoneadm -z my-zone boot
```

注意事項 ゾーンが正しく停止しない場合は、424 ページの「ゾーンが停止しない」でトラブルシューティングのヒントを参照してください。

▼ ゾーンをリブートする方法

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を持つユーザーである必要があります。307 ページの「ゾーンを停止処理する方法」も参照してください。

1 **root** になるか、同等の役割になります。

2 システムで稼働中のゾーンの一覧を表示します。

```
global# zoneadm list -v
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris    shared
 1  my-zone  running      /zones/my-zone                 solaris    excl
```

3 **zoneadm** コマンドを **-z reboot** オプションとともに使用することで、ゾーン **my-zone** をリブートします。

```
global# zoneadm -z my-zone reboot
```

4 システム内のゾーンの一覧を再度表示して、**my-zone** がリブートしたことを確認します。

```
global# zoneadm list -v
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris    shared
 2  my-zone  running      /zones/my-zone                 solaris    excl
```

ヒント – my-zone のゾーン ID が変更されていることに注目してください。通常、リブートするとゾーン ID は変更されます。

▼ ゾーンをアンインストールする方法

注意 – この手順は、注意して使用してください。ゾーンのルートファイルシステム内のファイルすべてを削除した後で、操作を元に戻すことはできません。

ゾーンは稼働状態であっていけません。uninstall 操作は、稼働中のゾーンに対しては無効です。

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を持つユーザーである必要があります。

1 **root** になるか、同等の役割になります。

2 システム内のゾーンの一覧を表示します。

```
global# zoneadm list -v
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris    shared
 -  my-zone  installed    /zones/my-zone                 solaris    excl
```

3 **zoneadm** コマンドを **-z uninstall** オプションとともに使用することで、ゾーン **my-zone** を削除します。

-F オプションを使用すると、処理を強制的に実行できます。このオプションが指定されていない場合、システムにより確認を求めるメッセージが表示されます。

```
global# zoneadm -z my-zone uninstall -F
```

zonepath として独自の ZFS ファイルシステムを保持しているゾーンをアンインストールすると、その ZFS ファイルシステムは破棄されます。

4 システム内のゾーンの一覧を再度表示して、**my-zone** が一覧に含まれていないことを確認します。

```
global# zoneadm list -iv
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris    shared
```

注意事項 ゾーンのアンインストールが中断した場合、ゾーンの状態は不完全なままになります。zoneadm uninstall コマンドを使用して、ゾーンを構成済みの状態にリセットしてください。

zonepath が削除されない場合、このゾーンが別のブート環境内にインストールされていることを示している可能性があります。特定の zonepath を保持するインストール済みゾーンを持つブート環境が存在するとき、zonepath やその zonepath データセット内に存在するさまざまなデータセットは削除されません。ブート環境に関する詳細は、beadm(1M) を参照してください。

実行した操作を元に戻すことはできないため、uninstall コマンドは慎重に使用してください。

同一システム上での非大域ゾーンのクローニング

クローニングは、ソースの zonepath からターゲットの zonepath にデータをコピーすることにより、システム上に新しいゾーンをプロビジョニングするために使用します。

ソースの zonepath とターゲットの zonepath が両方とも ZFS 上にあり、同じプールに含まれる場合、zoneadm clone コマンドは自動的に ZFS を使用してゾーンをクローンします。ただし、ZFS の zonepath のコピーは行い、ZFS のクローンは行わないように指定することもできます。

▼ ゾーンをクローンする方法

新規ゾーンをインストールする前に、そのゾーンを構成する必要があります。zoneadm create サブコマンドに渡されるパラメータは、クローンするゾーンの名前です。このソースゾーンを停止する必要があります。

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を持つユーザーである必要があります。

1 **root** になるか、同等の役割になります。

2 クローンされるソースゾーン (この手順では **my-zone**) を停止します。

```
global# zoneadm -z my-zone halt
```

3 ソースゾーン **my-zone** の構成をファイル (たとえば、**master**) にエクスポートすることにより、新規ゾーンの構成を開始します。

```
global# zonecfg -z my-zone export -f /zones/master
```

注 – 既存の構成を変更する代わりに、270 ページの「ゾーンの構成方法」で説明されている手順を使って、新規ゾーン構成を作成することもできます。この方法を使用する場合は、ゾーンを作成したあとで手順 6 に進みます。

---

4 **master** ファイルを編集します。複数のゾーンで同一であってはならないコンポーネントに対して、異なるプロパティーとリソースを設定します。たとえば、新しい **zonepath** を設定する必要があります。共有 **IP** ゾーンの場合は、各 **net** リソースの **IP** アドレスを変更する必要があります。排他的 **IP** ゾーンの場合は、各 **net** リソースの **physical** プロパティーを変更する必要があります。

5 *master* ファイル内のコマンドを使って、新規ゾーン **zone1** を作成します。

```
global# zonecfg -z zone1 -f /zones/master
```

6 **my-zone** をクローニングして、新規ゾーン **zone1** をインストールします。

```
global# zoneadm -z zone1 clone my-zone
```

システムには次のように表示されます。

```
Cloning zonepath /zones/my-zone...
```

7 (オプション) ストレージオブジェクトに既存のパーティション、**zpool**、または **UFS** ファイルシステムが含まれる場合は、**clone** が失敗し、エラーメッセージが表示されます。

操作を続行し、既存のデータを上書きするには、`zoneadm clone` に対して適切な `-x` オプションを使用します。`force` サブコマンドを使用する前に、ソースゾーンをアンインストールする必要があります。

-x force-zpool-import
-x force-zpool-create=*zpoolname*
-x force-zpool-create=*zpoolname1*,*zpoolname2*,*zpoolname3*
-x force-zpool-create-all

このオプションは、`zpool create -f` コマンドに似ています。

`-x force-zpool-create=`*zpoolname* オプションは、複数回使用できます。

`-x force` オプションを使用する前に、ソースゾーンを停止する必要があります。

8 システム内のゾーンの一覧を表示します。

```
ID  NAME     STATUS       PATH                          BRAND      IP
 0  global   running      /                             solaris    shared
 -  my-zone  installed    /zones/my-zone                solaris    excl
 -  zone1    installed    /zones/zone1                  solaris    excl
```

例 **19–3** システム構成プロファイルのクローンされたゾーンへの適用

構成プロファイルを含めるには、次のように入力します。

```
# zoneadm -z zone1 clone -c /path/config.xml my-zone
```

構成ファイルの絶対パスを指定する必要があります。

非大域ゾーンの移動

zonepath を変更してゾーンを同じシステムの新しい場所に移動する場合に、ここで説明する手順を使用します。ゾーンは、停止する必要があります。241 ページの「リソースタイプとプロパティー」で説明されている、zonepath の通常の基準が適用されます。

この情報は、solaris10 ブランドゾーンの移動にも適用されます。solaris10 ブランドゾーンの詳細は、パート III「Oracle Solaris 10 ゾーン」を参照してください。

注 – ほかの BE 内に存在しているゾーンを移動することはできません。最初にそれらの BE を削除でき、またそのゾーンをクローニングすることによって新しいパスに新しいゾーンを作成することもできます。

注 – rootzpool リソースを含む共有ストレージ上のゾーンをシステム上の別の場所に移動することはできません。zonepath の名前変更はサポートされています。

▼ 共有ストレージ上にないゾーンを移動する方法

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を持つユーザーである必要があります。

1 スーパーユーザーになるか、同等の承認を保持するようにします。

2 移動するゾーン (この手順では **db-zone**) を停止します。

```
global# zoneadm -z db-zone halt
```

3 **zoneadm** コマンドを **move** サブコマンドとともに使用して、ゾーンを新規の **zonepath** である **/zones/db-zone** に移動します。

```
global# zoneadm -z db-zone move /zones/db-zone
```

4 パスを確認します。

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris    shared
 -  my-zone  installed    /zones/my-zone                 solaris    excl
 -  db-zone  installed    /zones/db-zone                 solaris    excl
```

システムから非大域ゾーンを削除する

このセクションでは、システムからゾーンを完全に削除する手順を説明します。

▼ 非大域ゾーンを削除する方法

1 次のいずれかの方法を使用して、ゾーンの **my-zone** を停止処理します。**zoneadm shutdown** の方法をお勧めします。

- **zoneadm** を使用:

  ```
  global# zoneadm -z my-zone shutdown
  my-zone
  ```

- **zlogin** を使用:

  ```
  global# zlogin my-zone shutdown
  my-zone
  ```

2 **my-zone** のルートファイルシステムを削除します。

```
global# zoneadm -z my-zone uninstall -F
```

操作を強制実行する -F オプションは通常不要です。

3 **my-zone** の構成を削除します。

```
global# zonecfg -z my-zone delete -F
```

操作を強制実行する -F オプションは通常不要です。

4 システム内のゾーンの一覧を表示し、**my-zone** が一覧に含まれていないことを確認します。

```
global# zoneadm list -iv
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris    shared
```

◆ ◆ ◆ 第 20 章

20 非大域ゾーンへのログイン (概要)

この章では、大域ゾーンからゾーンへのログインについて説明します。

この章の内容は次のとおりです。

- 315 ページの「zlogin コマンド」
- 316 ページの「ゾーンの内部構成」
- 322 ページの「非大域ゾーンへのログイン方法」
- 324 ページの「対話型モードと非対話型モード」
- 323 ページの「フェイルセーフモード」
- 323 ページの「リモートログイン」

手順および使用法については、第 21 章「非大域ゾーンへのログイン (タスク)」を参照してください。使用できるオプションの一覧については、zlogin(1) のマニュアルページを参照してください。

zlogin コマンド

役割に基づくアクセス制御 (RBAC) を使用中の場合、ゾーンコンソールへのアクセスには、solaris.zone.manage/*zonename* の承認が必要です。スラッシュ文字 (/) が前に付いた特定の *zonename* の接尾辞はオプションです。省略すると、承認はすべてのゾーンと一致します。

-C オプションを使用してゾーンコンソールに接続しない限り、zlogin を使用してゾーンにログインすると、新しいタスクが開始されます。1 つのタスクを 2 つのゾーンで実行することはできません。

zlogin コマンドを使用して、大域ゾーンから稼働状態または準備完了状態にある任意のゾーンにログインします。

注 – 稼働状態にないゾーンへのログインに使用できるのは、zlogin コマンドと -C オプションの組み合わせだけです。

329 ページの「非対話型モードを使用してゾーンにアクセスする方法」に記述されているとおり、ゾーン内での実行を指示するコマンドを指定することで、zlogin コマンドを非対話型モードで使用できます。ただし、このコマンドおよびこのコマンドの処理対象となるファイルは、いずれも NFS 上に存在していてはなりません。開かれているファイルのいずれか、またはそのアドレス空間のいずれかの部分が NFS 上に存在する場合、コマンドは失敗します。アドレス空間には、コマンドの実行可能ファイル自体またはリンクされたライブラリが含まれます。

zlogin コマンドは、大域管理者または大域ゾーン内で操作している、適切な承認を持つユーザーだけによって使用できます。詳細は、zlogin(1) のマニュアルページを参照してください。

ゾーンの内部構成

システム構成データは、単一のプロファイル sc_profile.xml、または SMF プロファイルの profiles ディレクトリとして存在できます。単一のファイルまたはディレクトリの両方が、ゾーンのインストール時に自動化されたインストーラに渡される、ゾーンのシステム構成データについて記述します。ゾーンのインストール時に sc_profile.xml ファイルも profiles ディレクトリも提供されない場合、sysconfig 対話型ツールは、コンソールの zlogin コマンドが最初に使用されたときに、このデータについて管理者に確認します。

このリリースは、SMF を使用して構成情報を一元管理します。

Oracle Solaris インスタンスは、インストール中に作成されて構成されます。Oracle Solaris インスタンスは、大域ゾーン内または非大域ゾーン内のいずれかのブート環境として定義されます。sysconfig ユーティリティーを使用して、Oracle Solaris インスタンスに対して構成タスクを実行でき、また Oracle Solaris インスタンスを構成解除して、このインスタンスを再構成することもできます。sysconfig コマンドを使用して、SMF プロファイルを作成できます。

大域ゾーンまたは非大域ゾーン内での Oracle Solaris インスタンスのインストール後または作成後にシステム構成が必要な場合は、システム構成が自動的に実行されます。システム ID を保持するための -p オプションが指定される zoneadm clone 操作の場合、または -c*profile.xml*sysconfig ファイルオプションが指定されない attach 操作の場合では、システム構成は必要ありません。

次のような制御が可能になります。

- sysconfig configure コマンドを使用して、対象の Oracle Solaris インスタンスを再構成 (構成解除してから構成) します。

- `sysconfig configure` コマンドを使用して、対象の Oracle Solaris インスタンスを構成し、SCI ツールをコンソール上で開始します。

  ```
  # sysconfig configure
  ```

- `sysconfig configure` コマンドを使用して、大域ゾーンまたは非大域ゾーン内の構成解除済み Solaris インスタンスを構成します。

  ```
  # sysconfig configure -c sc_profile.xml
  ```

  このコマンドで既存の構成プロファイルを指定した場合、非対話型の構成が実行されます。このコマンドを使用して既存の構成プロファイルを指定しない場合、対話式システム構成 (SCI) ツールが実行されます。SCI ツールを使用すれば、対象の Oracle Solaris インスタンスに固有の構成情報を指定できます。

- 新しいシステム構成プロファイルを作成するには、`sysconfig create-profile` コマンドを使用できます。

`sysconfig` インタフェースについては、『Oracle Solaris 11.1 システムのインストール』の第 6 章「Oracle Solaris インスタンスの構成解除または再構成」および `sysconfig(1M)` のマニュアルページに記載されています。

対話式システム構成ツール

対話式システム構成 (SCI) ツールを使用すると、新規にインストールされた Oracle Solaris 11.1 インスタンスに構成パラメータを指定できます。

`-c` *profile.xml* オプションを指定しない `sysconfig configure` は、システムを構成解除したあと、SCI ツールを表示して管理者に確認してから `/etc/svc/profile/site/scit_profile.xml` にその構成を書き込みます。次に、このツールはこの情報を使用してシステムを構成します。

`sysconfig create-profile` は、管理者に確認してから、`/system/volatile/scit_profile.xml` に SMF プロファイルファイルを作成します。パラメータには、システムのホスト名、タイムゾーン、ユーザーと root のアカウント、ネームサービスが含まれます。

このツール内でナビゲートするには、次のようにします。

- 各画面の下部に一覧表示されたファンクションキーを使用して画面を移動し、ほかの操作を実行します。キーボードにファンクションキーがない場合や、キーが反応しない場合は、Esc キーを押します。画面下部の説明が変わり、ナビゲーションやほかの機能の各 Esc キーが表示されます。
- 選択を変更するか、または入力フィールド間で移動するには、上矢印/下矢印キーを使用します。

詳細は、『Oracle Solaris 11.1 システムのインストール』の第 6 章「Oracle Solaris インスタンスの構成解除または再構成」および `sysconfig(1M)` のマニュアルページを参照してください。

ゾーン構成プロファイルの例

自動構成を持つ排他的 IP ゾーン:

```
<!DOCTYPE service_bundle SYSTEM "/usr/share/lib/xml/dtd/service_bundle.dtd.1">
<service_bundle type="profile" name="sysconfig">
  <service version="1" type="service" name="system/config-user">
    <instance enabled="true" name="default">
      <property_group type="application" name="root_account">
        <propval type="astring" name="login" value="root"/>
        <propval type="astring" name="password"
         value="$5$KeNRy1zU$lqzy9rIsNloUhfvJFIWmVewE75aB5/EBA77kY7EP6F0"/>
        <propval type="astring" name="type" value="role"/>
      </property_group>
      <property_group type="application" name="user_account">
        <propval type="astring" name="login" value="admin1"/>
        <propval type="astring" name="password"
         value="$5$/g353K5q$V8Koe/XuAeR/zpBvpLsgVIqPrvc.9z0hYFYoyoBkE37"/>
        <propval type="astring" name="type" value="normal"/>
        <propval type="astring" name="description" value="admin1"/>
        <propval type="count" name="gid" value="10"/>
        <propval type="astring" name="shell" value="/usr/bin/bash"/>
        <propval type="astring" name="roles" value="root"/>
        <propval type="astring" name="profiles" value="System Administrator"/>
        <propval type="astring" name="sudoers" value="ALL=(ALL) ALL"/>
      </property_group>
    </instance>
  </service>
  <service version="1" type="service" name="system/timezone">
    <instance enabled="true" name="default">
      <property_group type="application" name="timezone">
        <propval type="astring" name="localtime" value="UTC"/>
      </property_group>
    </instance>
  </service>
  <service version="1" type="service" name="system/environment">
    <instance enabled="true" name="init">
      <property_group type="application" name="environment">
        <propval type="astring" name="LC_ALL" value="C"/>
      </property_group>
    </instance>
  </service>
  <service version="1" type="service" name="system/identity">
    <instance enabled="true" name="node">
      <property_group type="application" name="config">
        <propval type="astring" name="nodename" value="my-zone"/>
      </property_group>
    </instance>
  </service>
  <service version="1" type="service" name="system/keymap">
    <instance enabled="true" name="default">
      <property_group type="system" name="keymap">
        <propval type="astring" name="layout" value="US-English"/>
      </property_group>
    </instance>
  </service>
  <service version="1" type="service" name="system/console-login">
    <instance enabled="true" name="default">
```

```
      <property_group type="application" name="ttymon">
        <propval type="astring" name="terminal_type" value="vt100"/>
      </property_group>
    </instance>
  </service>
  <service version="1" type="service" name="network/physical">
    <instance enabled="true" name="default">
      <property_group type="application" name="netcfg">
        <propval type="astring" name="active_ncp" value="Automatic"/>
      </property_group>
    </instance>
  </service>
</service_bundle>
```

DNS と組み合わせない NIS を使用した静的構成を持つ排他的 IP ゾーン:

```
<!DOCTYPE service_bundle SYSTEM "/usr/share/lib/xml/dtd/service_bundle.dtd.1">
<service_bundle type="profile" name="sysconfig">
  <service version="1" type="service" name="system/config-user">
    <instance enabled="true" name="default">
      <property_group type="application" name="root_account">
        <propval type="astring" name="login" value="root"/>
        <propval type="astring" name="password"
         value="$5$m8OR3zqK$Ox5XGubRJdi4zj0JzNSmVJ3Ni4opDOGpxi2nK/GGzmC"/>
        <propval type="astring" name="type" value="normal"/>
      </property_group>
    </instance>
  </service>
  <service version="1" type="service" name="system/timezone">
    <instance enabled="true" name="default">
      <property_group type="application" name="timezone">
        <propval type="astring" name="localtime" value="UTC"/>
      </property_group>
    </instance>
  </service>
  <service version="1" type="service" name="system/environment">
    <instance enabled="true" name="init">
      <property_group type="application" name="environment">
        <propval type="astring" name="LC_ALL" value="C"/>
      </property_group>
    </instance>
  </service>
  <service version="1" type="service" name="system/identity">
    <instance enabled="true" name="node">
      <property_group type="application" name="config">
        <propval type="astring" name="nodename" value="my-zone"/>
      </property_group>
    </instance>
  </service>
  <service version="1" type="service" name="system/keymap">
    <instance enabled="true" name="default">
      <property_group type="system" name="keymap">
        <propval type="astring" name="layout" value="US-English"/>
      </property_group>
    </instance>
  </service>
  <service version="1" type="service" name="system/console-login">
    <instance enabled="true" name="default">
      <property_group type="application" name="ttymon">
```

```
        <propval type="astring" name="terminal_type" value="vt100"/>
      </property_group>
    </instance>
  </service>
  <service version="1" type="service" name="network/physical">
    <instance enabled="true" name="default">
      <property_group type="application" name="netcfg">
        <propval type="astring" name="active_ncp" value="DefaultFixed"/>
      </property_group>
    </instance>
  </service>
  <service version="1" type="service" name="network/install">
    <instance enabled="true" name="default">
      <property_group type="application" name="install_ipv4_interface">
        <propval type="astring" name="address_type" value="static"/>
        <propval type="net_address_v4" name="static_address" value="10.10.10.13/24"/>
        <propval type="astring" name="name" value="net0/v4"/>
        <propval type="net_address_v4" name="default_route" value="10.10.10.1"/>
      </property_group>
      <property_group type="application" name="install_ipv6_interface">
        <propval type="astring" name="stateful" value="yes"/>
        <propval type="astring" name="stateless" value="yes"/>
        <propval type="astring" name="address_type" value="addrconf"/>
        <propval type="astring" name="name" value="net0/v6"/>
      </property_group>
    </instance>
  </service>
  <service version="1" type="service" name="system/name-service/switch">
    <property_group type="application" name="config">
      <propval type="astring" name="default" value="files nis"/>
      <propval type="astring" name="printer" value="user files nis"/>
      <propval type="astring" name="netgroup" value="nis"/>
    </property_group>
    <instance enabled="true" name="default"/>
  </service>
  <service version="1" type="service" name="system/name-service/cache">
    <instance enabled="true" name="default"/>
  </service>
  <service version="1" type="service" name="network/dns/client">
    <instance enabled="false" name="default"/>
  </service>
  <service version="1" type="service" name="network/nis/domain">
    <property_group type="application" name="config">
      <propval type="hostname" name="domainname" value="example.net"/>
      <property type="host" name="ypservers">
        <host_list>
          <value_node value="192.168.224.11"/>
        </host_list>
      </property>
    </property_group>
    <instance enabled="true" name="default"/>
  </service>
  <service version="1" type="service" name="network/nis/client">
    <instance enabled="true" name="default"/>
  </service>
</service_bundle>
```

NIS を使用した動的構成を持つ排他的 IP ゾーン

```
<!DOCTYPE service_bundle SYSTEM "/usr/share/lib/xml/dtd/service_bundle.dtd.1">
<service_bundle type="profile" name="sysconfig">
 <service version="1" type="service" name="system/config-user">
   <instance enabled="true" name="default">
     <property_group type="application" name="root_account">
       <propval type="astring" name="login" value="root"/>
       <propval type="astring" name="password"
        value="$5$Iq/.A.K9$RQyt6RqsAY8TgnuxL9i0/84QwgIQ/nqcK8QsTQdvMy/"/>
       <propval type="astring" name="type" value="normal"/>
     </property_group>
   </instance>
 </service>
 <service version="1" type="service" name="system/timezone">
   <instance enabled="true" name="default">
     <property_group type="application" name="timezone">
       <propval type="astring" name="localtime" value="UTC"/>
     </property_group>
   </instance>
 </service>
 <service version="1" type="service" name="system/environment">
   <instance enabled="true" name="init">
     <property_group type="application" name="environment">
       <propval type="astring" name="LC_ALL" value="C"/>
     </property_group>
   </instance>
 </service>
 <service version="1" type="service" name="system/identity">
   <instance enabled="true" name="node">
     <property_group type="application" name="config">
       <propval type="astring" name="nodename" value="my-zone"/>
     </property_group>
   </instance>
 </service>
 <service version="1" type="service" name="system/keymap">
   <instance enabled="true" name="default">
     <property_group type="system" name="keymap">
       <propval type="astring" name="layout" value="US-English"/>
     </property_group>
   </instance>
 </service>
 <service version="1" type="service" name="system/console-login">
   <instance enabled="true" name="default">
     <property_group type="application" name="ttymon">
       <propval type="astring" name="terminal_type" value="sun-color"/>
     </property_group>
   </instance>
 </service>
 <service version="1" type="service" name="system/name-service/switch">
   <property_group type="application" name="config">
     <propval type="astring" name="default" value="files nis"/>
     <propval type="astring" name="printer" value="user files nis"/>
     <propval type="astring" name="netgroup" value="nis"/>
   </property_group>
   <instance enabled="true" name="default"/>
 </service>
 <service version="1" type="service" name="system/name-service/cache">
   <instance enabled="true" name="default"/>
```

```
  </service>
  <service version="1" type="service" name="network/dns/client">
    <instance enabled="false" name="default"/>
  </service>
  <service version="1" type="service" name="network/nis/domain">
    <property_group type="application" name="config">
      <propval type="hostname" name="domainname" value="special.example.com"/>
      <property type="host" name="ypservers">
        <host_list>
          <value_node value="192.168.112.3"/>
        </host_list>
      </property>
    </property_group>
    <instance enabled="true" name="default"/>
  </service>
  <service version="1" type="service" name="network/nis/client">
    <instance enabled="true" name="default"/>
  </service>
</service_bundle>
```

非大域ゾーンへのログイン方法

このセクションでは、ゾーンへのログインに使用可能な方法について説明します。

ゾーンコンソールログイン

各ゾーンは、仮想コンソール `/dev/console` を保持します。コンソール上で操作を実行することを、コンソールモードと呼びます。ゾーンがインストール済み状態の場合、ゾーンへのコンソールログインを使用できます。ゾーンコンソールは、システム上のシリアルコンソールに非常に似ています。コンソールへの接続は、ゾーンをリブートしても持続します。コンソールモードと `telnet` などのログインセッションとの違いを理解するには、323 ページの「リモートログイン」を参照してください。

ゾーンコンソールへの接続には、`zlogin` コマンドと `-C` オプション、および *zonename* を使用します。ゾーンを稼働状態にする必要はありません。

また、`-d` オプションを使用することもできます。このオプションは、ゾーンが停止した場合に、ゾーンがコンソールから切断されるよう指定します。このオプションは、`-C` オプションとともにのみ指定できます。

ゾーン内部のプロセスが、コンソールを開いてメッセージを書き込むことができます。`zlogin -C` プロセスが終了すると、別のプロセスがコンソールにアクセスできるようになります。

役割によるアクセス制御 (RBAC) を使用中の場合、ゾーンコンソールへのアクセスには `solaris.zone.manage/`*zonename* の承認が必要になります。スラッシュ文字 (`/`) が前に付いた特定の *zonename* の接尾辞はオプションです。省略すると、承認はすべてのゾーンと一致します。

ブート時に対話式システム構成 (SCI) ツールを開始するには、次のように入力します。

```
root@test2:~# sysconfig configure -s
```

ユーザーログインの方法

ユーザー名を使ってゾーンにログインする場合は、`zlogin` コマンドと `-l` オプション、ユーザー名、および *zonename* を使用します。たとえば、大域ゾーンの管理者は、`zlogin` に `-l` オプションを指定することで、通常のユーザーとして非大域ゾーンにログインできます。

```
global# zlogin -l user zonename
```

ユーザー `root` でログインするには、オプションを指定せずに `zlogin` を使用します。

フェイルセーフモード

ログインで問題が発生し、`zlogin` コマンドまたは `zlogin` コマンドと `-C` オプションを使用してゾーンにアクセスできない場合、代替手段が存在します。`zlogin` コマンドと `-S` (safe) オプションを使用することで、ゾーンに入ることができます。このモードは、ほかのログイン方法が成功しなかったときに、損傷を受けたゾーンを復元する場合にのみ使用してください。この最小環境では、ゾーンログインが失敗した理由を診断できる場合があります。

リモートログイン

ゾーンにリモートでログインする機能は、選択したネットワークサービスに依存します。必要な場合は、サービスの `pkg:/service/network/legacy-remote-utilities` を有効にして、`rlogin` および `telnet` によるログインを追加できます。

各ログインコマンドの詳細については、`rlogin(1)`、`ssh(1)`、および `telnet(1)` を参照してください

対話型モードと非対話型モード

zlogin コマンドを使ってゾーンにアクセスし、ゾーン内でコマンドを実行する方法がさらに 2 つ存在します。その方法が、対話型モードおよび非対話型モードです。

対話型モード

対話型モードでは、ゾーン内部で使用する新しい仮想端末が割り当てられます。コンソールデバイスへの排他的なアクセスが許可されるコンソールモードとは異なり、対話型モードでは、いつでも任意の数の zlogin セッションを開くことができます。対話型モードが有効になるのは、発行するコマンドが含まれていない場合です。エディタなどの端末デバイスを必要とするプログラムは、このモードで正常に動作します。

RBAC を使用中の場合、対話型ログインには、ゾーンに対する solaris.zone.login/*zonename* の承認が必要です。ゾーン内では、パスワード認証が実行されます。

非対話型モード

非対話型モードは、ゾーンを管理するシェルスクリプトを実行する場合に使用します。非対話型モードでは、新しい仮想端末は割り当てられません。ゾーン内部で実行されるコマンドを指定すると、非対話型モードが有効になります。

非対話型ログインの場合、またはパスワード認証を省略する場合は、solaris.zone.manage/ *zonename* の承認が必要です。

◆ ◆ ◆ 第 21 章

非大域ゾーンへのログイン (タスク)

この章では、インストール済みのゾーンの構成を完了して大域ゾーンからゾーンにログインし、ゾーンを停止処理する手順について説明します。また、この章では、zonename コマンドを使用して現在のゾーンの名前を出力する方法についても説明します。

ゾーンへのログイン処理の概要については、第 20 章「非大域ゾーンへのログイン (概要)」を参照してください。

初期ゾーンブートおよびゾーンログインの手順 (タスクマップ)

タスク	説明	参照先
内部構成を実行するか、またはゾーンを構成解除します。	システム構成は、テキストのユーザーインタフェースを使用して対話的に行うことができ、またプロファイルを使用して非対話的に行うこともできます。また、Solaris インスタンスを構成解除するために、sysconfig ユーティリティーも使用します。	『Oracle Solaris 11.1 システムのインストール』の第 6 章「Oracle Solaris インスタンスの構成解除または再構成」および sysconfig(1M) のマニュアルページを参照してください。

タスク	説明	参照先
ゾーンにログインします。	ゾーンへのログインには、コンソールを使用する、対話型モードを使って仮想端末を割り当てる、またはゾーン内で実行するコマンドを指定する方法があります。実行するコマンドを指定する場合、仮想端末は割り当てられません。ゾーンへの接続が拒否された場合は、フェイルセーフモードを使用してログインすることもできます。	326 ページの「ゾーンへのログイン」
非大域ゾーンから抜けます。	非大域ゾーンへの接続を切り離します。	329 ページの「非大域ゾーンから抜ける方法」
ゾーンを停止処理します。	`shutdown` ユーティリティーまたはスクリプトを使用して、ゾーンを停止処理します。	330 ページの「`zlogin` を使用してゾーンを停止処理する方法」
ゾーン名を出力します。	現在のゾーンの名前を出力します。	331 ページの「現在のゾーンの名前を出力する」

ゾーンへのログイン

`zlogin` コマンドを使用して、大域ゾーンから稼働状態または準備完了状態にある任意のゾーンへログインします。詳細は、`zlogin`(1) のマニュアルページを参照してください。

次の手順で説明されているように、ゾーンへのログインはさまざまな方法で実行できます。323 ページの「リモートログイン」で説明されているように、リモートでログインすることも可能です。

▼ 構成プロファイルを作成する方法

注意 – 必要なすべてのデータを提供する必要があることに注意してください。データが欠落したプロファイルを提供すると、ゾーンはデータが欠落した状態で構成されます。この構成によって、ユーザーがログインできなくなったり、ネットワークを実行できなくなったりする可能性があります。

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を持つユーザーである必要があります。

1 **root** になるか、同等の役割になります。

2 **sysconfig** ツールを使用してプロファイルを作成します。

- 排他的 **IP** ゾーンの場合:

```
# sysconfig create-profile -o /path/sysconf.xml
```

- 共有 **IP** ゾーンの場合:

```
# sysconfig create-profile -o /path/sysconf.xml -g location,identity,naming_services,users
```

3 ゾーンのインストール、クローン、または接続の操作中に、この作成したプロファイルを使用します。

```
# zoneadm -z my-zone install -c /path/sysconf.xml
```

構成ファイルが使用されている場合、システムは最初の `zlogin` 時にコンソール上で対話式システム構成 (SCI) ツールを開始しません。ファイルの引数は絶対パスで指定する必要があります。

▼ ゾーンコンソールにログインして初期ゾーン構成を行う方法

`config.xml` ファイルが `zoneadm clone`、`attach`、または `install` コマンドに渡されると、この構成ファイルがシステムを構成するために使用されます。`config.xml` ファイルが `clone`、`attach`、または `install` の操作中に提供されなかった場合、ゾーンの最初のブートではコンソール上で SCI ツールを開始します。

構成情報用の最初のプロンプトが表示されないことを避けるために、ゾーンが 2 回目のセッションでブートされる前に、`zlogin` が実行されるように 2 つの端末ウィンドウを使用することをお勧めします。

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を持つユーザーである必要があります。

1 **root** になるか、同等の役割になります。

2 **zlogin** コマンドを **-C** オプションと **my-zone** などのゾーン名とともに使用します。

```
global# zlogin -C my-zone
```

3 別の端末ウィンドウからゾーンをブートします。

```
global# zoneadm -z my-zone boot
```

次のような内容が、`zlogin` 端末ウィンドウに表示されます。

```
[NOTICE: Zone booting up]
```

4 新たにインストールしたゾーン用の構成パラメータに関する一連の質問に応答します。パラメータには、システムのホスト名、タイムゾーン、ユーザーと **root** アカウント、およびネームサービスが含まれます。**SCI** ツールは、デフォルトでは **/system/volatile/scit_profile.xml** に **SMF** プロファイルファイルを生成します。

注意事項 最初の SCI 画面が表示されない場合は、`Ctrl + L` を押してこの SCI 画面をリフレッシュできます。

▼ ゾーンコンソールへのログイン方法

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を持つユーザーである必要があります。

1 **root** になるか、同等の役割になります。

2 **zlogin** コマンドを **-C** オプション、**-d** オプション、および **my-zone** などのゾーンの名前とともに使用します。

```
global# zlogin -C -d my-zone
```

構成が実行されていない場合、`zlogin` コマンドを `-C` オプションとともに使用すると SCI ツールが開始されます。

3 ゾーンコンソールが表示されたら、**root** でログインし、**Return** キーを押します。プロンプトが表示されたら **root** のパスワードを入力します。

```
my-zone console login: root
Password:
```

▼ 対話型モードを使用してゾーンにアクセスする方法

対話型モードでは、ゾーン内部で使用する新しい仮想端末が割り当てられます。

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を持つユーザーである必要があります。

1 **root** になるか、同等の役割になります。

2 大域ゾーンからゾーン **(例: my-zone)** にログインします。

```
global# zlogin my-zone
```

次のような情報が表示されます。

```
[Connected to zone 'my-zone' pts/2]
Last login: Wed Jul  3 16:25:00 on console
```

3 **exit** と入力して、接続を閉じます。

次のようなメッセージが表示されます。

```
[Connection to zone 'my-zone' pts/2 closed]
```

▼ 非対話型モードを使用してゾーンにアクセスする方法

ゾーン内部で実行されるコマンドを指定すると、非対話型モードが有効になります。非対話型モードでは、新しい仮想端末は割り当てられません。

コマンドおよびコマンドの処理対象のファイルは、いずれも NFS 上に存在してはならないことに注意してください。

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を持つユーザーである必要があります。

1 **root** になるか、同等の役割になります。

2 大域ゾーンから **my-zone** ゾーンにログインして、コマンド名を入力します。

ここではコマンド zonename を使用します。

```
global# zlogin my-zone zonename
```

次の出力が表示されます。

```
my-zone
```

▼ 非大域ゾーンから抜ける方法

- 非大域ゾーンへの接続を切り離すには、次のいずれかの方法を使用します。
  - ゾーンの非仮想コンソールを終了するには、次の操作を行います。

    ```
    zonename# exit
    ```

  - ゾーンの仮想コンソールへの接続を切り離すには、次のようにチルダ (~) 文字とピリオドを使用します。

    ```
    zonename# ~.
    ```

    画面には、次のようなメッセージが表示されます。

    ```
    [Connection to zone 'my-zone' pts/6 closed]
    ```

注 – また、ssh のデフォルトのエスケープシーケンスも ~ で、これは ssh セッションを終了させます。ssh を使用してリモートからサーバーにログインする場合は、~~. を使用してゾーンを抜けます。

参照 zlogin コマンドのオプションの詳細については、zlogin(1) のマニュアルページを参照してください。

▼ フェイルセーフモードを使用してゾーンに入る方法

ゾーンへの接続が拒否された場合、zlogin コマンドと -S オプションを使用して、ゾーン内の最小環境に入ることができます。

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を持つユーザーである必要があります。

1 **root** になるか、同等の役割になります。

2 大域ゾーンから、**zlogin** コマンドと **-S** オプションを使用してゾーン (例: **my-zone**) にアクセスします。

```
global# zlogin -S my-zone
```

▼ zlogin を使用してゾーンを停止処理する方法

注 – 大域ゾーンで init 0 を実行して Oracle Solaris システムの完全な停止処理を実行すると、システム上のそれぞれの非大域ゾーンでも init 0 が実行されます。init 0 は、ローカルユーザーとリモートユーザーに対してシステムが停止する前にログオフするよう警告しないので、注意してください。

ゾーンを正しく停止処理するには、次の手順を実行します。停止処理スクリプトを実行せずにゾーンを停止する方法については、307 ページの「ゾーンの停止方法」を参照してください。

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を持つユーザーである必要があります。

1 **root** になるか、同等の役割になります。

2 停止処理を行うゾーン (例: **my-zone**) にログインし、ユーティリティーの名前として **shutdown** を、状態として **init 0** を指定します。

```
global# zlogin my-zone shutdown -i 0
```

サイトによっては、特定の環境に合わせた独自の停止処理スクリプトが存在する場合もあります。

サービスの有効化

ゾーン内にある個々のサービスを有効または無効にできます。

現在のゾーンの名前を出力する

zonename(1) のマニュアルページに説明されているように、zonename コマンドにより現在のゾーンの名前が出力されます。次に、大域ゾーン内で zonename を使用した場合の出力例を示します。

```
# zonename
global
```

◆ ◆ ◆ 第 22 章

22 ゾーンの移行と zonep2vchk ツールについて

この章では、次の概要について説明します。

- システムを非大域ゾーンに移行する Physical-To-Virtual 移行
- 既存のゾーンを新しいシステムに移行する Virtual-to-Virtual 移行

また、この章では、システムからゾーンへの移行で使用される `zonep2vchk` ツールについても説明します。

Physical-To-Virtual の概念と Virtual-to-Virtual の概念

P2V と V2V は、次の操作で使用できます。

- 多数のアプリケーションの単一サーバー上への統合
- 作業負荷のリバランス
- サーバーの交換
- 障害回復

移行方針の選択

新しいホスト上で `zonepath` が表示されるように SAN ベースのストレージを再構成できます。

1 つのシステム上のすべてのゾーンを別のシステムに移動しなければならない場合、レプリケーションストリームにはスナップショットとクローンを保持するという利点があります。スナップショットとクローンは、`pkg`、 `beadm create`、および `zoneadm clone` コマンドによって幅広く使用されます。

P2V または V2V の移行を行うには、5 つの手順があります。

1. P2V の場合、Oracle Solaris 構成のソースホストを次のように解析します。

- ネットワーク接続要件に基づき、非大域ゾーンの IP タイプ (排他的 IP または共有 IP) を判定します。
- ターゲットホストの大域ゾーンに追加の構成が必要かどうかを判定します。
- アプリケーションデータとファイルシステムの移行方法を決定します。

`b` オプションによって実行した `-zonep2vchk` の基本解析は、Oracle Solaris 構成に関連する問題またはソース大域ゾーンによって使用される機能を特定します。`s` オプションを使用した `-zonep2vchk` の静的解析は、ソース大域ゾーン上の特定のアプリケーションに関連する問題を特定するために役立ちます。`-r` によって実行した `zonep2vchk` の実行時解析は、現在実行中のアプリケーションに、ゾーン内で機能しない可能性がある操作がないかどうかを検査します。

2. ソースシステムまたはゾーンをアーカイブします。Oracle Solaris インスタンスのアーカイブは、場合によっては個別に移行予定のデータを除外することがあります。
   - Oracle Solaris 10 の大域ゾーンをアーカイブするには、`flarcreate` を使用できます。

     437 ページの「`flarcreate` を使用してイメージを作成する方法」を参照してください。
   - Oracle Solaris 10 システムと非大域ゾーンをアーカイブするには、`-R` または `-L` *archiver* を指定した `flarcreate` を使用して特定のファイルをアーカイブから除外できます。最初にゾーンを必ず停止してください。

     438 ページの「`flarcreate` を使用して特定のデータを除外する方法」を参照してください。
   - Oracle Solaris 11 の大域ゾーンの場合、`zfs send` を使用してルートプールをアーカイブできます。
   - Oracle Solaris 11 の非大域ゾーンの場合、`zfs send` を使用するとゾーンの `zonepath` データセットをアーカイブできます。
   - SAN などの共有ストレージ上の `zpool` 内に存在する `solaris10` または `solaris` ゾーンの場合、V2V 移行方針ではアーカイブの作成は必要ありません。新しいホスト上で `zonepath` が表示されるように SAN ベースのストレージを再構成できます。再構成するには:
     - `zpool` をエクスポートしてからターゲット大域ゾーン上でインポートします。
     - ターゲットシステム上で `zoneadm install` (推奨) または `attach` を使用します。(このセクションの手順 5 を参照。)

   共有ストレージ上のゾーンに関する内容も参照してください。
3. 次のような追加のデータとファイルシステム用の移行方針を選択します。
   - アーカイブにデータを含めます (このセクションの手順 2 を参照)。

- zfs send などの推奨のアーカイブフォーマットを使用してデータを個別にアーカイブし、移行後にゾーン内でデータを復元します。
- ターゲット大域ゾーンから SAN ストレージにアクセスして SAN のデータを移行し、zonecfg add fs を使用してそのデータをゾーンで使用できるようにします。
- ソースホスト上の zpool をエクスポートし、ストレージを移動し、ターゲット大域ゾーン上に zpool をインポートすることによって、ZFS zpools 内のストレージは移行できます。これらの ZFS ファイルシステムは、次に zonecfg add dataset または zonecfg add fs を使用してターゲットゾーンに追加できます。また、SAN ストレージデバイス上の zpools もこの方法で移行できます。

4. ターゲットホスト上のターゲットゾーンのゾーン構成 (zonecfg) を作成します。
   - P2V の場合は、zonep2vchk コマンドを -c オプションとともに使用して、構成の作成を支援します。
   - V2V の場合は、ソースホスト上で zonecfg -z source_zone export コマンドを使用します。Oracle Solaris 10 ゾーンに Oracle Solaris 10 コンテナを移行する場合は、必ずブランドを solaris10 に設定してください。

   必要に応じて (たとえば、ネットワーク接続リソースを更新するために)、エクスポートされた zonecfg を確認して変更します。
5. アーカイブを使用して、ターゲットホスト上でゾーンをインストールまたは接続します。新しい sysconfig プロファイルが指定できるようになっているか、または最初のブート時に sysconfig ユーティリティーを実行できます。

zonep2vchk ツールを使用したシステムの移行の準備

このセクションでは、zonep2vchk ツールについて説明します。このツールの主なドキュメントは、zonep2vchk(1M) のマニュアルページです。

zonep2vchk ツールについて

P2V の処理は、大域ゾーン (ソース) をアーカイブすることと、次にそのアーカイブを使用して非大域ゾーン (ターゲット) をインストールすることから構成されます。zonep2vchk ユーティリティーは、0 の実効ユーザー ID で実行する必要があります。

このユーティリティーは次のことを行います。

- ソースシステムの構成内の問題領域を特定します
- 必要になる手動の再構成作業を最小限に抑えます
- Oracle Solaris 10 システムイメージと Oracle Solaris 11 システムイメージの両方の Oracle Solaris 11 リリース上のゾーンへの移行をサポートします

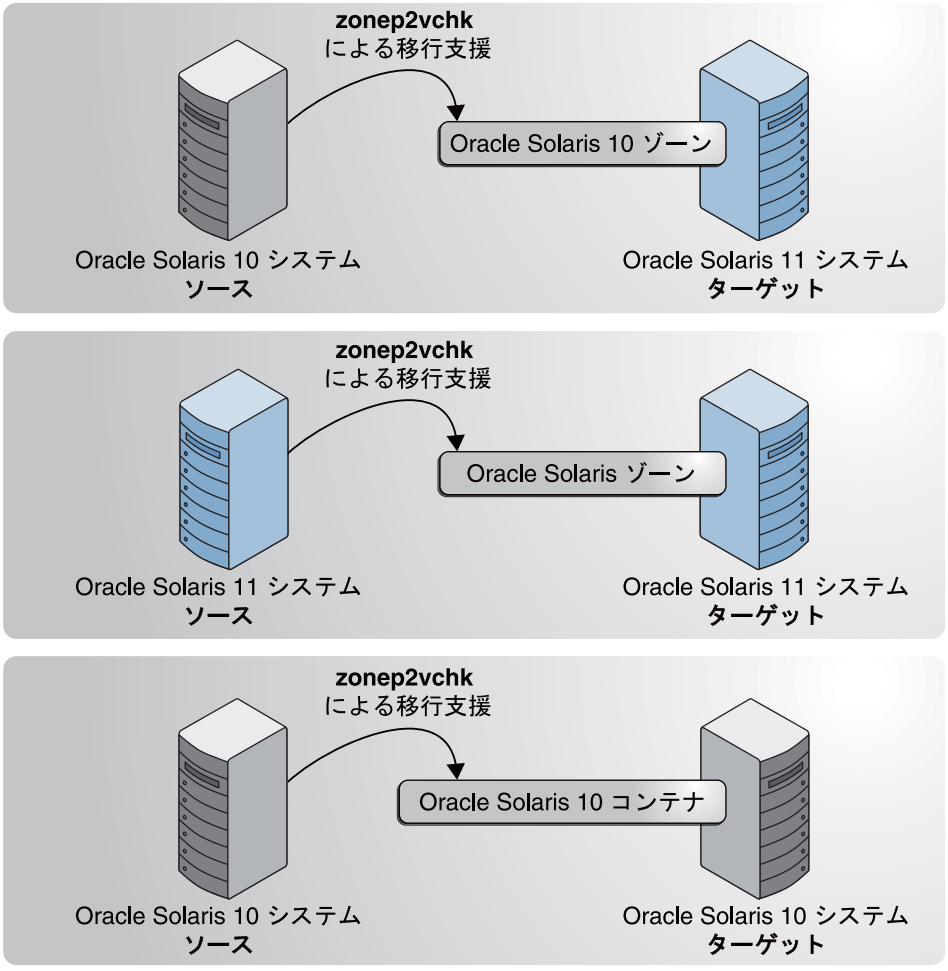
- 元のシステムイメージ内の複雑なネットワーク構成 (複数の IP インタフェース、IP マルチパス、VLAN など) をサポートします

このツールを使用すると、Oracle Solaris 11 の物理システムまたは Oracle Solaris 10 の物理システムを、このリリース上の非大域ゾーンに移行できます。

- Oracle Solaris 11 システムを `solaris` ブランドゾーンに移行します
- Oracle Solaris 10 システムを `solaris10` ブランドゾーンに移行します

Oracle Solaris 11 のターゲットシステムの場合、 `add anet` リソース (VNIC) は、ソースシステム上のネットワークリソースごとに `zonecfg` の出力に含まれます。デフォルトでは、Oracle Solaris 11 システム上の非大域ゾーンに Oracle Solaris 11 システムまたは Oracle Solaris 10 システムのいずれかを移行する場合のネットワークタイプは排他的 IP です。

図 22–1　zonep2vchk ユーティリティー



解析のタイプ

基本解析 (-b オプション) は、P2V 移行によって影響を受ける可能性がある使用中の Oracle Solaris の機能がないかどうかを検査します。

静的解析 (-s オプション) は、バイナリに、ゾーン内で機能しない可能性があるシステムコールとライブラリコールがないかどうかを検査します。

実行時解析 (-r オプション) は、現在実行中のアプリケーションに、ゾーン内で機能しない可能性がある操作がないかどうかを検査します。

生成される情報

次の 2 つの主なカテゴリの情報が解析によって提示されます。

- 特定のゾーンの構成または大域ゾーン内の構成変更によって対処できる問題
- ゾーン内で実行できない機能の特定

たとえば、アプリケーションがシステムクロックを設定する場合、適切な特権をゾーンに追加することによって有効にできますが、アプリケーションがカーネルメモリーにアクセスする場合、それはゾーン内部では許可されることはありません。出力では、これらの 2 つのクラスの問題を区別します。

このユーティリティーは、デフォルトでは人間が読める形式のメッセージを出力します。メッセージを機械解析可能な形式で出力するには、-P オプションが使用されます。使用できるオプションおよびコマンド起動と出力の詳細は、zonep2vchk(1M) のマニュアルページを参照してください。

◆ ◆ ◆ 第 23 章

Oracle Solaris システムの移行と非大域ゾーンの移行 (タスク)

この章では、Oracle Solaris 11 システムをターゲット Oracle Solaris 11 マシン上の非大域ゾーンに移行する方法について説明します。この章では、ソースシステムを移行する前にソースシステム上の既存の `solaris` ゾーンを新しいターゲットシステムに移行する方法についても説明します。

この情報は、`solaris10` ブランドゾーンの移行にも適用されます。`solaris10` ブランドゾーンについては、パート III「Oracle Solaris 10 ゾーン」を参照してください。

別のマシンへの非大域ゾーンの移行

ゾーンの移行について

`zonecfg` および `zoneadm` コマンドを使用して、既存の非大域ゾーンをあるシステムから別のシステムに移行できます。ゾーンは停止され、現在のホストから切り離されます。`zonepath` はターゲットホストに移動され、そこで接続されます。

ゾーンの移行には、次の要件が適用されます。

- 移行前に、ソースシステム上のアクティブでない BE をすべて削除する必要があります。
- ターゲットシステム上の大域ゾーンで、元のソースホストと同じかそれ以降の Oracle Solaris 11 リリースが稼働している必要があります。
- ゾーンが正常に動作することを保証するには、元のソースホストにインストールされている必須オペレーティングシステムパッケージと同じかまたはそれ以降のバージョンのものが、ターゲットシステムにインストールされている必要があります。

  サードパーティー製品のパッケージなど、ほかのパッケージは異なっていてもかまいません。

- ゾーンに依存するパッケージよりも後のバージョンが新しいホストにインストールされている場合、-u または -U のオプションを指定した zoneadm attach を使用すると、新しいホストに一致させるためにゾーン内のそれらのパッケージを更新します。接続時更新ソフトウェアは、移行対象のゾーンを調べ、新しいホストに一致するように更新する必要のあるパッケージを決定します。このようなパッケージだけが更新されます。残りのパッケージは、ゾーンによって異なる可能性があります。ゾーン内にインストールされ、大域ゾーンにはインストールされていないパッケージがある場合、それらのパッケージは無視され、そのままの状態になります。
- ストレージオブジェクトに既存のパーティション、zpool、または UFS ファイルシステムが含まれる場合は、attach が失敗し、エラーメッセージが表示されます。attach 操作を続行し、既存のデータを上書きするには、zoneadm attach に対して -x オプションを使用します。

  ```
  -x force-zpool-import
  -x force-zpool-create=zpoolname
  -x force-zpool-create=zpoolname1,zpoolname2,zpoolname3
  -x force-zpool-create-all
  ```

  このオプションは、zpool create -f コマンドに似ています。

  -x force-zpool-create=*zpoolname* オプションは、1 回または複数回使用できます。

zoneadm detach プロセスにより、別のシステムでゾーンを接続するのに必要な情報が作成されます。 zoneadm attach プロセスは、ターゲットマシンがゾーンのホストとして機能するための適正な構成を保持していることを確認します。

新規ホストで zonepath を使用可能にする方法は複数存在するため、あるシステムから別のシステムへの zonepath の実際の移動は、大域管理者が手動で行います。

新規システムへの接続時に、ゾーンはインストール済みの状態になります。

▼ ZFS アーカイブを使用して非大域ゾーンを移行する方法

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を持つユーザーである必要があります。

この例では、ゾーンのアーカイブを作成したあと、別のシステムにそのアーカイブを接続する方法について説明しています。ソースホストとターゲットホスト上の管理者が一時ファイルストレージ用の共有の NFS サーバーにアクセスできることを想定しています。共有の一時領域が利用できない場合、scp セキュアコピー (リモートファイルのコピープログラム) などのほかの手段を使用すると、ソースマシンとターゲットマシン間でファイルをコピーできます。パスワードまたはパスフレーズが認証に必要な場合、scp プログラムはそれらの入力を求めます。

1. **root** になるか、同等の役割になります。

2. 移行するゾーン (この手順では **my-zone**) を停止処理します。

   ```
   host1# zoneadm -z my-zone shutdown
   ```

3. (オプション) ゾーンを切り離します。

   ```
   host1# zoneadm -z my-zone detach
   ```

   切り離されたゾーンは、現在、構成済みの状態にあります。次回の大域ゾーンのブート時に、ゾーンは自動的にブートされません。

4. ゾーンの構成をエクスポートします。

   ```
   host1# mkdir /net/server/zonearchives/my-zone
   host1# zonecfg -z my-zone export > /net/server/zonearchives/my-zone/my-zone.zonecfg
   ```

5. **gzip** ZFS アーカイブを作成します。

```
host1# zfs list -H -o name /zones/my-zone
rpool/zones/my-zone
host1# zfs snapshot -r rpool/zones/my-zone@v2v
host1# zfs send -rc rpool/zones/my-zone@v2v | gzip > /net/server/zonearchives/my-zone/my-zone.zfs.gz
```

   圧縮の使用はオプションですが、アーカイブの書き込み後の読み取り中に入出力の実行がより少ないため、通常圧縮した方がより高速です。詳細は、『Oracle Solaris 11.1 の管理: ZFS ファイルシステム』を参照してください。

6. 新規ホスト上でゾーンを構成します。

   ```
   host2# zonecfg -z my-zone -f /net/server/zonearchives/my-zone/my-zone.zonecfg
   ```

   次のシステムメッセージが表示されます。

   ```
   my-zone: No such zone configured
   Use 'create' to begin configuring a new zone.
   ```

7. (オプション) 構成を表示します。

   ```
   host2# zonecfg:my-zone> info
   zonename: my-zone
   zonepath: /zones/my-zone
   autoboot: false
   pool:
   net:
           address: 192.168.0.90
           physical: bge0
   ```

8. 構成に必要な調整を加えます。

   たとえば、新規ホストではネットワーク物理デバイスが異なる場合があります。また、構成に含まれるデバイスの名前が新規ホストでは異なることもあります。

   ```
   host2# zonecfg -z my-zone
   zonecfg:my-zone> select net physical=bge0
   zonecfg:my-zone:net> set physical=e1000g0
   zonecfg:my-zone:net> end
   ```

9 構成を確定して終了します。

```
zonecfg:my-zone> commit
zonecfg:my-zone> exit
```

10 次のいずれかの方法を使用して、ゾーンを新しいホストにインストールします。**install** サブコマンドを使用することをお勧めします。

- **install** が正常に完了できるように、必要な最小の更新を実行して、ゾーンをインストールします。

```
host2# zoneadm -z my-zone install -p -a /net/server/zonearchives/my-zone/my-zone.zfs.gz
```

このリリースでは、接続が正常に完了できるように、必要な最小の更新を実行して、ゾーンの attach を行うこともできます。更新が許可されている場合は、zoneadm attach の実行中に発行元のカタログがリフレッシュされます。

```
host2# zoneadm -z my-zone attach -u -a /net/server/zonearchives/my-zone/my-zone.zfs.gz
```

- ゾーン内のすべてのパッケージを大域ゾーンと互換性のある最新バージョンに更新して、ゾーンをインストールします。

```
host2# zoneadm -z my-zone install -U -p -a /net/server/zonearchives/my-zone/my-zone.zfs.gz
```

このリリースでは、ゾーン内のすべてのパッケージを大域ゾーンと互換性のある最新バージョンに更新して、ゾーンの attach を行うこともできます。

```
host2# zoneadm -z my-zone install -U -a /net/server/zonearchives/my-zone/my-zone.zfs.gz
```

- ソフトウェアを更新せずに、ゾーンを新規ホストに接続します。

```
host2# zoneadm -z my-zone attach -a /net/server/zonearchives/my-zone/my-zone.zfs.gz
```

注 – attach サブコマンドの -a オプションと -d オプションは、Oracle Solaris の将来のリリースで削除される可能性があります。install サブコマンドを使用することをお勧めします。

注意事項 ストレージオブジェクトに既存のパーティション、zpool、または UFS ファイルシステムが含まれる場合は、install が失敗し、エラーメッセージが表示されます。インストールを続行し、既存のデータを上書きするには、zoneadm install に対して -x オプションを使用します。

使用できないマシンからのゾーンの移行

非大域ゾーンをホストするマシンが使用できなくなる場合があります。しかし、ゾーンが格納されている SAN などのストレージがまだ使用できる場合は、ゾーンを新しいホストに正常に移行できる可能性があります。ゾーンの `zonepath` を新しいホストに移動できます。SAN などのいくつかの場合には、`zonepath` データを実際には移動できないこともあります。SAN の場合は、`zonepath` が新規ホストに表示されるように、再構成が実行されるだけです。ゾーンが適切に切り離されなかったため、初めに `zonecfg` コマンドを使って新しいホストにゾーンを作成する必要があります。一度これを行なったあと、新しいホストでゾーンを接続してください。

このタスクの手順は、340 ページの「ZFS アーカイブを使用して非大域ゾーンを移行する方法」で説明されています。

非大域ゾーンへの Oracle Solaris システムの移行

ゾーンは入れ子にならないため、P2V プロセスは移行するシステムイメージの内部にある既存のゾーンを移行先ゾーンで使用できないようにします。大域ゾーンのシステムイメージを移行する前に、ソースシステム上の既存の非大域ゾーンを移行する必要があります。

solaris 非大域ゾーンへの Oracle Solaris システムの移行について

既存の Oracle Solaris 11 システムを Oracle Solaris 11 システム上の `solaris` ブランドゾーンに直接移行できます。ソースシステム上で `zonep2vchk` および `zfs` コマンドを使用して、移行の準備を行い、システムイメージをアーカイブします。`zonecfg` および `zoneadm` コマンドを使用して、アーカイブを構成し、それをターゲットシステム上の移行先ゾーンにインストールします。

非大域ゾーンへの大域ゾーンの移行には、次の制約が適用されます。

- ターゲットシステム上の大域ゾーンで、元のソースホストと同じかそれ以降の Oracle Solaris 11 リリースが稼働している必要があります。
- ゾーンが適切に実行されるようにするには、必要なオペレーティングシステムパッケージの同じまたは以降のバージョンがターゲットシステムに存在する必要があります。サードパーティー製品のパッケージなど、ほかのパッケージは異なっていてもかまいません。

詳細は、`zonep2vchk(1M)`、`zfs(1M)`、`zonecfg(1M)`、`zoneadm(1M)`、および `solaris(5)` のマニュアルページを参照してください。

▼ zonep2vchk を使用したソースシステムのスキャン

1 管理者になります。

2 **zonep2vchk** ツールを **-b** オプションとともに使用して基本解析を実行し、**P2V** 移行によって影響を受ける可能性がある使用中の **Oracle Solaris** の機能がないかどうかを検査します。

```
source# zonep2vchk -b 11
```

3 **-s** オプションを指定して **zonep2vchk** ツールを実行し、アプリケーションファイルの静的解析を実行します。これは、**ELF** バイナリに、ゾーン内の操作に影響を与える可能性があるシステムコールとライブラリコールがないかどうかを検査します。

```
source# zonep2vchk -s /opt/myapp/bin,/opt/myapp/lib
```

4 **-r** オプションを指定して **zonep2vchk** ツールを実行し、ゾーン内で正常に実行できない可能性があるプロセスを探す実行時チェックを実行します。

```
source# zonep2vchk -r 2h
```

5 ソースシステム上で **-c** オプションを指定して **zonep2vchk** ツールを実行し、テンプレート **zonecfg** スクリプトを生成します **(**この手順では **s11-zone.config** という名前を付けます**)**。

```
source# zonep2vchk -c > /net/somehost/p2v/s11-zone.config
```

この構成には、ソースホストの物理リソースとネットワーク構成に基づくリソース制限とネットワーク構成が含まれます。

▼ ネットワークデバイス上にシステムイメージのアーカイブを作成する方法

大域ゾーン内のファイルシステムをアーカイブします。ソースシステムに非大域ゾーンがインストールされていないことを確認します。`cpio`、`-x xustar` (XUSTAR) 形式で作成された `pax` アーカイブ、および `zfs` を含む複数のアーカイブ形式がサポートされています。このセクションの例では、アーカイブの作成に `zfs send` コマンドを使用します。これらの例では、ルートプールの名前を `rpool` としています。

1 管理者になります。

2 ルートプール全体のスナップショットを作成します **(**この手順では *rpool@p2v* という名前を付けます**)**。

```
source# zfs snapshot -r rpool@p2v
```

3 スワップおよびダンプデバイスに関連付けられたスナップショットは、ターゲットシステム上では不要なため、破棄します。

```
source# zfs destroy rpool/swap@p2v

source# zfs destroy rpool/dump@p2v
```

4 システムをアーカイブします。

- **gzip** で圧縮され、リモート **NFS** サーバーに保存された **ZFS** レプリケーションストリームのアーカイブを生成します。

  ```
  source# zfs send -R rpool@p2v | gzip > /net/somehost/p2v/s11-zfs.gz
  ```

- 次の代替コマンドを使用すると、中間のスナップショットが保存されなくなり、アーカイブのサイズを減らすことができます。

  ```
  source# zfs send -rc rpool@p2v
  ```

参照 詳細は、cpio(1)、pax(1)、および zfs(1M) のマニュアルページを参照してください。

▼ ターゲットシステム上のゾーンを構成する方法

zonep2vchk ツールによって生成されたテンプレート zonecfg スクリプトは、ソースシステムの構成のうち、移行先のゾーン構成でサポートされる必要がある部分を定義します。ゾーンを完全に定義するには、ターゲットシステムに依存するその他の情報を手動で指定する必要があります。

この手順では、構成ファイルの名前を s11-zone.config としています。

1 管理者になります。

2 **zonecfg** スクリプトの内容を確認して、ソースシステムの構成パラメータを理解します。

```
target# less /net/somehost/p2v/s11-zone.config
```

このスクリプトに含まれる zonepath の初期値は、ソースシステムのホスト名に基づいています。移行先ゾーンの名前がソースシステムのホスト名と異なる場合は、zonepath ディレクトリを変更できます。

コメントアウトされたコマンドには、メモリー容量、CPU の数、ネットワークカードの MAC アドレスなど、元の物理システム環境のパラメータが反映されます。これらの行をコメント解除して、ターゲットゾーンのリソースをきめ細かく制御することもできます。

3 ターゲットシステム上の大域ゾーンで次のコマンドを使用して、現在のリンク構成を表示します。

```
target# dladm show-link
target# dladm show-phys
target# ipadm show-addr
```

デフォルトでは、zonecfg スクリプトはソースシステム上で構成されたすべての物理ネットワークインタフェースに対して anet リソースを使用して排他的 IP ネットワーク構成を定義します。ターゲットシステムは、ゾーンのブート時に各 anet リソースに対して自動的に VNIC を作成します。VNIC を使用することで、複数のゾーンが同じ物理ネットワークインタフェースを共有できるようになります。anet リソースの下位リンク名は、zonecfg コマンドによって *change-me* に初期設定されます。このフィールドは手動でターゲットシステム上のいずれかのデータリンクの名前に設定する必要があります。VNIC の下位リンクとして有効な任意のリンクを指定できます。

4 **zonecfg** スクリプトをターゲットシステムにコピーします。

```
target# cp /net/somehost/p2v/s11-zone.config .
```

5 **vi** などのテキストエディタを使用して、構成ファイルに変更を加えます。

```
target# vi s11-zone.config
```

6 **zonecfg** コマンドを使用して *s11-zone* ゾーンを構成します。

```
target# zonecfg -z s11-zone -f s11-zone.config
```

▼ ターゲットシステムへのゾーンのインストール

この例では、インストール中に元のシステム構成は変更されません。

1 管理者になります。

2 ソースシステム上で作成されたアーカイブを使用してゾーンをインストールします。

```
target# zoneadm -z s11-zone install -a /net/somehost/p2v/s11-zfs.gz -p
```

◆ ◆ ◆ 第 24 章

ゾーンがインストールされている Oracle Solaris 11.1 システムでの自動インストールおよびパッケージ

AI クライアントインストールの一環として、非大域ゾーンのインストールおよび構成を指定できます。このリリースでは、Image Packaging System (IPS) がサポートされています。この章では、ゾーンをインストールするときに、IPS パッケージを使用してオペレーティングシステムをインストールおよび保守する方法について説明します。

`solaris10` ゾーンおよび `native` ゾーンで使用される SVR4 パッケージおよびパッチについては、『System Administration Guide: Oracle Solaris Containers-Resource Management and Oracle Solaris Zones』の第 25 章「ゾーンがインストールされている Oracle Solaris システムでのパッケージについて (概要)」および第 26 章「ゾーンがインストールされている Oracle Solaris システムでのパッケージとパッチの追加および削除 (タスク)」を参照してください。これは、このガイドの Oracle Solaris 10 バージョンです。

Oracle Solaris 11.1 リリースを実行しているシステムのイメージパッケージングシステムソフトウェア

グラフィカルツールおよびコマンド行ツールを使用すると、リポジトリからパッケージをダウンロードしてインストールできます。この章では、インストールされている非大域ゾーンにパッケージを追加するための情報を提供します。パッケージの削除に関する情報も含まれています。この章の内容は、既存の Oracle Solaris インストールおよびパッチに関するドキュメントの内容を補うものです。詳細は、『Oracle Solaris の管理: 一般的なタスク』および『Oracle Solaris 11.1 ソフトウェアパッケージの追加および更新』の第 4 章「ソフトウェアパッケージのインストールおよび更新」を参照してください。

ゾーンのパッケージの概要

solaris パッケージリポジトリは、ゾーン環境の管理に使用されます。

pkg コマンドを使用してシステムを新しいバージョンの Oracle Solaris にアップグレードすると、ゾーンは自動的に更新されます。

イメージパッケージングシステム (IPS) (pkg(5) に記載) は、パッケージのインストール、アップグレード、削除などのソフトウェアのライフサイクル管理のために提供されるフレームワークです。IPS を使用すると、ソフトウェアパッケージの作成、パッケージリポジトリの作成と管理、および既存のパッケージリポジトリのミラー化を行うことができます。

Oracle Solaris オペレーティングシステムを初期インストールしたあと、イメージパッケージングシステムの CLI および GUI (パッケージマネージャー) クライアントを使用して、追加のソフトウェアアプリケーションをパッケージリポジトリからインストールできます。

システムにパッケージをインストールすると、IPS クライアントを使用してパッケージを検索、アップグレード、および管理できるようになります。また、IPS クライアントを使用して、システム全体を新しい Oracle Solaris リリースにアップグレードしたり、リポジトリの作成および管理や、既存のリポジトリのミラー化を行なったりすることもできます。

IPS がインストールされているシステムからインターネットにアクセスできる場合、クライアントは Oracle Solaris 11.1 パッケージリポジトリ (デフォルトの solaris パブリッシャー) http://pkg.oracle.com/solaris/release/ にアクセスしてソフトウェアをインストールできます。

ゾーン管理者は、このドキュメントに記載されている制限の範囲内でパッケージツールを使用して、非大域ゾーンにインストールされたすべてのソフトウェアを管理できます。

ゾーンがインストールされている場合は、次の一般的な指針が適用されます。

- パッケージが大域ゾーンにインストールされている場合、非大域ゾーンでは大域ゾーンのシステムリポジトリサービスからパッケージをインストールすることができ、そのパッケージをインストールするためにネットワークを使用する必要はありません。そのパッケージが大域ゾーンにインストールされていない場合、ゾーンはゾーンプロキシサービスを使用してパブリッシャーにアクセスし、大域ゾーンを使用して、ネットワーク経由でパッケージをインストールする必要があります。
- 大域管理者または適切な承認を持つユーザーは、システムのすべてのゾーン内のソフトウェアを管理できます。

- 非大域ゾーンのルートファイルシステムは、Oracle Solaris パッケージツールを使用することで、大域ゾーンから管理できます。Oracle Solaris パッケージツールは、共通パッケージ (バンドル) 製品、スタンドアロン (別パッケージ) 製品、およびサードパーティー製品を管理するために、非大域ゾーン内でサポートされます。
- パッケージツールは、ゾーン対応の環境で動作します。このツールを使用すると、パッケージを非大域ゾーンにもインストールできます。

注 – 特定のパッケージ操作が行われている間、ゾーンはこの種のほかの操作から一時的にロックされます。システムはまた、要求された操作について管理者に確認してから処理を続行することがあります。

パッケージとゾーンについて

`solaris` ブランドゾーンにインストールされているソフトウェア (`brands(5)` に記載) は、大域ゾーンにインストールされているソフトウェアと互換性がある必要があります。`pkg` コマンドを使用すると、この互換性が自動的に適用されます。大域ゾーンで `pkg update` コマンドを実行してソフトウェアを更新すると、ゾーンも更新されて、ゾーンと大域ゾーンの同期が維持されます。非大域ゾーンと大域ゾーンには、異なるソフトウェアをインストールできます。ゾーンで `pkg` コマンドを使用して、ゾーン内のソフトウェアを管理することもできます。

`pkg update` コマンド (FMRI の指定なし) を大域ゾーンで実行すると、`pkg` はシステムの大域ゾーンと非大域ゾーンのすべてのソフトウェアを更新します。

`pkg install` の試行 (ドライランとも呼ばれる) インストール機能を Oracle Solaris ゾーンで使用できます。

ゾーンパッケージバリアントを使用すると、パッケージ内の各種のコンポーネントが、大域ゾーン (`global`) または非大域ゾーン (`nonglobal`) のいずれかのみにインストールされるように明確にタグ付けされます。指定されるパッケージには、非大域ゾーンにインストールされないようにタグ付けされたファイルを含めることができます。

非大域ゾーンのインストール時に、大域ゾーンにインストールされた Oracle Solaris パッケージの一部だけが完全にレプリケートされます。たとえば、非大域ゾーン内では、Oracle Solaris カーネルを含む多数のパッケージは必要ありません。すべての非大域ゾーンは、大域ゾーンと同一のカーネルを暗黙的に共有します。

詳細は、『Oracle Solaris 11.1 ソフトウェアパッケージの追加および更新』の「非大域ゾーンの操作」および『Oracle Solaris 11.1 システムのインストール』を参照してください。

注 – 非大域ゾーンを含むシステム上で大域ゾーンを更新すると、ゾーンに関するパッケージダウンロード情報が 2 回表示されているように見える場合があります。しかし、パッケージがダウンロードされるのは一度だけです。

ゾーンがインストールされているシステムへのパッケージの追加について

Oracle Solaris 11 リリースで pkg install コマンドを使用します。

```
# pkg install package_name
```

大域ゾーンでの pkg の使用

パッケージを大域ゾーンのみに追加するには、大域ゾーンで pkg install コマンドを使用します。パッケージはほかのゾーンには伝達されません。

非大域ゾーンでの pkg install コマンドの使用

pkg install コマンドは、パッケージを非大域ゾーンだけに追加するために、ゾーン管理者が非大域ゾーンで使用します。指定された非大域ゾーン内にパッケージを追加するには、ゾーン管理者として pkg install コマンドを実行します。

パッケージの依存性は IPS で自動的に処理されます。

カスタム AI マニフェストを使用したゾーンへのパッケージの追加

AI マニフェストを改訂して、インストール時にゾーンにソフトウェアを追加する処理を自動化できます。指定したパッケージと、それらが依存するパッケージがインストールされます。パッケージのデフォルトのリストは AI マニフェストから取得されます。デフォルトの AI マニフェストは /usr/share/auto_install/manifest/zone_default.xml です。パッケージの特定方法および操作方法については、『Oracle Solaris 11.1 ソフトウェアパッケージの追加および更新』を参照してください。

例 24–1 マニフェストの改訂

次の手順では、mercurial と vim エディタの完全インストールを *my-zone* という名前の構成済みゾーンに追加します。(デフォルトでは、solaris-small-server の一部である最小限の vim-core のみがインストールされます。)

例 24–1 マニフェストの改訂 (続き)

1. デフォルトの AI マニフェストをファイルの編集場所にコピーし、ファイルを書き込み可能にします。

```
# cp /usr/share/auto_install/manifest/zone_default.xml ~/my-zone-ai.xml
# chmod 644 ~/my-zone-ai.xml
```

2. ファイルを編集して、次のように mercurial および vim パッケージを software_data セクションに追加します。

```
      <software_data action="install">
              <name>pkg:/group/system/solaris-small-server</name>
              <name>pkg:/developer/versioning/mercurial</name>
              <name>pkg:/editor/vim</name>
            </software_data>
```

3. ゾーンをインストールします。

```
# zoneadm -z my-zone install -m ~/my-zone-ai.xml
```

システムには次のように表示されます。

```
A ZFS file system has been created for this zone.
Progress being logged to /var/log/zones/zoneadm.20111113T004303Z.my-zone.install
       Image: Preparing at /zones/my-zone/root.

 Install Log: /system/volatile/install.15496/install_log
 AI Manifest: /tmp/manifest.xml.XfaWpE
  SC Profile: /usr/share/auto_install/sc_profiles/enable_sci.xml
    Zonename: my-zone
Installation: Starting ...

              Creating IPS image
              Installing packages from:
                  solaris
                      origin:  http://localhost:1008/solaris/54453f3545de891d4daa841ddb3c844fe8804f55/

DOWNLOAD                                  PKGS       FILES    XFER (MB)
Completed                              169/169 34047/34047  185.6/185.6

PHASE                                        ACTIONS
Install Phase                              46498/46498

PHASE                                          ITEMS
Package State Update Phase                   169/169
Image State Update Phase                         2/2
Installation: Succeeded
...
```

ゾーン内でのパッケージの削除について

ゾーンがインストールされているシステムからパッケージを削除するには、pkg uninstall コマンドを使用します。

```
# pkg uninstall package_name
```

パッケージ情報の照会

ゾーンがインストールされているシステムでソフトウェアパッケージデータベースを照会するには、pkg info コマンドを使用します。

大域ゾーンでこのコマンドを使用すると、大域ゾーン内だけでソフトウェアパッケージデータベースを照会できます。非大域ゾーンでこのコマンドを使用すると、非大域ゾーン内だけでソフトウェアパッケージデータベースを照会できます。

インストール済みのゾーンがあるシステムでのプロキシ構成

『Oracle Solaris 11.1 ソフトウェアパッケージの追加および更新』の第 5 章「インストールされるイメージの構成」で説明されているように、--proxy オプションを使用してイメージに永続的なプロキシを設定するようにしてください。永続的なイメージプロキシ構成を使用せず、pkg コマンドの実行時に常に http_proxy および https_proxy 環境変数を使用してリポジトリにアクセスする場合は、SMF の system-repository サービスプロパティーを使用して、これらの同じプロキシを使用するように system-repository サービスも構成するようにしてください。pkg(1) のマニュアルページを参照してください。

大域ゾーンに構成されたリポジトリへのアクセスは、system-repository サービスを使用して非大域ゾーンに提供されます。大域ゾーンの起点に関するプロキシの更新は、system-repository の構成に対して自動的に行われます。この方法を使用する場合、system-repository SMF サービスに対する変更は必要ありません。

大域ゾーンの発行元に構成されているプロキシをオーバーライドして、system-repository SMF サービスによって使用されるプロキシを構成することも可能です。system-repository プロキシは、config/http_proxy または config/https_proxy SMF プロパティーを使用して設定できます。

詳細は、pkg.sysrepo(1M) のマニュアルページおよび『Oracle Solaris 11.1 ソフトウェアパッケージの追加および更新』を参照してください。

大域ゾーンでのプロキシの構成

大域ゾーンにはプロキシを直接構成することができ、大域ゾーンの起点に関するプロキシの更新は system-repository の構成に対して自動的に行われます。システムリポジトリサービスの変更は必要ありません。

例 24–2 大域ゾーンでのプロキシの構成

```
# pkg set-publisher --proxy http://www-proxy -g http://pkg-server pub
```

プロキシが 80 以外のポートで接続を受け入れる場合を除き、ポートを指定する必要はありません。

システム上にゾーンが存在する場合は、システムリポジトリサービスが再起動され、プロキシを使用して *pkg-server* へのアクセスが提供されます。

https_proxy および http_proxy を使用した system-repository プロキシのオーバーライド

イメージにプロキシを設定すること、および system-repository サービスプロキシのみを設定することをお勧めします。pkg コマンドを実行するときは、環境に https_proxy および http_proxy を設定するようにしてください。

このセクションの手順は、IPS 発行元リポジトリに直接接続していない内部サブネット上の system-repository サービスにプロキシを設定するために使用します。この手順を使用すると、大域ゾーンで pkg コマンドによって構成されたすべてのプロキシがオーバーライドされます。非大域ゾーンは、HTTP を介して system-repository と通信します。system-repository は、大域ゾーンに構成されたそのリポジトリ用のプロトコルを使用して発行元にアクセスします。

この構成によって、solaris 非大域ゾーンから大域ゾーンに設定されている発行元にも接続できるようになります。solaris ゾーンへの再帰的 pkg 操作は成功します。

例 24–3 https_proxy および http_proxy を使用した大域ゾーンのプロキシのオーバーライド

たとえば、solaris 非大域ゾーンを実行するシステム上のソフトウェアが IPS によって管理されていて、http と https の両方の URL にアクセスするためにプロキシサーバー http_proxy=http://*129.156.243.243:3128* を使用する必要があるとします。次の手順は、http_proxy および https_proxy 環境変数と SMF サービスプロパティーを使用して、大域ゾーンと非大域ゾーンから IPS リポジトリにアクセスできるようにする方法を示しています。

これらの変数は、ユーザーが非大域ゾーンから pkg コマンドを実行してシステム発行元の URI (Universal Resource Identifier) に接続しないかぎり、起点に設定されたプロキシ構成をオーバーライドします。この場合、コマンドは system-repository を経由します。

例 24–3 https_proxy および http_proxy を使用した大域ゾーンのプロキシのオーバーライド (続き)

解決可能なホスト名を使用することもできます。

1. 次の行を入力して、大域ゾーンの shell にプロキシを設定します。

```
# export http_proxy=http://129.156.243.243:3128
# export https_proxy=http://129.156.243.243:3128
```

プロキシを設定すると、pkg コマンドがプロキシサーバー経由で発行元に到達できるようになります。これは、https または http URL を使用し、大域ゾーンの system-repository を経由しない pkg 操作に影響を与えます。

2. システム上の solaris ゾーンが大域ゾーンから直接アクセスできる構成済みのシステム発行元を使用できるようにするには、次のコマンドを実行します。

```
# svccfg -s system-repository:default setprop config/http_proxy = http://129.156.243.243:3128
# svccfg -s system-repository:default setprop config/https_proxy = http://129.156.243.243:3128
```

3. ライブ SMF リポジトリ内で変更を有効にするには、次を実行します。

```
# svcadm refresh system-repository
```

4. 設定が機能していることを確認するには、次を実行します。

```
# svcprop -p config/http_proxy system-repository
# svcprop -p config/https_proxy system-repository
```

pkg コマンドの詳細は、pkg(1) のマニュアルページを参照してください。

ゾーンの並列更新

ゾーンは、順次ではなく並列的に更新されるように構成できます。並列更新により、システム上のすべてのゾーンを更新するのに必要な時間が大幅に短縮されます。詳細と構成例については、『Oracle Solaris 11.1 ソフトウェアパッケージの追加および更新』の「複数の非大域ゾーンの同時更新」を参照してください。

ゾーンの状態がパッケージの操作に与える影響

さまざまな状態の非大域ゾーンが存在しているシステムでパッケージコマンドを使用した結果を、次の表に示します。

ゾーンの状態	パッケージ操作への影響
構成済み	パッケージツールを実行できます。ソフトウェアはまだインストールされていません。

ゾーンの状態	パッケージ操作への影響
不完全	ゾーン上で zoneadm が動作している場合は、パッケージツールを使用しないようにしてください。ゾーン上で zoneadm プロセスが動作していない場合は、パッケージ操作を実行しても安全ですが、このゾーン内のソフトウェアは変更されず、ゾーン内のソフトウェアは依存性の解決に影響を与えません。
使用不可	ゾーン内のソフトウェアイメージにアクセスできません。ソフトウェアイメージは変更されず、依存性の解決にも影響を与えません。
インストール済み	パッケージツールを実行できます。 zoneadm -z *zonename* install が完了した直後にも、ゾーンはインストール済み状態に移行されます。
準備完了	パッケージツールを実行できます。
稼働	パッケージツールを実行できます。

非大域ゾーンのストレージにアクセスできない場合、または非大域ゾーンのイメージ (pkg(5) に記載) が大域ゾーンのイメージと同期していない場合、そのゾーンは使用不可の状態に移行します。この状態移行は、非大域ゾーンに影響を与えている問題によって大域ゾーンのパッケージ操作がブロックされないようにするために行われます。

ゾーンのストレージが一時的に使用できなくなり、インストール済みソフトウェアのバージョンを変更するパッケージ操作が行われたときは、ストレージの問題を修正したあとで、更新を可能にする solaris ブランドのいずれかの attach オプションを使用してゾーンを接続する必要がある場合があります。たとえば、大域ゾーンと使用不可の状態になっている非大域ゾーンの間で重要なソフトウェアのバージョンを同期するには、zoneadm -z *zonename* attach -u を使用する必要がある場合があります。

◆ ◆ ◆ 第 25 章

Oracle Solaris ゾーンの管理 (概要)

この章では、次の一般的なゾーン管理について説明します。

- 358 ページの「大域ゾーンの可視性とアクセス」
- 358 ページの「ゾーン内でのプロセス ID の可視性」
- 359 ページの「ゾーン内のシステム監視機能」
- 359 ページの「`zonestat` ユーティリティーを使用したアクティブなゾーンの統計情報の報告」
- 360 ページの「`fsstat` ユーティリティーを使用した非大域ゾーンの監視」
- 360 ページの「非大域ゾーンのノード名」
- 361 ページの「ファイルシステムと非大域ゾーン」
- 369 ページの「共有 IP 非大域ゾーンにおけるネットワーク」
- 371 ページの「排他的 IP 非大域ゾーンにおけるネットワーク」
- 373 ページの「非大域ゾーンでのデバイスの使用」
- 375 ページの「非大域ゾーンでのアプリケーションの実行」
- 376 ページの「非大域ゾーンで使用されるリソース制御」
- 377 ページの「ゾーンがインストールされている システムでの公平配分スケジューラ」
- 377 ページの「ゾーンがインストールされているシステムでの拡張アカウンティング」
- 378 ページの「非大域ゾーン内の特権」
- 383 ページの「ゾーン内での IP セキュリティーアーキテクチャーの使用」
- 383 ページの「ゾーン内での Oracle Solaris 監査の使用」
- 384 ページの「ゾーン内のコアファイル」
- 384 ページの「非大域ゾーン内での DTrace の実行」
- 385 ページの「ゾーンがインストールされている Oracle Solaris システムのバックアップについて」
- 387 ページの「非大域ゾーン内でバックアップするデータの決定」
- 389 ページの「ゾーンがインストールされているシステムで使用するコマンド」

`solaris10` ブランドゾーンの詳細は、パート III「Oracle Solaris 10 ゾーン」を参照してください。

大域ゾーンの可視性とアクセス

大域ゾーンは、システムのデフォルトゾーンとしても、システム規模の管理制御用ゾーンとしても機能します。この二重の役割と関係のある管理上の問題が存在します。ゾーン内部のアプリケーションはほかのゾーン内のプロセスおよびほかのシステムオブジェクトにアクセスするため、管理操作の効果が予期したものよりも大きい場合があります。たとえば、サービスの停止スクリプトでは、プロセスを終了させるためのシグナルの送信にしばしば `pkill` が使用されます。大域ゾーンからこの種のスクリプトを実行すると、システム内の該当するすべてのプロセスが、ゾーンに関係なくシグナルを送信します。

多くの場合、システム規模の範囲が必要になります。たとえば、システム規模のリソース使用状況を監視する場合、システム全体のプロセス統計情報を表示する必要があります。大域ゾーンのアクティビティーだけのビューには、システムリソースの一部または全体を共有可能なシステム内のほかのゾーンからの関連情報が欠落しています。この種のビューは、CPU などのシステムリソースがリソース管理機能を使用して厳密に区分されていない場合、特に重要です。

このため、大域ゾーン内のプロセスから、非大域ゾーン内のプロセスおよびほかのオブジェクトを監視できます。これにより、この種のプロセスがシステム規模の監視機能を備えることが可能になります。ほかのゾーン内のプロセスを制御したりシグナルを送信したりする機能は、`PRIV_PROC_ZONE` 特権により制限されます。この特権は、特権のないプロセスに設定された制限をオーバーライドできるため、`PRIV_PROC_OWNER` に類似しています。この場合の制限は、大域ゾーン内の特権のないプロセスはほかのゾーン内のプロセスにシグナルを送信したり制御したりすることはできない、というものです。これは、プロセスのユーザー ID が一致するか、動作しているプロセスが `PRIV_PROC_OWNER` 特権を保持している場合でも適用されます。`PRIV_PROC_ZONE` 特権を、そうでなければ特権の付与されたプロセスから削除して、大域ゾーンへの操作に制限できます。

`zoneidlist` を使用した照合プロセスの詳細は、`pgrep(1)` および `pkill(1)` のマニュアルページを参照してください。

ゾーン内でのプロセス ID の可視性

同一ゾーン内のプロセスだけが、`kill` や `priocntl` コマンドなどの、プロセス ID を指定するシステムコールインタフェースを介して表示されます。詳細については、`kill(1)` および `priocntl(1)` のマニュアルページを参照してください。

ゾーン内のシステム監視機能

ps コマンドに、次の変更が加えられました。

- 出力形式の指定には -o オプションを使用します。このオプションを使用すると、プロセスのゾーン ID またはプロセスを実行中のゾーンの名前を出力できます。
- 指定されたゾーン内のプロセスだけをリスト表示するには、-z *zonelist* オプションを使用します。ゾーンの指定には、ゾーン名またはゾーン ID を使用できます。このオプションは、大域ゾーン内でコマンドを実行する場合にのみ有用です。
- プロセスに関連するゾーンの名前を出力するには -Z オプションを使用します。列見出し ZONE の下に名前が出力されます。

詳細は、ps(1) のマニュアルページを参照してください。

-z *zonename* オプションが、次の Oracle Solaris ユーティリティーに追加されました。このオプションを使用して情報をフィルタ処理し、指定したゾーンだけを含めることができます。

- ipcs (ipcs(1) のマニュアルページを参照)
- pgrep (pgrep(1) のマニュアルページを参照)
- ptree (proc(1) のマニュアルページを参照)
- prstat (prstat(1M) のマニュアルページを参照)

コマンドに加えられた変更の全一覧については、表 25–5 を参照してください。

zonestat ユーティリティーを使用したアクティブなゾーンの統計情報の報告

zonestat ユーティリティーを使用する場合は、zonestat(1) のマニュアルページおよび397 ページの「非大域ゾーンでの zonestat ユーティリティーの使用」を参照してください。

zonestat ユーティリティーは、現在実行中のゾーンの CPU、メモリー、およびリソース制御の使用効率について報告します。zonestat ユーティリティーは、指定された間隔で一連のレポートを出力します。必要に応じて、このユーティリティーは 1 つ以上のサマリーレポートを出力できます。

zonestat ユーティリティーは、排他的 IP ゾーン内のネットワーク帯域幅の使用効率についても報告を生成します。排他的 IP ゾーンには、独自の IP に関連付けられた状態と 1 つ以上の専用のデータリンクが保持されます。

非大域ゾーンから実行すると、そのゾーンから参照可能なプロセッサセットのみが報告されます。非大域ゾーンの出力には、すべてのメモリーリソースおよび制限リソースが含まれます。

非大域ゾーンで zonestat サービスを使用するには、大域ゾーンの zonestat サービスがオンラインである必要があります。各非大域ゾーンの zonestat サービスは、大域ゾーンの zonestat サービスからシステム構成および使用効率データを読み取ります。

zonestatd システムデーモンは、システムブート時に開始されます。このデーモンは、ゾーンによるシステムリソースの使用を監視するほか、psrset プロセッサセット、プールプロセッサセット、リソース制御設定などのゾーンおよびシステム構成情報を監視します。構成できるコンポーネントはありません。

fsstat ユーティリティーを使用した非大域ゾーンの監視

fsstat ユーティリティーは、ゾーンごとの kstats (集計を含む) を収集して出力します。デフォルトでは、このユーティリティーは実行中のすべてのゾーンの集計を報告します。fstype ごとの kstat がそれぞれのゾーン用に生成されます。大域ゾーンの kstat は、その排他的なアクティビティーを報告します。大域ゾーンでは、システム上のすべてのゾーンの kstat を表示できます。非大域ゾーンでは、ユーティリティーが実行されているゾーンに関連付けられた kstats のみが表示されます。非大域ゾーンでは、ほかのゾーンのファイルシステムアクティビティーを監視できません。

詳細は、fsstat(1M) のマニュアルページおよび401 ページの「すべてのゾーンのゾーン別の fstype 統計情報の報告」を参照してください。

非大域ゾーンのノード名

ノード名は、システム名のローカルソースです。ノード名はゾーン名などのように一意である必要があります。ノード名は、ゾーン管理者が設定できます。

```
# hostname myhostname
```

ホスト名を表示するには、hostname と入力します。

```
# hostname
...
myhostname
```

ゾーンでの NFS サーバーの実行

ゾーンに NFS 共有を作成するには、NFS サーバーパッケージ svc:/network/nfs/server:default をゾーンにインストールする必要があります。NFS サーバーパッケージは、ゾーンの作成時にはインストールできません。

ゾーン内での NFS 共有を防止するため、ゾーン構成で sys_share 特権を禁止できます。表 25–1 を参照してください。

次の制限事項があります。

- ゾーンをまたぐ LOFS マウントはゾーンから共有できません。
- ゾーン内にマウントされたファイルシステムは、大域ゾーンから共有できません。
- リモートダイレクトメモリーアクセス (RDMA) を使用した NFS は、ゾーンでサポートされません。
- Oracle Sun Cluster HA for NFS (HANFS) のフェイルオーバーは、ゾーンでサポートされません。

『Oracle Solaris のシステム管理 (ネットワークサービス)』を参照してください。

ファイルシステムと非大域ゾーン

このセクションでは、ゾーンがインストールされている Oracle Solaris システムでファイルシステムを使用する場合の問題について説明します。各ゾーンは、ゾーンの root と呼ばれるディレクトリをルートとする、ファイルシステム階層の独自セクションを保持します。ゾーン内のプロセスは、ゾーンルート以下の階層部分内のファイルだけにアクセスできます。ゾーン内で chroot ユーティリティーを使用できますが、プロセスをゾーン内のルートパスに制限する場合だけです。chroot の詳細については、chroot(1M) のマニュアルページを参照してください。

-o nosuid オプション

mount ユーティリティーで -o nosuid オプションを指定する場合、次の機能を利用できます。

- nosetuid オプションを使用してマウントされたファイルシステム上の setuid バイナリに基づくプロセスは、setuid バイナリの権限では動作しません。プロセスは、バイナリを実行するユーザーの権限で動作します。

  たとえば、ユーザーが root の所有する setuid バイナリを実行する場合、プロセスはそのユーザーの権限で動作します。
- ファイルシステム内のデバイス特殊エントリを開くことはできません。この動作は、nodevices オプションを指定する場合と同じです。

mount(1M) のマニュアルページに記載されているように、このファイルシステム固有オプションは mount ユーティリティーでマウントできるすべての Oracle Solaris ファイルシステムで使用できます。これらのファイルシステムの一覧については、このガイドの 362 ページの「ゾーン内でのファイルシステムのマウント」を参照してください。マウント機能についても説明します。-o nosuid オプションの詳細については、『Oracle Solaris のシステム管理 (ネットワークサービス)』の「ネットワークファイルシステムへのアクセス (リファレンス)」を参照してください。

ゾーン内でのファイルシステムのマウント

ファイルシステムをゾーン内部でマウントする場合、nodevices オプションが適用されます。たとえば、ゾーンに、UFS ファイルシステムに対応するブロックデバイス (/dev/dsk/c0t0d0s7) および raw デバイス (/dev/rdsk/c0t0d0s7) へのアクセスが許可される場合、ゾーン内部からマウントを行うと、ファイルシステムのマウントで自動的に nodevices オプションが適用されます。この規則は、zonecfg 構成を使用して指定されたマウントには適用されません。

次の表で、非大域ゾーン内でファイルシステムをマウントする場合のオプションを説明します。これらの補助的なマウント方法の実行手順については、269 ページの「ゾーンを構成、検証、および確定する」および404 ページの「稼働中の非大域ゾーン内でファイルシステムをマウントする」を参照してください。

/usr/lib/*fstype*/mount 内にマウントバイナリが存在する場合、表に含まれない任意のファイルシステムタイプを構成内で指定できます。

非大域ゾーンの内部から HSFS と NFS 以外のファイルシステムタイプをマウントするには、zonecfg fs-allowed プロパティーを使用して、構成にファイルシステムタイプも追加します。

デフォルト以外のファイルシステムマウントを許可すると、ゾーン管理者によってシステムが危険にさらされることがあります。

ファイルシステム	非大域ゾーン内のマウントオプション
AutoFS	マウントに zonecfg を使用できません。ゾーン内部からマウントすることは可能です。
CacheFS	非大域ゾーン内では使用できません。
FDFS	zonecfg を使用してマウントできます。ゾーン内からマウントできます。
HSFS	zonecfg を使用してマウントできます。ゾーン内からマウントできます。
LOFS	zonecfg を使用してマウントできます。ゾーン内からマウントできます。

ファイルシステム	非大域ゾーン内のマウントオプション
MNTFS	マウントに zonecfg を使用できません。ゾーン内部からマウントすることは可能です。
NFS	マウントに zonecfg を使用できません。ゾーン内で現在サポートされているバージョンである V2、V3、および V4 を、ゾーン内からマウントできます。
PCFS	zonecfg を使用してマウントできます。ゾーン内からマウントできます。
PROCFS	マウントに zonecfg を使用できません。ゾーン内部からマウントすることは可能です。
TMPFS	zonecfg を使用してマウントできます。ゾーン内からマウントできます。
UDFS	zonecfg を使用してマウントできます。ゾーン内からマウントできます。

UFS	zonecfg を使用してマウントできます。ゾーン内からマウントできます。
	
	注 – quota コマンド (quota(1M) を参照) は、zonecfg add fs リソースを使用して追加された UFS ファイルシステムの割り当て情報の取得に使用することはできません。
	
	add fs を使用する場合は、system/file-system/ufs パッケージを大域ゾーンにインストールする必要があります。zonecfg コマンドを使用して非大域ゾーンで UFS ファイルシステムを使用する場合は、インストール後にこのパッケージをゾーンにインストールするか、AI マニフェストスクリプトを使用してインストールする必要があります。
	
	次を 1 行で入力します。
	
	`global# pkg -R /tank/zones/my-zone/root \`
	`install system/file-system/ufs`

VxFS	zonecfg を使用してマウントできます。ゾーン内からマウントできます。
ZFS	zonecfg dataset および fs リソースタイプを使用してマウントできます。

詳細は、270 ページの「ゾーンの構成方法」、404 ページの「稼働中の非大域ゾーン内でファイルシステムをマウントする」、および mount(1M) のマニュアルページを参照してください。

ゾーン内でのファイルシステムのアンマウント

ファイルシステムをアンマウントできるかどうかは、どの管理者がそのファイルシステムを最初にマウントしたかによって決まります。`zonecfg` コマンドを使用してゾーンを構成するときにファイルシステムが指定されている場合は、大域ゾーンがそのマウントの所有者となるので、非大域ゾーンの管理者はそのファイルシステムをアンマウントすることはできません。非大域ゾーンの `/etc/vfstab` ファイルにマウントを指定するなどの方法でファイルシステムが非大域ゾーンからマウントされている場合には、その非大域ゾーンの管理者はそのファイルシステムをアンマウントできます。

セキュリティーの制限およびファイルシステムの動作

ゾーン内部から特定のファイルシステムをマウントする場合、適用されるセキュリティーの制限が存在します。ほかのファイルシステムは、ゾーン内でマウントされたときに特有の動作を行います。変更されたファイルシステムの一覧を、次に示します。

AutoFS
: autofs は、自動的に適切なファイルシステムをマウントするためのクライアント側のサービスです。クライアントが現在マウントされていないファイルシステムにアクセスしようとすると、AutoFS ファイルシステムはその要求に介入し、`automountd` を呼び出して要求されたディレクトリをマウントします。ゾーン内で確立された AutoFS マウントは、そのゾーンだけで有効です。大域ゾーンを含むほかのゾーンからそのマウントにアクセスすることはできません。ゾーンが停止またはリブートすると、マウントは削除されます。AutoFS の詳細ついては、『Oracle Solaris のシステム管理 (ネットワークサービス)』の「autofs のしくみ」を参照してください。

  各ゾーンは、`automountd` の独自コピーを実行します。自動マップおよびタイムアウトは、ゾーン管理者により制御されます。大域ゾーンから非大域ゾーンの AutoFS マウントポイントを横断的に使用して、別のゾーン内でマウントを始動させることはできません。

  別のマウントが始動する際、カーネル内で特定の AutoFS マウントが作成されます。この種のマウントは、一括してマウントまたはアンマウントする必要があるため、通常の `umount` インタフェースを使用して削除することはできません。この機能は、ゾーンの停止処理用であることに注意してください。

MNTFS
: MNTFS は、ローカルシステムのマウント済みファイルシステムのテーブルに読み取り専用アクセスを提供する仮想ファイルシステムです。非大域ゾーン内から `mnttab` を使用して表示可能なファイルシステムのセットは、ゾーン内でマウント

されたファイルシステムセットおよびルート (/) のエントリで構成されます。/dev/rdsk/c0t0d0s0 などの、ゾーン内からアクセス不可能な特殊なデバイスを保持するマウントポイントは、マウントポイントと同じ特殊なデバイスセットを保持します。システム内のすべてのマウントが、大域ゾーンの /etc/mnttab テーブルから表示可能になります。MNTFS の詳細は、『Oracle Solaris 11.1 の管理: デバイスとファイルシステム』の「Oracle Solaris ファイルシステムのマウントおよびマウント解除」を参照してください。

NFS
: ゾーン内部で確立された NFS マウントは、そのゾーンでのみ有効です。大域ゾーンを含むほかのゾーンからそのマウントにアクセスすることはできません。ゾーンが停止またはリブートすると、マウントは削除されます。

  ゾーン内部からの NFS マウントは、nodevices オプションを使用してマウントされたかのように動作します。

  nfsstat コマンドの出力は、コマンドが実行されたゾーンにのみ関係があります。たとえば、コマンドが大域ゾーン内で実行される場合、大域ゾーンに関する情報だけが出力されます。nfsstat コマンドの詳細は、nfsstat(1M) のマニュアルページを参照してください。

PROCFS
: PROCFS とも呼ばれる /proc ファイルシステムは、プロセスの可視性とアクセス制限、およびプロセスのゾーン関連性に関する情報を提供します。/proc では、同じゾーン内のプロセスだけを表示できます。

  大域ゾーン内のプロセスから、非大域ゾーン内のプロセスおよびほかのオブジェクトを監視できます。これにより、この種のプロセスがシステム規模の監視機能を備えることが可能になります。

  ゾーン内部からは、procfs マウントは nodevices オプションを使用してマウントされたかのように動作します。procfs の詳細は、proc(4) のマニュアルページを参照してください。

LOFS
: LOFS でマウント可能なファイルシステムの範囲は、ゾーンで表示可能なファイルシステム部分に限定されています。このため、ゾーン内での LOFS マウントには制限はありません。

UFS、UDFS、PCFS、およびその他のストレージに基づいたファイルシステム
: zonecfg コマンドを使用して、UFS などの fsck バイナリを保持するストレージに基づいたファイルシステムを構成する場合、ゾーン管理者は raw パラメータを指定する必要があります。このパラメータは、/dev/rdsk/c0t0d0s7 などの raw (文字) デバイスを示します。zoneadmd デーモンは、fsck コマンドを preen モード (fsck -p) で自動的に実行します。このコマンドは、ファイルシステムをマウントする前に、非対話的にファイルシステムを検査および修正します。fsck が失敗した場合、zoneadmd を使用してゾーンを準備完了状態にすることはできません。raw により指定されるパスを、相対パスにするこ とはできません。

/usr/lib/fs/*fstype*/fsck 内で fsck バイナリを提供しないファイルシステムで、デバイスを fsck に指定するのは誤りです。また、そのファイルシステムに fsck バイナリが存在する場合に、デバイスを fsck に指定しないことも誤りです。

詳細は、292 ページの「zoneadmd デーモン」および fsck(1M) コマンドを参照してください。

ZFS

デフォルトのデータセット (229 ページの「ゾーンでマウントされるファイルシステム」に記載) に加えて、zonecfg コマンドに add dataset リソースを付けて実行することにより、ZFS データセットを非大域ゾーンに追加できます。データセットは非大域ゾーンでマウントされ可視になり、大域ゾーンでも可視になります。ゾーン管理者は、そのデータセット内のファイルシステムの作成と破棄、およびデータセットのプロパティーの変更を行うことができます。

zfs の zoned 属性は、データセットが非大域ゾーンに追加されたかどうかを示します。

```
# zfs get zoned tank/sales
NAME          PROPERTY    VALUE       SOURCE
tank/sales    zoned       on          local
```

データセットリソースを使用して非大域ゾーンに委任された各データセットには、別名が付けられます。データセットのレイアウトはゾーン内では表示されません。別名が付けられた各データセットは、プールのようにゾーンに表示されます。データセットのデフォルトの別名は、データセット名の最後のコンポーネントです。たとえば、委任されたデータセット tank/sales にデフォルトの別名が使用された場合、ゾーンには sales という仮想 ZFS プールが表示されます。別名が別の値になるようにカスタマイズするには、データセットリソース内の別名プロパティーを設定します。

各非大域ゾーンの zonepath データセット内には、rpool というデータセットが存在します。すべての非大域ゾーンで、ゾーンのこの rpool データセットは rpool として別名が付けられます。

```
my-zone# zfs list -o name,zoned,mounted,mountpoint
NAME                 ZONED  MOUNTED  MOUNTPOINT
rpool                   on       no  /rpool
rpool/ROOT              on       no  legacy
rpool/ROOT/solaris      on      yes  /
rpool/export            on       no  /export
rpool/export/home       on       no  /export/home
```

データセットの別名は、ZFS プールと同じ名前制限の対象となります。この制限については、zpool(1M) のマニュアルページに記載されています。

大域ゾーンのデータセットを共有する場合は、zonecfg コマンドとともに add fs サブコマンドを使用して、LOFS マウントした ZFS ファイルシステムを追加できます。データセットのプロパティーの設定および制御は、大域管理者または適切な承認を付与されたユーザーが担当します。

ZFS の詳細は、『Oracle Solaris 11.1 の管理: ZFS ファイルシステム』の第 9 章「Oracle Solaris ZFS の高度なトピック」を参照してください。

NFS クライアントとして機能する非大域ゾーン

ゾーンは、NFS クライアントとしても機能できます。バージョン 2、バージョン 3、およびバージョン 4 プロトコルがサポートされます。これらの NFS バージョンについては、『Oracle Solaris のシステム管理 (ネットワークサービス)』の「NFS サービスの機能」を参照してください。

デフォルトのバージョンは、NFS バージョン 4 です。次のいずれかの方法を使用して、クライアント上でほかの NFS バージョンを有効にできます。

- `sharectl`(1M) を使用すると、プロパティーを設定できます。`NFS_CLIENT_VERSMAX=`*number* を設定することで、指定したバージョンをゾーンのデフォルトとして使用できます。『Oracle Solaris のシステム管理 (ネットワークサービス)』の「NFS サービスの設定」を参照してください。『Oracle Solaris 11.1 でのネットワークファイルシステムの管理』の「クライアント上で異なるバージョンの NFS を選択する方法」の手順を使用します。
- バージョンマウントを手動で作成できます。この方法は、`sharectl` 設定をオーバーライドします。『Oracle Solaris のシステム管理 (ネットワークサービス)』の「NFS サービスの設定」を参照してください。『Oracle Solaris 11.1 でのネットワークファイルシステムの管理』の「クライアント上で異なるバージョンの NFS を選択する方法」の手順を使用します。

ゾーン内での `mknod` の使用禁止

`mknod`(1M) のマニュアルページに記載されているように、`mknod` コマンドを使用して、非大域ゾーン内で特殊ファイルを作成することはできません。

ファイルシステムの行き来

ゾーンのファイルシステム名前空間は、大域ゾーンからアクセス可能な名前空間の一部です。大域ゾーン内の特権のないプロセスが、非大域ゾーンのファイルシステム階層を行き来することはできません。これには、次のような理由があります。

- root だけが、ゾーンルートの親ディレクトリを所有、読み込み可能、書き込み可能、および実行可能に指定する
- `/proc` によりエクスポートされたディレクトリへのアクセスが制限される

別のゾーン用にマウントされた AutoFS ノードへのアクセスを試みても、失敗します。大域管理者は、その子孫がほかのゾーンに含まれる自動マップを保持してはいけません。

大域ゾーンから非大域ゾーンにアクセスする際の制限

非大域ゾーンのインストール後に、システムのバックアップユーティリティー以外のコマンドを使用して、大域ゾーンからそのゾーンに直接アクセスしてはいけません。また、非大域ゾーンを未知の環境に公開した後は、そのゾーンが安全であると考えることはできません。たとえば、公開されたネットワーク上に配置されたゾーンについて考えてみましょう。この場合、ゾーンのセキュリティーが低下し、ファイルシステムの内容が変更される可能性があります。セキュリティーが低下する可能性がある場合、大域管理者はゾーンを信頼できないものとして処理する必要があります。

次の両方の条件が当てはまる場合、`-R` オプションまたは `-b` オプション (またはこれと同等なオプション) を使って代替ルートを指定可能なコマンドを使用してはいけません。

- コマンドが大域ゾーン内で実行される。
- 代替ルートが非大域ゾーン内のいずれかのパスを参照している。現在動作中のシステムの大域ゾーンからの相対パスの場合や、代替ルート内の大域ゾーンからの相対パスの場合を含む。

たとえば、`pkgadd` ユーティリティーに `-R` *root_path* オプションを指定して、非大域ゾーンのルートパスを保持する大域ゾーンから実行する場合が、この条件に該当します。

次に、`-R` と代替ルートパスを使用するコマンド、プログラム、およびユーティリティーを一覧表示します。

- `auditreduce`
- `bart`
- `installf`
- `localeadm`
- `makeuuid`
- `metaroot`
- `pkg`
- `prodreg`
- `removef`
- `routeadm`
- `showrev`
- `syseventadm`

次に、`-b` と代替ルートパスを使用するコマンドおよびプログラムを一覧表示します。

- `add_drv`
- `pprosetup`

- rem_drv
- roleadd
- update_drv
- useradd

共有 IP 非大域ゾーンにおけるネットワーク

ゾーンがインストールされている Oracle Solaris システムでは、ゾーンはネットワーク経由で互いに通信できます。ゾーンはすべて別個の結合または接続を保持します。また、すべてのゾーンは独自のサーバーデーモンを実行できます。これらのデーモンは、同一のポート番号で競合することなく待機できます。IP スタックは、着信接続の IP アドレスを考慮に入れることで競合を解決します。IP アドレスにより、ゾーンが識別されます。

共有 IP タイプを使用するには、大域ゾーンのネットワーク構成は、自動ネットワーク構成ではなく ipadm を使用して行う必要があります。ipadm が使用されている場合、次のコマンドは DefaultFixed を返すはずです。

```
# svcprop -p netcfg/active_ncp svc:/network/physical:default
DefaultFixed
```

共有 IP ゾーンの区分化

共有 IP はデフォルトではありませんが、このタイプはサポートされます。

ゾーンをサポートするシステム内の IP スタックは、ゾーン間のネットワークトラフィックの分離を実装します。IP トラフィックを受け取るアプリケーションは、同じゾーンに送信されたトラフィックの受信だけを実行できます。

システム上の各論理インタフェースは、特定のゾーンに所属します。デフォルトは、大域ゾーンです。zonecfg ユーティリティー経由でゾーンに割り当てられた論理ネットワークインタフェースは、ネットワーク経由での通信に使用されます。各ストリームおよび接続は、それを開いたプロセスのゾーンに所属します。

上位層ストリームと論理インタフェース間の結合は、制限されます。ストリームが確立できるのは、同一ゾーン内の論理インタフェースへの結合だけです。同様に、論理インタフェースからのパケットを渡すことができるのは、論理インタフェースと同じゾーン内の上位層ストリームに対してだけです。

各ゾーンは、独自のバインドセットを保持します。アドレスが使用中であるため、各ゾーンは、同一のポート番号で待機する同じアプリケーションを、バインドが失敗することなく稼働可能です。各ゾーンは、次のような各種のネットワークサービスの固有のバージョンを実行できます。

- 完全な構成ファイルを保持するインターネットサービスデーモン (inetd(1M) のマニュアルページを参照)

- sendmail (sendmail(1M) のマニュアルページを参照)
- apache

大域ゾーン以外のゾーンは、ネットワークへのアクセスが制限されています。標準の TCP および UDP ソケットインタフェースが利用可能ですが、SOCK_RAW ソケットインタフェースは ICMP (Internet Control Message Protocol) に制限されています。ICMP は、ネットワークのエラー状況を検出および報告したり、ping コマンドを使用するのに必要です。

共有 IP ネットワークインタフェース

ネットワーク接続を必要とする非大域ゾーンには、それぞれ 1 つ以上の専用 IP アドレスがあります。これらのアドレスは、ゾーン内に配置可能な論理ネットワークインタフェースに関連付けられています。zonecfg により構成されるゾーンネットワークインタフェースは、ブート時に自動的に設定されてゾーン内に配置されます。ipadm コマンドを使用すると、ゾーンの稼働中に論理インタフェースを追加または削除できます。インタフェース構成およびネットワーク経路を変更できるのは、大域管理者または適切な承認を付与されたユーザーのみです。

非大域ゾーン内では、そのゾーンのインタフェースのみが ipadm コマンドで表示されます。

詳細については、ipadm(1M) および if_tcp(7P) のマニュアルページを参照してください。

同一マシン上の共有 IP ゾーン間の IP トラフィック

転送テーブルに宛先への使用可能な経路がある場合、共有 IP ゾーンは指定された IP 宛先に到達できます。転送テーブルを表示するには、ゾーン内から netstat コマンドに -r オプションを付けて実行します。IP 転送ルールは、別のゾーンまたは別のシステムの IP 宛先でも同じです。

共有 IP ゾーンでの Oracle Solaris IP フィルタ

Oracle Solaris IP フィルタは、ステートフルパケットフィルタリングとネットワークアドレス変換 (NAT) を行います。ステートフルパケットフィルタは、アクティブな接続の状態を監視し、取得した情報を使用して、ファイアウォールの通過を許可するネットワークパケットを決定することができます。Oracle Solaris IP フィルタには、ステートレスパケットフィルタリングと、アドレスプールの作成および管理を行う機能もあります。詳細は、『Oracle Solaris 11.1 でのネットワークのセキュリティー保護』の第 4 章「Oracle Solaris の IP フィルタ (概要)」を参照してください。

『Oracle Solaris 11.1 でのネットワークのセキュリティー保護』の第 5 章「IP フィルタ (タスク)」に記載されているとおり、ループバックフィルタリングをオンに設定することで、非大域ゾーンで Oracle Solaris IP フィルタを有効にすることができます。

Oracle Solaris IP フィルタは、オープンソースの IP Filter ソフトウェアを基にしています。

共有 IP ゾーン内の IP ネットワークマルチパス

IP ネットワークマルチパス (IPMP) は、同一の IP リンク上に複数のインタフェースを保持するシステムで、物理インタフェースの障害検出および透過的なネットワークアクセスフェイルオーバーを提供します。IPMP も、複数のインタフェースを保持するシステムについて、パケットの負荷分散を提供します。

すべてのネットワーク構成は、大域ゾーン内で行われます。大域ゾーン内で IPMP を構成した後で、この機能を非大域ゾーンに拡張できます。ゾーンの構成時に、ゾーンのアドレスを IPMP グループ内に配置することでこの機能は拡張されます。その後、大域ゾーン内のいずれかのインタフェースで障害が発生すると、非大域ゾーンアドレスが別のインタフェースカードに移されます。

指定された非大域ゾーンで `ipadm` コマンドを使用すると、ゾーンに関連するインタフェースだけが表示されます。

409 ページの「IP ネットワークマルチパス機能を共有 IP 非大域ゾーンに拡張する方法」を参照してください。ゾーンの構成手順については、270 ページの「ゾーンの構成方法」を参照してください。IPMP の機能、コンポーネント、および使用法については、『Oracle Solaris 11.1 ネットワークパフォーマンスの管理』の第 5 章「IPMP の概要」を参照してください。

排他的 IP 非大域ゾーンにおけるネットワーク

排他的 IP ゾーンは、IP 関連の状態を独自に保持します。ゾーンの構成時に、独自のデータリンクセットがゾーンに割り当てられます。

パケットは、物理リンクに送信されます。その後、Ethernet スイッチや IP ルーターなどのデバイスがパケットを宛先へ転送します。この送信先は、送信元と同じマシン上の別のゾーンである可能性もあります。

仮想リンクの場合、パケットは最初に仮想スイッチに送信されます。宛先のリンクが同じデバイス (同じ物理リンクまたは etherstub 上の VNIC など) を経由する場合、パケットは宛先の VNIC に直接送信されます。それ以外の場合、パケットは VNIC のベースとなる物理リンクに送信されます。

排他的 IP 非大域ゾーンで使用できる機能については、227 ページの「排他的 IP 非大域ゾーン」を参照してください。

排他的 IP ゾーンの区分化

排他的 IP ゾーンはそれぞれ個別の TCP/IP スタックを持っているため、下位のデータリンク層まで分離されます。排他的 IP ゾーンには、1 つ以上のデータリンク名が大域管理者によって割り当てられます。データリンク名は、NIC または NIC 上の VLAN の場合があります。ゾーン管理者は、大域ゾーンの場合と同じ柔軟性とオプションで、これらのデータリンクの IP を構成できます。

排他的 IP データリンクインタフェース

1 つのデータリンク名は、1 つのゾーンだけに割り当てる必要があります。

`dladm show-link` コマンドを使用して、実行中のゾーンに割り当てられているデータリンクを表示できます。

```
sol-t2000-10{pennyc}1: dladm show-link
LINK                CLASS     MTU    STATE    OVER
vsw0                phys      1500   up       --
e1000g0             phys      1500   up       --
e1000g2             phys      1500   up       --
e1000g1             phys      1500   up       --
e1000g3             phys      1500   up       --
zoneA/net0          vnic      1500   up       e1000g0
zoneB/net0          vnic      1500   up       e1000g0
aggr1               aggr      1500   up       e1000g2 e1000g3
vnic0               vnic      1500   up       e1000g1
zoneA/vnic0         vnic      1500   up       e1000g1
vnic1               vnic      1500   up       e1000g1
zoneB/vnic1         vnic      1500   up       e1000g1
vnic3               vnic      1500   up       aggr1
vnic4               vnic      1500   up       aggr1
zoneB/vnic4         vnic      1500   up       aggr1
```

詳細は、`dladm(1M)` を参照してください。

同一マシン上の排他的 IP ゾーン間の IP トラフィック

排他的 IP ゾーン間での IP パケットの内部ループバックはありません。すべてのパケットが下位のデータリンクまで送信されます。これは通常、パケットがネットワークインタフェース上に送信されることを意味します。その後、Ethernet スイッチや IP ルーターなどのデバイスがパケットを宛先へ転送します。この送信先は、送信元と同じマシン上の別のゾーンである可能性もあります。

排他的 IP ゾーンにおける Oracle Solaris IP フィルタ

排他的 IP ゾーンでは、大域ゾーンのものと同じ IP フィルタ機能を使用できます。排他的 IP ゾーンでの IP フィルタの構成方法も大域ゾーンと同じです。

排他的 IP ゾーン内の IP ネットワークマルチパス

IP ネットワークマルチパス (IPMP) は、同一の IP リンク上に複数のインタフェースを保持するシステムで、物理インタフェースの障害検出および透過的なネットワークアクセスフェイルオーバーを提供します。IPMP も、複数のインタフェースを保持するシステムについて、パケットの負荷分散を提供します。

データリンク構成は、大域ゾーン内で行われます。最初に、`zonecfg` を使用して、複数のデータリンクインタフェースをゾーンに割り当てます。複数のデータリンクインタフェースを同じ IP サブネットに接続する必要があります。その後、排他的 IP ゾーン内からゾーン管理者が IPMP を構成することができます。

非大域ゾーンでのデバイスの使用

あるゾーンのプロセスが別のゾーンで実行中のプロセスに干渉することがないよう、ゾーン内部で利用可能なデバイスセットには制限が課されています。たとえば、ゾーン内のプロセスが、カーネルメモリーおよびルートディスクの内容を変更することはできません。このため、デフォルトでは、ゾーン内で安全に利用可能であると見なされる特定の仮想デバイスだけを使用できます。`zonecfg` ユーティリティーを使用すると、利用可能なデバイスを特定のゾーンに追加できます。

`/dev` および `/devices` 名前空間

`devfs(7FS)` のマニュアルページで説明されているように、Oracle Solaris システムでは `/devices` の管理に `devfs` ファイルシステムが使用されます。この名前空間内の各要素は、ハードウェアデバイス、仮想デバイス、またはネクサスデバイスへの物理パスを表します。名前空間には、デバイスツリーが反映されます。したがって、ファイルシステムは、ディレクトリおよびデバイス特殊ファイルの階層により生成されます。

デバイスは相対 `/dev` 階層に従ってグループ化されます。たとえば、大域ゾーンの `/dev` の下のすべてのデバイスは、大域ゾーンのデバイスとしてグループ化されます。非大域ゾーンの場合、デバイスはゾーンのルートパスの下の `/dev` ディレクトリでグループ化されます。各グループは、`/dev` ディレクトリの下にマウントされた `/dev` ファイルシステムのインスタンスです。したがって、大域ゾーンのデバイスは `/dev` の下にマウントされ、`my-zone` という非大域ゾーンのデバイスは `/my-zone/root/dev` の下にマウントされます。

/dev ファイル階層は、dev ファイルシステム (dev(7FS) のマニュアルページに記載) によって管理されます。

注意 – /devices パス名に依存するサブシステムは、非大域ゾーンで実行できません。/dev パス名を使用するようにサブシステムを更新する必要があります。

注意 – 非大域ゾーンに /dev/zvol 内のデバイスと一致するデバイスリソースがある場合、非大域ゾーン内で名前空間の競合が発生する可能性があります。詳細は、dev(7FS) のマニュアルページを参照してください。

排他使用のデバイス

デバイスを特定のゾーンに割り当てることが必要な場合があります。特権のないユーザーがブロックデバイスにアクセスできるようにすると、これらのデバイスの使用が許可されて、システムパニックやバスリセットなどの不具合が生じる場合があります。この種の割り当てを行う前に、次の点を考慮してください。

- SCSI テープデバイスを特定のゾーンに割り当てる前に、sgen(7D) のマニュアルページを参照してください。
- 物理デバイスを複数のゾーンに配置する場合、ゾーン間に隠れたチャネルが作成される場合があります。大域ゾーンアプリケーションでこの種のデバイスを使用すると、非大域ゾーンによるデータ整合性の損失や、データの破壊が発生する危険があります。

デバイスドライバの管理

modinfo(1M) のマニュアルページに記載されているように、非大域ゾーン内で modinfo コマンドを使用して、読み込まれたカーネルモジュールのリストを検査できます。

カーネル、デバイス、およびプラットフォームの管理に関係した大半の操作は、非大域ゾーンの内部では機能しません。これは、プラットフォームハードウェア構成を変更すると、ゾーンのセキュリティーモデルに違反するためです。これらの操作には、次のことが含まれます。

- ドライバの追加および削除
- カーネルモジュールの明示的な読み込みおよび読み込み解除
- 動的再構成 (DR) 操作の開始
- 物理プラットフォームの状態に影響を与える機能の使用

非大域ゾーンで動作しないか、変更されるユーティリティー

非大域ゾーンで動作しないユーティリティー

次のユーティリティーは、通常は使用できないデバイスに依存しているため、ゾーン内では動作しません。

- `add_drv` (`add_drv(1M)` のマニュアルページを参照)
- `disks` (`disks(1M)` のマニュアルページを参照)
- `prtconf` (`prtconf(1M)` のマニュアルページを参照)
- `prtdiag` (`prtdiag(1M)` のマニュアルページを参照)
- `rem_drv` (`rem_drv(1M)` のマニュアルページを参照)

SPARC: 非大域ゾーンでの使用に合わせて変更されたユーティリティー

`eeprom` ユーティリティーをゾーン内で使用して、設定を表示できます。このユーティリティーを使用して、設定を変更することはできません。詳細については、`eeprom(1M)` および `openprom(7D)` のマニュアルページを参照してください。

セキュリティーを考慮することにより許可されるユーティリティー

`allowed-raw-io` が有効にされている場合、次のユーティリティーをゾーンで使用できます。セキュリティーに関する考慮事項を評価する必要があります。デバイスを追加する場合、制限およびセキュリティーの懸念事項については、373 ページの「非大域ゾーンでのデバイスの使用」、375 ページの「非大域ゾーンでのアプリケーションの実行」、および378 ページの「非大域ゾーン内の特権」を参照してください。

- `cdrecord` (`cderecord(1)` のマニュアルページを参照)。
- `cdrw` (`cdrw(1)` のマニュアルページを参照)。
- `rmformat` (`rmformat(1)` のマニュアルページを参照)。

非大域ゾーンでのアプリケーションの実行

通常は、すべてのアプリケーションを非大域ゾーンで実行できます。ただし、次のタイプのアプリケーションは、この環境に適さない場合があります。

- システム全体に影響を与える特権付きの操作を実行するアプリケーション。大域システムクロックを設定したり、物理メモリーをロックダウンする操作が、これに該当します。

- /dev/kmem などの、非大域ゾーン内に存在しない特定のデバイスに依存するいくつかのアプリケーション。
- 共有 IP ゾーンで、/dev/ip 内のデバイスに依存するアプリケーション。

非大域ゾーンで使用されるリソース制御

ゾーン内でのリソース管理機能の使用に関する追加情報については、パート I「Oracle Solaris のリソース管理」の、この機能について説明した章を参照してください。

リソース管理の章に記載されたリソース制御および属性はすべて、大域ゾーンおよび非大域ゾーンの /etc/project ファイル、NIS マップ、または LDAP ディレクトリサービスで設定できます。指定されたゾーンの設定は、そのゾーンにのみ影響を及ぼします。異なるゾーン内で自動実行中のプロジェクトは、ゾーンごとに別個の制御セットを保持できます。たとえば、大域ゾーン内のプロジェクト A を project.cpu-shares=10 に設定し、非大域ゾーン内の Project A を project.cpu-shares=5 に設定できます。それぞれが該当するゾーン内でのみ機能する、rcapd のインスタンスをシステム上で複数実行できます。

ゾーン内部のプロジェクト、タスク、およびプロセスを制御するため、ゾーン内で使用するリソースの制御および属性は、プールおよびゾーン規模のリソース制御に関する追加要件に従います。

1 つの非大域ゾーンにはリソースプールを 1 つだけ関連付けることができますが、プールは特定のゾーンに排他的に割り当てる必要はありません。1 つのプール内のリソースを複数の非大域ゾーンが共有してもかまいません。ただし、十分な特権を付与されたプロセスを使って、大域ゾーン内のプロセスを任意のプールにバインドすることが可能です。リソースコントローラ poold は、大域ゾーン内だけで動作します。大域ゾーン内には、リソースコントローラが動作するプールが複数存在します。poolstat ユーティリティーを非大域ゾーンで実行すると、そのゾーンに関連付けられているプールの情報だけが表示されます。非大域ゾーンで引数なしで pooladm コマンドを実行すると、そのゾーンに関連付けられているプールの情報だけが表示されます。

ゾーン規模のリソース制御が project ファイルで設定されている場合、そのリソース制御は有効にはなりません。ゾーン規模のリソース制御は、zonecfg ユーティリティーを使って設定されます。

ゾーンがインストールされている システムでの公平配分スケジューラ

このセクションでは、ゾーンで公平配分スケジューラ (FSS) を使用する方法について説明します。

大域ゾーンまたは非大域ゾーン内の FSS 配分分割

ゾーンの FSS CPU 配分は、階層的です。大域ゾーンおよび非大域ゾーンの配分は、ゾーン規模のリソース制御 `zone.cpu-shares` を使って大域管理者が設定します。次に、そのゾーン内のプロジェクトごとにリソース制御 `project.cpu-shares` を定義して、ゾーン規模の制御で設定された配分をさらに分割できます。

`zonecfg` コマンドを使用してゾーンに配分を割り当てる方法については、284 ページの「大域ゾーンの zone.cpu-shares を設定する方法」を参照してください。`project.cpu-shares` の詳細は、78 ページの「使用可能なリソース制御」を参照してください。配分を一時的に設定する方法を示す手順例については、413 ページの「ゾーンがインストールされている Oracle Solaris システムでの公平配分スケジューラの使用」も参照してください。

ゾーン間の配分均衡

`zone.cpu-shares` を使用して、FSS 配分を大域ゾーンと非大域ゾーンに割り当てることができます。FSS がシステムのデフォルトのスケジューラになっている場合で、配分が割り当てられていないときは、各ゾーンにはデフォルトで 1 つの配分が付与されます。システムに 1 つの非大域ゾーンが存在し、`zone.cpu-shares` を使ってこのゾーンに 2 つの配分を付与する場合、これにより非大域ゾーンが大域ゾーンとの関連で受ける CPU の比率が定義されます。2 つのゾーン間の CPU 比率は 2:1 です。

ゾーンがインストールされているシステムでの拡張アカウンティング

拡張アカウンティングサブシステムを大域ゾーンで実行した場合、非大域ゾーンを含むシステム全体の情報が収集および報告されます。大域管理者は、ゾーンごとのリソース消費を決定することもできます。

拡張アカウンティングサブシステムを使用すると、プロセスおよびタスクに基づいたアカウンティングに対応した異なるアカウンティング設定およびファイルを、ゾーン単位で指定することが可能になります。`exacct` レコードに、プロセスの場合はゾーン名 `EXD_PROC_ZONENAME` のタグを付け、タスクの場合はゾーン名 `EXD_TASK`

ZONENAME のタグを付けることができます。アカウンティングレコードは、大域ゾーンのアカウンティングファイルおよびゾーン単位のアカウンティングファイルに書き込まれます。EXD TASK HOSTNAME、EXD PROC HOSTNAME、および EXD HOSTNAME レコードには、大域ゾーンのノード名の代わりに、プロセスまたはタスクが実行されたゾーンの uname -n 値が含まれます。

IPQoS フローアカウンティングについては、『Oracle Solaris 11.1 での IP サービス品質の管理』の第 5 章「フローアカウンティングの使用と統計情報の収集 (タスク)」を参照してください。

非大域ゾーン内の特権

プロセスは、特権の一部に制限されています。特権を制限することで、ほかのゾーンに影響を及ぼす可能性がある操作がゾーンで実行されないようにします。特権セットにより、特権が付与されたユーザーがゾーン内で実行可能な機能が制限されます。指定されたゾーン内で利用可能な特権のリストを表示するには、ppriv ユーティリティーを使用します。

次の表に、Oracle Solaris の特権すべて、およびゾーン内での各特権のステータスを示します。省略可能な特権は、デフォルト特権セットの一部ではありませんが、limitpriv プロパティーを使って指定できます。必須の特権が、生成される特権セットに含まれている必要があります。禁止された特権を、生成される特権セットに含めることはできません。

表 25–1 ゾーン内の特権のステータス

特権	ステータス	注意事項
cpc_cpu	任意	特定の cpc(3CPC) カウンタへのアクセス
dtrace_proc	任意	fasttrap および pid プロバイダ。plockstat(1M)
dtrace_user	任意	profile および syscall プロバイダ
graphics_access	任意	ioctl(2) による agpgart_io(7I) へのアクセス
graphics_map	任意	mmap(2) による agpgart_io(7I) へのアクセス
net_rawaccess	共有 IP ゾーンではオプション。 排他的 IP ゾーンではデフォルト。	raw PF_INET/PF_INET6 パケットアクセス
proc_clock_highres	任意	高解像度タイマーの使用

表 25–1 ゾーン内の特権のステータス (続き)

特権	ステータス	注意事項
proc_priocntl	任意	スケジューリングの制御。priocntl(1)
sys_ipc_config	任意	IPC メッセージキューのバッファーサイズの引き上げ
sys_time	任意	システム時間の操作。xntp(1M)
dtrace_kernel	禁止	現在、未サポート
proc_zone	禁止	現在、未サポート
sys_config	禁止	現在、未サポート
sys_devices	禁止	現在、未サポート
sys_dl_config	禁止	現在、未サポート
sys_linkdir	禁止	現在、未サポート
sys_net_config	禁止	現在、未サポート
sys_res_config	禁止	現在、未サポート
sys_smb	禁止	現在、未サポート
sys_suser_compat	禁止	現在、未サポート
proc_exec	必須、デフォルト	init(1M) の起動に使用
proc_fork	必須、デフォルト	init(1M) の起動に使用
sys_mount	必須、デフォルト	必須ファイルシステムのマウントに必要
sys_flow_config	必須、排他的 IP ゾーンではデフォルト 共有 IP ゾーンでは禁止	フローの構成に必要
sys_ip_config	必須、排他的 IP ゾーンではデフォルト 共有 IP ゾーンでは禁止	ゾーンのブートおよび排他的 IP ゾーンの IP ネットワークの初期化に必要
sys_iptun_config	必須、排他的 IP ゾーンではデフォルト 共有 IP ゾーンでは禁止	IP トンネルリンクの構成
contract_event	デフォルト	契約ファイルシステムで使用
contract_identity	デフォルト	プロセス規約テンプレートのサービス FMRI 値の設定
contract_observer	デフォルト	UID とは無関係な契約観察

表 25–1 ゾーン内の特権のステータス (続き)

特権	ステータス	注意事項
file_chown	デフォルト	ファイル所有権の変更
file_chown_self	デフォルト	所有するファイルの所有者/グループの変更
file_dac_execute	デフォルト	モード/ACL に依存しない実行アクセス
file_dac_read	デフォルト	モード/ACL に依存しない読み取りアクセス
file_dac_search	デフォルト	モード/ACL に依存しない検索アクセス
file_dac_write	デフォルト	モード/ACL に依存しない書き込みアクセス
file_link_any	デフォルト	所有者に依存しないリンクアクセス
file_owner	デフォルト	所有者に依存しないその他のアクセス
file_setid	デフォルト	setid、setgid、 setuid ファイルのアクセス権の変更
ipc_dac_read	デフォルト	モードに依存しない IPC 読み取りアクセス
ipc_dac_owner	デフォルト	モードに依存しない IPC 書き込みアクセス
ipc_owner	デフォルト	モードに依存しないその他の IPC アクセス
net_icmpaccess	デフォルト	ICMP パケットアクセス: ping(1M)
net_privaddr	デフォルト	特権ポートへのバインド
proc_audit	デフォルト	監査レコードの生成
proc_chroot	デフォルト	root ディレクトリの変更
proc_info	デフォルト	プロセスの検査
proc_lock_memory	デフォルト	メモリーのロック。shmctl(2) および mlock(3C) この特権がシステム管理者によって非大域ゾーンに割り当てられている場合、ゾーンがすべてのメモリーをロックするのを防ぐために zone.max-locked-memory リソース制御の設定も検討してください。
proc_owner	デフォルト	所有者に依存しないプロセス制御

表 25–1 ゾーン内の特権のステータス (続き)

特権	ステータス	注意事項
proc_session	デフォルト	セッションに依存しないプロセス制御
proc_setid	デフォルト	ユーザー/グループ ID の任意設定
proc_taskid	デフォルト	呼び出し元へのタスク ID の割り当て
sys_acct	デフォルト	アカウンティングの管理
sys_admin	デフォルト	単純なシステム管理タスク
sys_audit	デフォルト	監査の管理
sys_nfs	デフォルト	NFS クライアントのサポート
sys_ppp_config	排他的 IP ゾーンではデフォルト 共有 IP ゾーンでは禁止	PPP (sppp) インタフェースの作成および削除、PPP トンネル (sppptun) の構成
sys_resource	デフォルト	リソース制限の操作
sys_share	デフォルト	ファイルシステムの共有に必要な sharefs システムコールを許可します。ゾーン内での NFS 共有を防止するために、ゾーン内で特権を禁止にすることができます。

次の表に、Oracle Solaris Trusted Extensions の特権すべて、および各特権のゾーン内のステータスを示します。省略可能な特権は、デフォルト特権セットの一部ではありませんが、limitpriv プロパティーを使って指定できます。

注 – Oracle Trusted Solaris 特権が解釈されるのは、システムが Oracle Trusted Extensions を使って構成されている場合だけです。

表 25–2 ゾーン内での Oracle Solaris Trusted Extensions の特権のステータス

Oracle Solaris Trusted Extensions の特権	ステータス	注意事項
file_downgrade_sl	任意	ファイルまたはディレクトリの機密ラベルを、既存の機密ラベルを優先する機密ラベルに設定します。
file_upgrade_sl	任意	ファイルまたはディレクトリの機密ラベルを、既存の機密ラベルよりも優先される機密ラベルに設定します。
sys_trans_label	任意	機密ラベルの制御下にないラベルの変換

表 25–2 ゾーン内での Oracle Solaris Trusted Extensions の特権のステータス (続き)

Oracle Solaris Trusted Extensions の特権	ステータス	注意事項
win_colormap	任意	カラーマップ制限のオーバーライド
win_config	任意	X サーバーにより常時保持されるリソースの構成または破棄
win_dac_read	任意	クライアントのユーザー ID が所有していないウィンドウリソースからの読み取り
win_dac_write	任意	クライアントのユーザー ID が所有していないウィンドウリソースへの書き込みまたは作成
win_devices	任意	入力デバイスでの操作の実行
win_dga	任意	ダイレクトグラフィックスアクセス X プロトコル拡張機能の使用。フレームバッファー特権が必要
win_downgrade_sl	任意	ウィンドウリソースの機密ラベルを、既存ラベルの制御下にある新規ラベルに変更
win_fontpath	任意	フォントパスの追加
win_mac_read	任意	クライアントのラベルを制御するラベルを使用した、ウィンドウリソースからの読み取り
win_mac_write	任意	クライアントのラベルと同等ではないラベルを使用した、ウィンドウリソースへの書き込み
win_selection	任意	確認者の介入なしでの要求データの移動
win_upgrade_sl	任意	ウィンドウリソースの機密ラベルを、既存ラベルの制御下にない新規ラベルに変更
net_bindmlp	デフォルト	マルチレベルポート (MLP) へのバインドの許可
net_mac_aware	デフォルト	NFS を使用した読み取りの許可

非大域ゾーン構成内の特権を変更する方法については、269 ページの「ゾーンを構成、検証、および確定する」を参照してください。

特権セットを検査する方法については、395 ページの「ppriv ユーティリティーの使用」を参照してください。特権の詳細は、ppriv(1) のマニュアルページおよび『Solaris のシステム管理 (セキュリティサービス)』を参照してください。

ゾーン内での IP セキュリティーアーキテクチャーの使用

IP データグラム保護を提供する IPsec (Internet Protocol Security Architecture) については、『Oracle Solaris 11.1 でのネットワークのセキュリティー保護』の第 8 章「IP セキュリティーアーキテクチャー (リファレンス)」に記載されています。IKE (Internet Key Exchange) プロトコルを使用して、認証および暗号化に必要な鍵材料が自動的に管理されます。

詳細については、ipsecconf(1M) および ipseckey(1M) のマニュアルページを参照してください。

共有 IP ゾーン内の IP セキュリティーアーキテクチャー

IPsec は、大域ゾーン内で使用できます。ただし、非大域ゾーン内では、IPsec で IKE を使用することはできません。そのため、大域ゾーンで Internet Key Exchange (IKE) プロトコルを使用して、非大域ゾーンの IPsec キーおよび IPSec ポリシーを管理する必要があります。構成中の非大域ゾーンに対応するソースアドレスを使用します。

排他的 IP ゾーン内の IP セキュリティーアーキテクチャー

IPsec は、排他的 IP ゾーン内で使用できます。

ゾーン内での Oracle Solaris 監査の使用

監査レコードには、システムへのログイン、ファイルへの書き込みなどのイベントが記載されます。Oracle Solaris 監査は、ゾーンを実行しているシステムで次の 2 つの監査モデルを提供します。

- すべてのゾーンを大域ゾーンから一元的に監査します。このモデルは、すべてのゾーンが大域ゾーンによって管理される場合 (たとえば、ゾーンを使用してサービスの隔離を実現する場合) に使用されます。
- 各ゾーンを大域ゾーンから独立して監査します。このモデルは、各ゾーンが別々に管理される場合 (たとえば、ゾーンごとにサーバーの統合を実現する場合) に使用されます。

Oracle Solaris 監査については、『Oracle Solaris 11.1 の管理: セキュリティーサービス』の第 26 章「監査 (概要)」に記載されています。監査に関するゾーンの考慮事項については、『Oracle Solaris 11.1 の管理: セキュリティーサービス』の「Oracle Solaris ゾーンを含むシステムでの監査」および『Oracle Solaris 11.1 の管理: セキュリティーサービス』の「ゾーンでの監査サービスの構成 (タスク)」を参照してください。詳細については、`auditconfig(1M)`、`auditreduce(1M)`、`usermod(1M)`、および `user_attr(4)` のマニュアルページを参照してください。

注 – 一時的に有効にされるが、リポジトリには設定されない監査ポリシーを使用することもできます。

詳細は、『Oracle Solaris 11.1 の管理: セキュリティーサービス』の「監査ポリシーを変更する方法」の次の例を参照してください。

ゾーン内のコアファイル

`coreadm` コマンドを使用して、異常終了するプロセスにより生成されるコアファイルの名前と場所を指定できます。`%z` 変数を指定することで、プロセスが実行されたゾーンの *zonename* を含むコアファイルパスを生成できます。パス名は、ゾーンのルートディレクトリに対する相対パスです。

詳細は、`coreadm(1M)` および `core(4)` のマニュアルページを参照してください。

非大域ゾーン内での DTrace の実行

`dtrace_proc` 特権と `dtrace_user` 特権だけを必要とする DTrace プログラムは、非大域ゾーンで実行できます。非大域ゾーンで使用できる特権のセットにこれらの特権を追加するには、`zonecfg limitpriv` プロパティーを使用します。手順については、402 ページの「DTrace を使用する方法」を参照してください。

`dtrace_proc` によってサポートされるプロバイダは、`fasttrap` と `pid` です。`dtrace_user` によってサポートされるプロバイダは、`profile` と `syscall` です。DTrace のプロバイダおよびアクションの有効範囲は、ゾーンに制限されます。

詳細は、378 ページの「非大域ゾーン内の特権」も参照してください。

ゾーンがインストールされている Oracle Solaris システムのバックアップについて

非大域ゾーンを個別にバックアップしたり、大域ゾーンからシステム全体をバックアップしたりできます。

ループバックファイルシステムのディレクトリのバックアップ

非大域ゾーン内からループバックファイルシステム (`lofs`) をバックアップしないでください。

非大域ゾーン内から読み取り/書き込みループバックファイルシステムをバックアップおよび復元する場合、これらのファイルシステムは大域ゾーンおよび読み取り/書き込みマウントされているほかのすべてのゾーンからも書き込むことができます。複数のコピーを防止するために、これらのファイルシステムは大域ゾーンからのみバックアップおよび復元します。

大域ゾーンからのシステムのバックアップ

次のような場合は、大域ゾーンからバックアップを実行することをお勧めします。

- 非大域ゾーンの構成をアプリケーションデータと一緒にバックアップする場合。
- 障害から回復することがもっとも重要である場合。使用しているゾーンのルートファイルシステムおよびそれらの構成データ、使用している大域ゾーン内のデータなど、システム上のすべてまたはほぼすべての情報を復元する必要がある場合。このような場合は、大域ゾーンでバックアップを実行してください。
- 市販のネットワークバックアップソフトウェアを使用する場合。

注 – ネットワークバックアップソフトウェアを使用するときには、可能であれば、継承された `lofs` ファイルシステムはすべてスキップするように構成することをお勧めします。バックアップは、ゾーンとそのアプリケーションがバックアップ対象のデータを休止させた状態のときに、行うことをお勧めします。

システム上の非大域ゾーンを個別にバックアップ

次のような場合は、非大域ゾーン内でバックアップを実行することをお勧めします。

- 非大域ゾーンの管理者が、重大度の低い障害から回復する機能、またはゾーンに固有のアプリケーションデータまたはユーザーデータを復元する機能を必要とする場合。
- tar や cpio など、ファイル単位でバックアップを行うプログラムを使用する場合。tar(1) および cpio(1) のマニュアルページを参照してください。
- ゾーン内で動作する特定のアプリケーションまたはサービスのバックアップソフトウェアを使用する場合。ディレクトリパスやインストール済みソフトウェアなどのアプリケーション環境が大域ゾーンと非大域ゾーンとの間で異なっている場合には、バックアップソフトウェアを大域ゾーンから実行するのが困難な場合があります。

  アプリケーションが非大域ゾーンごとのバックアップスケジュールに基づいてスナップショットを実行し、そのバックアップデータを大域ゾーンからエクスポートした書き込み可能なディレクトリに格納できる場合には、大域ゾーンの管理者は大域ゾーンからバックアップする処理の中でそれら個々のバックアップデータを個別に選択できます。

Oracle Solaris ZFS バックアップの作成

ZFS send コマンドを実行すると、ZFS スナップショットのストリーム表現が作成され、標準出力に書き込まれます。デフォルトでは、完全なストリームが生成されます。この出力は、ファイルまたは別のシステムにリダイレクトできます。ZFS receive コマンドを実行すると、ストリームに内容が指定されているスナップショットが作成され、標準入力に渡されます。ストリーム全体を受信する場合、新しいファイルシステムも作成されます。これらのコマンドを使えば、ZFS スナップショットデータを送信したり、ZFS スナップショットデータやファイルシステムを受信したりできます。

ZFS send および receive コマンド以外に、tar や cpio コマンドなどのアーカイブユーティリティーを使用して、ZFS ファイルを保存することもできます。これらのユーティリティーは、ZFS ファイル属性とアクセス制御リスト (ACL) を保存して復元します。マニュアルページで tar および cpio コマンドの適切なオプションを確認してください。

詳細と例については、『Oracle Solaris 11.1 の管理: ZFS ファイルシステム』の第 6 章「Oracle Solaris ZFS のスナップショットとクローンの操作」を参照してください。

非大域ゾーン内でバックアップするデータの決定

非大域ゾーン内のデータは、すべてバックアップできます。ゾーンの構成が頻繁に変更されない場合には、アプリケーションデータだけをバックアップすることもできます。

アプリケーションデータのみのバックアップ

アプリケーションデータがファイルシステムの特定の場所に格納されている場合には、このデータだけを定期的にバックアップすることもできます。ゾーンのルートファイルシステムはそれほど頻繁には変更されないため、頻繁にバックアップする必要がない場合もあります。

アプリケーションファイルがどこに格納されているかを確認する必要があります。アプリケーションファイルは次のような場所に格納されている可能性があります。

- ユーザーのホームディレクトリ
- `/etc` (構成データファイルの場合)
- `/var`

アプリケーション管理者がデータの格納場所を認識している場合には、ゾーンごとに書き込み可能ディレクトリを割り当てるように、システムを作成することもできます。バックアップがゾーンごとに格納されるので、大域管理者または適切な承認を付与されたユーザーは、その場所をシステム上のバックアップ対象の 1 つとして選択することができます。

一般的なデータベースバックアップ操作

データベースアプリケーションデータがデータベース固有のディレクトリに存在していない場合には、次の規則が適用されます。

- 最初にデータベースが安定した状態にあることを確認します。

  データベースが休止している必要があります。内部バッファーのデータがディスクにフラッシュされるためです。大域ゾーンからバックアップを開始する前に、非大域ゾーン内のデータベースが安定した状態になっていることを確認してください。
- ゾーンごとに、ファイルシステム機能を使用してデータのスナップショットを作成してから、そのスナップショットを大域ゾーンから直接バックアップします。

  このようにすると、バックアップのための時間が短縮されるだけでなく、すべてのゾーンのクライアントやモジュールをバックアップする必要がなくなります。

テープによるバックアップ

非大域ゾーンだけが使用するファイルシステムについては、そのゾーンにとって都合のよい時間帯に、アプリケーションがわずかに休止している時間を利用して、スナップショットを作成することもできます。それらのスナップショットは、アプリケーションがサービスに戻ったあとに大域ゾーンからバックアップしてテープに格納できます。

この方法により、次の利点が得られます。

- 必要なテープデバイスが少なくてすみます。
- 非大域ゾーンの間で調整する必要がなくなります。
- デバイスを直接ゾーンに割り当てる必要がないため、セキュリティー機能が向上します。
- 大域ゾーンでシステム管理を続けることができるので、通常はこの方法をお勧めします。

非大域ゾーンの復元について

大域管理者または適切な承認を付与されたユーザーは、大域ゾーンから実行したバックアップを復元するときには、関係するゾーンを再インストールしてから、そのゾーンのファイルを復元できます。この処理を行うときには、ゾーンが次の状態である必要があります。

- 復元するゾーンの構成が、バックアップしたときと同じ構成であること。
- バックアップしてからゾーンを復元するまでの間に、大域ゾーンをアップデートしていないこと。

これらの前提を満たしていない場合は、一部のファイルが復元によって上書きされ、手作業でマージしなければならないことがあります。

注 – 大域ゾーンのすべてのファイルシステムが失われた場合には、大域ゾーンのすべてのファイルを復元すると、非大域ゾーンも復元されます。ただし、非大域ゾーンの各ルートファイルシステムがバックアップに含まれている必要があります。

ゾーンがインストールされているシステムで使用するコマンド

表 25–3 に示すコマンドにより、ゾーン機能に対する主要な管理インタフェースが提供されます。

表 25–3 ゾーンの管理および監視に使用するコマンド

コマンド	説明
zlogin(1)	非大域ゾーンにログインします
zonename(1)	現在のゾーンの名前を出力します
zonestat(1)	ゾーンのリソース使用率の監視に使用します
zoneadm(1M)	システムのゾーンを管理します
zonecfg(1M)	ゾーン構成の設定に使用されます
getzoneid(3C)	ゾーン ID と名前のマッピングに使用されます
zones(5)	ゾーン機能の説明を提供します
zcons(7D)	ゾーンコンソールのデバイスドライバ

zoneadmd デーモンは、ゾーンの仮想プラットフォームを管理する基本プロセスです。zoneadmd デーモンのマニュアルページは、zoneadmd(1M) です。このデーモンは、プログラミングインタフェースの構成要素ではありません。

次の表に示すコマンドは、リソース上限デーモンとともに使用されます。

表 25–4 rcapd とともに使用されるコマンド

コマンド	説明
rcapstat(1)	上限が定義されたプロジェクトのリソース使用効率を監視します。
rcapadm(1M)	リソース上限デーモンを構成します。構成済みのリソース上限デーモンの現在のステータスを表示します。リソース上限制御を有効または無効にします。
rcapd(1M)	リソース上限デーモン。

次の表で示すコマンドは、ゾーンがインストールされている Oracle Solaris システムで使用できるように変更されています。これらのコマンドには、ゾーンに固有のオプションが用意されています。指定するオプションによって異なる情報が表示されます。コマンドは、マニュアルページのセクション別に記載されています。

表 25–5 ゾーンがインストールされている Oracle Solaris システムで使用するために変更されたコマンド

コマンド	説明
ipcrm(1)	-z *zone* オプションが追加されました。このオプションは、大域ゾーン内でコマンドを実行する場合にのみ有用です。
ipcs(1)	-z *zone* オプションが追加されました。このオプションは、大域ゾーン内でコマンドを実行する場合にのみ有用です。
pgrep(1)	-z *zoneidlist* オプションが追加されました。このオプションは、大域ゾーン内でコマンドを実行する場合にのみ有用です。
ppriv(1)	-l オプションとともに使用する式 zone が追加されました。これを使用すると、現在のゾーン内で使用可能なすべての特権が一覧表示されます。また、zone に -v オプションを指定して、冗長出力を取得できます。
priocntl(1)	*idlist* と -i *idtype* でゾーン ID を使用することで、プロセスを指定できます。priocntl -i *zoneid* コマンドを使用すると、実行中のプロセスを非大域ゾーン内の別のスケジューリングクラスに移動できます。
proc(1)	ptree だけに -z *zone* オプションが追加されました。このオプションは、大域ゾーン内でコマンドを実行する場合にのみ有用です。
ps(1)	-o オプションが使用する、認識される format 名リストに、*zonename* と *zoneid* が追加されました。 指定したゾーン内のプロセスだけを一覧表示するため、-z *zonelist* が追加されました。ゾーンの指定には、ゾーン名またはゾーン ID を使用できます。このオプションは、大域ゾーン内でコマンドを実行する場合にのみ有用です。 プロセスに関連するゾーンの名前を出力するため、-Z が追加されました。名前は、追加された列ヘッダー ZONE の下に出力されます。
renice(1)	有効な引数をリスト表示するため、-i オプションとともに使用する *zoneid* が追加されました。
sar(1)	プール機能が有効な非大域ゾーン内で実行する際、-b、-c、-g、-m、-p、-u、-w、および -y オプションを指定すると、ゾーンのバインド先プールのプロセッサセット内のプロセッサの値だけが表示されます。
auditconfig(1M)	zonename トークンが追加されました。
auditreduce(1M)	-z *zone-name* オプションが追加されました。ゾーンの監査ログを取得する機能が追加されました。
coreadm(1M)	プロセスが実行されたゾーンを識別するための変数 %z が追加されました。

表 25–5 ゾーンがインストールされている Oracle Solaris システムで使用するために変更されたコマンド (続き)

コマンド	説明
df(1M)	すべての可視ゾーン内のマウントを表示する -Z オプションが追加されました。このオプションは、非大域ゾーンでは効果がありません。
dladm(1M)	デフォルトのコマンド出力にゾーン列を追加する -Z オプションが show サブコマンドに追加されました。ゾーン列は、現在、リソースが割り当てられているゾーンを示します。
dlstat(1M)	デフォルトのコマンド出力にゾーン列を追加する -Z オプションが show サブコマンドに追加されました。ゾーン列は、現在、リソースが割り当てられているゾーンを示します。
fsstat(1M)	ゾーン別のファイルシステムアクティビティーについて報告する -z オプションが追加されました。複数の -z オプションを使用して、選択したゾーンのアクティビティーを監視できます。このオプションは、fstypes ではなく mountpoints のみを監視する場合に使用しても効果はありません。 指定した fstypes のファイルシステムアクティビティーのすべてのゾーンにわたる集計を報告する -A オプションが追加されました。これは、-z または -Z オプションがどちらも使用されなかった場合のデフォルトの動作です。-A オプションは、fstypes ではなく mountpoints のみを監視する場合に使用しても効果はありません。 -A オプションを -z または -Z オプションとともに使用すると、指定した fstypes のすべてのゾーンにわたる集計が別個の行に表示されます。 システム上のすべてのゾーンのファイルシステムアクティビティーを報告する -Z オプションが追加されました。このオプションは、-z オプションとともに使用しても効果はありません。このオプションは、fstypes ではなく mountpoints のみを監視する場合に使用しても効果はありません。
iostat(1M)	プール機能が有効な非大域ゾーン内で実行すると、ゾーンのバインド先プールのプロセッサセット内のプロセッサ情報だけが提供されます。
ipadm(1M)	インターネットプロトコルネットワークインタフェースおよび TCP/IP 調整可能パラメータの構成from-gz タイプは、非大域ゾーンでのみ表示され、非大域排他的 IP ゾーンに構成されている allowed-address プロパティーに基づいて大域ゾーンから構成されたアドレスであることを示します。zone アドレスプロパティーは、allowed-address によって参照されるすべてのアドレスが配置されるゾーンを指定します。ゾーンは、共有 IP ゾーンとして構成する必要があります。

表 25–5 ゾーンがインストールされている Oracle Solaris システムで使用するために変更されたコマンド (続き)

コマンド	説明
kstat(1M)	大域ゾーンで実行すると、すべてのゾーンの kstat が表示されます。非大域ゾーンで実行すると、一致する *zoneid* を持つ kstat だけが表示されます。
mpstat(1M)	プール機能が有効な非大域ゾーン内で実行すると、ゾーンのバインド先プールのプロセッサセット内のプロセッサの行だけが表示されます。
ndd(1M)	大域ゾーン内で使用された場合、すべてのゾーンの情報を表示します。排他的 IP ゾーンの TCP/IP モジュールに対して ndd を実行すると、そのゾーンの情報だけが表示されます。
netstat(1M)	現在のゾーンのみの情報を表示します。
nfsstat(1M)	現在のゾーンのみの統計情報を表示します。
poolbind(1M)	*zoneid* リストが追加されました。ゾーンとリソースプールの併用については、141 ページの「ゾーンで使用されるリソースプール」も参照してください。
prstat(1M)	-z *zoneidlist* オプションが追加されました。また、-Z オプションも追加されました。 プール機能が有効な非大域ゾーンで実行した場合、ゾーンのバインド先プールのプロセッサセット内のプロセッサだけを対象にして、プロセスが使用した最新の CPU 時間の比率が表示されます。 -a、-t、-T、-J、および -Z の各オプションを指定すると、出力にはサイズ列の代わりにスワップ列が表示されます。報告されるスワップは、ゾーンのプロセスと tmpfs マウントで消費されるスワップの合計量です。この値により、各ゾーンで予約されているスワップを監視しやすくなり、適切な zone.max-swap 設定を選択することができます。
psrinfo(1M)	非大域ゾーン内で実行した場合、ゾーンで表示可能なプロセッサの情報だけが表示されます。
traceroute(1M)	使用方法が変更されました。非大域ゾーン内から指定した場合、-F オプションをしても効果はありません。理由は、「断片化しない」というビットが常に設定されているためです。
vmstat(1M)	プール機能が有効な非大域ゾーン内で実行すると、ゾーンのバインド先プールのプロセッサセット内のプロセッサ統計情報だけが報告されます。-p オプション指定時の出力と、page、faults、および cpu 報告フィールドに適用されます。
priocntl(2)	P_ZONEID *id* 引数が追加されました。

表 25–5 ゾーンがインストールされている Oracle Solaris システムで使用するために変更されたコマンド (続き)

コマンド	説明
processor_info(2)	呼び出し側が非大域ゾーン内にあり、プール機能が有効であるが、プロセッサがゾーンのバインド先プールのプロセッサセット内にない場合、エラーが返されます。
p_online(2)	呼び出し側が非大域ゾーン内にあり、プール機能が有効であるが、プロセッサがゾーンのバインド先プールのプロセッサセット内にない場合、エラーが返されます。
pset_bind(2)	*idtype* として P_ZONEID が追加されました。P_MYID 仕様の選択肢にゾーンが追加されました。EINVAL エラー説明内の有効な *idtype* リストに P_ZONEIDが追加されました。
pset_info(2)	呼び出し側が非大域ゾーン内にあり、プール機能が有効であるが、プロセッサがゾーンのバインド先プールのプロセッサセット内にない場合、エラーが返されます。
pset_list(2)	呼び出し側が非大域ゾーン内にあり、プール機能が有効であるが、プロセッサがゾーンのバインド先プールのプロセッサセット内にない場合、エラーが返されます。
pset_setattr(2)	呼び出し側が非大域ゾーン内にあり、プール機能が有効であるが、プロセッサがゾーンのバインド先プールのプロセッサセット内にない場合、エラーが返されます。
sysinfo(2)	PRIV_SYS_CONFIG が PRIV_SYS_ADMIN に変更されました。
umount(2)	*file* が参照しているファイルが絶対パスでない場合、ENOENT が返されます。
getloadavg(3C)	呼び出し側が非大域ゾーン内にあり、プール機能が有効な場合、PS_MYID の psetid を使用して呼び出した場合と動作は同じになります。
getpriority(3C)	ゾーン ID が、指定可能なターゲットプロセスに追加されました。ゾーン ID が EINVAL のエラー説明に追加されました。
priv_str_to_set(3C)	呼び出し側のゾーン内部で使用可能なすべての特権セットで、「zone」文字列が追加されました。
pset_getloadavg(3C)	呼び出し側が非大域ゾーン内にあり、プール機能が有効であるが、プロセッサがゾーンのバインド先プールのプロセッサセット内にない場合、エラーが返されます。
sysconf(3C)	呼び出し側が非大域ゾーン内にあり、プール機能が有効な場合、sysconf(_SC_NPROCESSORS_CONF) および sysconf(_SC_NPROCESSORS_ONLN) は、ゾーンのバインド先プールのプロセッサセット内の合計プロセッサ数およびオンラインのプロセッサ数を返します。

表 25–5 ゾーンがインストールされている Oracle Solaris システムで使用するために変更されたコマンド (続き)

コマンド	説明
ucred_get(3C)	ucred_getzoneid() 関数が追加されました。この関数は、プロセスのゾーン ID を返します。ただし、ゾーン ID を取得できなかった場合は -1 を返します。
core(4)	n_type: NT_ZONENAME が追加されました。このエントリには、プロセスが実行されていたゾーンの名前を示す文字列が含まれます。
pkginfo(4)	ゾーンのサポート内でオプションパラメータおよび環境変数が提供されるようになりました。
proc(4)	ゾーン内で実行中のプロセスに関する情報を取得する機能が追加されました。
audit_syslog(5)	in<*zone name*> フィールドが追加されました。このフィールドは、zonename 監査ポリシーが設定されている場合に使用されます。
privileges(5)	プロセスによる追跡またはほかのゾーン内のプロセスへのシグナル送信を可能にする PRIV_PROC_ZONE が追加されました。zones(5) を参照してください。
if_tcp(7P)	ゾーンの ioctl() 呼び出しが追加されました。
cmn_err(9F)	ゾーンパラメータが追加されました。
ddi_cred(9F)	cr の指し示すユーザー資格からゾーン ID を返す、crgetzoneid() が追加されました。

◆ ◆ ◆ 第 26 章

Oracle Solaris ゾーンの管理 (タスク)

この章では、一般的な管理タスクおよびその使用例を示します。

- 395 ページの「ppriv ユーティリティーの使用」
- 397 ページの「非大域ゾーンでの zonestat ユーティリティーの使用」
- 402 ページの「非大域ゾーン内での DTrace の使用」
- 404 ページの「稼働中の非大域ゾーン内でファイルシステムをマウントする」
- 407 ページの「大域ゾーン内の特定のファイルシステムへのアクセス権を非大域ゾーンに追加する」
- 409 ページの「ゾーンがインストールされている Oracle Solaris システムでの IP ネットワークマルチパスの使用」
- 410 ページの「排他的 IP 非大域ゾーンでのデータリンクの管理」
- 413 ページの「ゾーンがインストールされている Oracle Solaris システムでの公平配分スケジューラの使用」
- 414 ページの「ゾーン管理での権利プロファイルの使用」
- 414 ページの「ゾーンがインストールされている Oracle Solaris システムのバックアップ」
- 416 ページの「非大域ゾーンの再作成」

一般的なゾーン管理のトピックについては、第 25 章「Oracle Solaris ゾーンの管理 (概要)」を参照してください。

ppriv ユーティリティーの使用

ppriv ユーティリティーを使用してゾーンの特権を表示します。

▼ 大域ゾーンでの Oracle Solaris の特権を一覧表示する方法

ppriv ユーティリティーを -l オプションとともに使用して、システムで使用可能な特権を一覧表示します。

- プロンプトで **ppriv -l zone** と入力し、ゾーンで使用可能な特権セットを表示します。

```
global# ppriv -l zone
```

次のような内容が表示されます。

```
contract_event
contract_observer
cpc_cpu
.
.
.
```

▼ 非大域ゾーンの特権セットの表示方法

ppriv ユーティリティーを -l オプションおよび式 zone とともに使用して、ゾーンの特権を表示します。

1. 非大域ゾーンにログインします。この例では、*my-zone* という名前のゾーンを使用します。
2. プロンプトで **ppriv -l zone** と入力し、ゾーンで使用可能な特権セットを表示します。

```
my-zone# ppriv -l zone
```

次のような内容が表示されます。

```
contract_event
contract_identity
contract_observer
file_chown

.
.
.
```

▼ 非大域ゾーンの特権セットを冗長出力で表示する方法

ppriv ユーティリティーを -l オプション、式 zone、および -v オプションとともに使用して、ゾーンの特権を一覧表示します。

1. 非大域ゾーンにログインします。この例では、*my-zone* という名前のゾーンを使用します。
2. プロンプトで **ppriv -l -v zone** と入力して、ゾーン内で使用可能な特権セットおよび各特権の説明を出力します。

```
my-zone# ppriv -lv zone
```

次のような内容が表示されます。

```
contract_event
        Allows a process to request critical events without limitation.
        Allows a process to request reliable delivery of all events on
        any event queue.
contract_identity
        Allows a process to set the service FMRI value of a process
        contract template.
contract_observer
        Allows a process to observe contract events generated by
        contracts created and owned by users other than the process's
        effective user ID.
        Allows a process to open contract event endpoints belonging to
        contracts created and owned by users other than the process's
        effective user ID.
file_chown
        Allows a process to change a file's owner user ID.
        Allows a process to change a file's group ID to one other than
        the process' effective group ID or one of the process'
        supplemental group IDs.
.
.
.
```

非大域ゾーンでの zonestat ユーティリティーの使用

zonestat ユーティリティーは、現在実行中のゾーンの CPU、メモリー、ネットワーク、およびリソース制御の使用効率について報告します。使用例はこのあとにあります。

詳細は、zonestat(1) を参照してください。

zonestat ネットワークコンポーネントは、PHYS、AGGR、Etherstub、および SIMNET データリンクの仮想ネットワーク (VNIC) リソースの使用率をゾーンごとに

表示します。その他のデータリンク (ブリッジ、トンネルなど) の情報は、ネットワークユーティリティー (dladm(1M) および dlstat(1M) のマニュアルページに記載) を使用して取得できます。

ゾーンの統計情報を表示するために、非大域ゾーン内ですべての zonestat オプションおよびリソースタイプを呼び出すことができます。

```
root@zoneA:~# zonestat -z global -r physical-memory 2
```

注 – zonestat を非大域ゾーンで使用すると、ほかのすべてのゾーン (大域ゾーンを含む) を合計したリソース使用状況が、大域ゾーンによって使用されたものとして報告されます。非大域ゾーンの zonestat ユーザーは、システムを共有しているほかのゾーンを認識しません。

▼ zonestat ユーティリティーを使用して、CPU およびメモリーの使用効率のサマリーを表示する方法

1. **root** になるか、同等の役割になります。
2. **CPU** およびメモリー使用効率のサマリーを **5** 秒ごとに表示します。

```
# zonestat -z global -r physical-memory 5
Collecting data for first interval...
Interval: 1, Duration: 0:00:05
PHYSICAL-MEMORY              SYSTEM MEMORY
mem_default                          2046M
                                ZONE  USED %USED   CAP  %CAP
                             [total] 1020M 49.8%     -     -
                            [system]  782M 38.2%     -     -
                              global  185M 9.06%     -     -

Interval: 2, Duration: 0:00:10
PHYSICAL-MEMORY              SYSTEM MEMORY
mem_default                          2046M
                                ZONE  USED %USED   CAP  %CAP
                             [total] 1020M 49.8%     -     -
                            [system]  782M 38.2%     -     -
                              global  185M 9.06%     -     -
...
```

▼ zonestat ユーティリティーを使用して、デフォルトの pset について報告する方法

1 **root** になるか、同等の役割になります。

2 デフォルトの **pset** について **1** 秒ごとに **1** 分間報告します。

```
# zonestat -r default-pset 1 1m
Collecting data for first interval...
Interval: 1, Duration: 0:00:01
PROCESSOR_SET                   TYPE  ONLINE/CPUS     MIN/MAX
pset_default            default-pset          2/2         1/-
                                ZONE  USED   PCT   CAP  %CAP   SHRS  %SHR %SHRU
                             [total]  0.02 1.10%     -     -      -     -     -
                            [system]  0.00 0.19%     -     -      -     -     -
                              global  0.01 0.77%     -     -      -     -     -
                               zone1  0.00 0.07%     -     -      -     -     -
                               zone2  0.00 0.06%     -     -      -     -     -

...
Interval: 60, Duration: 0:01:00
PROCESSOR_SET                   TYPE  ONLINE/CPUS     MIN/MAX
pset_default            default-pset          2/2         1/-
                                ZONE  USED   PCT   CAP  %CAP   SHRS  %SHR %SHRU
                             [total]  0.06 3.26%     -     -      -     -     -
                            [system]  0.00 0.18%     -     -      -     -     -
                              global  0.05 2.94%     -     -      -     -     -
                               zone1  0.00 0.06%     -     -      -     -     -
                               zone2  0.00 0.06%     -     -      -     -     -
```

▼ zonestat を使用した合計および最大使用効率の報告

1 **root** になるか、同等の役割になります。

2 表示なしで **10** 秒ごとに **3** 分間監視したあと、合計使用効率および最大使用効率の報告を生成します。

```
# zonestat -q -R total,high 10s 3m 3m
Report: Total Usage
    Start: Fri Aug 26 07:32:22 PDT 2011
      End: Fri Aug 26 07:35:22 PDT 2011
    Intervals: 18, Duration: 0:03:00
SUMMARY                  Cpus/Online: 2/2   PhysMem: 2046M  VirtMem: 3069M
                    ---CPU----  --PhysMem-- --VirtMem-- --PhysNet--
                ZONE  USED %PART  USED %USED  USED %USED PBYTE %PUSE
             [total]  0.01 0.62% 1020M 49.8% 1305M 42.5%    14 0.00%
            [system]  0.00 0.23%  782M 38.2% 1061M 34.5%     -     -
              global  0.00 0.38%  185M 9.06%  208M 6.77%     0 0.00%
               test2  0.00 0.00% 52.4M 2.56% 36.6M 1.19%     0 0.00%
```

```
Report: High Usage
    Start: Fri Aug 26 07:32:22 PDT 2011
      End: Fri Aug 26 07:35:22 PDT 2011
    Intervals: 18, Duration: 0:03:00
SUMMARY                   Cpus/Online: 2/2   PhysMem: 2046M  VirtMem: 3069M
                      ---CPU----  --PhysMem-- --VirtMem-- --PhysNet--
                ZONE  USED %PART  USED %USED  USED %USED PBYTE %PUSE
             [total]  0.01 0.82% 1020M 49.8% 1305M 42.5%  2063 0.00%
            [system]  0.00 0.26%  782M 38.2% 1061M 34.5%     -     -
              global  0.01 0.55%  185M 9.06%  207M 6.77%     0 0.00%
               test2  0.00 0.00% 52.4M 2.56% 36.6M 1.19%     0 0.00%
```

▼ 排他的 IP ゾーンでのネットワーク帯域幅の使用効率を取得する方法

zonestat コマンドに -r オプションと network リソースタイプを付けて実行すると、各ネットワークデバイスのゾーンごとの使用効率が表示されます。

各ゾーンによって使用された VNIC 形式のデータリンク帯域幅の量を表示するには、この手順を使用します。たとえば、e1000g0 の下に表示されている zoneB は、このゾーンが VNIC 形式の e1000g0 のリソースを消費したことを示しています。-x オプションを追加すると、特定の VNIC を表示できます。

1. ルート管理者になります。

2. **-r** オプションを付けた **zonestat** コマンドに **network** リソースタイプを指定して実行し、使用効率を一度に表示します。

```
    # zonestat -r network 1 1
Collecting data for first interval...
Interval: 1, Duration: 0:00:01

    NETWORK-DEVICE                  SPEED        STATE        TYPE
    aggr1                         2000mbps           up        AGGR
                    ZONE TOBYTE  MAXBW %MAXBW PRBYTE %PRBYTE POBYTE %POBYTE
                  global  1196K      -      -   710K   0.28%   438K   0.18%

    e1000g0                       1000mbps           up        PHYS
                    ZONE TOBYTE  MAXBW %MAXBW PRBYTE %PRBYTE POBYTE %POBYTE
                 [total]  7672K      -      -  6112K   4.89%  1756K   1.40%
                  global  5344K  100m*  42.6%  2414K   1.93%  1616K   1.40%
                   zoneB   992K   100m  15.8%  1336K   0.76%   140K   0.13%
                   zoneA  1336K    50m  10.6%   950K   1.07%      0   0.00%

    e1000g1                       1000mbps           up        PHYS
                    ZONE TOBYTE  MAXBW %MAXBW PRBYTE %PRBYTE POBYTE %POBYTE
                  global   126M      -      -    63M   6.30%    63M   6.30%

    etherstub1                         n/a          n/a   ETHERSTUB
                    ZONE TOBYTE  MAXBW %MAXBW PRBYTE %PRBYTE POBYTE %POBYTE
                 [total]  3920K      -      -      0       -      0       -
                  global  1960K  100M*  1.96%      0       -      0       -
```

```
                zoneA  1960K    50M  3.92%      0       -      0       -
```

参考 非大域ゾーンでのコマンドの例

非大域ゾーンで使用されるコマンド

```
root@zoneA:~# zonestat -r network -x 1 1
```

すべてのゾーンのゾーン別の **fstype** 統計情報の報告

ゾーン別のファイルシステムアクティビティーについて報告するには、-z オプションを使用します。複数の -z オプションを使用して、選択したゾーンのアクティビティーを監視できます。

指定した fstypes のファイルシステムアクティビティーのすべてのゾーンにわたる集計を報告するには、-A オプションを使用します。これは、-z または -Z オプションがどちらも使用されなかった場合のデフォルトの動作です。

-A オプションを -z または -Z オプションとともに使用すると、指定した fstypes のすべてのゾーンにわたる集計が別個の行に表示されます。

システム上のすべてのゾーンのファイルシステムアクティビティーを報告するには、-Z オプションを使用します。このオプションは、-z オプションとともに使用しても効果はありません。このオプションは、fstypes ではなく mountpoints のみを監視する場合に使用しても効果はありません。

▼ **-z** オプションを使用して特定のゾーンのアクティビティーを監視する方法

- *s10* ゾーンと *s10u9* ゾーンのアクティビティーを監視するには、複数の **-z** オプションを使用します。

```
$ fsstat -z s10 -z s10u9 zfs tmpfs
 new  name   name  attr  attr lookup rddir  read read  write write
 file remov  chng   get   set    ops   ops   ops bytes   ops bytes
   93    82     6  163K   110   507K   148 69.7K 67.9M 4.62K 13.7M zfs:s10
  248   237   158  188K   101   612K   283 70.6K 68.6M 4.71K 15.2M zfs:s10u9
12.0K 1.90K 10.1K 35.4K    12  60.3K     4 25.7K 29.8M 36.6K 31.0M tmpfs:s10
12.0K 1.90K 10.1K 35.6K    14  60.2K     2 28.4K 32.1M 36.5K 30.9M tmpfs:S10u9
```

▼ すべてのゾーンのゾーン別の fstype 統計情報を表示する方法

- システム上で実行されているゾーンごとに **tmpfs** および **zfs** ファイルシステムタイプのゾーン別の統計情報を取得し、**tmpfs** および **zfs** ファイルシステムタイプのシステム全体の集計も表示します。

```
$  fsstat -A -Z zfs tmpfs
 new  name   name  attr  attr lookup rddir  read read  write write
 file remov  chng   get   set    ops   ops   ops bytes   ops bytes
 360K 1.79K 20.2K 4.20M 1.02M  25.0M  145K 5.42M 2.00G 1.07M 8.10G zfs
 359K 1.48K 20.1K 4.04M 1.02M  24.5M  144K 5.31M 1.88G 1.06M 8.08G zfs:global
   93    82     6 74.8K   107   250K   144 54.8K 60.5M 4.61K 13.7M zfs:s10
  248   237   158 90.2K   101   336K   283 53.0K 58.3M 4.71K 15.2M zfs:s10u9
60.0K 41.9K 17.7K  410K   515   216K   426 1022K 1.02G  343K  330M tmpfs
49.4K 38.1K 11.0K  366K   489   172K   420  968K  979M  283K  273M tmpfs:global
5.28K 1.90K 3.36K 21.9K    12  21.7K     4 25.7K 29.8M 29.9K 28.3M tmpfs:s10
5.25K 1.90K 3.34K 22.1K    14  21.6K     2 28.4K 32.1M 29.8K 28.2M tmpfs:s10u9
```

この出力では、システム上の非大域ゾーンは `S10` と `S10u9` です。

非大域ゾーン内での DTrace の使用

384 ページの「非大域ゾーン内での DTrace の実行」で説明されている DTrace 機能を使用するには、次の手順を実行します。

▼ DTrace を使用する方法

1. **zonecfg limitpriv** プロパティーを使用して、**dtrace_proc** 特権と **dtrace_user** 特権を追加します。

```
global# zonecfg -z my-zone
zonecfg:my-zone> set limitpriv="default,dtrace_proc,dtrace_user"
zonecfg:my-zone> exit
```

注 – 必要に応じて、どちらか一方の特権を追加することも、両方の特権を追加することもできます。

---

2. ゾーンをブートします。

```
global# zoneadm -z my-zone boot
```

3. ゾーンにログインします。

```
global# zlogin my-zone
```

4. DTrace プログラムを実行します。

```
my-zone# dtrace -l
```

非大域ゾーンの SMF サービスのステータスの確認

非大域ゾーンの SMF サービスのステータスを確認するには、`zlogin` コマンドを使用します。

▼ コマンド行から SMF サービスのステータスを確認する方法

1. **root** になるか、同等の役割になります。
2. コマンド行から次のように入力して、無効になっているサービスも含むすべてのサービスを表示します。

```
global# zlogin my-zone svcs -a
```

参照　詳細は、第 21 章「非大域ゾーンへのログイン (タスク)」および svcs(1) を参照してください。

▼ ゾーン内から SMF サービスのステータスを確認する方法

1. **root** になるか、同等の役割になります。
2. ゾーンにログインします。

```
global# zlogin my-zone
```

3. **svcs** コマンドに **-a** オプションを付けて実行して、無効になっているサービスも含むすべてのサービスを表示します。

```
my-zone# svcs -a
```

参照　詳細は、第 21 章「非大域ゾーンへのログイン (タスク)」および svcs(1) を参照してください。

稼働中の非大域ゾーン内でファイルシステムをマウントする

稼働中の非大域ゾーン内でファイルシステムをマウントできます。具体的には、次の作業について説明しています。

- 大域管理者または大域ゾーン内で適切な承認を付与されたユーザーとして、raw デバイスおよびブロックデバイスを非大域ゾーンにインポートできます。デバイスのインポート後に、ゾーン管理者はディスクにアクセスできます。その後、ゾーン管理者はディスク上に新しいファイルシステムを作成して、次のいずれかの操作を実行できます。
  - ファイルシステムを手動でマウントします
  - ファイルシステムがゾーンのブート時にマウントされるように、`/etc/vfstab` 内に配置します
- 大域管理者または適切な承認を付与されたユーザーとして、大域ゾーンから非大域ゾーンにファイルシステムをマウントすることもできます。

  大域ゾーンから非大域ゾーンにファイルシステムをマウントする前に、非大域ゾーンは準備完了状態であるか、ブートされている必要があります。それ以外の場合、次回ゾーンをブートまたは準備完了状態にしようとすると失敗します。また、大域ゾーンから非大域ゾーンにマウントされたファイルシステムは、ゾーンが停止されるときにアンマウントされます。

▼ LOFS を使用してファイルシステムをマウントする方法

LOFS マウントを使用すると、大域ゾーンと非大域ゾーンの間でファイルシステムを共有できます。この手順では、`zonecfg` コマンドを使用して、大域ゾーンの `/export/datafiles` ファイルシステムの LOFS マウントを `my-zone` 構成に追加します。この例では、マウントオプションをカスタマイズしません。

この手順を実行するには、大域管理者またはゾーンセキュリティー権利プロファイルを持つ大域ゾーンのユーザーである必要があります。

1. **root** になるか、同等の役割になります。
2. **zonecfg** コマンドを使用します。

   ```
   global# zonecfg -z my-zone
   ```

3. ファイルシステムを構成に追加します。

   ```
   zonecfg:my-zone> add fs
   ```

4 ファイルシステムのマウントポイントに **my-zone** の **/datafiles** を設定します。

```
zonecfg:my-zone:fs> set dir=/datafiles
```

5 大域ゾーンの **/export/datafiles** が、**my-zone** で **/datafiles** としてマウントされるように指定します。

```
zonecfg:my-zone:fs> set special=/export/datafiles
```

6 ファイルシステムのタイプを設定します。

```
zonecfg:my-zone:fs> set type=lofs
```

7 指定を終了します。

```
zonecfg:my-zone:fs> end
```

8 構成を検証および確定します。

```
zonecfg:my-zone> verify
zonecfg:my-zone> commit
```

参考 一時的なマウント

非大域ゾーンをリブートせずに、大域ゾーンから LOFS ファイルシステムのマウントを追加できます。

```
global# mount -F lofs /export/datafiles /export/my-zone/root/datafiles
```

ゾーンをブートするたびにこのマウントが行われるようにするには、`zonecfg` コマンドを使用してゾーンの構成を変更する必要があります。

▼ ZFS データセットを非大域ゾーンに委任する方法

ZFS データセットを非大域ゾーンに委任するには、この手順を使用します。

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を付与されたユーザーである必要があります。

1 **root** になるか、同等の役割になります。

2 大域ゾーンから、**poolA** という既存の **ZFS** プールに **fs2** という新しい **ZFS** ファイルシステムを作成します。

```
global# zfs create poolA/fs2
```

3 **(オプション) poolA/fs2** ファイルシステムの **mountpoint** プロパティーに **/fs-del/fs2** を設定します。

```
global# zfs set mountpoint=/fs-del/fs2 poolA/fs2
```

mountpoint を設定する必要はありません。mountpoint プロパティーを指定しない場合、データセットはデフォルトでゾーン内の /alias にマウントされます。mountpoint および canmount プロパティーをデフォルト以外の値にすると、zfs(1M) のマニュアルページで説明されているように、この動作が変更されます。

4 このファイルシステムの **mountpoint** プロパティーのソースが、現在 **local** であることを確認します。

```
global# zfs get mountpoint poolA/fs2
NAME       PROPERTY    VALUE        SOURCE
poolA/fs2  mountpoint  /fs-del/fs2  local
```

5 **poolA/fs2** ファイルシステムを委任するか、別名のデータセットを指定します。

- **poolA/fs2** ファイルシステムをゾーンに委任します。

```
# zonecfg -z my-zone
zonecfg:my-zone> add dataset
zonecfg:my-zone:dataset> set name=poolA/fs2
zonecfg:my-zone:dataset> end
```

- 別名のデータセットを指定します。

```
# zonecfg -z my-zone
zonecfg:my-zone> add dataset
zonecfg:my-zone:dataset> set name=poolA/fs2
zonecfg:my-zone:dataset> set alias=delegated
zonecfg:my-zone:dataset> end
```

6 ゾーンをリブートして、すべての **poolA** ファイルシステムの **zoned** プロパティーを表示します。

```
global# zfs get -r zoned poolA
NAME      PROPERTY  VALUE   SOURCE
poolA     zoned     off     default
poolA/fs2 zoned     on      default
```

poolA/fs2 の zoned プロパティーが on に設定されます。この ZFS ファイルシステムは、非大域ゾーンに委任されてゾーンにマウントされ、ゾーン管理者の制御下にあります。ZFS は、zoned プロパティーを使用して、データセットがある時点で非大域ゾーンに委任されたことを示します。

大域ゾーン内の特定のファイルシステムへのアクセス権を非大域ゾーンに追加する

▼ 非大域ゾーンで CD または DVD メディアにアクセスする権限を追加する方法

この手順を使用して、非大域ゾーンで CD または DVD メディアに読み取り専用のアクセスを行う権限を追加できます。メディアをマウントするときには、大域ゾーンでボリューム管理ファイルシステムが使用されます。アクセス権を追加したら、CD または DVD を使用して製品を非大域ゾーンにインストールできます。この手順では、`jes_05q4_dvd` という DVD を使用します。

1. **root** になるか、同等の役割になります。
2. 大域ゾーンでボリューム管理ファイルシステムが稼働しているかどうかを調べます。

```
global# svcs rmvolmgr
STATE          STIME    FMRI
online         Sep_29   svc:/system/filesystem/volfs:default
```

3. (オプション) 大域ゾーンでボリューム管理ファイルシステムが稼働していない場合は、起動します。

```
global# svcadm rmvolmgr enable
```

4. メディアを挿入します。
5. ドライブにメディアが入っているかどうかを確認します。

```
global# volcheck
```

6. **DVD** が自動マウントされているかどうかをテストします。

```
global# ls /media
```

次のような情報が表示されます。

```
cdrom   cdrom1   jes_05q4_dvd
```

7. **ro,nodevices** オプション (読み取り専用、デバイスなし) を指定して、非大域ゾーンでファイルシステムをループバックマウントします。

```
global# zonecfg -z my-zone
zonecfg:my-zone> add fs
zonecfg:my-zone:fs> set dir=/cdrom
zonecfg:my-zone:fs> set special=/cdrom
zonecfg:my-zone:fs> set type=lofs
```

```
zonecfg:my-zone:fs> add options [ro,nodevices]
zonecfg:my-zone:fs> end
zonecfg:my-zone> commit
zonecfg:my-zone> exit
```

8 非大域ゾーンをリブートします。

```
global# zoneadm -z my-zone reboot
```

9 **zoneadm list** コマンドに **-v** オプションを指定して、ステータスを確認します。

```
global# zoneadm list -v
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris    shared
 1  my-zone  running      /zones/my-zone                 solaris    excl
```

10 非大域ゾーンにログインします。

```
global#  my-zone
```

11 **DVD-ROM** がマウントされているかを確認します。

```
my-zone# ls /cdrom
```

次のような内容が表示されます。

```
cdrom   cdrom1   jes_05q4_dvd
```

12 製品のインストールガイドで説明されているとおりに、製品をインストールします。

13 非大域ゾーンから抜けます。

```
my-zone# exit
```

ヒント – /cdrom ファイルシステムを非大域ゾーンに残すこともできます。マウントするときには、常に CD-ROM ドライブの現在の内容が反映されます。つまり、ドライブが空の場合は、ディレクトリは空になります。

---

14 (オプション) 非大域ゾーンから **/cdrom** ファイルシステムを削除する場合は、次の手順を使用します。

```
global# zonecfg -z my-zone
zonecfg:my-zone> remove fs dir=/cdrom
zonecfg:my-zone> commit
zonecfg:my-zone> exit
```

ゾーンがインストールされている Oracle Solaris システムでの IP ネットワークマルチパスの使用

▼ 排他的 IP 非大域ゾーンで IP ネットワークマルチパスを使用する方法

排他的 IP ゾーンの IP ネットワークマルチパス (IPMP) は、大域ゾーンと同じ方法で構成します。IPMP を使用するには、排他的 IP ゾーンに少なくとも 2 つの `zonecfg add net` リソースが必要です。IPMP は、これらのデータリンク上のゾーン内から構成されます。

IP マルチパスグループ (IPMP グループ) に 1 つ以上の物理インタフェースを構成できます。IPMP を構成すると、IPMP グループのインタフェースに障害が発生していないかどうかをシステムが自動的に監視します。グループのインタフェースに障害が発生した場合や、保守のためにインタフェースが削除された場合、IPMP は自動的に、そのインタフェースの IP アドレスを移行して処理を継続します。フェイルオーバーされたアドレスは、障害が発生したインタフェースの IPMP グループ内の機能中のインタフェースが受け取ります。IPMP のフェイルオーバーコンポーネントは、接続を保持し、既存の接続の切断を防止します。さらに、IPMP は、ネットワークトラフィックを自動的に IPMP グループ内のインタフェースのセットに分散することによって、ネットワークパフォーマンス全体を向上させます。この処理は負荷分散と呼ばれます。

1. **root** になるか、同等の役割になります。
2. 『**Oracle Solaris 11.1 ネットワークパフォーマンスの管理**』の「**IPMP グループの構成**」の説明に従って **IPMP** グループを構成します。

▼ IP ネットワークマルチパス機能を共有 IP 非大域ゾーンに拡張する方法

大域ゾーン内で IPMP を構成し、IPMP 機能を非大域ゾーンに拡張する場合に、ここで説明する手順を使用します。

ゾーンの構成時に、各アドレスつまり論理インタフェースを非大域ゾーンと関連付ける必要があります。手順については、237 ページの「`zonecfg` コマンドの使用」および 270 ページの「ゾーンの構成方法」を参照してください。

この手順を実行すると、次のことが達成されます。

- カード `bge0` および `hme0` がグループ内でともに構成されます。

- アドレス 192.168.0.1 が非大域ゾーン *my-zone* と関連付けられます。
- `bge0` カードが物理インタフェースとして設定されます。このため、IP アドレスが `bge0` および `hme0` カードを含むグループ内に収容されます。

稼働中のゾーンで、`ipadm` コマンドを使用して関連付けを行うことができます。詳細は、370 ページの「共有 IP ネットワークインタフェース」および `ipadm(1M)` のマニュアルページを参照してください。

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を付与されたユーザーである必要があります。

1. **root** になるか、同等の役割になります。
2. 大域ゾーンで、『**Oracle Solaris 11.1 ネットワークパフォーマンスの管理**』の「**IPMP グループの構成**」の説明に従って **IPMP** グループを構成します。
3. **zonecfg** コマンドを使用してゾーンを構成します。**net** リソースを構成する際、アドレス **192.168.0.1** および物理インタフェース **bge0** をゾーン *my-zone* に追加します。

```
zonecfg:my-zone> add net
zonecfg:my-zone:net> set address=192.168.0.1
zonecfg:my-zone:net> set physical=bge0
zonecfg:my-zone:net> end
```

非大域ゾーン *my-zone* 内に `bge0` だけが表示されます。

参考 bge0 が連続して失敗した場合

`bge0` が連続して失敗し、`bge0` データアドレスが大域ゾーン内の `hme0` に引き継がれる場合、*my-zone* アドレスも移行します。

アドレス `192.168.0.1` が `hme0` に移動する場合、非大域ゾーン *my-zone* 内で `hme0` だけが表示されます。このカードは、アドレス `192.168.0.1` に関連付けられ、`bge0` は表示されなくなります。

排他的 IP 非大域ゾーンでのデータリンクの管理

データリンクを管理するには、大域ゾーンから `dladm` コマンドを使用します。

▼ dladm show-linkprop の使用方法

`dladm` コマンドを `show-linkprop` サブコマンドとともに使用して、実行中の排他的 IP ゾーンに対するデータリンクの割り当てを表示できます。

データリンクを管理するには、大域管理者または適切な承認を付与された大域ゾーンのユーザーである必要があります。

1. **root** になるか、同等の役割になります。
2. システムのデータリンクの割り当てを表示します。

```
global# dladm show-linkprop
```

例 26–1 dladm コマンドを show-linkprop サブコマンドとともに使用する

1. ゾーン 49bge には bge0 が割り当てられていますが、最初の画面ではこのゾーンはブートしていません。

```
global# dladm show-linkprop
LINK     PROPERTY        PERM   VALUE        DEFAULT        POSSIBLE
bge0     zone            rw     --           --             --
vsw0     speed           r-     1000         1000           --
vsw0     autopush        rw     --           --             --
vsw0     zone            rw     --           --             --
vsw0     duplex          r-     full         full           half,full
vsw0     state           r-     up           up             up,down
vsw0     adv_autoneg_cap --     --           0              1,0
vsw0     mtu             rw     1500         1500           1500
vsw0     flowctrl        --     --           no             no,tx,rx,bi,pfc,
                                                            auto
...
```

2. ゾーン 49bge をブートします。

```
global# zoneadm -z 49bge boot
```

3. コマンド dladm show-linkprop を再度実行します。すると、bge0 リンクが 49bge に割り当てられていることに注目してください。

```
global# dladm show-linkprop
LINK     PROPERTY        PERM   VALUE        DEFAULT        POSSIBLE
bge0     zone            rw     49bge        --             --
vsw0     speed           r-     1000         1000           --
vsw0     autopush        rw     --           --             --
vsw0     zone            rw     --           --             --
vsw0     duplex          r-     full         full           half,full
vsw0     state           r-     up           up             up,down
vsw0     adv_autoneg_cap --     --           0              1,0
vsw0     mtu             rw     1500         1500           1500
vsw0     flowctrl        --     --           no             no,tx,rx,bi,pfc,
                                                            auto
...
```

例 26–2 バニティーネーミングを使用している場合に、データリンク名および物理的な場所を表示する方法

デバイスの物理的な場所は、LOCATION フィールドに表示されます。デバイスのデータリンク名および物理的な場所の情報を表示するには、-L オプションを使用します。

```
global# dladm show-phys -L
LINK         DEVICE       LOCATION
net0         e1000g0      MB
```

```
net1          e1000g1       MB
net2          e1000g2       MB
net3          e1000g3       MB
net4          ibp0          MB/RISER0/PCIE0/PORT1
net5          ibp1          MB/RISER0/PCIE0/PORT2
net6          eoib2         MB/RISER0/PCIE0/PORT1/cloud-nm2gw-2/1A-ETH-2
net7          eoib4         MB/RISER0/PCIE0/PORT2/cloud-nm2gw-2/1A-ETH-2
```

▼ dladm を使用して、一時的なデータリンクを割り当てる方法

dladm コマンドを set-linkprop サブコマンドとともに使用して、実行中の排他的 IP ゾーンに対してデータリンクを一時的に割り当てることができます。持続的な割り当ては zonecfg コマンドを使用して行う必要があります。

データリンクを管理するには、大域管理者または適切な承認を付与された大域ゾーンのユーザーである必要があります。

1. **root** になるか、同等の役割になります。
2. **dladm set-linkprop** を **-t** とともに使用して、**zoneA** という実行中のゾーンに **bge0** を追加します。

```
global# dladm set-linkprop -t -p zone bge0
LINK     PROPERTY        PERM   VALUE        DEFAULT        POSSIBLE
bge0     zone            rw     zoneA        --             --
```

ヒント – -p オプションを使用すると、マシンで構文解析できる安定した形式の表示が生成されます。

▼ dladm reset-linkprop の使用方法

dladm コマンドを reset-linkprop サブコマンドとともに使用して、bge0 リンクの値を未設定にリセットできます。

1. **root** になるか、同等の役割になります。
2. **dladm reset-linkprop** を **-t** オプションとともに使用して、ゾーンへの **bge0** デバイスの割り当てを解除します。

```
global# dladm reset-linkprop -t -p zone bge0
LINK     PROPERTY        PERM   VALUE        DEFAULT        POSSIBLE
bge0     zone            rw     zoneA        --             --
```

ヒント – -p オプションを使用すると、マシンで構文解析できる安定した形式の表示が生成されます。

注意事項 実行中のゾーンでそのデバイスが使用中の場合、再割り当ては失敗し、エラーメッセージが表示されます。423 ページの「排他的 IP ゾーンがデバイスを使用しているために dladm reset-linkprop が失敗する」を参照してください。

ゾーンがインストールされている Oracle Solaris システムでの公平配分スケジューラの使用

prctl コマンドで指定された制限には持続性がありません。システムがリブートされると、制限は無効になります。ゾーンの配分を持続的に設定する方法については、270 ページの「ゾーンの構成方法」および 284 ページの「大域ゾーンの zone.cpu-shares を設定する方法」を参照してください。

▼ prctl コマンドを使用して大域ゾーンの FSS 配分を設定する方法

大域ゾーンには、デフォルトで 1 つの配分が付与されます。ここで説明する手順を使って、デフォルトの割り当てを変更できます。システムをリブートするたびに prctl コマンドで割り当てた配分を設定し直す必要があります。

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を付与されたユーザーである必要があります。

1. **root** になるか、同等の役割になります。
2. **prctl** ユーティリティーを使用して、**2** つの配分を大域ゾーンに割り当てます。

```
# prctl -n zone.cpu-shares -v 2 -r -i zone global
```

3. (オプション) 大域ゾーンに割り当てられた配分の数を確認するには、次のように入力します。

```
# prctl -n zone.cpu-shares -i zone global
```

参照 prctl ユーティリティーの詳細は、prctl(1) のマニュアルページを参照してください。

▼ ゾーンの zone.cpu-shares 値を動的に変更する方法

この手順は、大域ゾーンまたは非大域ゾーンで使用できます。

1 root になるか、同等の役割になります。

2 `prctl` コマンドを使って `cpu-shares` の新しい値を指定します。

```
# prctl  -n zone.cpu-shares -r -v value -i zone zonename
```

idtype は *zonename*、*zoneid* のいずれかです。*value* は新しい値です。

ゾーン管理での権利プロファイルの使用

このセクションでは、非大域ゾーンで権利プロファイルを使用することに関連したタスクについて説明します。

▼ Zone Management プロファイルを割り当てる方法

ユーザーは、Zone Management プロファイルを使用することで、システムの非大域ゾーンをすべて管理できます。

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を付与されたユーザーである必要があります。

1 スーパーユーザーになるか、同等の承認を保持するようにします。

役割の詳細は、『Oracle Solaris 11.1 の管理: セキュリティーサービス』のパート III「役割、権利プロファイル、特権」を参照してください。

2 Zone Management 権利プロファイルを含む役割を作成し、その役割をユーザーに割り当てます。

ゾーンがインストールされている Oracle Solaris システムのバックアップ

次の手順を使用して、ゾーン内のファイルをバックアップできます。ゾーンの構成ファイルもバックアップしてください。

▼ ZFSsend を使用してバックアップを実行する方法

1. **root** になるか、同等の役割になります。
2. ゾーンの **zonepath** を取得します。

```
global# zonecfg -z my-zone info zonepath
zonepath: /zones/my-zone
```

3. **zfs list** コマンドを使用して、**zonepath** データセットを取得します。

```
global# zfs list -H -o name /zones/my-zone
rpool/zones/my-zone
```

4. **ZFS** スナップショットを使用して、ゾーンのアーカイブを作成します。

```
global# zfs  snapshot -r rpool/zones/my-zone@snap
global# zfs snapshot -r rpool/zones/my-zone@snap
global# zfs zfs send -rc rpool/zones/my-zone@snap > /path/to/save/archive
global# zfs destroy -r rpool/zones/my-zone@snap
```

次のような情報が表示されます。

```
-rwxr-xr-x   1 root     root     99680256 Aug 10 16:13 backup/my-zone.cpio
```

▼ ゾーン構成のコピーを出力する方法

非大域ゾーン構成のバックアップファイルを作成することをお勧めします。必要に応じて、このバックアップを使用してゾーンをあとで再作成できます。ゾーンにはじめてログインして `sysidtool` からの質問に回答したあとに、ゾーンの構成のコピーを作成します。この手順では、処理をわかりやすく説明するために、`my-zone` という名前のゾーンと `my-zone.config` という名前のバックアップファイルを使用します。

1. **root** になるか、同等の役割になります。
2. ゾーン **my-zone** の構成を **my-zone.config** というファイルに出力します。

```
global# zonecfg -z my-zone export > my-zone.config
```

非大域ゾーンの再作成

▼ 非大域ゾーンを個別に再作成する方法

必要に応じて、非大域ゾーン構成のバックアップファイルを使用して非大域ゾーンを再作成できます。この手順では、ゾーンを再作成する処理をわかりやすく説明するために、`my-zone` という名前のゾーンと `my-zone.config` という名前のバックアップファイルを使用します。

1. **root** になるか、同等の役割になります。
2. ゾーン **my-zone** を再作成するために、**my-zone.config** を **zonecfg** コマンドファイルとして使用することを指定します。

   ```
   global# zonecfg -z my-zone -f my-zone.config
   ```

3. ゾーンをインストールします。

   ```
   global# zoneadm -z my-zone install -a /path/to/archive  options
   ```

4. ゾーン固有のファイル (アプリケーションデータなど) を復元する場合は、新しく作成したゾーンのルートファイルシステムに対してバックアップのファイルを手動で復元します。手作業でのマージが必要な場合があります。

◆ ◆ ◆ 第 27 章

27 不変ゾーンの構成と管理

不変ゾーンは、solaris 非大域ゾーンに読み取り専用ファイルシステムのプロファイルを提供します。

読み取り専用ゾーンの概要

読み取り専用ゾーンルートを持つゾーンは、不変ゾーンと呼ばれます。solaris 不変ゾーンは、非大域ゾーンに読み取り専用の root ファイルシステムを実装することにより、ゾーンの構成を維持します。このゾーンでは、実行時環境に追加の制限を加えることにより、ゾーンの安全な実行時の境界が拡張されます。特定の保守操作として実行される場合を除き、システムバイナリまたはシステム構成への変更はブロックされます。

必須書き込みアクセス制御 (MWAC) カーネルポリシーを使用すると、zonecfg file-mac-profile プロパティーを介して、ファイルシステム書き込み権限が適用されます。大域ゾーンは MWAC ポリシーの対象ではないため、大域ゾーンはインストール、イメージの更新、および保守のために非大域ゾーンのファイルシステムに書き込むことができます。

ゾーンが準備完了状態になると、MWAC ポリシーがダウンロードされます。このポリシーはゾーンのブート時に有効になります。インストール後のアセンブリと構成を行うには、一時的な書き込み可能ルートファイルシステムのブート処理を使用します。ゾーンの MWAC 構成への変更は、ゾーンのリブートによってのみ有効になります。

ゾーンの構成、インストール、およびブートの一般的な情報については、第 17 章「非大域ゾーンの計画と構成 (タスク)」および第 19 章「非大域ゾーンのインストール、ブート、停止処理、停止、アンインストール、およびクローニング (タスク)」を参照してください。

読み取り専用ゾーンの構成

zonecfg の file-mac profile プロパティー

デフォルトでは、zonecfg の file-mac-profile プロパティーは非大域ゾーンに設定されません。ゾーンは、書き込み可能なルートデータセットを持つように構成されます。

solaris 読み取り専用ゾーンでは、読み取り専用ゾーンルートを構成するために、file-mac-profile プロパティーが使用されます。読み取り専用ルートは、ゾーン内からの実行時環境へのアクセスを制限します。

zonecfg ユーティリティーを使用して、file-mac-profile を次のいずれかの値に設定できます。none 以外のすべてのプロファイルでは、/var/pkg ディレクトリとその内容がゾーンの内部から読み取り専用になります。

none	既存のゾーン境界のほかに追加の保護がない標準の読み取り書き込み非大域ゾーン。値を none に設定するのは、file-mac profile プロパティーを設定しないのと同等になります。
strict	読み取り専用ファイルシステム。例外はありません。 ■ IPS パッケージはインストールできません。 ■ 持続的に有効な SMF サービスは固定されます。 ■ SMF マニフェストはデフォルトの場所から追加できません。 ■ 構成ファイルのロギングおよび監査は固定されます。データはリモートからのみロギングできます。
fixed-configuration	システム構成コンポーネントが含まれているディレクトリを除く、/var/*ディレクトリへの更新を許可します。 ■ IPS パッケージ (新しいパッケージを含む) はインストールできません。 ■ 持続的に有効な SMF サービスは固定されます。 ■ SMF マニフェストはデフォルトの場所から追加できません。 ■ 構成ファイルのロギングおよび監査はローカルに行うことができます。syslog および監査の構成は固定されます。
flexible-configuration	/etc/*ディレクトリ内のファイルの変更、ルートのホームディレクトリの変更、および /var/*ディレクトリへの更新を許可します。この構成は、Oracle Solaris 10 の

ネイティブな疎ルートゾーン (『System Administration Guide: Oracle Solaris Containers-Resource Management and Oracle Solaris Zones』を参照) に近い機能を提供します。これは、このガイドの Oracle Solaris 10 バージョンです。

- IPS パッケージ (新しいパッケージを含む) はインストールできません。
- 持続的に有効な SMF サービスは固定されます。
- SMF マニフェストはデフォルトの場所から追加できません。
- 構成ファイルのロギングおよび監査はローカルに行うことができます。`syslog` および監査の構成は変更できます。

zonecfg add dataset リソースポリシー

`add dataset` リソースを使用してゾーンに追加されたデータセットは、MWAC ポリシーの対象にはなりません。追加のデータセットを委任されたゾーンは、それらのデータセットを完全に制御できます。プラットフォームのデータセットは表示されますが、ゾーンが読み取り/書き込みでブートされないかぎり、そのデータとプロパティーは読み取り専用です。

zonecfg add fs リソースポリシー

`add fs` リソースを使用してゾーンに追加されたファイルシステムは、MWAC ポリシーの対象にはなりません。ファイルシステムは読み取り専用でマウントできます。

読み取り専用ゾーンの管理

ディスク上の構成は大域ゾーンからのみ管理できます。実行中のゾーン内では、ゾーンが書き込み可能でブートされた場合を除き、管理は実行時状態の設定に制限されます。SMF コマンド (`svcadm(1M)` および `svccfg(1M)` のマニュアルページに記載) を使用して行われた構成変更は、ディスク上の SMF データベースではなく、一時的な使用中の SMF データベースにのみ適用できます。ゾーンの MWAC 構成への変更は、ゾーンのリブート時に有効になります。

初期インストールまたはアップデートのあと、self-assembly-complete マイルストーンに達するまで、ゾーンは一時的な読み取り/書き込みでブートされます。その後、ゾーンは読み取り専用モードでリブートされます。

zoneadm list -p の表示

解析可能な出力に、R/W 列および file-mac-profile 列が表示されます。

```
global# zoneadm list -p
0:global:running:/:UUID:solaris:shared:-:none
    5:testzone2:running:/export/zones/testzone2:UUID \
        :solaris:shared:R:fixed-configuration
    12:testzone3:running:/export/zones/testzone3:UUID \
        :solaris:shared:R:fixed-configuration
    13:testzone1:running:/export/zones/testzone1:UUID \
        :solaris:excl:W:fixed-configuration
    -:testzone:installed:/export/zones/testzone:UUID \
        :solaris:excl:-:fixed-configuration
```

R および W オプションは、次のように定義されます。

- R は、読み取り専用でブートされる file-mac-profile を持つゾーンを示します。
- W は、読み取り/書き込みでブートされる file-mac-profile を持つゾーンを示します。
- - は、ゾーンが実行されていないか、file-mac-profile がないことを示します。

書き込み可能なルートファイルシステムを持つ読み取り専用ゾーンをブートするオプション

zoneadm boot サブコマンドを使用すると、書き込み可能なルートファイルシステムまたは一時的に書き込み可能なルートファイルシステムのいずれかを持つ読み取り専用ゾーンを手動でブートできる 2 つのオプションが大域ゾーン管理者に提供されます。ゾーンは次にリブートが行われるまで書き込み可能モードとなります。

-w　書き込み可能なルートファイルシステムを持つゾーンを手動でブートします。

-W　一時的に書き込み可能なルートファイルシステムを持つゾーンを手動でブートします。システムは、self-assembly-complete マイルストーンに達すると自動的にリブートされます。

リブートすると、ゾーンがふたたび MWAC ポリシーの制御下に置かれます。このオプションは、ゾーンが MWAC ポリシー none を持っている場合に許可されます。

-W および -w オプションは、非 ROZR ゾーンでは無視されます。

◆ ◆ ◆ 第 28 章

Oracle Solaris ゾーンで発生するさまざまな問題のトラブルシューティング

この章では、ゾーンのトラブルシューティングについて説明します。

排他的 IP ゾーンがデバイスを使用しているために dladm reset-linkprop が失敗する

次のエラーメッセージが表示された場合

```
dladm: warning: cannot reset link property 'zone' on 'bge0': operation failed
```

412 ページの「dladm reset-linkprop の使用方法」で説明されているように、dladm reset-linkprop を使用しようとして失敗しました。実行中のゾーン excl がそのデバイスを使用しています。

値をリセットするには、次の手順に従います。

1. 次のように入力します。

   ```
   global# ipadm delete-ip bge0
   ```
2. dladm コマンドを再実行します。

ゾーン構成内に不正な特権セットが指定されている

ゾーンの特権セットに許可されない特権が含まれる場合、必須の特権が欠落している場合、または不明な特権名が含まれる場合、ゾーンの検証、準備、またはブートの試行は失敗し、次のようなエラーメッセージが表示されます。

```
zonecfg:zone5> set limitpriv="basic"
.
.
.
```

```
global# zoneadm -z zone5 boot
     required privilege "sys_mount" is missing from the zone’s privilege set
     zoneadm: zone zone5 failed to verify
```

ゾーンが停止しない

ゾーンに関連付けられたシステム状態を破棄できない場合には、停止処理は中途で失敗します。このため、稼働中とインストール済みの間で、ゾーンが中間状態のままになります。この状態では、アクティブなユーザープロセスやカーネルスレッドは存在せず、何も作成できません。停止操作が失敗した場合は、手動で処理を完了する必要があります。

障害のもっとも一般的な原因は、システムがすべてのファイルシステムをアンマウントできないことです。システム状態が破棄される Oracle Solaris システムの従来の停止処理とは異なり、ゾーンでは、ゾーンのブートまたは操作時に実行されたマウントがゾーンの停止後に残らないことを保証する必要があります。`zoneadm` によりゾーン内で実行中のプロセスが存在しないことが確認されても、大域ゾーン内のプロセスによりゾーン内のファイルが開かれた場合には、アンマウント操作が失敗することがあります。`proc`(1) (`pfiles` に記載) および `fuser`(1M) のマニュアルページに記載されているツールを使用してこれらのプロセスを検索し、適切な処理を実行してください。これらのプロセスを処理したあとで `zoneadm halt` を再度呼び出すと、ゾーンが完全に停止します。

パート III

Oracle Solaris 10 ゾーン

Oracle Solaris 10 ゾーンとは `solaris10` ブランドゾーンです。このゾーンでは、Oracle Solaris 11 カーネル上で実行される x86 および SPARC の Solaris 10 9/10 (または以降にリリースされた Solaris 10 アップデート) のユーザー環境をホストします。カーネルパッチの 142909-17 (SPARC) または 142910-17 (x86/x64)、あるいはそれらよりも後のバージョンを最初に元のシステムにインストールしている場合は、10/09 よりも前の Oracle Solaris 10 リリースを使用できます。

◆ ◆ ◆ 第 29 章

29 Oracle Solaris 10 ゾーンの紹介

BrandZ は、ブランドゾーンを作成するためのフレームワークを提供します。ブランドゾーンは、Oracle Solaris 11 環境で実行できないアプリケーションを実行するために使用します。ここで説明するブランドは `solaris10` ブランドで、これは Oracle Solaris 10 ゾーンです。これらの `solaris10` ブランドゾーン内で実行されるワークロードでは、Oracle Solaris カーネルに追加された拡張機能を活用したり、Oracle Solaris 11 リリースでのみ利用できる革新的な技術 (仮想 NIC (VNIC)、ZFS の複製解除など) を使用したりできます。

注 – `solaris10` ブランドゾーンをすぐに作成する場合は、第 30 章「Oracle Solaris 10 システムの評価とアーカイブの作成」に進んでください。

---

solaris10 ブランドについて

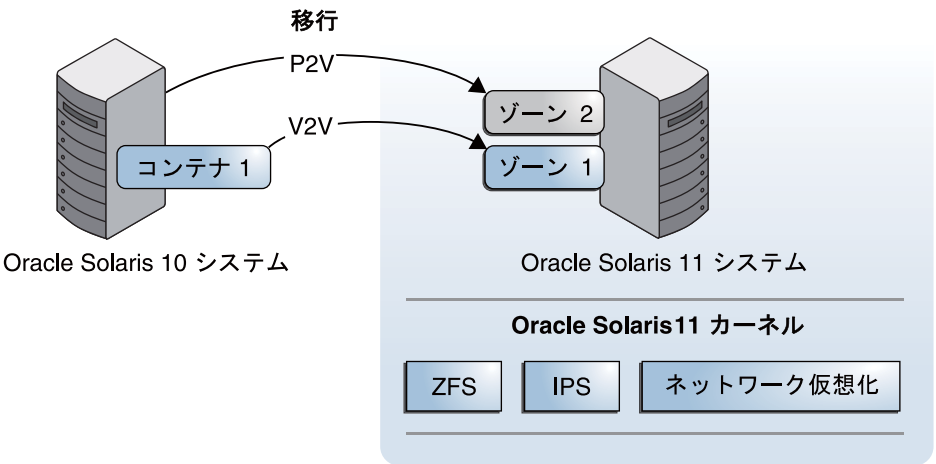
`solaris10` ブランドゾーン (`solaris10(5)` のマニュアルページに記載) は、Oracle Solaris 10 9/10 オペレーティングシステムまたは以降にリリースされたアップデートを実行している SPARC および x86 マシン上の Oracle Solaris 10 アプリケーションの完全な実行時環境です。Oracle Solaris 10 9/10 より前の Oracle Solaris 10 リリースを実行している場合は、最初にカーネルパッチ 142909-17 (SPARC) または 142910-17 (x86/x64) (または以降のバージョン) を元のシステムにインストールすると、以前のアップデートリリースを使用できます。ゾーンのインストールに使用するアーカイブを作成する前に、このパッチをインストールする必要があります。Oracle Solaris 10 ゾーンへの移行の前提条件は、完全な Oracle Solaris 10 9/10 または以降のリリースではなく、リリースのカーネルパッチです。パッチのソフトウェアダウンロードサイトは、My Oracle Support (`https://support.oracle.com`) です。「パッチと更新 (Patches & Updates)」タブをクリックします。そのサイトでは、ダウンロード手順を確認し、イメージをダウンロードすることができます。パッチに関する追加情報については、サポートプロバイダにお問い合わせください。

単一の大域ゾーンインスタンス内で実行されている非大域ゾーンは、Oracle Solaris 11.1 リリースのサポート対象プラットフォームとして定義されているすべてのアーキテクチャーでサポートされます。

ブランドには、Oracle Solaris 10 システムイメージを非大域ゾーンにインストールする際に必要となるツールが含まれています。`solaris10` ブランドゾーンは、Oracle Solaris 10 のメディアから直接インストールできません。既存のシステムをターゲットシステム上の非大域ゾーンに直接移行するには、P2V (Physical-To-Virtual) 機能を使用します。`zonep2vchk` ツールは、P2V プロセスに必要な情報の生成、およびターゲットシステムで使用するテンプレート zonecfg コマンド行の出力を行うために使用します。このユーティリティーは、ソースシステムの構成と一致するゾーンを作成します。ソース Oracle Solaris 10 1/13 システムでは、ユーティリティー `/usr/sbin/zonep2vchk` がこのリリースに含まれています。Oracle Solaris 10 の以前のバージョンでこのユーティリティーを使用するには、Oracle Technology Network (OTN) (`http://www.oracle.com/technetwork/server-storage/solaris10/downloads`) からアンバンドルのパッケージをダウンロードします。

またブランドでは、Oracle Solaris 10 ネイティブゾーンを `solaris10` ブランド非大域ゾーンに移行する際に使用されるツールもサポートされています。Oracle Solaris 10 のネイティブな非大域ゾーンを `solaris10` ブランドゾーンに移行する V2V (Virtual-To-Virtual) プロセスでは、P2V と同じアーカイブ形式がサポートされます。詳細は、第 31 章「(オプション) Oracle Solaris 10 ゾーンへの Oracle Solaris 10 native 非大域ゾーンの移行」を参照してください。

図 29–1　Oracle Solaris 10 ゾーンへの Oracle Solaris 10 コンテナの移行



solaris10 ゾーンのサポート

solaris10 ブランドゾーンは完全ルート非大域ゾーンモデルをサポートします。必須の Oracle Solaris 10 ソフトウェアパッケージおよび追加パッケージのすべてが、ゾーン固有のファイルシステムにインストールされます。

非大域ゾーンは固有の ZFS データセット上に構築される必要があり、ZFS だけがサポートされています。ZFS データセットは、ゾーンのインストール時または接続時に自動的に作成されます。ZFS データセットを作成できない場合、ゾーンのインストールや接続は行われません。ゾーンパスの親ディレクトリも ZFS データセットである必要があります。それ以外の場合は、ファイルシステムの作成が失敗します。

ネイティブな Oracle Solaris 10 非大域ゾーンで実行されるアプリケーションまたはプログラムは、solaris10 ブランドゾーンでも動作します。

ゾーンでは、静的にリンクされたバイナリはサポートされません。

注 – solaris10 ブランドゾーンは、ラベルが有効にされた Oracle Solaris Trusted Extensions システムに作成およびインストールできますが、このシステム構成のブランドゾーンをブートできるのは、ブートするブランドがラベル付きのブランドである場合のみです。Oracle Solaris 10 システムで Oracle Solaris Trusted Extensions を使用しているお客様は、認定されている Oracle Solaris システム構成に移行する必要があります。

Oracle Solaris 10 ゾーンの SVR4 パッケージおよびパッチ

solaris10 ブランドゾーンでのパッケージおよびパッチの使用について

SVR4 パッケージのメタデータは、ゾーンの内部で使用可能であり、パッケージおよびパッチのコマンドは正常に動作します。正常に動作させるためには、アーカイブを作成する前に、パッチ 119254-75 (SPARC) または 119255-75 (x86/x64) (または以降のバージョン) を Oracle solaris 10 システムにインストールする必要があります。パッチのソフトウェアダウンロードサイトは、My Oracle Support (https://support.oracle.com) です。ダウンロードの説明を表示して、イメージをダウンロードするには、「パッチと更新 (Patches & Updates)」タブをクリックします。パッチに関する追加情報については、サポートプロバイダにお問い合わせください。

solaris10 ブランドゾーンは完全ルートゾーンであるため、すべてのパッケージ操作およびパッチ操作は、マニュアルページおよびその他のドキュメントで説明されているとおりに機能します。パッケージまたはパッチのカーネルコンポーネントはイ

ンストールに使用されません。SVR4 パッケージは、現在のゾーンにのみインストールされます。solaris10 および native ゾーンで使用される SVR4 パッケージについては、『System Administration Guide: Oracle Solaris Containers-Resource Management and Oracle Solaris Zones』の第 25 章「ゾーンがインストールされている Solaris システムでのパッケージについて (概要)」および第 26 章「ゾーンがインストールされている Solaris システムでのパッケージとパッチの追加および削除 (タスク)」を参照してください。これは、このガイドの Oracle Solaris 10 バージョンです。

システムのリリースレベルについては、第 29 章「Oracle Solaris 10 ゾーンの紹介」を参照してください。

リモートからのパッケージ操作およびパッチ操作の実行について

パッチ操作が Oracle Solaris 10 ゾーン内から開始された場合、リモートシステムが別の solaris10 ゾーンであれば、パッチ操作は正常に動作します。ただし、リモートシステムがミニルート、または solaris10 ゾーンではない Oracle Solaris 10 システムである場合、この操作では未定義の結果が生成されます。同様に、パッチツールを使用して、Oracle Solaris 10 ゾーンではなくミニルートまたは物理システムから Oracle Solaris 10 ゾーンにパッチを適用すると、未定義の結果が生成されます。

一般に、patchadd および patchrm ツールでは、管理者はパッチ操作の実行時に代替ルートを指定できます。この機能により、管理者はリモートシステム (NFS 経由で root ディレクトリを表示できる Oracle Solaris 10 ミニルートおよび Oracle Solaris 10 物理システムなど) にパッチを適用できます。たとえば、Oracle Solaris 10 システムのルートディレクトリがローカルシステムの /net/a-system ディレクトリに NFS マウントされている場合は、ローカルシステムからリモートの Oracle Solaris 10 システムにパッチを適用できます。

リモートシステムにパッチ 142900-04 (または以降のバージョン) をインストールするには、次の手順に従います。

```
# patchadd -R /net/a-system 142900-04
```

詳細については、『man pages section 1M: System Administration Commands』の次のマニュアルページを参照してください。

- patchadd(1M)、-R および -C オプション
- patchrm(1M)

NFS クライアントとして機能する非大域ゾーン

ゾーンは、NFS クライアントとしても機能できます。バージョン 2、バージョン 3、およびバージョン 4 プロトコルがサポートされます。これらの NFS バージョンについては、『Oracle Solaris のシステム管理 (ネットワークサービス)』の「NFS サービスの機能」を参照してください。

デフォルトのバージョンは、NFS バージョン 4 です。次のいずれかの方法を使用して、クライアント上でほかの NFS バージョンを有効にできます。

- `/etc/default/nfs` を編集して `NFS_CLIENT_VERSMAX=`*number* を設定することで、指定したバージョンをゾーンのデフォルトとして使用できます。『Oracle Solaris のシステム管理 (ネットワークサービス)』の「NFS サービスの設定」を参照してください。タスクマップの「`/etc/default/nfs` ファイルを変更して、クライアント上で異なるバージョンの NFS を選択する方法」に記載された手順を実行します。
- バージョンマウントを手動で作成できます。この方法を使用すると、`/etc/default/nfs` の内容がオーバーライドされます。『Oracle Solaris のシステム管理 (ネットワークサービス)』の「NFS サービスの設定」を参照してください。タスクマップの「コマンド行を使用して、クライアント上で異なるバージョンの NFS を選択する方法」に記載された手順を実行します。

ゾーンの一般的な概念

次に示すリソース管理およびゾーンの概念 (このガイドのパート I「Oracle Solaris のリソース管理」およびパート II「Oracle Solaris ゾーン」を参照) について、よく理解しておくようにしてください。

- `zonep2vchk` ツール
- サポートされる機能とサポートされない機能
- 利用可能なシステムリソースをアプリケーションでどのように使用するかを管理者が制御できるようにするリソース制御
- ゾーンの構成、インストール、および管理に使用されるコマンド。主に `zonecfg`、`zoneadm`、および `zlogin`
- `zonecfg` リソースおよびプロパティータイプ
- 大域ゾーンおよび非大域ゾーン
- 完全ルート非大域ゾーンモデル
- 承認は、`zonecfg` ユーティリティーを使用して付与します。
- 大域管理者とゾーン管理者
- ゾーンの状態モデル
- ゾーン隔離の特徴
- 特権

- ネットワーク処理
- anet リソースを使用した IPoIB の構成
- ゾーンの共有 IP および排他的 IP タイプ
- リソースプールなどのリソース管理機能をゾーンで使用する方法
- 公平配分スケジューラ (FSS)。これは、配分に基づいて CPU 時間を割り当てることができるようにするスケジューリングクラスです
- リソース上限デーモン (rcapd)。これを大域ゾーンから使用して、ブランドゾーンの常駐セットサイズ (RSS) 使用量を制御できます

このリリースの Oracle Solaris 10 ゾーンについて

操作に関する制限事項

/dev/sound のデバイスは solaris10 ブランドゾーンに構成できません。

読み取り専用ゾーンの作成に使用する file-mac-profile プロパティーは使用できません。

quota コマンド (quota(1M) を参照) は、solaris10 ブランドゾーンの内部で使用される UFS ファイルシステムの割り当て情報を取得するために使用することはできません。

solaris10 ブランドゾーンを NFS サーバーにすることはできません。

Oracle Solaris 10 ゾーンのネットワーク

次のセクションでは、Oracle Solaris 10 ゾーンで使用できない、または Oracle Solaris 10 ゾーンでは異なる Oracle Solaris 10 ネットワークコンポーネントを示します。

サポートされないネットワークコンポーネント

- atun STREAMS モジュールを使用した自動トンネルはサポートされません。
- 次のチューニング可能な ndd パラメータは、solaris10 ブランドゾーンではサポートされません。
  - ip_squeue_fanout
  - ip_soft_rings_cnt
  - ip_ire_pathmtu_interval
  - tcp_mdt_max_pbufs

異なるネットワーク機能

排他的 IP 構成を持つ solaris10 ブランドゾーンでは、次の機能が物理的な Oracle Solaris 10 システムと異なります。

- モバイル IP は Oracle Solaris 11 リリースで使用できないため、使用できません。
- solaris10 ブランドゾーンでは、tcp、udp、または icmp ソケットが開いている場合、autopush 構成は無視されます。デフォルトでは、これらのソケットは、STREAMS デバイスではないモジュールにマップされます。autopush を使用するには、soconfig および sock2path.d ユーティリティー (soconfig(1M) および sock2path.d(4) のマニュアルページに記載) を使用して、これらのソケットを明示的に STREAMS ベースのデバイスにマップします。
- Oracle Solaris 10 9/10 またはそれ以前のアップデートを実行している物理システムからアーカイブされた solaris10 ブランドゾーンでは、/dev/net のリンク (VNIC など) はデータリンクプロバイダインタフェースライブラリ (libdlpi) によってサポートされません。これらのリンクは、Oracle Solaris 10 8/11 でサポートされています。このライブラリについては、libdlpi(3LIB) のマニュアルページに記載されています。

  Oracle Solaris 10 8/11 の libdlpi ライブラリ、libpcap バージョン 1.0.0、またはそれ以上のライブラリを使用しないアプリケーションは、/dev/net のリンク (VNIC など) にアクセスできません。
- Oracle Solaris 10 ゾーンの IP ネットワークマルチパス (IPMP) は Oracle Solaris 11 リリースを基にしているため、Oracle Solaris 10 オペレーティングシステムのコマンド出力と比較すると、ifconfig コマンドの出力に違いがあります。ただし、ドキュメントに記載されている ifconfig コマンドおよび IPMP の機能は変更されません。そのため、記載されているインタフェースを使用する Oracle Solaris 10 アプリケーションは、変更なしで Oracle Solaris 10 ゾーンで引き続き動作します。

  次の例は、データアドレス 192.168.1.3 の IPMP グループ ipmp0 と、テストアドレス 192.168.1.1 および 192.168.1.2 のベースとなるインタフェース e1000g1 および e1000g2 に対する solaris10 ブランドゾーンの ifconfig コマンドの出力をそれぞれ示しています。

```
% ifconfig -a
e1000g1:
flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
mtu 1500 index 8
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
        ether 0:11:22:45:40:a0
e1000g2:
flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
mtu 1500 index 9
        inet 192.162.1.2 netmask ffffff00 broadcast 192.168.1.255
        ether 0:11:22:45:40:a1
ipmp0: flags=8011000803<UP,BROADCAST,MULTICAST,IPv4,FAILED,IPMP> mtu 68
index 10
        inet 192.168.1.3 netmask ffffff00 broadcast 192.168.1.255
        groupname ipmp0
```

- Oracle Solaris 10 システムで生成される表示と異なり、Oracle Solaris 10 コンテナの `ifconfig` コマンドでは、IP アドレスへのベースとなるインタフェースの割り当ては表示されません。この情報は、`arp` コマンドを `-an` オプションとともに実行することにより取得できます。
- インタフェースが IPv6 に接続され、アドレスの構成が成功すると、そのインタフェースに固有のグローバルアドレスが与えられます。Oracle Solaris 10 システムでは、IPMP グループの各物理インタフェースは固有のグローバルアドレスを保有し、IPMP グループはインタフェースと同じ数のグローバルアドレスを保有します。Oracle Solaris 10 ゾーンでは、固有のグローバルアドレスを保有するのは IPMP インタフェースのみです。ベースとなるインタフェースは、固有のグローバルアドレスを保有しません。
- Oracle Solaris 10 オペレーティングシステムと異なり、IPMP グループにインタフェースが 1 つしかない場合、テストアドレスおよびデータアドレスは同じにできません。

`arp`(1M) および `ifconfig`(1M) のマニュアルページおよび373 ページの「排他的 IP ゾーン内の IP ネットワークマルチパス」を参照してください。

ネイティブ非大域ゾーンがインストールされている場合

Oracle Solaris 10 9/10 (または以降にリリースされたアップデート) のソース物理システムに `native` ゾーンがある場合は、P2V プロセスで追加のステップが発生します。ゾーンは入れ子にならないため、これらのシステムの P2V プロセスは、ブランドゾーン内部の既存のゾーンを使用できないようにします。ゾーンがインストールされると、既存のゾーンが検出され、入れ子になったゾーンは使用不可であることと、ディスク領域が回復されることを示す警告が発行されます。これらのゾーンは、最初に V2V プロセス (第 31 章「(オプション) Oracle Solaris 10 ゾーンへの Oracle Solaris 10 `native` 非大域ゾーンの移行」に記載) を使用して移行できます。

以前のリリースを実行しているシステムにカーネルパッチを適用する場合は、既存のゾーンを移行する前にパッチを適用します。

◆ ◆ ◆ 第 30 章

Oracle Solaris 10 システムの評価とアーカイブの作成

この章では、Oracle Solaris 10 10/09 (または以降にリリースされたアップデート) システムに関する情報を取得する方法、およびシステムのアーカイブを作成する方法について説明します。既存の Oracle Solaris システムをターゲットシステム上の非大域ゾーンに直接移行するには、P2V (Physical-To-Virtual) 機能を使用します。ターゲットシステムで必須となるパッケージに関する情報も提供されます。

ソースシステムおよびターゲットシステムの前提条件

Oracle Solaris 10 のパッケージツールおよびパッチツールの有効化

Oracle Solaris 10 ゾーンで Oracle Solaris 10 のパッケージツールおよびパッチツールを使用するには、イメージを作成する前に、ソースシステムのアーキテクチャーに次のパッチをインストールします。

- 119254-75、119534-24、および 140914-02 (SPARC)
- 119255-75、119535-24、および 140915-02 (x86/x64)

P2V プロセスはこれらのパッチを適用しなくても動作しますが、パッケージツールおよびパッチツールは `solaris10` ブランドゾーン内で正常に動作しません。

ターゲットシステムへの必須 Oracle Solaris パッケージのインストール

使用しているシステムで Oracle Solaris 10 ゾーンを使用するには、Oracle Solaris 11 を実行しているシステムに `pkg:/system/zones/brand/brand-solaris10` をインストールする必要があります。

リポジトリの詳細は、347 ページの「Oracle Solaris 11.1 リリースを実行しているシステムのイメージパッケージングシステムソフトウェア」を参照してください。

パッケージのインストール手順については、『Oracle Solaris 11.1 ソフトウェアパッケージの追加および更新』を参照してください。

zonep2vchk ユーティリティーを使用して移行されるシステムの評価

既存の Oracle Solaris 10 9/10 システム (または以降にリリースされた Solaris 10 アップデート) は、Oracle Solaris 11 システムの `solaris10` ブランドゾーンに直接移行できます。

はじめに、`zonep2vchk` ツール (`zonep2vchk(1M)` および第 22 章「ゾーンの移行と zonep2vchk ツールについて」を参照) を使用して、ソースシステムを検査し、必要な情報を収集します。このツールは、移行するシステムを評価し、`zonecfg` テンプレート (ネットワーク構成を含む) を生成するために使用されます。

元のシステムで実行されていたサービスに応じて、大域管理者または適切な承認を付与されたユーザーは、インストール後にゾーンを手動でカスタマイズする必要が生じる場合があります。たとえば、ゾーンに割り当てられた特権の変更が必要になる場合があります。この操作は自動的に実行されることはありません。また、すべてのシステムサービスがゾーン内で動作するとは限らないため、すべての Oracle Solaris 10 システムがゾーンへの移行に適しているわけではありません。

注 – 移行するシステムにネイティブな非大域ゾーンがある場合、これらのゾーンは、削除するか、最初にアーカイブして新しいターゲットシステムのゾーンに移動する必要があります。疎ルートゾーンの場合、アーカイブは準備完了状態のゾーンで作成する必要があります。移行の詳細については、第 31 章「(オプション) Oracle Solaris 10 ゾーンへの Oracle Solaris 10 native 非大域ゾーンの移行」を参照してください。疎ルートゾーンの詳細については、Oracle Solaris 10 ドキュメントのゾーンの概要を参照してください。

Oracle Solaris 10 システムのみ: zonep2vchk ユーティリティーの取得

`zonep2vchk` ユーティリティーは、Oracle Solaris 10 1/13 システムで使用できます。

Oracle Solaris 10 システムの以前のバージョンでこのユーティリティーを使用するには、OTN (`http://www.oracle.com/technetwork/server-storage/solaris10/downloads`) からアンバンドルのパッケージをダウンロードできます。別パッケージは `/opt/SUNWzonep2vchk` にインストールされます。

注 – このアンバンドルのパッケージは、引き続きシステムをアップグレードするか、またはシステムにパッチを適用すれば、Oracle Solaris 10 1/13 によってもたらされたバージョンと干渉しません。アンバンドル版は、`/opt/SUNWzonep2vchk` にインストールされます。Oracle Solaris 10 1/13 のアップグレードまたはパッチにより、`/usr/sbin` にバンドル版が追加されます。以前に取得したアンバンドルのパッケージはアンインストールできます。

Oracle Solaris 10 システムをゾーンに直接移行するためのイメージの作成

インストール済みの Solaris システムのイメージを Oracle Solaris Flash アーカイブツールを使用して作成し、ゾーンに直接移行することができます。

イメージを作成する前に、ゾーンで実行するすべてのソフトウェアを含む完全なシステムを構成できます。このイメージは、ゾーンのインストール時にインストーラによって使用されます。

▼ flarcreate を使用してイメージを作成する方法

ZFS ルートを持つシステムでは、`flarcreate` コマンド (`flarcreate(1M)` の Oracle Solaris 10 マニュアルページに記載) を使用して、システムイメージを作成できます。デフォルトでは、作成される `flar` は ZFS 送信ストリームです (『Oracle Solaris 11.1 の管理: ZFS ファイルシステム』の「ZFS データを送信および受信する」に記載)。

この手順例では、ターゲットの Oracle Solaris 11 システムにフラッシュアーカイブを配置するために NFS を使用しますが、ほかの方法でファイルを移動することもできます。

この手順を実行するには、大域管理者または大域ゾーン内で必要な権利プロファイルを持つユーザーである必要があります。

1. **root** になるか、同等の役割になります。
2. アーカイブするソース **Oracle Solaris 10** システムにログインします。
3. ディレクトリをルートディレクトリに変更します。

```
# cd /
```

4 ソースシステムで **flarcreate** を使用して **s10-system** という名前のフラッシュアーカイブイメージファイルを作成し、そのアーカイブをターゲットの **Oracle Solaris 11** システムに配置します。

```
source-system # flarcreate -n s10-system /net/target/export/archives/s10-system.flar
```

▼ flarcreate を使用して特定のデータを除外する方法

ZFS データセットの境界にないデータをアーカイブから除外するには、cpio または pax に flarcreate を付けて実行する必要があります。-L archiver オプションを使用すると、ファイルをアーカイブする方法として cpio または pax を指定できます。

この手順例では、ターゲットの Oracle Solaris 11 システムにフラッシュアーカイブを配置するために NFS を使用しますが、ほかの方法でファイルを移動することもできます。

この手順を実行するには、大域管理者または大域ゾーン内で必要な権利プロファイルを持つユーザーである必要があります。

1 **root** になるか、同等の役割になります。

2 アーカイブするソース **Oracle Solaris 10** システムにログインします。

3 ディレクトリをルートディレクトリに変更します。

```
# cd /
```

4 ソースシステムで **flarcreate** を使用して **s10-system** という名前のフラッシュアーカイブイメージファイルを作成し、そのアーカイブをターゲットの **Oracle Solaris 11** システムに配置します。

```
source-system # flarcreate -S -n s10-system -x /path/to/exclude -L cpio
                /net/target/export/archives/s10-system.flar
Determining which filesystems will be included in the archive...
Creating the archive...
cpio: File size of "etc/mnttab" has
increased by 435
2068650 blocks
1 error(s)
Archive creation complete.
```

ヒント – 場合によっては、flarcreate の実行時に cpio コマンドからのエラーが表示されることがあります。もっとも多いのは、「File size of etc/mnttab has increased by 33」のようなメッセージです。これらのメッセージがログファイルまたはシステム状態を反映するファイルに関連するものであれば、無視してもかまいません。必ずすべてのエラーメッセージを確認してください。

ほかのアーカイブ作成方法

別の方法を使用してアーカイブを作成することもできます。インストーラは次のアーカイブフォーマットを受け入れることができます。

- cpio アーカイブ
- gzip で圧縮された cpio アーカイブ
- bzip2 で圧縮された cpio アーカイブ
- -x xustar (XUSTAR) 形式で作成された pax アーカイブ
- ufsdump レベル 0 (完全) バックアップ

また、インストーラは、ファイルのアクセス権、所有権、およびリンクを保存および復元するアーカイブユーティリティーを使用して作成されたファイルのディレクトリのみを受け入れることができます。

詳細は、cpio(1)、pax(1)、bzip2 (1)、gzip(1)、および ufsdump(1M) のマニュアルページを参照してください。

注 – フラッシュアーカイブ以外の方法を使用して P2V 用アーカイブを作成する場合は、アーカイブを作成する前に、ソースシステム上でプロセッサ依存の libc.so.1 の、lofs でマウントされたハードウェア機能 (hwcap) ライブラリをアンマウントする必要があります。それらをマウント解除しなければ、そのアーカイブを使用してインストールしたゾーンがターゲットシステムでブートしない可能性があります。アーカイブの作成後に、lofs と mount -O オプションを使用して、/lib/libc.so.1 上に適切なハードウェア機能ライブラリを再マウントできます。

```
source-system# umount /lib/libc.so.1
source-system# mount -O -F lofs /lib/libc.so.1
```

ホストIDのエミュレーション

スタンドアロンの Oracle Solaris システムから新しいシステム上のゾーンにアプリケーションを移行すると、hostid は新しいマシンの hostid に変更されます。

場合によっては、アプリケーションが元の hostid に依存しており、アプリケーション構成を更新できないことがあります。このような場合は、元のシステムの hostid を使用するようにゾーンを構成することができます。そのためには、270 ページの「ゾーンの構成方法」の説明に従って、zonecfg プロパティーを設定して hostid を指定します。値としては、元のシステムで hostid コマンドを実行した場合の出力を使用してください。インストール済みゾーンで hostid を表示する場合も、hostid コマンドを使用します。

ホスト ID の詳細は、hostid(1) のマニュアルページを参照してください。

◆ ◆ ◆ 第 31 章

(オプション) Oracle Solaris 10 ゾーンへの Oracle Solaris 10 native 非大域ゾーンの移行

この章では、Oracle Solaris 10 9/10 (または以降にリリースされたアップデート) システムの native 非大域ゾーンを Oracle Solaris 11 リリースを実行しているシステムの Oracle Solaris 10 ゾーンに移行する方法について説明します。

この章は、移行するシステムにネイティブな非大域ゾーンがある場合にのみ参照してください。これらのゾーンは、最初にアーカイブして新しいターゲットシステムのブランドゾーンに移動する必要があります。

アーカイブに関する考慮事項

Oracle Solaris 10 システムの疎ルートゾーンは、システムによって solaris10 ブランドゾーン移行用の完全ルートモデルに変換されます。疎ルートゾーンは、V2V プロセスが実行される前に、ソースシステムで準備完了状態である必要があります。これにより、アーカイブが作成される前に、すべての inherited-pkg-dir リソースがマウントされます。これらの概念の詳細は、Oracle Solaris 10 バージョンのこのガイドのゾーンの概要を参照してください。

ゾーンのブランドは、プロセスの一環として変更されます。

solaris10 ゾーン移行プロセスの概要

Oracle Solaris 10 のネイティブゾーンを solaris10 ブランドゾーンに移行する V2V (Virtual-To-Virtual) プロセスは、P2V と同じアーカイブ形式をサポートします。このプロセスは、zoneadm install サブコマンドを使用します。solaris10 ブランドの install サブコマンドでは、attach サブコマンドの同じオプションに対応する次のオプションを使用します。

注 – attach サブコマンドの -a オプションと -d オプションは、Oracle Solaris の将来のリリースで削除される可能性があります。install サブコマンドを使用することをお勧めします。

オプション	説明
-a *path*	ゾーンに展開するアーカイブのパスを指定します。完全なフラッシュアーカイブと pax、cpio、gzip で圧縮された cpio、bzip で圧縮された cpio、およびレベル 0 ufsdump がサポートされています。
-d *path*	ファイルのツリーへのパスをインストールのソースとして指定します。
-d –	-d オプションとダッシュ (-) パラメータを使用して、zonepath で既存のディレクトリレイアウトが使用されるように指定します。このため、インストールの前に管理者が手動で zonepath ディレクトリを設定する場合に、-d – オプションを使用してそのディレクトリがすでに存在するかどうかを示すことができます。

solaris10 ゾーンの切り離しおよび接続について

solaris10 ゾーンは、ターゲットシステムにゾーンを構成してから、zoneadm コマンドに detach および attach サブコマンドを指定し、アーカイブを接続する場合は -a オプション、zonepath を指定する場合は -d オプションを付けて実行することにより、Oracle Solaris ホストに移行できます。この手順については、339 ページの「ゾーンの移行について」および340 ページの「ZFS アーカイブを使用して非大域ゾーンを移行する方法」で説明しています。

注 – attach サブコマンドの -a オプションと -d オプションは、Oracle Solaris の将来のリリースで削除される可能性があります。install サブコマンドを使用することをお勧めします。

solaris10 ブランドゾーンの移行

zonecfg および zoneadm コマンドを使用すると、あるシステムから別のシステムに既存の非大域ゾーンを移行できます。ゾーンは停止され、現在のホストから切り離されます。zonepath はターゲットホストに移動され、そこで接続されます。

zoneadm detach プロセスにより、別のシステムでゾーンを接続するのに必要な情報が作成されます。 zoneadm attach プロセスは、ターゲットマシンがゾーンのホストとして機能するための適正な構成を保持していることを確認します。

新規ホストで zonepath を使用可能にする方法は複数存在するため、あるシステムから別のシステムへの zonepath の実際の移動は、大域管理者が手動で行います。

新規システムへの接続時に、ゾーンはインストール済みの状態になります。

例 31–1 attach コマンドの例

```
host2# zoneadm -z zonename attach -a /net/machine_name/s10-system.flar
```

Oracle Solaris 10 システムの既存のゾーンの移行

物理システムを移行する前に、最初にシステムの既存の非大域ゾーンをアーカイブして新しいターゲットシステムのゾーンに移動する必要があります。

▼ 既存のネイティブ非大域ゾーンを移行する方法

V2V プロセスを使用して、Solaris 10 システムの既存のゾーンを Oracle Solaris 11 リリースを実行しているシステムの solaris10 ブランドゾーンに移行します。

1. 既存のゾーンの構成を出力します。この情報は、宛先システムでゾーンを再作成するために必要になります。

```
source# zonecfg -z my-zone info
zonename: my-zone
zonepath: /zones/my-zone
brand: native
autoboot: false
bootargs:
pool:
limitpriv:
scheduling-class:
ip-type: shared
hostid: 1337833f
inherit-pkg-dir:
         dir: /lib
inherit-pkg-dir:
         dir: /platform
inherit-pkg-dir:
         dir: /sbin
inherit-pkg-dir:
         dir: /usr
net:
         address: 192.168.0.90
         physical: bge0
```

2. ゾーンを停止します。

```
source# zoneadm -z my-zone halt
```

ゾーン内のアプリケーションデータまたはシステムデータが整合性のない状態で取得される可能性があるため、実行中のゾーンをアーカイブしないようにしてください。

3 (オプション) ゾーンが **inherit-pkg-dir** 設定を持つ疎ルートゾーンである場合、継承されたディレクトリがアーカイブされるように、最初にゾーンを **ready** 状態にします。

```
source# zoneadm -s my-zone ready
```

4 **zonepath** が **/zones/my-zone** のゾーンをアーカイブします。

- そのゾーンの **gzip** で圧縮された **cpio** アーカイブを **my-zone.cpio.gz** という名前で作成します。ターゲットシステムでも、これには **my-zone** という名前が付けられます。

  ```
  source# cd /zones
  source# find my-zone -print | cpio -oP@ | gzip >/zones/my-zone.cpio.gz
  ```

- ターゲットシステムでゾーンの名前を変更する場合は、**zonepath** 内からアーカイブを作成します。

  ```
  source# cd /zones/my-zone
  source# find root -print | cpio -oP@ | gzip >/zones/my-zone.cpio.gz
  ```

5 次のようなファイルをコピーするためのファイル転送メカニズムを使用して、アーカイブをターゲットの **Oracle Solaris 11.1** システムに転送します。

- `sftp` コマンド (`sftp(1)` のマニュアルページに記載)
- NFS マウント
- ファイルをコピーするためのその他のファイル転送メカニズム。

6 ターゲットシステムでゾーンを再作成します。

```
target# zonecfg -z my-zone
my-zone: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:my-zone> create -t SYSsolaris10
zonecfg:my-zone> set zonepath=/zones/my-zone
...
```

注 – ゾーンのブランドは `solaris10` である必要があり、元のゾーンが疎ルートゾーンとして構成されていても、ゾーンは `inherit-pk -dir` 設定を使用できません。`inherit-pkg-dir` リソースの詳細は、Part II, Oracle Solaris ゾーンを参照してください。

宛先システムに、異なるハードウェア、異なるネットワークインタフェース、またはゾーンで構成する必要がある別のデバイスまたはファイルシステムがある場合は、ゾーンの構成を更新する必要があります。第 16 章「非大域ゾーンの構成 (概要)」、第 17 章「非大域ゾーンの計画と構成 (タスク)」、および339 ページの「ゾーンの移行について」を参照してください。

---

7 ゾーンの構成の表示

```
target# zonecfg -z my-zone info
zonename: my-zone
zonepath: /zones/my-zone
brand: solaris10
autoboot: false
bootargs:
pool:
limitpriv:
scheduling-class:
ip-type: shared
hostid: 1337833f
net:
         address: 192.168.0.90
         physical: bge0
```

8 アーカイブを宛先システムの **/zones** ディレクトリに転送し、ソースシステムで作成されたアーカイブからゾーンをインストールします。

```
target# zoneadm -z my-zone install -a /zones/my-zone.cpio.gz
```

ゾーンのインストールが正常に完了したら、ゾーンをブートできます。

ゾーンのアーカイブをあとで使用する場合に備えて保存するか、システムから削除します。

宛先システムからアーカイブを削除するには、次の手順に従います。

```
target# rm /zones/myzone.cpio.gz
```

◆ ◆ ◆ 第 32 章

solaris10 ブランドゾーンの構成

この章では、solaris10 ブランドゾーンを構成する方法について説明します。

事前構成タスク

次のものが必要です。

- Oracle Solaris 11 リリースが実行されているサポートされる SPARC または x86 システム。
- デフォルトは、anet リソースを持つ排他的 IP タイプです。ネットワーク接続が必要な共有 IP ゾーンの場合は、作成する各ゾーンに 1 つ以上の一意の IPv4 アドレスを指定する必要があります。物理インタフェースも指定する必要があります。
- solaris 10 コンテナに移行する Oracle Solaris 10 10/09 (または以降にリリースされたアップデート) オペレーティングシステムを実行しているマシン。適切なカーネルパッチを使用して以前の更新を移行できます。既存のシステムから独自のイメージを生成することができます。この手順は、437 ページの「Oracle Solaris 10 システムをゾーンに直接移行するためのイメージの作成」で説明しています。

デフォルトで構成に含まれているリソース

ブランドゾーン内のデバイス、ファイルシステム、および特権は、デフォルトで構成に含められます。

solaris10 ブランドゾーンで構成されるデバイス

各ゾーンでサポートされるデバイスについては、そのブランドに関するマニュアルページやほかのガイドに記載されています。solaris10 ゾーンでは、サポートされていないデバイスや認識されないデバイスの追加は許可されません。サポートされて

いないデバイスを追加しようとすると、フレームワークによって検出されます。ゾーン構成が検証不可能であることを示すエラーメッセージが生成されます。

非大域ゾーンのデバイスに関する考慮事項の詳細は、373 ページの「非大域ゾーンでのデバイスの使用」を参照してください。

solaris10 ブランドゾーンで定義される特権

プロセスは、特権の一部に制限されています。特権を制限することで、ほかのゾーンに影響を及ぼす可能性がある操作がゾーンで実行されないようにします。特権セットにより、特権が付与されたユーザーがゾーン内で実行可能な機能が制限されます。

デフォルトの特権、必須のデフォルト特権、省略可能な特権、および禁止される特権が各ブランドによって定義されます。270 ページの「ゾーンの構成方法」の手順 8 に示されているとおり、`limitpriv` プロパティーを使用して特定の特権を追加または削除することもできます。表 25–1 に、Solaris の特権すべて、およびゾーン内での各特権のステータスを示します。

特権の詳細は、`ppriv(1)` のマニュアルページおよび『Solaris のシステム管理 (セキュリティサービス)』を参照してください。

solaris10 ブランドゾーンの構成処理

`zonecfg` コマンドは、次の処理を行うために使用されます。

- ゾーンのブランドを設定します。
- `solaris10` ゾーンの構成を作成します。
- 指定されたリソースおよびプロパティーが仮定のシステムで許可されており内部的に一貫しているかどうかを調べるために、構成を確認します。
- ブランド固有の確認を実行します。

ゾーン構成は `zonep2vchk` ユーティリティーを使用して作成できます。

特定の構成について `zonecfg verify` コマンドで実行される検査では、次のことが確認されます。

- ゾーンパスが指定されていること
- 各リソースの必須プロパティーがすべて指定されていること
- ブランドの要件が満たされていること

`zonecfg` コマンドの詳細は、`zonecfg(1M)` のマニュアルページを参照してください。

ターゲットゾーンの構成

Oracle Solaris 11 システムに `pkg:/system/zones/brand/brand-solaris10` をインストールする必要があります。

`zonecfg` コマンドを使用して、ターゲットシステムに新しいゾーン構成を作成します。

`zonecfg` のプロンプトは次のような形式です。

```
zonecfg:zonename>
```

ファイルシステムなど、特定のリソースタイプの構成を行うときは、そのリソースタイプもプロンプトに表示されます。

```
zonecfg:zonename:fs>
```

ヒント – CD または DVD を使用してアプリケーションを `solaris10` ブランドゾーンにインストールする予定の場合は、ブランドゾーンを最初に構成するときに `add fs` を使用して、大域ゾーンの CD または DVD メディアに読み取り専用のアクセスを行う権限を追加します。アクセス権を追加したら、CD または DVD を使用して製品をブランドゾーンにインストールできます。詳細は、407 ページの「非大域ゾーンで CD または DVD メディアにアクセスする権限を追加する方法」を参照してください。

▼ 排他的 IP solaris10 ブランドゾーンの構成方法

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を持つユーザーである必要があります。

1. **root** になるか、同等の役割になります。
2. 排他的 **IP** `solaris10` ゾーンを `s10-zone` というゾーン名で作成します。

   ```
   global# zonecfg -z s10-zone
   ```

   このゾーンの初回構成時には、次のシステムメッセージが表示されます。

   ```
   s10-zone: No such zone configured
   Use 'create' to begin configuring a new zone.
   ```

3. `SYSsolaris10` テンプレートを使用して、新しい `solaris10` ゾーン構成を作成します。

   ```
   zonecfg:s10-zone> create -t SYSsolaris10
   ```

   `SYSsolaris10` プロファイルによって、自動 `anet` リソースを含む排他的 IP ゾーンがデフォルトで作成されます。

4 ゾーンパス (この手順では、**/zones/s10-zone**) を設定します。

```
zonecfg:s10-zone> set zonepath=/zones/s10-zone
```

5 **autoboot** 値を設定します。

```
zonecfg:s10-zone> set autoboot=true
```

`true` に設定すると、大域ゾーンのブート時にこのゾーンが自動的にブートします。デフォルト値は `false` です。ゾーンを自動的にブートするには、ゾーンサービス `svc:/system/zones:default` も有効になっている必要があります。ゾーンサービスを有効にするには、`svcadm` コマンドを使用します。

6 大域ゾーンと共有する **ZFS** ファイルシステムを追加します。

```
zonecfg:s10-zone> add fs
```

a. タイプを **zfs** に設定します。

```
zonecfg:s10-zone:fs> set type=zfs
```

b. 大域ゾーンからマウントするディレクトリを設定します。

```
zonecfg:s10-zone:fs> set special=share/zone/s10-zone
```

c. マウントポイントを指定します。

```
zonecfg:s10-zone:fs> set dir=/opt/shared
```

d. 指定を終了します。

```
zonecfg:s10-zone:fs> end
```

この手順を複数回実行することで、複数のファイルシステムを追加できます。

7 ストレージプール *tank* の *sales* という **ZFS** データセットを委任します。

```
zonecfg:my-zone> add dataset
```

a. **ZFS** データセット *sales* のパスを指定します。

```
zonecfg:my-zone> set name=tank/sales
```

b. データセットの指定を終了します。

```
zonecfg:my-zone> end
```

8 **hostid** としてソースシステムの **hostid** を使用するように設定します。

```
zonecfg:my-zone> set hostid=80f0c086
```

9 ゾーンの構成を検証します。

```
zonecfg:s10-zone> verify
```

10 ゾーンの構成を確定します。

```
zonecfg:s10-zone> commit
```

11 **zonecfg** コマンドを終了します。

```
zonecfg:s10-zone> exit
```

プロンプトで commit コマンドを明示的に入力しなくても、exit を入力するか EOF が発生すると、commit の実行が自動的に試みられます。

12 **info** サブコマンドを使用して、ブランドが **solaris10** に設定されていることを確認します。

```
global# zonecfg -z s10-zone info
```

13 (オプション) **info** サブコマンドを使用して、**hostid** を確認します。

```
global# zonecfg -z s10-zone info hostid
```

次の手順

ヒント – ゾーンを構成したあとは、ゾーンの構成のコピーを作成することをお勧めします。このバックアップを使用して、あとでゾーンを再作成できます。適切なプロファイルを持つ root または管理者として、ゾーン *s10-zone* の構成をファイルに出力します。次の例では、s10-zone.config というファイルを使用しています。

```
global# zonecfg -z s10-zone export > s10-zone.config
```

参照 zonecfg を使用して構成可能な追加コンポーネントについては、第 16 章「非大域ゾーンの構成 (概要)」を参照してください。このガイドには、zonecfg コマンドをコマンド行モードまたはコマンドファイルモードで使用する方法も説明されています。共有 IP ゾーンの場合は、zonecfg net リソースに静的アドレスを割り当てる必要があります。ZFS ファイルシステムを追加する方法については、『Oracle Solaris 11.1 の管理: ZFS ファイルシステム』の「ZFS ファイルシステムを非大域ゾーンに追加する」を参照してください。

▼ 共有 IP solaris10 ブランドゾーンの構成方法

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を持つユーザーである必要があります。

1 **root** になるか、同等の役割になります。

2 共有 IP **solaris10** ゾーンを **s10-zone** というゾーン名で作成します。

```
global# zonecfg -z s10-zone
```

このゾーンの初回構成時には、次のシステムメッセージが表示されます。

```
s10-zone: No such zone configured
Use ’create’ to begin configuring a new zone.
```

3 新しい **solaris10** ゾーン構成を作成します。

```
zonecfg:s10-zone> create -b
set brand=solaris10
```

注 – **create -t SYSsolaris10-shared-ip** を使用して IP タイプを設定しないでください。

---

4 ゾーンパス (この手順では、**/zones/s10-zone**) を設定します。

```
zonecfg:s10-zone> set zonepath=/zones/s10-zone
```

5 **autoboot** 値を設定します。

`true` に設定すると、大域ゾーンのブート時にこのゾーンが自動的にブートします。ゾーンを自動的にブートするには、ゾーンサービス `svc:/system/zones:default` も有効になっている必要があります。デフォルト値は `false` です。

```
zonecfg:s10-zone> set autoboot=true
```

6 ネットワーク仮想インタフェースを持つ共有 **IP** ゾーンを作成します。

```
zonecfg:my-zone> set ip-type=shared

zonecfg:my-zone> add net
```

a. ネットワークインタフェースの物理デバイスタイプ (この手順では **bge** デバイス) を指定します。

```
zonecfg:my-zone:net> Set physical=bge0
```

b. **IP** アドレス (この手順では *10.6.10.233/24*) を設定します。

```
zonecfg:my-zone:net> Set address=10.6.10.233/24
```

c. 指定を終了します。

```
zonecfg:my-zone:net> end
```

この手順を複数回実行することで、複数のネットワークインタフェースを追加できます。

7 大域ゾーンと共有する **ZFS** ファイルシステムを追加します。

```
zonecfg:s10-zone> add fs
```

a. タイプを **zfs** に設定します。

```
zonecfg:s10-zone:fs> set type=zfs
```

b. 大域ゾーンからマウントされるディレクトリを設定します。

```
zonecfg:s10-zone:fs> set special=share/zone/s10-zone
```

c. マウントポイントを指定します。

```
zonecfg:s10-zone:fs> set dir=/opt/shared
```

d. 指定を終了します。

```
zonecfg:s10-zone:fs> end
```

この手順を複数回実行することで、複数のファイルシステムを追加できます。

8 ストレージプール *tank* の *sales* という **ZFS** データセットを委任します。

```
zonecfg:my-zone> add dataset
```

a. **ZFS** データセット *sales* のパスを指定します。

```
zonecfg:my-zone> set name=tank/sales
```

b. データセットの指定を終了します。

```
zonecfg:my-zone> end
```

9 **hostid** としてソースシステムの **hostid** を使用するように設定します。

```
zonecfg:my-zone> set hostid=80f0c086
```

10 ゾーンの構成を検証します。

```
zonecfg:s10-zone> verify
```

11 ゾーンの構成を確定します。

```
zonecfg:s10-zone> commit
```

12 **zonecfg** コマンドを終了します。

```
zonecfg:s10-zone> exit
```

プロンプトで `commit` コマンドを明示的に入力しなくても、`exit` を入力するか EOF が発生すると、`commit` の実行が自動的に試みられます。

13 **info** サブコマンドを使用して、ブランドが **solaris10** に設定されていることを確認します。

```
global# zonecfg -z s10-zone info
```

14 (オプション) **info** サブコマンドを使用して、**hostid** を確認します。

```
global# zonecfg -z s10-zone info hostid
```

次の手順

ヒント – ゾーンを構成したあとは、ゾーンの構成のコピーを作成することをお勧めします。このバックアップを使用して、あとでゾーンを再作成できます。適切なプロファイルを持つ root または管理者として、ゾーン *s10-zone* の構成をファイルに出力します。次の例では、`s10-zone.config` というファイルを使用しています。

```
global# zonecfg -z s10-zone export > s10-zone.config
```

参照 `zonecfg` を使用して構成可能な追加コンポーネントについては、第 16 章「非大域ゾーンの構成 (概要)」を参照してください。このガイドには、`zonecfg` コマンドをコマンド行モードまたはコマンドファイルモードで使用する方法も説明されています。共有 IP ゾーンの場合は、`zonecfg net` リソースに静的アドレスを割り当てる必要があります。ZFS ファイルシステムを追加する方法については、『Oracle Solaris 11.1 の管理: ZFS ファイルシステム』の「ZFS ファイルシステムを非大域ゾーンに追加する」を参照してください。

◆ ◆ ◆ 第 33 章

33 solaris10 ブランドゾーンのインストール

この章では、solaris10 ブランドゾーンのインストールについて説明します。

ゾーンのインストールイメージ

システムイメージの種類

- ゾーンで実行されるすべてのソフトウェアとともに、完全に構成された Oracle Solaris システムのイメージを使用できます。437 ページの「Oracle Solaris 10 システムをゾーンに直接移行するためのイメージの作成」を参照してください。zoneadm install -a コマンドは、物理システムのアーカイブを取得します。
- 物理システムのイメージの代わりに、既存の Oracle Solaris 10 のネイティブゾーンのイメージを使用できます。第 31 章「(オプション) Oracle Solaris 10 ゾーンへの Oracle Solaris 10 native 非大域ゾーンの移行」を参照してください。zoneadm install -a コマンドはゾーンのアーカイブまたは物理システムのアーカイブを取得し、zoneadm attach -a コマンドはゾーンのアーカイブを取得します。

イメージ sysidcfg のステータス

インストール完了後のゾーンの構成で使用するために sysidcfg ファイルを渡すには、-c を使用できます。

既存のシステムから Oracle Solaris 10 システムアーカイブを作成していて、ゾーンをインストールするときに -p (sysidcfg を維持する) オプションを使用する場合は、イメージの作成に使用されたシステムと同じ ID がゾーンに割り当てられます。

-u (sys-unconfig) および -c オプションを使用してターゲットゾーンをインストールすると、ホスト名やネームサービスの構成されていないゾーンが作成されます。

solaris10 ブランドゾーンのインストール

zoneadm コマンド (パート II「Oracle Solaris ゾーン」および zoneadm(1M) のマニュアルページに記載) は、非大域ゾーンをインストールおよび管理するための主要なツールです。zoneadm を使用する操作は、ターゲットシステムの大域ゾーンから実行する必要があります。

インストールプロセスは、アーカイブからファイルを展開するほかに、ゾーンがホスト上で最適に実行されることを保証するために、検査や必要な後処理などの機能を実行します。

既存のシステムから Oracle Solaris システムアーカイブを作成した場合は、-p (sysidcfg を維持する) オプションを使用してゾーンをインストールすると、イメージの作成に使用されたシステムと同じ ID がゾーンに設定されます。

-u (sys-unconfig) オプションを使用してターゲットゾーンをインストールすると、ホスト名やネームサービスの構成されていないゾーンが作成されます。

注意 – *p* オプションまたは -u オプションのどちらかを使用する必要があります。-これら 2 つのオプションの 1 つを指定しないとエラーになります。

インストーラオプション

オプション	説明
-a	システムイメージのコピー元となるアーカイブの場所。完全なフラッシュアーカイブと pax、cpio、gzip で圧縮された cpio、bzip で圧縮された cpio、およびレベル 0 ufsdump がサポートされています。
-c *path*	インストール完了後のゾーンの構成に使用するため、sysidcfg ファイルを渡します。
-d *path*	システムイメージのコピー元となるディレクトリの場所。
-d –	-d オプションとダッシュ (-) パラメータを使用して、zonepath で既存のディレクトリレイアウトが使用されるように指定します。このため、インストールの前に管理者が手動で zonepath ディレクトリを設定する場合に、-d – オプションを使用してそのディレクトリがすでに存在するかどうかを示すことができます。
-p	システム ID を維持します。 -p または -u のいずれかを使用する必要があります。
-s	サイレントインストールします。

オプション	説明
-u	ゾーンに対して sys-unconfig を実行します。 -p または -u のいずれかを使用する必要があります。 インストール完了後のゾーンの構成で使用する sysidcfg ファイルを渡すために、-u オプションに加えて --c を使用できます。
-v	詳細情報を出力します。

-a オプションと -d オプションは相互に排他的です。

▼ solaris10 ブランドゾーンをインストールする方法

構成された solaris10 ブランドゾーンは、install サブコマンドとともに zoneadm コマンドを実行することによりインストールします。

Oracle Solaris 10 システムのイメージを作成する方法については、437 ページの「Oracle Solaris 10 システムをゾーンに直接移行するためのイメージの作成」を参照してください。作成したシステムイメージを変更することなく、そのイメージの sysidcfg ID を維持するには、install サブコマンドのあとに -p オプションを使用します。作成したシステムイメージを変更することなく、そのイメージのシステム ID を削除するには、-u オプションを使用します。ターゲットゾーンに対して sys-unconfig が実行されます。インストール完了後のゾーンの構成で使用する情報を含む sysidcfg ファイルを取り込むために、-c オプションを使用できます。

この手順例では、物理的にインストールされた Oracle Solaris 10 システムで作成されたアーカイブイメージに -a オプションを使用する方法を示します。

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を持つユーザーである必要があります。

1. **root** になるか、同等の役割になります。
2. **zoneadm install** コマンドに **-p** オプション、**-a** オプション、およびアーカイブのパスを指定して、構成済みゾーン **s10-zone** をインストールします。

```
global# zoneadm -z s10-zone install -p -a /net/machine_name/s10-system.flar -u
```

インストールの完了につれてさまざまなメッセージが表示されます。これにはしばらく時間がかかることがあります。

3 (オプション)エラーメッセージが表示され、ゾーンのインストールが失敗した場合は、**zoneadm list** コマンドに **-c** および **-v** オプションを付けて実行して、ゾーンの状態を取得します。

```
global# zoneadm list -civ

ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris    shared
 -  s10-zone configured   /zones/s10-zone                solaris10  shared
```

- 状態が構成済みであると表示された場合は、メッセージに示された修正を行い、zoneadm install コマンドを再度実行します。
- 状態が不完全であると表示された場合は、最初に次のコマンドを実行します。

  ```
  global# zoneadm -z my-zone uninstall
  ```

  次にメッセージに示された修正を行い、zoneadm install コマンドを再度実行します。

4 インストールが完了したら、**list** サブコマンドに **-i** オプションおよび **-v** オプションを指定してインストール済みのゾーンを一覧表示し、ステータスを確認します。

```
global# zoneadm list -iv
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris     shared
 -  s10-zone  installed    /zones/s10-zone               solaris10   shared
```

例 33–1 solaris10 ゾーンのインストール

```
# zoneadm -z s10sepvar install -p -a /net/data13/tmp/s10u10_sparc_sepvar.flar -u
The following ZFS file system(s) have been created:
    rpool/zones/s10sepvar
Progress being logged to /var/log/zones/zoneadm.20120519T151123Z.s10sepvar.install
    Installing: This may take several minutes...
```

注意事項

インストールが失敗した場合は、ログファイルを確認してください。成功した場合、ログファイルはゾーン内の /var/log にあります。失敗した場合、ログファイルは大域ゾーン内の /var/log/zones にあります。

ゾーンのインストールが中断または失敗した場合は、ゾーンの状態は不完全なままになります。uninstall コマンドに -F オプションを付けて実行し、ゾーンを構成済みの状態にリセットします。

◆ ◆ ◆ 第 34 章

34 ゾーンのブート、ログイン、ゾーンの移行

この章では、インストール済みゾーンをブートし、zlogin を使用して内部ゾーン構成を完了する方法について説明します。この章では、ゾーンを別のマシンに移行する方法についても説明します。

solaris10 ブランドゾーンのブートについて

ゾーンをブートすると、ゾーンが稼働状態になります。ゾーンは、準備完了状態またはインストール済み状態からブートできます。ブートしたインストール済み状態のゾーンは、準備完了状態から稼働状態に透過的に移行します。稼働状態のゾーンに対してはゾーンへのログインが可能です。

最初のブート後に未構成のゾーンにはじめてログインするときに、内部ゾーン構成を実行します。

イメージ sysidcfg プロファイル

既存のシステムから Oracle Solaris 10 システムアーカイブを作成していて、ゾーンをインストールするときに -p (sysidcfg を維持する) オプションを使用する場合は、イメージの作成に使用されたシステムと同じ ID がゾーンに割り当てられます。

インストール完了後のゾーンを構成するときに使用する sysidcfg ファイルを取り込むために、-c オプションを使用できます。solaris10 ゾーンをインストールするには、コマンド行で sysidcfg ファイルを使用します。このファイルへのフルパスを指定する必要があります。

```
# zoneadm -z s10-zone install -a /net/machine_name/s10-system.flar -u -c /path_to/sysidcfg
```

次の例の sysidcfg ファイルでは、net0 ネットワーク名と timezone を使用して、静的 IP 構成の排他的 IP ゾーンが構成されます。

```
system_locale=C
 terminal=xterm
 network_interface=net0 {
    hostname=test7
    ip_address=192.168.0.101
    netmask=255.255.255.0
    default_route=NONE
    protocol_ipv6=no
 }
 name_service=NONE
 security_policy=NONE
 timezone=US/Pacific
 timeserver=localhost
 nfs4_domain=dynamic
 root_password=FSPXl81aZ7Vyo
 auto_reg=disable
```

次の例の sysidcfg ファイルは、共有 IP ゾーンを構成するために使用されます。

```
system_locale=C
terminal=dtterm
network_interface=primary {
hostname=my-zone
}
security_policy=NONE
name_service=NIS {
domain_name=special.example.com
name_server=bird(192.168.112.3)
}
nfs4_domain=domain.com
timezone=US/Central
root_password=m4qtoWN
```

次の例の sysidcfg ファイルは、静的 IP 構成の排他的 IP ゾーンを構成するために使用されます。

```
system_locale=C
terminal=dtterm
network_interface=primary {
hostname=my-zone
default_route=10.10.10.1
ip_address=10.10.10.13
netmask=255.255.255.0
}
nfs4_domain=domain.com
timezone=US/Central
root_password=m4qtoWN
```

次の例の sysidcfg ファイルは、DHCP および IPv6 オプションで排他的 IP ゾーンを構成するために使用されます。

```
system_locale=C
terminal=dtterm
network_interface=primary {
dhcp protocol_ipv6=yes
}
```

```
security_policy=NONE
name_service=DNS {
domain_name=example.net
name_server=192.168.224.11,192.168.224.33
}
nfs4_domain=domain.com
timezone=US/Central
root_password=m4qtoWN
```

▼ solaris10 ブランドゾーンの内部構成

プロファイルがない場合は、最初に `zlogin -C` を使用したときに構成ツールが起動します。

この手順でのゾーン名は `s10-zone` です。

1 **root** になるか、同等の役割になります。

2 一方の端末ウィンドウで、ゾーンをブートする前に次のコマンドを使用して、ゾーンコンソール (この手順では **s10-zone**) に接続します。

```
# zlogin -C s10-zone
```

3 2 番目のウィンドウで、461 ページの「**solaris10** ブランドゾーンをブートする方法」の説明に従ってゾーンをブートします。

▼ solaris10 ブランドゾーンをブートする方法

この手順を実行するには、大域管理者または大域ゾーン内で適切な承認を持つユーザーである必要があります。

1 **root** になるか、同等の役割になります。

2 **zoneadm** コマンドを **-z** オプション、ゾーン名 (**s10-zone**)、および **boot** サブコマンドとともに使用することで、ゾーンをブートします。

```
global# zoneadm -z s10-zone boot
```

3 ブートが完了したら、**list** サブコマンドに **-v** オプションを指定してステータスを確認します。

```
global# zoneadm list -v
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris     shared
 1  s10-zone running      /zone/s10-zone                 solaris10   shared
```

参照　ゾーンのブートおよびブートオプションの詳細は、第 19 章「非大域ゾーンのインストール、ブート、停止処理、停止、アンインストール、およびクローニング (タスク)」を参照してください。

別のホストへの **solaris10** ブランドゾーンの移行

`solaris10` ゾーンは、`zoneadm` コマンドに `detach` および `attach` サブコマンドを指定して実行することにより、別のホストに移行できます。この手順については、339 ページの「ゾーンの移行について」および340 ページの「ZFS アーカイブを使用して非大域ゾーンを移行する方法」で説明しています。

`zoneadm attach -a` コマンドは、ゾーンのアーカイブを取得し、物理システムのアーカイブは取得しません。

用語集

blessed	Perl では、この用語はオブジェクトのクラスのメンバーシップを示すために使用されます。
cap	システムリソース使用率に対する規制。
FSS	公平配分スケジューラを参照してください。
Oracle Solaris 10 ゾーン	Oracle Solaris 11 リリースを実行しているシステムの `solaris10` ブランドゾーンで実行される Solaris 10 アプリケーション用の完全な実行時環境。
Oracle Solaris ゾーン	オペレーティングシステムサービスを仮想化し、アプリケーションを実行するための分離およびセキュリティー保護された環境を提供するソフトウェア区分技術。
RSS	常駐セットサイズを参照してください。
WSS	作業セットサイズを参照してください。
アカウンティングの拡張	Solaris オペレーティングシステムで、タスクまたはプロセスに基づくリソース消費量を柔軟に記録できる方法。
完全ルートゾーン	必要なすべてのシステムソフトウェアおよび追加パッケージがゾーン固有のファイルシステムにインストールされる非大域ゾーンのタイプ。
局所有効範囲	制御値を超えようとしているプロセスに対して行われる局所アクション。
公平配分スケジューラ	公平さを基準に CPU 時間を割り当てるスケジューリングクラス (FSS とも呼ぶ)。配分は、システムの CPU リソースのうちプロジェクトに割り当てる部分を定義します。
作業セットサイズ	作業セットのサイズ。作業セットとは、プロジェクトの作業負荷がその処理サイクル中にアクティブに使用するページのことです。
作業負荷	アプリケーションまたはアプリケーショングループのプロセスすべての合計。
上限制御	システムリソース使用率を規制するプロセス。
常駐セットサイズ	常駐セットのサイズ。常駐セットとは、物理メモリーに常駐するページのことです。
スキャナ	あまり使用されていないページを識別するカーネルスレッド。メモリーが少ない状況では、スキャナは最近使用されていないページを再生します。
静的プール構成	管理者が、リソースプール機能に関してシステムを構成する方法を表現したもの。
ゾーン管理者	ゾーン管理者の特権は、非大域ゾーンに対してのみ有効です。

	大域管理者も参照してください。
ゾーン状態	非大域ゾーンのステータス。ゾーンの状態は、構成済み、不完全、インストール済み、準備完了、使用不可、稼働、または停止処理のいずれかになります。
大域管理者	root ユーザーまたは root 役割を持つ管理者大域ゾーンにログインすると、大域管理者または適切な承認が付与されたユーザーは、システム全体を監視したり制御したりできます。 ゾーン管理者も参照してください。
大域ゾーン	すべての Oracle Solaris システムに含まれるゾーン。非大域ゾーンを使用しているときには、大域ゾーンはシステムのデフォルトゾーンであると同時に、システム規模の管理制御に使用されるゾーンでもあります。 非大域ゾーンも参照してください。
大域有効範囲	システム上のすべてのリソース制御のリソース制御値に適用されるアクション。
タスク	リソース管理において、長時間にわたる作業の集合を表すプロセスの集まり。各タスクは 1 つのプロジェクトに関連付けられます。
データリンク	OSI プロトコルスタックのレイヤー 2 インタフェース。システム内で STREAMS DLPI (v2) インタフェースとして表されます。このインタフェースは、TCP/IP などのプロトコルスタックで接続できます。Oracle Solaris 10 ゾーンでは、データリンクは物理インタフェース、アグリゲーション、または VLAN でタグ付けされたインタフェースです。データリンクは、物理インタフェースとして参照されることもあります (たとえば、NIC または VNIC を指す場合)。
デフォルトプール	プールが有効に設定される際にシステムにより作成されるプール。 リソースプールも参照してください。
デフォルトプロセッサセット	プールが有効に設定される際にシステムにより作成されるプロセッサセット。 プロセッサセットも参照してください。
動的構成	ある時点における、指定されたシステムのリソースプールフレームワーク内部のリソース配置に関する情報。
動的再構成	SPARC ベースのシステムで、システムの稼働中にハードウェアを再構成する機能。DR とも呼ばれます。
ネームサービスデータベース	このドキュメントの「プロジェクトとタスク (概要)」の章では、LDAP コンテナと NIS マップの両方を指して使用されます。
ヒープ	プロセス内で割り当てられたスクラッチメモリー。
非大域ゾーン	Oracle Solaris オペレーティングシステムの単一インスタンス内に作成された仮想オペレーティングシステム環境。Oracle Solaris ゾーンソフトウェア区分技術を使用して、オペレーティングシステムサービスが仮想化されます。

非大域ゾーン管理者	ゾーン管理者を参照してください。
プール	リソースプールを参照してください。
プールデーモン	リソースの動的割り当てが必要な場合にアクティブになる `poold` システムデーモン。
ブランド	BrandZ 機能のインスタンス。アプリケーションの実行に使用される、ネイティブでないオペレーティング環境が含まれている非大域ゾーンを提供します。
ブランドゾーン	ネイティブでないアプリケーションを非大域ゾーンで実行するための隔離された環境。
プロジェクト	ネットワーク全体の関連作業に対する管理識別子。
プロセッサセット	互いに素である CPU のグループ。各プロセッサセットには、0 以上のプロセッサを含めることができます。プロセッサセットは、リソースプール構成内でリソース要素として表されます。これは pset とも呼ばれます。 素も参照してください。
メモリー上限実行しきい値	リソース上限デーモンが上限を制限するときのシステム上における物理メモリーの使用効率 (パーセンテージ)。
素	セットのメンバーが重複しないセットのタイプ。
読み取り専用ゾーン	読み取り専用ルートを持つように構成された不変ゾーン。
リソース	アプリケーションの動作を変更する目的で操作されるコンピューティングシステムの側面。
リソース管理	アプリケーションが利用可能なシステムリソースをどのように使用するかを制御する機能。
リソース上限デーモン	リソース上限が定義されたプロジェクト内で動作するプロセスが消費する物理メモリーを規制するデーモン。
リソース消費者	基本的には Solaris のプロセス。プロジェクトやタスクなどのプロセスモデルエンティティーにより、集計済みのリソース消費に関して考察できます。
リソース制御	プロセスごと、タスクごと、またはプロジェクトごとのリソース消費量に対する制限。
リソースセット	プロセスをバインド可能なリソース。たいていの場合、カーネルサブシステムにより構築され、ある種の区分化を提供するオブジェクトを指して使用されます。リソースセットの例には、スケジューリングクラスやプロセッサセットが含まれます。
リソースパーティション	リソースの排他的な一部。リソースのパーティションすべての合計は、実行中の単一の Solaris インスタンスで利用可能なリソースの総量を表します。
リソースプール	マシンのリソース区分化に使用する構成メカニズム。リソースプールは、区分化可能なリソースグループ間の関係を表します。
ロックされたメモリー	ページング不可能なメモリー。

索引

A

acctadm コマンド，69
ゾーンの admin 承認，219
allowed-addresses, 排他的 IP ゾーン，227
anet リソース，215
autoboot，218

B

bootargs プロパティー，242
BrandZ，200, 427

C

capped-cpu リソース，220, 244
capped-memory，244
capped-memory リソース，221
CPU 配分の構成，108

D

dedicated-cpu リソース，219, 243
defrouter，255
　排他的 IP ゾーン，227
DHCP, 排他的 IP ゾーン，227
DRP，141
dtrace_proc，243, 384, 402
dtrace_user，243, 384, 402

E

/etc/project
　エントリの形式，41
　ファイル，40
/etc/user_attr ファイル，38
exacct ファイル，60

F

flarcreate
　cpio，438
　pax，438
　ZFS ルート，437
　データの除外，438
　デフォルトのイメージ，437
force-zpool-import，292
FSS，「公平配分スケジューラ (FSS)」を参照
fsstat，401, 402
fsstat ユーティリティー，360

H

hostid，231
ゾーン内の hostid プロパティー，439

I

ip-type プロパティー，243
ipkg ゾーン, solaris へのマップ，197
ipkg ゾーン, 変換，200

IPMP, 排他的 IP ゾーン，227
IPoIB，255
IPsec, ゾーン内で使用，383
IP フィルタ, 排他的 IP ゾーン，227
IP ルーティング, 排他的 IP ゾーン，227

L
libexacct ライブラリ，60
limitpriv プロパティー，243
linkmode，252
リムーバブル lofi デバイス，231
lofi デバイス, リムーバブル，231

M
MWAC，417

N
net リソース
　共有 IP ゾーン，226
　排他的 IP ゾーン，227
NFS サーバー，361

O
Oracle Solaris 10 ゾーン，427
　制限，432
　ネットワーク，432
Oracle Solaris 監査, ゾーン内で使用，383
Oracle Solaris クラスタ, ゾーンクラスタ，24
Oracle Solaris リソースマネージャー，24

P
P2V
　flarcreate，437, 438
　zonep2vchk，436
　イメージの作成，437
P2V (続き)
　システム評価，436
P2V でのシステム評価，436
PAM (プラグイン可能認証モジュール), アイデンティティー管理，40
Perl インタフェース，64
pkey，252, 255
pkgupdate 中の autoboot の無効化，218
poold
　cpu-pinned プロパティー，149
　構成可能なコンポーネント，153
　制御範囲，158
　制約，149
　説明，148
　同期制御違反，158
　動的リソース割り当て，141
　非同期制御違反，158
　目標，150
　ログ情報，154
poolstat
　出力形式，160
　使用例，182
　説明，159
pool プロパティー，246
project.cpu-shares，108
project.pool 属性，146
project データベース，40
putacct システムコール，61

R
rcap.max-rss 属性，122
rcapadm コマンド，123
rcapd
　構成，123
　サンプリング間隔，127
　走査間隔，127
rcapd デーモン，121
rcapstat コマンド，127
rctls，75
　「リソース制御」を参照
RDS (Reliable Datagram Sockets)，228
rlimits,「リソース制限」を参照
rootzpool リソース，222

S

scheduling-class プロパティー，243
SMF サービス
　大域ゾーン，211
　非大域ゾーン，211
solaris10 ゾーンのブート，459
solaris10 ネイティブゾーン, 移行，455
solaris10 ブランド，427
　SVR4 パッケージ，429
solaris10 ブランドゾーン，427
　V2V，441
　構成，449, 451
　構成の概要，448
　サポートされるデバイス，447
　接続，442, 462
　定義された特権，448
　ブート手順，459
solaris10 ブランドゾーンの接続，442, 462
solaris10 ブランドのインストール，455
solaris ゾーン, 手動の同期，348
solaris 非大域ゾーン, Oracle Solaris 11.1，197
SVR4 パッケージ, solaris10 ブランドの，429
プロジェクト system,「プロジェクト 0」を参照

V

/var/adm/exacct ディレクトリ，62

Z

ZFS
　クローン，310
　スナップショット，310
　データセット，244
zone, 移行，442
zone.cpu-cap リソース制御，234
大域ゾーンの zone.cpu-shares，284
zone.cpu-shares リソース制御，234
zone.max-locked-memory リソース制御，234
zone.max-lofi リソース制御，234
zone.max-lwps リソース制御，234
zone.max-msg-ids リソース制御，235
zone.max-processes リソース制御，234
zone.max-sem-ids リソース制御，235
zone.max-shm-ids リソース制御，235
zone.max-shm-memory リソース制御，235
zone.max-swap リソース制御，235
zoneadm, mark サブコマンド，303
zoneadmd デーモン，292
zoneadm コマンド，287
zonecfg
　admin 承認，219
　solaris10 ブランドゾーンの処理，448
　一時プール，219
　エンティティー，241
　サブコマンド，238
　操作，218
　大域ゾーンでの，269
　大域ゾーン内，237
　手順，269
　モード，238
　有効範囲，238
　有効範囲、大域，238
　有効範囲、リソース固有，238
zonecfg コマンド，269
zonep2vchk, 移行ツール，335
Oracle Solaris 10 での zonep2vchk の取得，436
zonep2vchk ユーティリティー, Oracle Solaris 10 での取得，436
zonepath, ZFS に自動的に作成，300
zonestat，397
zonestat ユーティリティー，359
zpool リソース，224
zsched プロセス，293

い

移行
　solaris10 ネイティブゾーン，455
　zonep2vchk の使用，335
　システム，333
移行先ゾーン, zonecfg，449
一時プール，219
イメージの作成, P2V，437
インストール, solaris10 ブランド，455

え
エントリの形式, /etc/project ファイル，41

か
拡張アカウンティング
  SMF，62
  概要，59
  課金，60
  起動，68–71
  コマンド，63
  ステータスの表示，69
  ファイル形式，60
拡張アカウンティングステータスの表示，69
拡張アカウンティングの起動，68–71

き
機能, 排他的 IP ゾーン，227
共有 IP ゾーン，226

く
クローン, ZFS，310

こ
構成, rcapd，123
構成可能な特権、ゾーン，232
構成、リソース制御の，77
公平配分スケジューラ (FSS)，104, 220
  project.cpu-shares，104
  構成，117
  とプロセッサセット，110
  配分の定義，104
コマンド
  拡張アカウンティング，63
  公平配分スケジューラ (FSS)，113
  ゾーン，389
  プロジェクトとタスク，45
  リソース制御，91

さ
サーバーの統合，33

し
しきい値、リソース制御，85
システム, 移行，333

す
スケジューリングクラス，112
スナップショット, ZFS，310
スワップ領域の上限，221

せ
制限, Oracle Solaris 10 ゾーン，432

そ
ゾーン
  anet，250
  anet，244
  bootargs プロパティー，242
  capped-cpu，244
  capped-memory，221, 244
  dedicated-cpu，243
  DTrace の実行，384
  ip-type，243
  IPoIB，250
  IPsec，383
  limitpriv，243
  net，244
  NFS サーバー，361
  Oracle Solaris 11.1 の制限と機能，197
  Oracle Solaris 監査，383
  pool，246
  rootzpool，247
  scheduling-class，243
  solaris、更新，349
  solaris、パッケージ，349

ゾーン (続き)
UUID, 302
zonep2vchk, 333
zonep2vchk ツール, 335
zonestat ユーティリティー, 397
アンインストール, 309
移行, 339
移動, 312–313
インストール, 300
監視, 212
機能, 212
共有 IP, 226
クローン, 296, 310
検証, 299
権利、役割、プロファイル, 218
構成, 237
構成可能な特権, 232
構成の概要, 218
サイズの制限, 266
削除, 313
作成, 206
種類別の特性, 205
準備完了状態, 303
状態, 207
状態モデル, 207
使用されるコマンド, 389
使用できないマシンからの移行, 343
シングルユーザーモードでブート, 305
接続時アップグレード, 340
接続時のアップグレード, 442
ゾーン, ゾーン規模のリソース制御, 241
ゾーン
対話型モード, 324
データセット, 244
データの取り込み, 288
データリンクの管理, 410
ディスク形式のサポート, 232
ディスク領域, 265
定義, 196
停止, 294, 307
停止処理, 293, 307
デフォルト以外の, 200
特権, 378
内部構成, 316
ゾーン (続き)
名前の変更, 282
ネットワークアドレス, 267
ネットワーク、共有 IP, 369
ネットワーク、排他的 IP, 371
ノード名, 360
排他的 IP, 227
パッケージ, 348
パッケージの削除, 352
パッケージの追加, 350
非対話型モード, 324
ブート, 304
ブート引数, 294, 305
不変ゾーン, 417
ブランド, 200, 427
プロキシ構成, 352
プロパティータイプ, 241
リスト, 300
リソース制御, 233
リソースタイプ, 241
リソースタイプのプロパティー, 247
リブート, 294, 308
ログインの概要, 315
ゾーン ID, 204
ゾーン管理者, 206
ゾーン管理プロファイル, 414
ゾーン規模のリソース制御, 233
ゾーン構成, スクリプト, 276
ゾーンコンソールログイン, コンソールログインモード, 322
ゾーンサイズの制限, 266
ゾーンでの DTrace の実行, 384, 402
ゾーン内の特権, 378
ゾーンのアンインストール, 309
ゾーンの移行, 339, 442
ゾーンの移動, 312–313
ゾーンのインストール, 299, 300
概要, 287
タスク, 299
ゾーンのクローニング, 310
ゾーンのクローン, 296
ゾーンの検証, 299
ゾーンの構成
タスク, 261

ゾーンのコマンド，389
ゾーンの削除，313
ゾーンの準備完了，303
ゾーンの停止，294, 307
　トラブルシューティング，294
ゾーンの停止処理，293, 307
ゾーンの名前の変更，282
ゾーンのブート，304
ゾーンの並列更新，354
ゾーンのホスト名，266
ゾーンのリスト，300
ゾーンのリブート，294, 308
ゾーンへのデータの取り込み，288
ゾーンへのログイン
　フェイルセーフモード，323
　リモート，323
ゾーン名，204
属性, project.pool，146

た
大域管理者，204, 206
大域ゾーン，204
大域ゾーンのプロキシ，353
大域ゾーンプロキシのオーバーライド，353
タスク, リソース管理，44

て
データリンク，225
データリンクの管理，410
ディスク形式のサポート, ゾーン，232
デフォルト以外の, ゾーン，200
デフォルトプロジェクト，38
デフォルトプロセッサセット，140
デフォルトリソースプール，140

と
動的なプール構成，143
動的リソースプール
　無効化，165
動的リソースプール (続き)
　有効化，165
動的リソースプールの無効化，165
動的リソースプールの有効化，165
特権レベル、しきい値，85

ね
ネットワーク, Oracle Solaris 10 ゾーン，432
ネットワーク、共有 IP，369
ネットワーク、排他的 IP，371

の
ノード名, ゾーン，360

は
排他的 IP ゾーン，227
　anet，215

ひ
非大域ゾーン，204
非大域ゾーン管理者，204

ふ
ブート引数とゾーン，305
プール，140
物理メモリーの上限，221
不変ゾーン，417
　読み取り専用ゾーン，200
プラグイン可能認証モジュール，「PAM」を参照
ブランド，427
ブランドゾーン，200, 427
　実行中のプロセス，201
　デバイスのサポート，447
　特権，447
　ファイルシステムのサポート，447

プロキシ構成, ゾーン, 352
プロキシの構成, 352
プロジェクト
  アイドル状態, 105
  アクティブ状態, 105
  定義, 38
  配分がゼロ, 105
プロジェクト 0, 109
プロセス間通信 (IPC), 「リソース制御」を参照

ほ
ゾーン内の ホスト ID, 439

む
無効化、リソース上限制御, 135

め
メモリー上限実行しきい値, 124

よ
読み取り専用ゾーン, 417
  add dataset ポリシー, 419
  add fs ポリシー, 419
  file-mac-profile, 219, 418
  管理, 419
  構成, 418
  ブート, 420
読み取り専用ゾーンの管理, 419
読み取り専用ゾーンのブート, 420
読み取り専用ゾーンルート, 219, 417, 418

り
リソース管理
  区分, 32
  スケジューリング, 31
リソース管理 (続き)
  制約, 31
  タスク, 44
  定義, 29
リソース上限, 121
リソース上限制御
  無効化, 135
  有効, 134
リソース上限制御を有効にする, 134
リソース上限デーモン, 121
リソース制御
  inf 値, 89
  一時的な変更, 90
  一時的に更新, 90
  一覧, 78
  概要, 75
  局所アクション, 77, 86
  構成, 77
  しきい値, 77, 85, 86
  ゾーン規模の, 233
  大域アクション, 85
  定義, 75
  プロセス間通信 (IPC), 76
リソース制御の一時的な変更, 90
リソース制御を一時的に更新, 90
リソース制限, 76
リソースプール, 140
  /etc/pooladm.conf, 143
  管理, 161
  結合, 180
  構成の起動, 178
  構成の削除, 179
  構成の要素, 144
  削除, 179
  作成, 146
  実装, 145
  静的なプール構成, 143
  動的再構成, 146
  プロパティー, 144
  無効化, 165
  有効化, 165
リソースプール属性の設定, 180
リソースプールの管理, 161
リソースプールの削除, 179

リソースプールの作成，146
リソースプールの実装，145
リソースプールの無効化，165
リソースプールの有効化，165
リソースプールへの結合，180
リモートゾーンへのログイン，323

ろ

ログイン, リモートゾーン，323
ロックされたメモリーの上限，222