Solaris ボリュームマネージャの管理

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 819–0375–12
2006 年 5 月

原典: Solaris Volume Manager Administration Guide

Part No: 816-4520-12

Revision A

060623@14986

目次

表目次

図目次

はじめに

『Solaris ボリュームマネージャの管理』では、 Solaris™ ボリュームマネージャを使用してシステムに必要な記憶装置を管理する方法について説明します。Solaris ボリュームマネージャを使用すると、RAID-0 (連結方式とストライプ方式) ボリューム、RAID-1 (ミラー) ボリューム、 RAID-5 ボリューム、およびソフトパーティションを作成、変更、および使用できます。

注 – このリリースでは、SPARC® および x86 系列のプロセッサアーキテクチャー (UltraSPARC®、SPARC64、AMD64、Pentium、Xeon EM64T) を使用するシステムをサポートします。サポートされるシステムについては、Solaris 10 Hardware Compatibility List (`http://www.sun.com/bigadmin/hcl`) を参照してください。本書では、プラットフォームにより実装が異なる場合は、それを特記します。

本書の x86 に関連する用語については、以下を参照してください。

- 「x86」は、64 ビットおよび 32 ビットの x86 互換製品系列を指します。
- 「x64」は、AMD64 または EM64T システムに関する 64 ビット特有の情報を指します。
- 「32 ビット x86」は、x86 をベースとするシステムに関する 32 ビット特有の情報を指します。

サポートされるシステムについては、Solaris 10 Hardware Compatibility List を参照してください。

対象読者

システムおよび記憶装置の管理者は、本書で下記の内容を学ぶことができます。

- Solaris ボリュームマネージャがサポートする作業
- Solaris ボリュームマネージャを使用して、より信頼性が高くアクセスしやすいデータを提供する方法

内容の紹介

本書は、次のように構成されています。

第 1 章では、本書で説明する概念や作業の詳細な「ロードマップ」を示します。この章によって、このマニュアルの内容全体を理解できます。

第 2 章では、記憶装置の管理にまだ精通していない読者のために、その一般的な概念を紹介します。

第 3 章では、Solaris ボリュームマネージャについて説明します。この章では、この製品に関連する重要な概念とともに、Solaris ボリュームマネージャの各種ツールの利用方法について説明します。

第 4 章では、複数所有者ディスクセット について紹介します。複数所有者ディスクセットにより、Sun™ Cluster 環境で Solaris ボリュームマネージャを拡張することができます。

第 5 章では、このマニュアル全体で使用する記憶装置の構成例を示します。これは、Solaris ボリュームマネージャ 製品を理解しやすくするための事例です。

第 6 章では、状態データベースと状態データベースの複製に関連する概念について説明します。

第 7 章では、状態データベースと状態データベースの複製に関連する作業の実行方法について説明します。

第 8 章では、RAID-0 (ストライプ方式と連結方式) ボリュームに関連する概念について説明します。

第 9 章では、RAID-0 (ストライプ方式と連結方式) ボリュームに関連する作業の実行方法について説明します。

第 10 章では、RAID-1 (ミラー) ボリュームに関連する概念について説明します。

第 11 章では、RAID-1 (ミラー) ボリュームに関連する作業の実行方法について説明します。

第 12 章では、Solaris ボリュームマネージャのソフトパーティション機能に関連する概念について説明します。

第 13 章では、ソフトパーティションに関連する作業の実行方法について説明します。

第 14 章では、RAID-5 ボリュームに関連する概念について説明します。

第 15 章では、RAID-5 ボリュームに関連する作業の実行方法について説明します。

第 16 章では、ホットスペアとホットスペア集合に関連する概念について説明します。

第 17 章では、ホットスペアとホットスペア集合に関連する作業の実行方法について説明します。

第 18 章では、ディスクセットに関連する概念について説明します。

第 19 章では、ディスクセットに関連する作業の実行方法について説明します。

第 20 章では、特定の Solaris ボリュームマネージャ コンポーネントに関連しない、一般的な保守作業について説明します。

第 21 章では、Solaris ボリュームマネージャを使用して構築できる最善の記憶装置構成について説明します。

第 23 章では、Solaris ボリュームマネージャの自動 (トップダウン) ボリューム作成機能に関連する作業の概念について説明します。

第 24 章では、Solaris ボリュームマネージャ SNMP エージェントやその他のエラーチェック手法の概念と使用手順について説明します。

第 25 章では、Solaris ボリュームマネージャ環境におけるトラブルシューティングと一般的な問題の解決方法を示します。

付録 A では、Solaris ボリュームマネージャの重要なファイルの一覧を示します。

付録 B では、コマンドとその他の有用な情報を要約した表を示します。

付録 C では、WBEM 準拠の管理ツールからオープンな Solaris ボリュームマネージャを利用するための CIM/WBEM API について簡単に紹介します。

関連情報

Solaris ボリュームマネージャは、Solaris オペレーティングシステムで使用できるシステム管理ツールの 1 つです。システム管理の全体的な特徴や機能、および関連するツールについては、次のマニュアルを参照してください。

- 『Solaris のシステム管理 (基本編)』
- 『Solaris のシステム管理 (上級編)』
- 『Solaris のシステム管理 (デバイスとファイルシステム)』

マニュアル、サポート、およびトレーニング

Sun の Web サイトでは、次のサービスに関する情報も提供しています。

- マニュアル (`http://jp.sun.com/documentation/`)
- サポート (`http://jp.sun.com/support/`)
- トレーニング (`http://jp.sun.com/training/`)

表記上の規則

このマニュアルでは、次のような字体や記号を特別な意味を持つものとして使用します。

表 P–1 表記上の規則

字体または記号	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。 system%
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。	system% **su** password:
AaBbCc123	変数を示します。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。
『』	参照する書名を示します。	『コードマネージャ・ユーザーズガイド』を参照してください。
「」	参照する章、節、ボタンやメニュー名、強調する単語を示します。	第 5 章「衝突の回避」を参照してください。 この操作ができるのは、「スーパーユーザー」だけです。
\	枠で囲まれたコード例で、テキストがページ行幅を超える場合に、継続を示します。	sun% **grep '^#define ** **XV_VERSION_STRING'**

コード例は次のように表示されます。

- C シェル

```
machine_name% command y|n [filename]
```

- C シェルのスーパーユーザー

```
machine_name# command y|n [filename]
```

- Bourne シェルおよび Korn シェル

```
$ command y|n [filename]
```

- Bourne シェルおよび Korn シェルのスーパーユーザー

```
# command y|n [filename]
```

[ ] は省略可能な項目を示します。上記の例は、*filename* は省略してもよいことを示しています。

| は区切り文字 (セパレータ) です。この文字で分割されている引数のうち 1 つだけを指定します。

キーボードのキー名は英文で、頭文字を大文字で示します (例: Shift キーを押します)。ただし、キーボードによっては Enter キーが Return キーの動作をします。

ダッシュ (-) は 2 つのキーを同時に押すことを示します。たとえば、Ctrl-D は Control キーを押したまま D キーを押すことを意味します。

一般規則

- このマニュアルでは、英語環境での画面イメージを使っています。このため、実際に日本語環境で表示される画面イメージとこのマニュアルで使っている画面イメージが異なる場合があります。本文中で画面イメージを説明する場合には、日本語のメニュー、ボタン名などの項目名と英語の項目名が、適宜併記されています。

◆ ◆ ◆ 第 1 章

1 Solaris ボリュームマネージャで行う作業の概要

『Solaris ボリュームマネージャの管理』では、高度な可用性、柔軟性、および信頼性を備えた記憶装置を構成するために、Solaris ボリュームマネージャを使用してシステムを設定および管理する方法を説明します。

この章は、記憶容量の設定など、Solaris ボリュームマネージャの特定の作業に関する情報を見つけるためのガイドとして役立ちます。この章では、Solaris ボリュームマネージャを使用する必要があるすべての作業について説明しているわけではありません。新機能の概要を紹介し、Solaris ボリュームマネージャの概念に関連した一般的な作業手順の参照先を示します。

この章では、次のようなロードマップを示します。

注意 – Solaris ボリュームマネージャの使用法が正しくないと、データを破壊してしまうおそれがあります。Solaris ボリュームマネージャは、ディスクとディスク上のデータを確実に管理するための強力な手段です。しかし、データのバックアップは、常に取るようにしてください。特に、Solaris ボリュームマネージャの実際の構成を変更するときには、バックアップが必要です。

Solaris ボリュームマネージャのロードマップ—新機能

作業	説明	参照先
1 つまたは複数のコンポーネントが 1T バイトを超える記憶領域の管理	1T バイトを超えるサイズの物理論理番号 (LUN) を使用するか、1T バイトを超える論理ボリュームを作成します。	49 ページの「Solaris ボリュームマネージャにおけるマルチテラバイトサポートの概要」
あるシステムから別のシステムにディスクセットをインポートする	`metaimport` コマンドを使用して、ディスクセットをインポートします。ほかのシステムで作成されたディスクセットもインポートできます。このコマンドは、拡張されたデバイス ID サポートを使用して、名前付きディスクセット内におけるディスクの移動を自動的に追跡します。	208 ページの「ディスクセットのインポート」 213 ページの「ディスクセット内で非同期的に共有される記憶領域」
複数所有者ディスクセットを作成して管理する	`metaset -M` を使用して、Sun Cluster 環境の複数所有者ディスクセットを管理します。	56 ページの「複数所有者ディスクセットに関連する作業」

Solaris ボリュームマネージャのロードマップ— 記憶容量

作業	説明	参照先
記憶領域を設定します	RAID-0 または RAID-5 ボリュームを作成することによって、複数のスライスにまたがる記憶領域を作成します。この RAID-0 または RAID-5 ボリュームは、ファイルシステムや raw デバイスにアクセスする任意のアプリケーション (データベースなど) に使用できます。	88 ページの「RAID-0 (ストライプ方式) ボリュームを作成するには」 90 ページの「RAID-0 (連結方式) ボリュームを作成するには」 111 ページの「未使用のスライスから RAID-1 ボリュームを作成するには」 113 ページの「ファイルシステムから RAID-1 ボリュームを作成するには」 176 ページの「RAID-5 ボリュームを作成するには」

作業	説明	参照先
既存のファイルシステムを拡張します	RAID-0 (連結方式) ボリュームを作成して、そのボリュームにスライスを追加することによって、既存のファイルシステムの容量を拡張します。	91 ページの「既存のデータの記憶容量を拡張するには」
既存の RAID-0 (連結方式またはストライプ方式) ボリュームを拡張します	既存の RAID-0 ボリュームにスライスを連結することによって、RAID-0 ボリュームを拡張します。	93 ページの「既存の RAID-0 ボリュームを拡張するには」
RAID-5 ボリュームを拡張します	既存の RAID-5 ボリュームにスライスを連結することによって、RAID-5 ボリュームの容量を拡張します。	178 ページの「RAID-5 ボリュームを拡張するには」
拡張されたボリューム上の UFS ファイルシステムを拡張します	`growfs` コマンドを使って UFS のサイズを拡張することによって、ファイルシステムを拡張します。この作業は、UFS ファイルシステムをマウントしたまま実行できるので、データへのアクセスを中断する必要はありません。	245 ページの「ファイルシステムを拡張するには」
スライスまたは論理ボリュームをより小さいパーティションに分割し、8 スライスというハードパーティション制限を取り除きます	ソフトパーティションを使用して論理ボリュームまたはスライスをさらに分割します。	160 ページの「ソフトパーティションを作成するには」
ファイルシステムを作成します	RAID-0 (ストライプ方式および連結方式)、RAID-1 (ミラー)、RAID-5、またはソフトパーティション上でファイルシステムを作成します。	『Solaris のシステム管理 (デバイスとファイルシステム)』の第 18 章「UFS、TMPFS、LOFS ファイルシステムの作成 (手順)」

Solaris ボリュームマネージャのロードマップ — 可用性

作業	説明	参照先
データの可用性を最大限強化	Solaris ボリュームマネージャのミラー化機能を使ってデータの複数のコピーを保持します。データの作成に先立って未使用のスライスから RAID-1 ボリュームを作成できます。あるいは、root (/) や /usr など、既存のファイルシステムをミラー化できます。	111 ページの「未使用のスライスから RAID-1 ボリュームを作成するには」 113 ページの「ファイルシステムから RAID-1 ボリュームを作成するには」
最小限のハードウェアコストでデータの可用性を強化	Solaris ボリュームマネージャの RAID-5 ボリュームを使用することによって、最小限のハードウェアでデータの可用性を高めます。	176 ページの「RAID-5 ボリュームを作成するには」
既存の RAID-1 または RAID-5 ボリュームのデータ可用性の向上	ホットスペア集合を作成して RAID-1 ボリュームまたは RAID-5 ボリュームのサブミラーと対応付けることによって、RAID-1 または RAID-5 ボリュームのデータ可用性を高めます。	192 ページの「ホットスペア集合の作成」 195 ページの「ホットスペア集合とボリュームの対応付け」

Solaris ボリュームマネージャのロードマップ — 入出力性能

作業	説明	参照先
RAID-1 ボリュームの読み書きポリシーの調整	所定の構成で入出力性能が向上するように、RAID-1 ボリュームの読み書きポリシーを指定します。	104 ページの「RAID-1 ボリュームの読み取りおよび書き込みポリシー」 140 ページの「RAID-1 ボリュームオプションを変更するには」

作業	説明	参照先
デバイス性能の最適化	RAID-0 (ストライプ) ボリュームを作成して、ストライプを構成するデバイスの入出力性能を最適化します。飛び越し値は、ランダムアクセスまたは順次アクセス用に最適化できます。	88 ページの「RAID-0 (ストライプ方式) ボリュームの作成」
RAID-0 (ストライプ方式) ボリュームにおけるデバイス性能の維持	領域が足りなくなったストライプや連結に新しいコンポーネントを連結することによって、ストライプや連結を拡張します。ストライプの連結の方がスライスの連結より、入出力性能が向上します。	91 ページの「記憶容量の拡張」

Solaris ボリュームマネージャのロードマップ—管理

作業	説明	参照先
ボリューム管理構成のグラフィカル管理	Solaris 管理コンソールのグラフィカルユーザーインタフェース (GUI) を使用して、ボリューム管理構成を管理します。	Solaris 管理コンソールアプリケーションの Solaris ボリュームマネージャ (拡張ストレージ) ノード内から使用するオンラインヘルプ
スライスやファイルシステムのグラフィカル管理	Solaris 管理コンソールの GUI を使って、ディスクやファイルシステムを管理します。ここでは、ディスクのパーティション分割や UFS ファイルシステムの構築といった作業を行います。	Solaris 管理コンソールアプリケーション内から使用するオンラインヘルプ
Solaris ボリュームマネージャの最適化	Solaris ボリュームマネージャの性能は、構成が適切に設定されているかどうかによって決まります。構成を作成したら、その構成を監視および調整する必要があります。	47 ページの「Solaris ボリュームマネージャ構成の指針」 241 ページの「構成ファイルの使用」
将来の拡張に対する備え	ファイルシステムの領域を使い切ってしまうことがあるため、ファイルシステムを連結で構成することによって将来の拡張に備えることができます。	90 ページの「RAID-0 (連結方式) ボリューム」 91 ページの「記憶容量の拡張」

Solaris ボリュームマネージャのロードマップ — トラブルシューティング

作業	説明	参照先
不良スライスの交換	ディスクに障害が発生したときは、Solaris ボリュームマネージャ構成で使用されているスライスを交換する必要があります。RAID-0 ボリュームの場合、新しいスライスを使用し、ボリュームを削除して作成し直したうえで、バックアップからデータを復元する必要があります。RAID-1 と RAID-5 ボリュームの場合は、スライスの交換と再同期を行なってもデータが失われることはありません。	142 ページの「RAID-1 ボリュームのコンポーネント障害に対する処置」 180 ページの「RAID-5 ボリューム内のコンポーネントを置き換えるには」
ブート障害からの回復	ハードウェア障害やオペレータの操作ミスによって、システムのブート時に特殊な問題が起ることがあります。	301 ページの「`/etc/vfstab` 内の不適切なエントリを修正するには」 307 ページの「状態データベースの複製数の不足から回復するには」 303 ページの「ブートデバイスの障害から回復するには」

◆ ◆ ◆ 第 2 章

記憶装置管理の概念

この章では、一般的な記憶装置管理の概念について簡単に紹介します。

この章では、次の内容について説明します。

記憶装置管理の紹介

記憶装置をどのように管理するかによって、システム上のアクティブデータが格納されるデバイスの制御方法が決まります。ハードウェア障害やソフトウェア障害などの予期せぬ事態が発生したあとでも、アクティブなデータを以前と同じ状態で利用できるようにする必要があります。

記憶装置ハードウェア

データを格納するには、さまざまなデバイスを利用できます。記憶装置の要件にもっとも適したデバイスは、主に次の 3 つの要素を考慮して決定します。

- 性能
- 可用性
- コスト

Solaris ボリュームマネージャを使用すると、性能、可用性、コストのバランスを管理できます。つまり、Solaris ボリュームマネージャを使用することで、多くの場合に諸条件を考慮する必要性が減ります。

Solaris ボリュームマネージャは、Solaris オペレーティングシステムが稼働するシステム上でサポートされているすべての記憶装置に対して使用できます。

RAID レベル

RAID は Redundant Array of Inexpensive (または Independent) Disks の略語です。RAIDとは、ユーザーからは 1 つの大きなディスクドライブに見えるディスク群 (アレイまたはボリュームと呼ばれる) を意味します。構成に応じて、このアレイは信頼性、応答時間、または記憶容量の向上をもたらします。

技術的には、6 つの RAID レベル (0 から 5) があります。各レベルは、データの冗長性を確保しながらデータの分散を図る方法に対応しています。(RAID レベル 0 は、データの冗長性を提供しませんが、通常は RAID の分類に含まれます。RAID レベル 0 は、現在使用されているほとんどの RAID 構成の基礎になっています。)RAID レベル 2、3、4 をサポートしている記憶装置環境はほとんど存在しないため、それらの環境の説明は省略します。

Solaris ボリュームマネージャは、次の RAID レベルをサポートします。

- **RAID** レベル **0** – ストライプと連結は冗長性を備えていませんが、通常、これらのボリュームは RAID-0 と呼ばれています。基本的に、データは、複数の物理ディスクに交互かつ均一に割り当てられる比較的小さな同じ大きさの領域に分散されます。ドライブの 1 つに障害が発生しただけで、データは失われます。RAID-0 では高いデータ転送速度や入出力スループットが得られますが、信頼性や可用性は単一ディスクよりも劣ります。
- **RAID** レベル **1** – ミラー化では、同じ容量の複数のディスクにデータとそのコピー (ミラー) を別々に格納します。データは、2 つ以上の物理ディスクに複製 (またはミラー化) されます。両方のドライブから同時にデータを読み取ることができる、つまり、どちらのドライブも要求に応えることができるため、性能が向上します。1 つの物理ディスクに障害が発生しても、性能の低下やデータの損失なしにミラーを引き続き使用できます。

  Solaris ボリュームマネージャは、使用するボリュームによっては、RAID-0+1 ミラー化と (透過的に) RAID-1+0 ミラー化の両方をサポートします。詳細については、99 ページの「RAID-1+0 と RAID-0+1 の提供」を参照してください。
- **RAID** レベル **5** – RAID-5 は、ストライプ化によってアレイ内の複数のディスクにデータを分散します。また RAID-5 では、パリティー情報を格納することによって、データの冗長性を確保します。RAID-5 ボリュームは、使用している装置で障害が発生しても、耐えることができます。RAID-5 をホットスペアと組み合わせて使用すると、ボリュームは複数の障害に耐えられるようになります。RAID-5 ボリュームは、障害の発生した装置で動作させると、性能が著しく低下します。

  RAID-5 モデルの各デバイスには、パリティーストライプを含む 1 つの領域と、データを含むそれ以外の領域があります。パリティーはアレイ内のすべてのディスクに分散されるため、書き込み時間が短縮されます。書き込み時間が短縮されるのは、パリティー専用ディスクがデータを受け入れることができるようになるまで待機する必要がないためです。

構成計画の指針

記憶装置管理の構成を計画するときには、どのような構成であっても、性能、可用性、ハードウェアコストの間にトレードオフがあることを念頭に置いてください。最適な構成を見つけるためには、さまざまな要素の組み合わせを試してみる必要があります。

この節では、次のタイプのボリュームを使用するための指針を示します。

- RAID-0 (連結方式とストライプ方式) ボリューム
- RAID-1 (ミラー) ボリューム
- RAID-5 ボリューム
- ソフトパーティション
- Solaris ボリュームマネージャのボリューム上に構築されるファイルシステム

記憶装置の選択

特定の記憶方式を実装する前に、どのような記憶デバイスを使用するかを決める必要があります。この一連の指針では、選択が容易になるように、各種の記憶装置を比較します。Solaris ボリュームマネージャで実装する場合、記憶装置のタイプごとに適用される指針がほかにもあります。詳細は、個々のボリュームタイプに関する章を参照してください。

注 – 下記の記憶装置のタイプは、排他的ではありません。つまり、これらのボリュームを組み合わせて使用することによって、複数の目的を達成することができます。たとえば、まず、RAID-1 ボリュームを作成して冗長性を確保します。次に、RAID-1 ボリューム上にソフトパーティションを作成すると、独立したファイルシステムの数を増やすことができます。

次の表で、記憶装置のタイプ別に使用できる機能を比較します。

表 2–1 記憶装置のタイプ別比較

要件	RAID-0 (連結)	RAID-0 (ストライプ)	RAID-1 (ミラー)	RAID-5	ソフトパーティション
データの冗長性	不可	不可	可能	可能	不可
読み取り性能の改善	不可	可能	使用するデバイスによって異なる	可能	不可
書き込み性能の改善	不可	可能	不可	不可	不可

表 2–1 記憶装置のタイプ別比較　(続き)

要件	RAID-0 (連結)	RAID-0 (ストライプ)	RAID-1 (ミラー)	RAID-5	ソフトパーティション
デバイス当たり 8 個以上のスライスの作成	不可	不可	不可	不可	可能
使用可能な記憶装置の増加	可能	可能	不可	可能	不可

次の表に、RAID-1 と RAID-5 のボリュームについて、書き込み操作、ランダムな読み取り、ハードウェアコストのトレードオフの概略を示します。

表 2–2 冗長性の最適化

	RAID-1 (ミラー)	RAID-5
書き込み操作	高速	低速
ランダム読み込み	高速	低速
ハードウェアコスト	高い	少ない

表の情報について簡単に説明します。

- RAID-0 ボリューム (ストライプ方式および連結方式) とソフトパーティションは、データの冗長性を提供しません。
- 連結は、小規模なランダム入出力操作に適しています。
- ストライプ化は、大規模な順次入出力操作や、ランダムな入出力操作に適しています。
- ミラー化によって読み取り性能が向上することもありますが、書き込み性能は例外なく低下します。
- 読み取り - 変更 - 書き込みという RAID-5 ボリュームの性質上、書き込みが 20 パーセントを超えるボリュームでは、RAID-5 を使用してはなりません。冗長性が必要な場合は、ミラー化を検討してください。
- RAID-5 の書き込み性能は、ミラー化の書き込み性能よりも常に低くなります。ミラー化の書き込みは、書き込みを保護しない方式よりも常に低速です。
- ソフトパーティションは、非常に大規模な記憶デバイスの管理に有効です。

注 – Solaris ボリュームマネージャを使用して冗長性を備えたデバイスをサポートする方法については、これらの一般的な記憶方式のほかにも、47 ページの「ホットスペア集合」を参照してください。

性能に関する一般的な指針

記憶装置の構成を決めるときは、次の性能に関する指針を検討してください。

- 一般に、もっとも高い性能が得られるのはストライプ化ですが、ストライプ化はデータの冗長性を提供しません。書き込みが多いアプリケーションでは、一般に、RAID-1 ボリュームの方が RAID-5 ボリュームよりも高い性能を提供します。
- RAID-1 と RAID-5 ボリュームではデータの可用性は向上しますが、どちらのボリュームタイプでも、一般に書き込み操作の性能は低下します。ミラー化は、ランダムな読み取り性能を向上させます。
- RAID-5 ボリュームのハードウェアコストは RAID-1 ボリュームよりも少なくなります。RAID-0 ボリュームでは、追加のハードウェアコストは発生しません。
- ストライプと RAID-5 ボリュームではデータが複数のディスクドライブに分散されるため、入出力負荷が均一化されます。
- もっともアクセス頻度の高いデータを特定し、そのデータに対するアクセス帯域幅をミラー化またはストライプ化によって拡張します。
- 性能監視機能や一般的なツール (`iostat` コマンドなど) を使って、もっともアクセス頻度の高いデータを特定します。そのデータに対するアクセス帯域幅を、ストライプ化、RAID-1 ボリューム、または RAID-5 ボリュームを使って拡張します。
- ソフトパーティションのサイズを複数回変更すると、性能が低下することがあります。
- 書き込み操作の場合、RAID-5 ボリュームはストライプよりも性能が低くなります。これは、RAID-5 ボリュームのパリティーを計算および格納するために、複数の入出力操作が必要となるためです。
- ランダム raw 入出力の読み取りの場合には、ストライプと RAID-5 ボリュームの性能は同等です。ストライプでも RAID-5 ボリュームでも、データは複数のディスクに分割されます。また、スライスに障害が発生した場合を除き、RAID-5 ボリュームのパリティー計算は、読み取り性能に影響を与える要因にはなりません。
- ランダム raw 入出力の書き込みでは、ストライプの方が RAID-5 ボリュームよりも優れています。

Solaris ボリュームマネージャ固有の構成の指針については、47 ページの「Solaris ボリュームマネージャ構成の指針」を参照してください。

ランダム入出力と順次入出力の最適化

ここでは、構成の最適化方法について説明します。

作成しようとしている Solaris ボリュームマネージャのボリューム上で順次入出力とランダム入出力のどちらが多用されるのかわからない場合は、性能のチューニングに関する以下の手法を適用しないでください。実装が適切でないと、性能が低下することがあります。

以下に示す最適化のための手法では、RAID-0 ボリュームを最適化するものとします。通常は、RAID-0 ボリュームを最適化してからそのボリュームをミラー化して、最適な性能とデータの冗長性を同時に達成します。

ランダム入出力

データベースや汎用ファイルサーバーに使用されるランダム入出力環境では、すべてのディスクにおいて、入出力要求の応答にかかる時間が同じであることが望まれます。

たとえば、データベースアプリケーション用に 40G バイトの記憶領域があるとします。10G バイトのディスクスピンドルを 4 つ使ってストライプ化を行っている場合、入出力がランダムで、ボリューム間で均一に分散されていれば、各ディスクの使用率が均一になり、通常、性能は向上します。

ランダム入出力性能を最大化するためには、ディスクの使用率を 35 パーセント以下に抑えるようにします (`iostat` コマンドで調べることができる)。ディスクの使用率が定常的に 65 パーセントを超える場合には問題となります。90 パーセントを超える場合には重大な問題が発生します。このような問題を解決するためには、さらに多くのディスク (スピンドル) を追加して、新しい RAID-0 ボリュームを作成する必要があります。

注 – 既存のボリュームにディスクを追加するだけでは、性能を向上することはできません。適切なパラメータを設定して新しいボリュームを作成し、性能の最適化を図る必要があります。

データをすべてのディスクに分散する場合には、飛び越し値のサイズは重要ではありません。一般的な入出力要求よりも大きい飛び越し値を指定するだけで十分です。

順次アクセス入出力

テーブル全体のスキャンが頻繁に行われる DBMS サーバーまたはデータの非常に多い環境で使用される NFS サーバーなどの順次入出力環境で、構成の性能を最適化できます。順次入出力環境を活用するには、飛び越し値を通常の入出力要求のサイズより小さく設定してください。

たとえば、通常の入出力要求のサイズが 256K バイトで、ストライプが 4 スピンドルに渡っているとします。この場合の適切なストライプユニットサイズは

256K バイト / 4 = 64K バイトまたはそれ以下です。

この場合には、通常の入出力要求が複数のディスクスピンドルに分散されるため、順次入出力の帯域幅が増加します。

注 – 順次入出力環境では、シーク時間と回転待ち時間は実質的にゼロです。順次入出力の最適化では、ディスクの内部転送速度が最も重要な要素になります。

順次アプリケーションの通常の入出力サイズには大きい値を使用します (128K バイト以上、場合によっては 1M バイト以上)。ここでは、通常の入出力要求のサイズが 256K バイトで、ストライプが 4 ディスクスピンドルに渡っているとします。

256K バイト / 4 = 64K バイトであるため、

飛び越しサイズは 32–64K バイトにするにが適切です。

◆ ◆ ◆ 第 3 章

Solaris ボリュームマネージャの概要

この章では、Solaris ボリュームマネージャの構造の全体について説明します。この章では、次の内容について説明します。

Solaris ボリュームマネージャの新機能

この節では、今回の Solaris リリースで導入された、Solaris ボリュームマネージャの新機能について説明します。

Solaris の新機能の一覧と Solaris のリリースの説明については、『Solaris 10 の概要』を参照してください。

Solaris ボリュームマネージャの概要

Solaris ボリュームマネージャは、多数のディスクとそれらのディスク上のデータを管理するためのソフトウェア製品です。Solaris ボリュームマネージャにはいろいろな用途がありますが、ほとんどの場合、次の目的で使用されます。

- 記憶容量を増加する
- データ可用性を向上する
- 大規模な記憶デバイスの管理を容易にする

状況によっては、Solaris ボリュームマネージャを使用すると、入出力性能が向上することもあります。

Solaris オペレーティングシステムでサポートされるディスクの種類については、『Solaris のシステム管理 (デバイスとファイルシステム)』の第 11 章「ディスクの管理 (概要)」を参照してください。

Solaris ボリュームマネージャによる記憶装置の管理方法

Solaris ボリュームマネージャは、仮想ディスクを使用して、物理ディスクとその関連データの管理を行います。Solaris ボリュームマネージャでは、仮想ディスクをボリュームと呼びます。ただし、従来からの慣習により、コマンド行ユーティリティーではボリュームをメタデバイスと呼ぶこともあります。

アプリケーションまたはファイルシステムから見た場合、ボリュームはディスクと同じ機能です。Solaris ボリュームマネージャは、ボリュームに対する入出力要求を、そのボリュームを構成するメンバーディスクに対する入出力要求に変換します。

Solaris ボリュームマネージャのボリュームは、ディスクスライスまたはほかの Solaris ボリュームマネージャボリュームから作成されます。Solaris 管理コンソールに組み込まれているグラフィカルユーザーインタフェース (GUI) を使えば、ボリュームを簡単に作成できます。Solaris 管理コンソール内の「拡張ストレージ」を使用することで、存在しているすべてのボリュームの情報を表示できます。管理者は、ウィザードのステップに従うだけで、Solaris ボリュームマネージャのあらゆる種類のボリュームとコンポーネントを簡単に作成できます。また、Solaris ボリュームマネージャのコマンド行ユーティリティーを使用しても、ボリュームの作成や変更が行えます。

たとえば、大きな記憶領域を 1 つのボリュームとして作成したい場合、Solaris ボリュームマネージャを使用すると、複数のスライスの集まりを単一の大容量ボリュームとして扱うように、システムを設定できます。このような複数のスライスから作成した単一のボリュームは、ただちに、「実」スライスまたは「実」デバイスと同じように使用できます。

ボリュームの詳細については、42 ページの「ボリュームの概要」を参照してください。

Solaris ボリュームマネージャでは、RAID-1 (ミラー) ボリュームや RAID-5 ボリュームを使ってデータの信頼性と可用性を高めることができます。Solaris ボリュームマネージャのホットスペアを使用すれば、ミラーや RAID-5 ボリュームのデータ可用性をさらに向上できます。

構成の設定が完了したら、Solaris 管理コンソール内の「拡張ストレージ」を使って動作状況を検査できます。

Solaris ボリュームマネージャとの対話方法

Solaris ボリュームマネージャを管理するには、次のどちらかの方法を使用します。

- Solaris 管理コンソール – このツールは、ボリューム管理機能を使用するための GUI を提供します。Solaris 管理コンソール内の「拡張ストレージ」を使用します。「拡張ストレージ」ツールの例については、図 3–1 を参照してください。このインタフェースは、ボリュームや、ホットスペア、状態データベースの複製など、Solaris ボリュームマネージャのコンポーネントをグラフィカルに表示します。このインタフェースではウィザードを使って Solaris ボリュームマネージャコンポーネントを操作できるため、ディスクの構成や既存の構成を簡単に変更できます。
- コマンド行 – ボリューム管理機能を実行するためのコマンドが用意されています。Solaris ボリュームマネージャのコアコマンドは、`metainit` や `metastat` のように `meta` で始まります。Solaris ボリュームマネージャのコマンドの一覧については、付録 B を参照してください。

注 – Solaris ボリュームマネージャを管理するときには、コマンド行と GUI を同時に使用しないでください。構成に対して矛盾した変更が加えられ、予測していない動作が生じることがあります。どちらのツールを使っても、Solaris ボリュームマネージャを管理することは可能ですが、両方のツールを同時に使用することはできません。

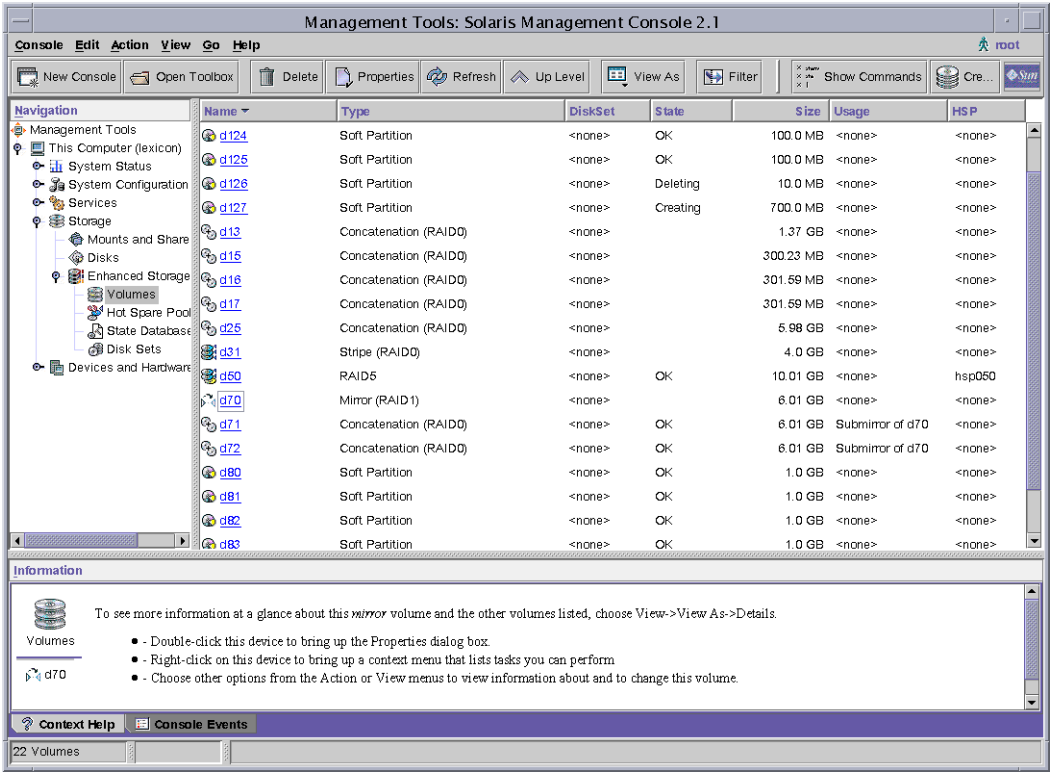

図 3–1 Solaris 管理コンソール内の「拡張ストレージ (Enhanced Storage)」ツール (Solaris ボリュームマネージャ) の使用例

▼ Solaris ボリュームマネージャグラフィカルユーザーインタフェース (GUI) を使用するには

Solaris ボリュームマネージャ GUI (Enhanced Storage) は、Solaris 管理コンソールに組み込まれています。この GUI にアクセスするには、次の手順に従います。

1. 次のコマンドを使って、ホストシステム上で **Solaris** 管理コンソールを起動します。

   ```
   % /usr/sbin/smc
   ```

2. ナビゲーション区画で「このコンピュータ **(This Computer)**」をダブルクリックします。
3. ナビゲーション区画で「ストレージ **(Storage)**」をダブルクリックします。
4. ナビゲーション区画で「拡張ストレージ **(Enhanced Storage)**」をダブルクリックして、**Solaris** ボリュームマネージャツールを起動します。

5 ログインを促すプロンプトが表示されたら、**root** または同等のアクセス権をもつユーザーとしてログインします。

6 適切なアイコンをダブルクリックして、ボリュームや、ホットスペア集合、状態データベースの複製、ディスクセットを管理します。

ヒント – Solaris 管理コンソールの各ツールは、コンソールウィンドウの下部またはウィザードパネルの左側に情報を表示します。このインタフェースでの作業について情報が必要な場合は、いつでも「ヘルプ (Help)」を選択できます。

Solaris ボリュームマネージャの要件

Solaris ボリュームマネージャを使用するための要件は、次のとおりです。

- Solaris ボリュームマネージャを管理するためには root 権限が必要です。Solaris 管理コンソールの User Profile 機能によって同等の権限が与えられていれば、Solaris 管理コンソールを使って管理を行うことができます。ただし、Solaris ボリュームマネージャのコマンド行インタフェースを使用できるのは、スーパーユーザーだけです。
- Solaris ボリュームマネージャを使ってボリュームを作成するためには、少なくとも 3 つの状態データベースの複製が、Solaris ボリュームマネージャを稼働するシステム上に存在していなければなりません。状態データベースの複製には、すべてのボリューム、ホットスペア、およびディスクセットの構成と状態に関する情報が格納されています。最大限の信頼性を確保するためには、これらの複製を異なるコントローラと異なるディスクに配置するようにします。状態データベースの複製の詳細については、65 ページの「Solaris ボリュームマネージャの状態データベースと状態データベースの複製について」を参照してください。状態データベースの複製を作成する手順については、72 ページの「状態データベースの複製の作成」を参照してください。

Solaris ボリュームマネージャコンポーネントの概要

Solaris ボリュームマネージャで作成できるコンポーネントの基本的なタイプは、ボリューム、ソフトパーティション、ディスクセット、状態データベースの複製、ホットスペア集合の 5 つです。次の表に、Solaris ボリュームマネージャの機能の概要を示します。

表 3–1 Solaris ボリュームマネージャの機能の要約

Solaris ボリュームマネージャの機能	定義	目的	参照先
■ RAID-0 ボリューム (ストライプ、連結、連結ストライプ) ■ RAID-1 (ミラー) ボリューム ■ RAID-5 ボリューム	システム上で単一の論理デバイスとして扱われる物理スライスの集まり	記憶容量、性能、またはデータの可用性を高める。	42 ページの「ボリュームの概要」
ソフトパーティション	物理スライスまたは論理ボリュームを分割したもので、より小さく管理しやすい記憶ユニットを提供します	大規模な記憶ボリュームを管理し易くする。	第 12 章
状態データベース (状態データベースの複製)	すべてのボリューム、ホットスペア、およびディスクセットの構成と状態に関する情報が格納されているデータベース。状態データベースの複製を作成しないと、Solaris ボリュームマネージャは動作しません。	Solaris ボリュームマネージャ構成の状態に関する情報を格納	46 ページの「状態データベースと状態データベースの複製」
ホットスペア集合	予約されているスライス (ホットスペア) の集合。サブミラーまたは RAID-5 ボリュームのコンポーネントで障害が発生すると、これらのスライスが自動的に代用されます。	RAID-1 と RAID-5 ボリュームのデータ可用性を高める。	47 ページの「ホットスペア集合」
ディスクセット	個別の名前空間をもつ共有ディスクドライブの集まり。ボリュームとホットスペアを含み、複数のホストによって排他的に共有されます	データの冗長性と可用性を提供し、また個別の名前空間を提供することによって管理を容易にする。	47 ページの「ディスクセット」

ボリュームの概要

「ボリューム」とは、単一の論理デバイスとしてシステムに認識される物理スライスの集合です。 実際には、ボリュームは標準 UNIX® の擬似または仮想デバイスと同義です。

注 – これまで、Solstice DiskSuite™ 製品ではこのような論理デバイスを「メタデバイス」と呼んでいましたが、このマニュアルでは、簡潔性と標準化のために「ボリューム」と呼びます。

ボリュームクラス

ボリュームは、RAID-0 (連結方式またはストライプ方式) ボリューム、RAID-1 (ミラー) ボリューム、RAID-5 ボリューム、 またはソフトパーティションとして作成します。

ボリュームの作成や管理には、Solaris 管理コンソール内の「拡張ストレージ」かコマンド行ユーティリティーを使用します。

次の表に、ボリュームクラスの要約を示します。

表 3–2 ボリュームクラス

ボリューム	説明
RAID-0 (ストライプ方式または連結方式)	そのまま使用することも、ミラーの基本的な構築ブロックとして使用することもできます。RAID-0 ボリューム自体にはデータの冗長性はありません。
RAID-1 (ミラー)	複数のコピーを保持することによってデータを複製します。RAID-1 ボリュームは、サブミラーと呼ばれる 1 つまたは複数の RAID-0 ボリュームから構成されます。
RAID-5	パリティー情報を使ってデータを複製します。ディスクに障害が発生すると、利用可能なデータとパリティー情報から失われたデータが復元されます。RAID-5 ボリュームは、通常、複数のスライスで構成されています。1 スライスに相当する領域がパリティー情報に割り当てられますが、パリティーは RAID-5 ボリュームのすべてのスライスに分散されます。
ソフトパーティション	スライスまたは論理ボリュームをより小さい 1 つまたは複数の拡張可能ボリュームに分割します。

ボリュームの使用方法

ボリュームを使用すると、記憶領域を拡張したり、性能やデータの可用性を高めたりできます。状況によっては、ボリュームを使用すると、入出力性能が向上することもあります。機能的には、ボリュームとスライスは同じように動作します。エンドユーザーや、アプリケーション、ファイルシステムからは、ボリュームとスライスは同じように見えます。物理デバイスと同じように、ボリュームはブロックまたは raw デバイス名を使ってアクセスされます。ボリューム名は、ブロックデバイスを使用するか raw デバイスを使用するかによって異なります。ボリューム名の詳細については、45 ページの「ボリューム名」を参照してください。

ボリューム上では、`mkfs`、`mount`、 `umount`、`ufsdump`、`ufsrestore` など、ファイルシステムのほとんどのコマンドを使用できます。ただし、`format` コマンドは使用できません。ボ

リューム上にマウント済みのファイルシステムが存在すれば、ボリュームとの間でファイルの読み取り、書き込み、コピーを行うことができます。

例 — 2 つのスライスから構成されるボリューム

図 3–2 に、Disk A のスライス 1 つと Disk B のスライス 1 つの合計 2 つのスライスからなるボリュームを示します。アプリケーションや UFS は、このボリュームを 1 つの物理ディスクであるかのように扱います。ボリュームにさらに多くのスライスを追加すれば、記憶容量を増やすことができます。

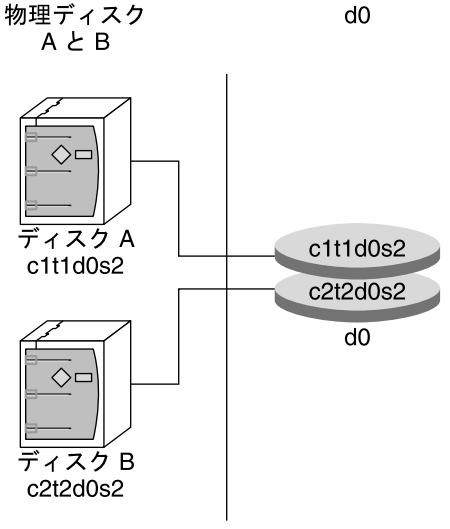

図 3–2 ボリューム、物理ディスク、スライスの関係

growfs コマンドによるボリュームとディスク領域の拡張

Solaris ボリュームマネージャでは、ボリュームにスライスを追加することによってボリュームを拡張できます。既存のボリュームにスライスを追加するには、Solaris 管理コンソール内の「拡張ストレージ」かコマンド行インタフェースを使用します。

ボリュームに含まれている UFS ファイルシステムは、マウントされているかどうかに関係なく、システムを停止したりバックアップを取らなくても拡張できます。ただし、どのような場合でも、データのバックアップを取ることを推奨します。ボリュームを拡張したあとは、`growfs` コマンドを使用してファイルシステムを拡張します。

注 – いったん拡張したファイルシステムを縮小することはできません。ファイルシステムを縮小できないことが UFS の制約です。同じように、いったん拡張した Solaris ボリュームマネージャのパーティションを縮小することはできません。

---

raw ボリュームを使用するアプリケーションやデータベースは、独自の方法で領域を拡張し、それを認識できなければなりません。Solaris ボリュームマネージャには、この機能はありません。

ボリュームのディスク領域を拡張するには、次の方法を利用できます。

- RAID-0 ボリュームに 1 つまたは複数のスライスを追加する
- RAID-1 ボリュームのすべてのサブミラーに 1 つまたは複数のスライスを追加する
- RAID-5 ボリュームに 1 つまたは複数のスライスを追加する
- パーティションを構成するコンポーネントから領域を追加することによって、ソフトパーティションを拡張する

`growfs` コマンドは、サービスの中断やデータを失うことなく、UFS ファイルシステムを拡張します。ただし、`growfs` コマンドが実行している間は、ボリュームへの書き込みアクセスはできません。ファイルシステムは、それを格納しているスライスまたはボリュームのサイズまで拡張できます。

追加ディスク領域の一部だけを使ってファイルシステムを拡張する場合は、`growfs` コマンドに `-s` *size* オプションを指定します。

注 – ミラーを拡張すると、ミラーを構成しているすべてのサブミラーに領域が追加されます。`growfs` コマンドは RAID-1 ボリューム上で実行されます。一般的な規則としては、ボリュームを構成するデバイスに領域を追加してから、最上位のデバイスに対して `growfs` コマンドを実行します。

ボリューム名

物理スライスと同じように、ボリュームにも、ファイルシステムで使用される論理名があります。論理ボリューム名は、`/dev/md/dsk` ディレクトリ (ブロックデバイスの場合) または `/dev/md/rdsk` ディレクトリ (raw デバイスの場合) に作成されます。`meta*` コマンドでは、`/dev/md/dsk/`*volume-name* のような完全なボリューム名を指定する代わりに、`d1` のような省略形のボリューム名を使用することができます。名前を変更しようとするボリュームが現在使用されておらず、かつ、新しい名前がほかのボリュームで使用されていなければ、ボリューム名はいつでも変更できます。詳細については、239 ページの「ボリューム名の交換」を参照してください。

以前は、ボリューム名は「d」で始まり、そのあとに数字が続く形式にする必要がありました (たとえば、`d0`)。この形式は現在も使用可能です。次に、「d*」命名構文を使用するボリューム名の例を示します。

`/dev/md/dsk/d0` ブロック型ボリューム `d0`

`/dev/md/dsk/d1` ブロック型ボリューム `d1`

`/dev/md/rdsk/d126` raw ボリューム `d126`

`/dev/md/rdsk/d127` raw ボリューム `d127`

ボリューム名に関する指針

ボリューム名を標準化すると、管理が容易になるだけでなく、ボリュームタイプが一目でわかるようになります。次の推奨事項を考慮してください。

- ボリュームタイプごとに範囲を使用します。たとえば、RAID-1 ボリュームには 0 から 20、RAID-0 ボリュームには 21 から 40、などのように割り当てます。
- ミラーの名前を相互に関連付けます。たとえば、ミラーの名前を 0 で終わる数字にし、サブミラーの名前を 1 や 2 で終わる数字にします。これに従えば、最初のミラーは `d10`、そのサブミラーは `d11` と `d12` になります。さらに、次のミラーは `d20`、そのサブミラーは `d21`、 `d22`、 `d23`、`d24` になります。
- スライス番号とディスク番号がボリューム番号に対応するような命名方法を使用します。

状態データベースと状態データベースの複製

「状態データベース」は、Solaris ボリュームマネージャの構成の状態に関する情報を格納するデータベースです。状態データベースは、構成に対して加えられた変更を記録および管理します。Solaris ボリュームマネージャは、構成や状態に変化があると、状態データベースを自動的に更新します。たとえば、新しいボリュームの作成は構成の変更であり、サブミラーの障害は状態の変化を意味します。

状態データベースは、実際には複製された複数のデータベースコピーの集まりです。各コピーは、状態データベースの複製と呼ばれ、データベース内のデータが常に有効であることを保証します。状態データベースのコピーを複数持つことにより、単一点障害からデータを保護することができます。状態データベースは、既知の状態データベースの複製の格納場所と状態をすべて記録しています。

状態データベースとその状態データベースの複製が作成されるまで、Solaris ボリュームマネージャは動作できません。Solaris ボリュームマネージャ構成には、正常に動作する状態データベースが必要です。

構成を設定するときは、状態データベースの複製を次のどちらかに配置できます。

- 専用のスライス上
- 後でボリュームの一部となるスライス上

Solaris ボリュームマネージャは、状態データベースの複製が割り当てられているスライスを認識し、そのスライスがボリューム内で使用されている場合には、その複製部分を自動的にスキップします。状態データベースの複製用に予約されているスライスの部分を、他の目的に使用することはできません。

複数の状態データベースのコピーを 1 つのスライス上に置くこともできますが、そのようにすると、システムは単一点障害に対して脆弱になります。

すべての状態データベースの複製が削除された場合でも、Solaris オペレーティングシステムは正常に動作します。しかし、状態データベースの複製がディスク上にまったくない状態でシステムをリブートすると、すべての Solaris ボリュームマネージャの構成データが失われます。

ホットスペア集合

「ホットスペア集合」は、障害のあるコンポーネントと自動的に交換できるように、Solaris ボリュームマネージャが予約しているスライス (「ホットスペア」) の集合です。ホットスペアは、サブミラーまたは RAID-5 ボリュームのどちらでも使用できます。RAID-1 ボリュームと RAID-5 ボリュームでは、ホットスペアを使用することによってデータの可用性が向上します。ホットスペア集合は、Solaris 管理コンソール内の「拡張ストレージ」とコマンド行インタフェースのどちらでも作成できます。

コンポーネントエラーが発生すると、Solaris ボリュームマネージャは、障害のあるコンポーネントと同じかそれより大きいサイズのホットスペアを探します。該当するホットスペアが見つかると、Solaris ボリュームマネージャは自動的にそのコンポーネントを交換して、データの再同期をとります。適切なサイズのスライスがホットスペア集合にないと、サブミラーまたは RAID-5 ボリュームは使用不能とみなされます。詳細については、第 16 章を参照してください。

ディスクセット

ディスクセットとは、論理ボリュームとホットスペアを含む物理的な記憶領域ボリュームの集まりのことです。ボリュームやホットスペア集合は、そのディスクセット内のドライブだけで構成される必要があります。ディスクセットに作成したボリュームは、物理スライスと同じように使用できます。

クラスタ環境では、ディスクセットの使用によってデータの可用性が向上します。一方のホストに障害が発生しても、そのディスクセットを他方のホストが引き継ぐことができます。このタイプの構成は「フェイルオーバー構成」と呼ばれます。さらに、ディスクセットを使用することにより、Solaris ボリュームマネージャの名前空間の管理や、ネットワーク接続されている記憶装置へのアクセスが容易になります。

詳細については、第 18 章を参照してください。

Solaris ボリュームマネージャ構成の指針

Solaris ボリュームマネージャの構成が適切に設定されていないと、性能が低下することがあります。この節では、Solaris ボリュームマネージャの性能を最適に保つためのヒントについて説明します。記憶装置構成の性能に関する指針については、33 ページの「性能に関する一般的な指針」を参照してください。

一般的な指針

- ディスクとコントローラ – 1 つのボリュームに複数のドライブがある場合は、それぞれ別のドライブパスに配置します。なお、SCSI ドライブの場合は、それぞれ別のホストアダプタに配置します。入出力負荷をいくつかのコントローラに分散することによって、ボリュームの性能と可用性が向上します。
- システムファイル – `/etc/lvm/mddb.cf` や `/etc/lvm/md.cf` ファイルは変更したり削除したりしないでください。

  これらのファイルは、定期的にバックアップしてください。
- ボリュームの整合性 – スライスをボリュームとして定義したら、その元となるスライスをダンプデバイスなどの別の目的に使用してはなりません。
- ディスクとパーティションに関する情報 – 不良ディスクを再フォーマットしたり、Solaris ボリュームマネージャの構成を作成し直したりしなければならない場合に備えて、`prtvtoc` と `metastat -p` コマンドの出力コピーを保管してください。

ファイルシステムに関する指針

ボリュームを構成しているスライス上にファイルシステムをマウントしないでください。スライスがボリュームを構成するために使用されている場合は、そのスライスをファイルシステムとしてマウントしてはなりません。可能であれば、ボリュームとして使用する予定の物理デバイスは、アクティブにする前にマウント解除してください。

Solaris ボリュームマネージャコンポーネントの作成についての概要

Solaris ボリュームマネージャのコンポーネントを作成するときは、物理スライスを Solaris ボリュームマネージャの論理名 (たとえば `d0`) に割り当てます。作成できる Solaris ボリュームマネージャのコンポーネントには、次のものがあります。

- 状態データベースの複製
- ボリューム (RAID-0 (ストライプ、連結)、RAID-1 (ミラー)、RAID-5、およびソフトパーティション)
- ホットスペア集合
- ディスクセット

注 – ボリューム名の付け方のヒントについては、45 ページの「ボリューム名」を参照してください。

Solaris ボリュームマネージャコンポーネントを作成するための前提条件

Solaris ボリュームマネージャコンポーネントを作成するための前提条件は、次のとおりです。

- 初期状態データベースの複製を作成します。作成手順については、72 ページの「状態データベースの複製の作成」を参照してください。
- Solaris ボリュームマネージャで使用できるスライスを特定します。必要に応じて、`format` コマンド、`fmthard` コマンド、または Solaris 管理コンソールを使って、既存ディスクのパーティションを再分割します。
- root 権限を持っていることを確認します。
- すべてのデータの最新バックアップを取ります。
- GUI を使用する場合は、Solaris 管理コンソールを起動して Solaris ボリュームマネージャ機能に移動します。詳細については、40 ページの「Solaris ボリュームマネージャグラフィカルユーザーインタフェース (GUI) を使用するには」を参照してください。

Solaris ボリュームマネージャにおけるマルチテラバイトサポートの概要

Solaris 9 4/03 リリース以降、Solaris ボリュームマネージャは、64 ビットカーネルが動作しているシステム上で、1 テラバイト (T バイト) より大きな記憶装置や論理ボリュームをサポートします。

注 – 64 ビットカーネルがシステムで動作しているかどうかを知るには、`isainfo -v` を使用します。「64-bit」という文字列が表示された場合、64 ビットカーネルが動作しています。

Solaris ボリュームマネージャを使用すると、次の作業を行うことができます。

- 1T バイトを超えるサイズの論理記憶装置ユニット (LUN) で (あるいは、LUN から) 構築された論理ボリュームを、作成、変更、および削除できます。
- 1T バイトを超えるサイズの論理ボリュームを、作成、変更、および削除できます。

大容量ボリュームのサポートが自動化されます。1T バイトを超えるデバイスを作成する場合、Solaris ボリュームマネージャが適切に構成するので、ユーザーの介入は不要です。

大容量ボリュームのサポートの制限

Solaris ボリュームマネージャが大容量ボリューム (1T バイト超) をサポートするのは、64 ビットカーネルが動作している Solaris 9 4/03 以降のリリースに限られます。これより前の Solaris 9 リリースの 32 ビットカーネルが動作しているシステムで大容量ボリュームを扱うと、Solaris ボリュームマネージャの機能に影響が出ます。特に、次の点に注意してください。

- 大容量ボリュームを持つシステムを Solaris 9 4/03 以降の 32 ビットカーネルでリブートした場合、大容量ボリュームは `metastat` の出力では見えますが、その大容量ボリュームにアクセス、変更、または削除することはできません。さらに、新しい大容量ボリュームは作成できません。大容量ボリューム上のボリュームまたはファイルシステムも利用できません。
- 大容量ボリュームを持つシステムを Solaris 9 4/03 より前のリリースでリブートした場合、Solaris ボリュームマネージャは起動しません。大容量ボリュームに対応していない Solaris プラットフォームで Solaris ボリュームマネージャを実行する前には、すべての大容量ボリュームを削除する必要があります。

注意 – 32 ビットカーネルで Solaris ソフトウェアを動作させる予定がある場合、または Solaris 9 4/03 リリースより前のバージョンの Solaris OS を使用する場合は、大容量ボリュームを作成してはなりません。

大容量ボリュームの使用

Solaris ボリュームマネージャのすべてのコマンドは大容量ボリュームで機能します。大容量ボリュームサポートを利用する場合でも、構文は同じです。特別な作業も不要です。したがって、Solaris ボリュームマネージャに精通しているシステム管理者であれば、Solaris ボリュームマネージャでただちに大容量ボリュームを扱うことができます。

ヒント – 大容量ボリュームを作成したあとで、Solaris 9 4/03 より前のリリースまたは Solaris 9 4/03 以降の 32 ビットカーネルで Solaris ボリュームマネージャを使用する必要ができた場合、大容量ボリュームを削除する必要があります。Solaris 9 4/03 より前のリリースまたは 32 ビットカーネルでリブートする前に、64 ビットカーネルで `metaclear` コマンドを使用して、Solaris ボリュームマネージャ構成から大容量ボリュームを削除してください。

Solaris ボリュームマネージャへのアップグレード

Solstice DiskSuite バージョン 4.1、4.2、および 4.2.1 から Solaris ボリュームマネージャへは、シームレスなアップグレードが可能です。ただし、すべてのボリュームが「正常 (Okay)」状態である (「保守が必要 (Needs Maintenance)」や「最後にエラー (Last Erred)」の状態でない) ことが必要です。さらに、ホットスペアが使用されていてはなりません。

これ以上、アップグレードのために Solaris ボリュームマネージャに必要な作業はありません。つまり、構成の変更やルートミラーの解除は不要です。システムをアップグレードすると、Solstice DiskSuite の構成が繰り越され、Solaris ボリュームマネージャの各種ツールを通してアクセスできるようになります。

Solaris 10 OS では、サービス管理機能 (SMF) が導入されました。この機能が提供するインフラストラクチャーによって、従来の UNIX 起動スクリプト、`init` 実行レベル、および構成ファイルを補強できます。Solaris OS の以前のバージョンからアップグレードする場合は、Solaris ボリュームマネージャに関連する SMF サービスがオンラインであることを確認してください。このような SMF サービスがオンラインでない場合、Solaris ボリュームマネージャを管理するときに問題が発生する可能性があります。

Solaris ボリュームマネージャに関連する SMF サービスを確認するには、次の形式の `svcs` コマンドを使用します。

```
# svcs -a |egrep "md|meta"
disabled       12:05:45 svc:/network/rpc/mdcomm:default
disabled       12:05:45 svc:/network/rpc/metamed:default
disabled       12:05:45 svc:/network/rpc/metamh:default
online         12:05:39 svc:/system/metainit:default
online         12:05:46 svc:/network/rpc/meta:default
online         12:05:48 svc:/system/fmd:default
online         12:05:51 svc:/system/mdmonitor:default
```

Solaris ボリュームマネージャ構成がローカルセットだけで構成されている場合、次のサービスはオンラインであるべきです。

```
svc:/system/metainit
svc:/network/rpc/meta
svc:/system/mdmonitor
```

Solaris ボリュームマネージャ構成にディスクセットが含まれている場合、次のサービスもオンラインであるべきです。

```
svc:/network/rpc/metamed
svc:/network/rpc/metamh
```

Solaris ボリュームマネージャ構成にマルチノードのディスクセットが含まれている場合、上記のサービスに加えて、次のサービスもオンラインであるべきです。

```
svc:/network/rpc/mdcomm
```

SMF の詳細については、『Solaris のシステム管理 (基本編)』の第 14 章「サービスの管理 (概要)」を参照してください。

◆ ◆ ◆ 第 4 章

Solaris Volume Manager for Sun Cluster (概要)

この章では、Solaris Volume Manager for Sun Cluster の概要について説明します。

この章では、次の内容について説明します。

Solaris Volume Manager for Sun Cluster の紹介

Solaris 9 9/04 リリースから、Solaris ボリュームマネージャでは Sun Cluster 環境の記憶装置を複数所有者ディスクセットを使用して管理できるようになりました。「複数所有者ディスクセット」を使用すると、複数のノードがディスクセットの所有権を共有したり、共有ディスクに同時に書き込むことができます。従来の共有ディスクセットは、参加しているすべてのホストから認識できても、アクセスできるのは 1 度に 1 つのホストだけでした。複数所有者ディスクセットは Sun Cluster および Oracle9 Real Application Clusters などのアプリケーションの組み合わせで機能します。

複数所有者ディスクセットと Solaris ボリュームマネージャの共有ディスクセットは同じノードに共存できます。ただし、これら 2 つの構成間でディスクセットを移動することはできません。

注 – 複数所有者ディスクセット用の Solaris Volume Manager for Sun Cluster デバイス ID サポートは利用できません。従って、現在のところ、あるシステムから別のシステムへ、複数所有者ディスクセットをインポートすることはできません。

Solaris Volume Manager for Sun Cluster は、Solaris ボリュームマネージャで作成できるものと同じコンポーネント、つまり、ストライプ、連結、ミラー、ソフトパーティション、ホットスペアを作成できます。Solaris Volume Manager for Sun Cluster は RAID-5 とトランザクションボリュームはサポートしません。

次の図に、典型的なクラスタ構成におけるソフトウェアと共有記憶装置間の関係を示します。

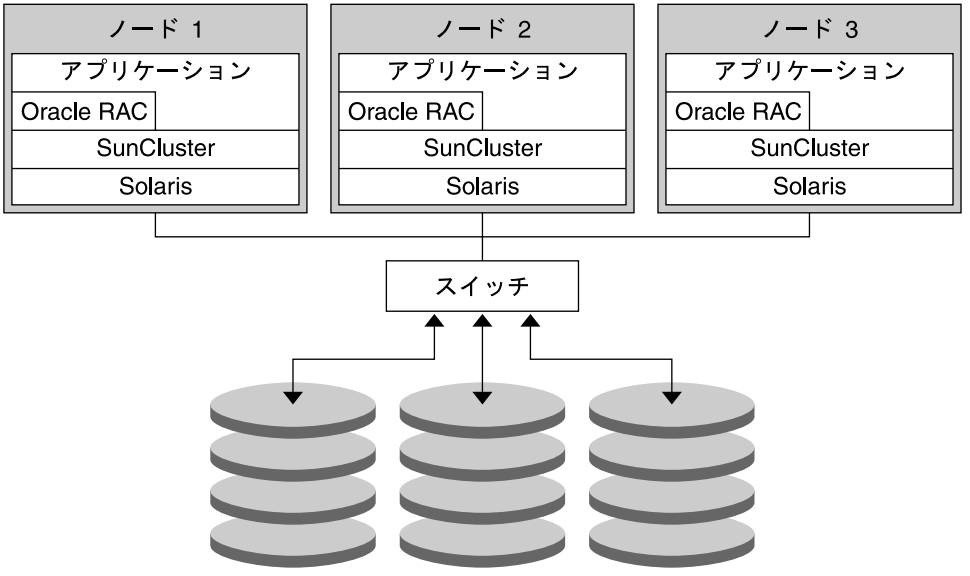

図 4–1 サンプルのクラスタ構成

各ノードはローカル記憶装置を持っており、また、共有記憶装置へのパスを少なくとも 1 つ持っています。クラスタ環境の複数所有者ディスクセットは、Solaris オペレーティングシステム (Solaris OS) に組み込まれている Solaris Volume Manager for Sun Cluster によって管理します。

前提条件: 複数所有者ディスクセット機能に必要なソフトウェアコンポーネント

Solaris Volume Manager for Sun Cluster を使用するには、Solaris OS とともに次のソフトウェアをインストールする必要があります。

- Sun Cluster 初期クラスタフレームワーク
- Sun Cluster Support for Oracle Real Application Clusters ソフトウェア
- Oracle Real Application Clusters ソフトウェア

注 – Sun Cluster と Oracle Real Application Clusters ソフトウェアの設定については、『Sun Cluster ソフトウェアのインストール (Solaris OS 版)』および『Sun Cluster Data Service for Oracle Real Application Clusters ガイド (Solaris OS 版)』を参照してください。

複数所有者ディスクセットの概念

Solaris Volume Manager for Sun Cluster が管理する記憶装置は、複数所有者ディスクセットとしてグループ分けされます。複数所有者ディスクセットを使用すると、複数のノードがディスクセットの所有権を共有したり、共有ディスクに同時に書き込むことができます。Oracle Real Application Clusters といったアプリケーションのインスタンスがクラスタ内の各ノードで動作するので、複数所有者ディスクセットは拡張性を提供します。アプリケーションの各インスタンスは共有記憶装置に直接アクセスするため、複数所有者ディスクセットを使用すると、アプリケーションの性能も上がります。

注 – 複数所有者ディスクセット機能が有効になるのは、Sun Cluster 環境においてのみです。「ノード」とは、Sun Cluster システムの一部である物理的なマシンのことです。

どの複数所有者ディスクセットにも、ノードのリストが関連付けられています。これらのノードがこのディスクセットの所有権を共有しています。次の `metaset -s` *disk-set* コマンドは、複数所有者ディスクセットの出力を表示します。

```
# metaset -s blue

Multi-owner Set name = blue, Set number = 1, Master = nodeone

Host                Owner          Member
  nodeone           multi-owner   Yes
  nodetwo           multi-owner   Yes

Drive    Dbase

d9       Yes

d13      Yes
```

この出力は、ノードリスト内の `nodeone` と `nodetwo` がディスクセットの所有権を共有していることを示しています。また、`nodeone` が「マスターノード」であることも示しています。

どの複数所有者ディスクセットにも、マスターノードが 1 つあります。ディスクセットの作成後、最初のディスクを追加したノードが、そのディスクセットのマスターノードになります。マスターノードは、ディスクセットの状態データベースの複製を作成、削除、および更新します。

注 – 状態データベースの複製の詳細については、第 6 章を参照してください。

Solaris Volume Manager for Sun Cluster では、ディスクセットのノードリストが互いに異なっていても、さらには重複があっても、共存が可能です。ディスクセットごとにマスターノードが存在するため、同じクラスタ上に複数のマスターが共存することもあります。

次の metaset コマンドからの出力は、最初のディスクをディスクセットに追加した時点で、nodeone がマスターノードになることを示しています。

```
nodeone# metaset -s red
Multi-owner Set name = red, Set number = 1, Master =

Host                Owner          Member
  nodeone                           Yes
  nodetwo                           Yes
nodeone# metaset -s red -a /dev/did/dsk/d9
nodeone# metaset -s red

Multi-owner Set name = red, Set number = 1, Master = nodeone

Host                Owner          Member
  nodeone           multi-owner     Yes
  nodetwo           multi-owner     Yes

Drive     Dbase

d9        Yes
```

Solaris Volume Manager for Sun Cluster では、ディスクセットのノードリストが互いに異なっていても、さらには重複があっても、共存が可能です。ディスクセットごとにマスターノードが存在するため、同じクラスタ上に複数のマスターが共存することもあります。

複数所有者ディスクセットに関連する作業

注意 – 複数所有者ディスクセットを構成する前に、Solaris OS とともに次のソフトウェアをインストールしておく必要があります。

- Sun Cluster 初期クラスタフレームワーク
- Sun Cluster Support for Oracle Real Application Clusters ソフトウェア
- Oracle Real Application Clusters ソフトウェア

Sun Cluster と Oracle Real Application Clusters ソフトウェアの設定については、『Sun Cluster ソフトウェアのインストール (Solaris OS 版)』および『Sun Cluster Data Service for Oracle Real Application Clusters ガイド (Solaris OS 版)』を参照してください。

Solaris Volume Manager for Sun Cluster でディスクセットに関連する作業を実行するには、通常、Solaris ボリュームマネージャと同じコマンドを使用します。metaset コマンドに、複数所有者ディスクセット固有のコマンドオプションが追加されています。たとえば、複数所有者ディスクセットを作成する作業には、metaset コマンドに -M オプションを指定する必要があります。次の出力は、metaset -s *diskset-name* -a -M -h *hostname* コマンドを使用して、複数所有者ディスクセットを作成する方法を示しています。

```
# metaset -s red -a -M -h nodeone
# metaset
Multi-owner Set name = red, Set number = 1, Master =

Host                Owner          Member
  nodeone                          Yes
```

また、metaset コマンドオプションの中には、ディスクセットを取得または解放するコマンドなど、複数所有者ディスクセットでは使用できないものもあります。詳細については、metaset(1M) のマニュアルページを参照してください。

ディスクを処理する際も、Sun Cluster 環境では作業方法が異なります。Sun Cluster は各ディスクに一意のディスク ID (DID) 番号を割り当てます。ディスクを識別するときには、*c*t*n*d*n* 形式を使用するのではなく、Sun Cluster の DID パス名 /dev/did/dsk/d*N* を使用します。変数 *N* は、Sun Cluster によって割り当てられたデバイス番号です。

次の出力は、複数所有者ディスクセットにディスクを追加するときに、metaset -s *diskset-name* -a *disk-name* コマンドと Sun Cluster の DID パス名を使用してディスクを識別する方法を示しています。

```
nodeone# metaset -s red
Multi-owner Set name = red
Multi-owner Set name = red, Set number = 1, Master =

Host                Owner          Member
  nodeone                           Yes
  nodetwo                           Yes
nodeone# metaset -s red -a /dev/did/dsk/d13
nodeone# metaset -s red
Multi-owner Set name = red, Set number = 1, Master = nodeone

Host                Owner          Member
  nodeone            multi-owner   Yes

Drive Dbase

d13   Yes
```

Oracle Real Application Clusters に対応する複数所有者ディスクセットの作成方法については、『Sun Cluster Data Service for Oracle Real Application Clusters ガイド (Solaris OS 版)』の「Oracle Real Application Clusters データベース用の Solaris Volume Manager for Sun Cluster にマルチオーナーディスクセットを作成」を参照してください。

ディスクセットに関連する作業については、第 19 章を参照してください。

Solaris Volume Manager for Sun Cluster の構成

Solaris Volume Manager for Sun Cluster は次の構成をサポートします。

- Solaris Volume Manager for Sun Cluster は最大で 32 個のディスクセットをサポートします。これらのディスクセットは、複数所有者ディスクセット、共有ディスクセット、およびローカルディスクセットの任意の組み合わせにできます。

  注 – 異なる種類のディスクセットの詳細については、204 ページの「ディスクセットのタイプ」を参照してください。

- 複数所有者ディスクセットでは、ディスクセットごとに最大で 8192 個のボリュームをサポートします。
- 状態データベースの複製のデフォルトサイズは 16M バイトです。最小サイズは 16M バイトです。最大サイズは 256M バイトです。

Sun Cluster Support for Oracle Real Application Clusters で拡張されたプロパティーの多くは、再構成プロセスの手順でタイムアウトを指定するものです。タイムアウトの設定については、『Sun Cluster Data Service for Oracle Real Application Clusters ガイド (Solaris OS 版)』の「Sun Cluster Support for Oracle Real Application Clusters の調整」を参照してください。

複数所有者ディスクセットにおける RAID–1 (ミラー) ボリューム

複数所有者ディスクセットに作成された RAID-1 ボリューム (つまり、ミラー) は、Solaris ボリュームマネージャ共有ディスクセットの RAID-1 ボリュームと同じように機能します。ただし、複数所有者ディスクセットの RAID-1 ボリュームには、ほかにも機能があります。

複数所有者ディスクセットにおけるミラー所有権

ミラー所有権の概念は、複数所有者ディスクセットに固有です。Solaris ボリュームマネージャの共有ディスクセット用の RAID-1 ボリュームとは異なり、複数所有者ディスクセット用の RAID-1 ボリュームには通常、1 つの所有者が関連付けられています。 ミラーボリュームの所有権は、ボリュームマネージャによって選択されます。ディスクセットのノードリストに記載されたノードの 1 つがこのボリュームの所有者になります。このボリュームに書き込むことができるのは、RAID-1 ボリュームの所有者だけです。所有者でないノードがそのボリュームに書き込もうとした場合、書き込み操作を行おうとしているこのノードへ、所有者が切り換わります。次の `metastat -s` *diskset-name* コマンドの出力は、 `nodeone` が RAID-1 ボリューム `d24` の所有者であることを示しています。

```
# metastat -s red
red/d24: Mirror
```

```
    Submirror 0: red/d20
      State: Okay
    Submirror 1: red/d21
      State: Okay
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Resync option: optimizedresync
    Owner: nodeone
    Size: 825930 blocks (403 MB)
```

データの管理と回復のプロセス

Solaris ボリュームマネージャ用の RAID-1 ボリュームと同様に、Solaris Volume Manager for Sun Cluster 用の RAID-1 ボリュームはデータの整合性を保つための操作を実行します。Solaris Volume Manager for Sun Cluster は、データの管理と回復のための 2 つのオプションを RAID-1 ボリュームに提供します。

Solaris Volume Manager for Sun Cluster 用の最適化された再同期

Solaris Volume Manager for Sun Cluster 用の最適化された再同期は、Solaris ボリュームマネージャ用の最適化された再同期と同じように機能します。しかし、複数所有者ディスクセットは、再同期オプションが最適化された再同期に設定された RAID-1 ボリュームは常に、ミラー所有者を持ちます。次の `metastat -s` *diskset-name* コマンドからの出力は、最適化オプションが `optimizedresync` (最適化された再同期) に設定されていることを示しています。

```
# metastat -s red
red/d24: Mirror
    Submirror 0: red/d20
      State: Okay
    Submirror 1: red/d21
      State: Okay
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Resync option: optimizedresync
    Owner: nodeone
    Size: 825930 blocks (403 MB)
```

最適化された再同期の詳細については、101 ページの「再同期の最適化」を参照してください。

アプリケーションベースの回復と指定されたミラー読み取り

Solaris Volume Manager for Sun Cluster のデータ回復を最適化するためには、Oracle Real Application Clusters などのアプリケーションは、データの回復を管理および制御する機能を必要とします。アプリケーションが回復を制御できると、回復の性能が上がります。ioctl の `DKIOGETVOLCAP`、`DKIOSETVOLCAP`、および `DKIODMR` は、クラスタ環境におけるアプリケーションのデータ管理回復のサポートを提供します。これらの ioctl は、アプリケーションに次の機能を提供します。

- アプリケーションベースの回復 (ABR)— アプリケーションがミラーボリューム上のデータの回復を制御できるようにします。
- 指定されたミラー読み取り — アプリケーションが特定のサブミラーに対する読み取りを指定して、データの状態を調べることができるようにします。

アプリケーションベースのデータ管理回復で使用される ioctl の詳細については、`dkio(7I)` のマニュアルページを参照してください。

再同期オプションをアプリケーションベースの回復に設定した RAID-1 ボリュームがミラー所有者を持つのは、アプリケーションベースの回復プロセスの間だけです。次の `metastat -s` *diskset-name* コマンドからの出力は、RAID-1 ボリュームの通常の状態を示しています。再同期オプションはアプリケーションベースの回復に設定されています。ミラー所有者は存在しません。

```
# metastat -s red
red/d24: Mirror
    Submirror 0: red/d20
      State: Okay
    Submirror 1: red/d21
      State: Okay
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Resync option: application based
    Owner: None
    Size: 825930 blocks (403 MB)
```

◆ ◆ ◆ 第 5 章

Solaris ボリュームマネージャの構成と使用

『Solaris ボリュームマネージャの管理』では、可能なかぎり、1 つの記憶装置構成に基づいた使用例を示しながら、説明を進めます。この章では、例で使用するシナリオについて説明します。また、以降の各章で使用する記憶装置の初期構成について、詳しく説明します。

この章では、次の内容について説明します。

シナリオの背景情報

このマニュアルに示されているさまざまなシナリオや使用例の多くは、1 つの構成に基づいています。この構成自体は説明を簡単にするために小規模なものになっていますが、その概念はより大規模な記憶装置環境にも適用できます。

ハードウェア構成

ハードウェアシステムは、次のように構成されているとします。

- 物理的に分離された 3 つのコントローラが使用されている (`c0` – IDE、`c1` – SCSI、および `c2` – SCSI)。
- 各 SCSI コントローラは、6 つの内臓 9G バイトディスク (`c1t1` から `c1t6` と `c2t1` から `c2t6`) を持つ MultiPack に接続されている。これによって、ミラー構成が作成される。
- 個々のコントローラ/ターミネータペア (c*n*t*n*) の使用可能な記憶容量は 8.49G バイトである。
- ルート (`/`) ドライブ `c0t0d0` の記憶領域は 6 つのパーティションに分割されている。

この構成を図に示すと次のようになります。

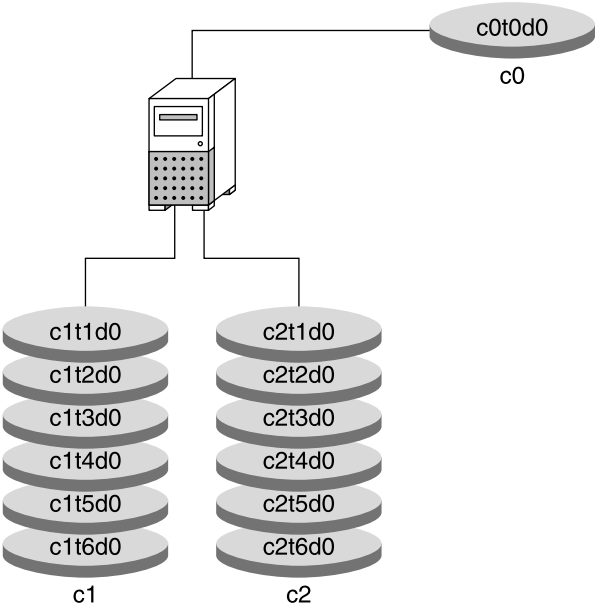

図 5–1 記憶装置のシナリオによる基本的なハードウェア図

物理的記憶領域の初期構成

Solaris ボリュームマネージャを構成する前の記憶装置構成を示します。

- SCSI コントローラ/ターミネータペア (*cntn*) の記憶容量はおよそ 20G バイトである。
- 各ディスク (たとえば、`c1t1d0`) の記憶領域は、7 つのパーティションに分割されている (*cntn*`d0s0` から *cntn*`d0s6`)。

  ディスクをパーティションに分割する方法については、『Solaris のシステム管理 (デバイスとファイルシステム)』の「ディスクのフォーマット」を参照してください。

Solaris ボリュームマネージャの最終的な構成

このマニュアルでは、作業ごとに特定のシナリオが用意されています。しかし、このマニュアルで使用される例を理解しやすくするために、`metastat -p` コマンドが出力する最終構成は、ほぼ次のとおりになっています。

```
[root@lexicon:/]$ metastat -p
d50 -r c1t4d0s5 c1t5d0s5 c2t4d0s5 c2t5d0s5 c1t1d0s5 c2t1d0s5 -k -i 32b
d1 1 1 c1t2d0s3
d2 1 1 c2t2d0s3
d12 1 1 c1t1d0s0
```

```
d13 1 1 c2t1d0s0
d16 1 1 c1t1d0s1
d17 1 1 c2t1d0s1
d25 2 2 c1t1d0s3 c2t1d0s3 -i 32b \
         1 c0t0d0s3
d31 1 2 c1t4d0s4 c2t4d0s4 -i 8192b
d80 -p d70 -o 1 -b 2097152
d81 -p d70 -o 2097154 -b 2097152
d82 -p d70 -o 4194307 -b 2097152
d83 -p d70 -o 6291460 -b 2097152
d84 -p d70 -o 8388613 -b 2097152
d85 -p d70 -o 10485766 -b 2097152
d70 -m d71 d72 1
d71 3 1 c1t3d0s3 \
         1 c1t3d0s4 \
         1 c1t3d0s5
d72 3 1 c2t3d0s3 \
         1 c2t3d0s4 \
         1 c2t3d0s5
d123 -p c1t3d0s6 -o 1 -b 204800
d124 -p c1t3d0s6 -o 204802 -b 204800
d125 -p c1t3d0s6 -o 409603 -b 204800
d126 -p c1t3d0s7 -o 3592 -b 20480
d127 -p c2t3d0s7 -o 3592 -b 1433600
hsp010
hsp014 c1t2d0s1 c2t2d0s1
hsp050 c1t2d0s5 c2t2d0s5
hsp070 c1t2d0s4 c2t2d0s4
```

`-p`オプションの詳細については、`metastat(1M)` コマンドを参照してください。

◆ ◆ ◆ 第 6 章

状態データベース (概要)

この章では、状態データベースの複製の概念について説明します。関連する作業の実行手順については、第 7 章を参照してください。

この章では、次の内容について説明します。

Solaris ボリュームマネージャの状態データベースと状態データベースの複製について

Solaris ボリュームマネージャの状態データベースには、すべてのボリューム、ホットスペア、およびディスクセットの構成と状態に関する情報が格納されています。Solaris ボリュームマネージャは、冗長性を確保し、システムクラッシュ時のデータの損失を防止するために、複数の状態データベースのコピー (複製) を保持しています (データクラッシュ時に損傷を受けるデータベースのコピーはせいぜい 1 つです)。

状態データベースの複製は、状態データベースのデータが常に有効であることを保証します。状態データベースが更新されると、個々の状態データベースの複製も更新されます。ただし、システムクラッシュによってすべての更新が失われるのを防ぐために、更新は一度に 1 つずつ行われます。

システムから 1 つの状態データベースの複製が失われると、Solaris ボリュームマネージャは、どの状態データベースの複製に有効なデータが格納されているかを判断する必要があります。そのために、Solaris ボリュームマネージャは多数決アルゴリズムを使用します。このアルゴリズムでは、過半数 (半数 + 1) の複製が使用可能であり、一致していれば、それらの複製を有効であるとみなします。この多数決アルゴリズムの要件とし

て、ディスク構成を設定するときに、3 つ以上の状態データベースの複製を作成する必要があります。3 つの状態データベースの複製のうち少なくとも 2 つが有効であれば、コンセンサスが得られたことになります。

ブート時には、Solaris ボリュームマネージャは、損傷した状態データベースの複製を無視します。場合によっては、損傷した複製を Solaris ボリュームマネージャが作成し直すこともありますが、そうでなければ、そのような複製は管理者が修正するまで無視されます。使用しているスライスに障害が発生して複製が損傷した場合は、スライスを修理または交換してから複製を有効にする必要があります。

注意 – ファブリック接続型記憶領域 (FAS)、SAN などの、システムに直接接続されていない記憶領域に、状態データベースの複製を格納しないでください。Solaris ボリュームマネージャをブートできなくなる可能性があります。複製は、従来の SCSI または IDE ドライブにブートプロセス時に使用できる記憶デバイスに格納しなければなりません。

すべての状態データベースの複製が失われると、理論的には、Solaris ボリュームマネージャのボリュームに格納されているすべてのデータが失われます。そのため、十分な数の複製を別々のドライブとコントローラに分散させて作成し、最悪の事態を回避するようにします。さらに、最初の Solaris ボリュームマネージャ構成情報とディスクパーティション情報を保存しておくのも良い方法です。

システムに状態データベースの複製を追加する方法については、第 7 章を参照してください。状態データベースの複製が失われた場合に回復する方法については、307 ページの「状態データベースの複製の障害からの回復」を参照してください。

状態データベースの複製は、RAID-1 ボリュームの再同期領域でも使用されます。ミラーの数に比べて状態データベースの複製の数が少なすぎると、複製の入出力が RAID-1 ボリュームの性能に影響を与えることがあります。ミラーの数が多い場合は、RAID-1 ボリューム当たり少なくとも 2 つの状態データベースの複製 (ディスクセット当たりの複製の最大数は 50) を用意してください。

デフォルトでは、ボリューム、ローカルセット、およびディスクセットの各状態データベースの複製は 4M バイト (8192 個のディスクセクター) のディスク記憶領域を使用します。複数所有者ディスクセットの状態データベースの複製のデフォルトサイズは 16M バイトです。

複製は、次のデバイスに格納できます。

- 専用のローカルディスクパーティション
- ボリュームの一部となるローカルパーティション
- UFS ロギングデバイスの一部となるローカルパーティション

ルート (/)、 `swap`、または `/usr` スライスに複製を格納することはできません。また、ファイルシステムまたはデータがすでに存在するスライスに複製を格納することもできません。ただし、複製を格納した後で、同じスライスにボリュームやファイルシステムを置くことができます。

多数決アルゴリズムとは

複製されたデータベース特有の問題は、どのデータベースが有効でデータが正しいかの判別が困難だということです。この問題を解決するために、Solaris ボリュームマネージャでは多数決アルゴリズムが使用されます。このアルゴリズムでは、過半数の複製のコンセンサスが得られないかぎり、いずれの複製も有効なものとみなされません。したがって、このアルゴリズムでは、始めから 3 つ以上の複製が存在していなければなりません。3 つの複製のうち少なくとも 2 つが使用可能であれば、コンセンサスが得られたことになります。一方、複製が 1 つしか存在していないときに、システムがクラッシュすると、すべてのボリューム構成データが失われてしまう可能性があります。

データを保護するために、Solaris ボリュームマネージャは、半数の複製が使用可能でなければ、動作しません。これにより、データの損傷が防止されます。

多数決アルゴリズムによって、システムは多数決アルゴリズムに従って次のように動作します。

- 状態データベースの複製の半分以上が使用可能であれば、システムは引き続き動作する
- 使用可能な複製が半数を下回ると、システムはパニックを起こす
- システムは、過半数の複製が使用可能でなければ、マルチユーザーモードでリブートできない

使用可能な状態データベースの複製の数が足りない場合は、シングルユーザーモードでブートし、損傷または失われた複製を削除して、規定数を満たす有効な複製を確保する必要があります。307 ページの「状態データベースの複製数の不足から回復するには」を参照してください。

注 – 状態データベースの複製の総数が奇数の場合は、その値を 2 で割り、端数を切り捨てた整数値に 1 を加えることによって過半数値が計算されます。たとえば、複製が 7 つあるシステムの過半数値は 4 です (7 を 2 で割って端数を切り捨てると 3 になり、それに 1 を足すと 4 になる)。

状態データベースの複製管理

- 状態データベースの複製のデフォルトサイズは 4M バイト (8192 ブロック) です。状態データベースの複製は、複製ごとに 4M バイト以上の容量を持つ専用スライス上に作成します。ディスクスライスのサイズがこれより大きい場合は、状態データベースの複製を格納できるように、スライスのサイズを変更できます。スライスのサイズの変更については、『Solaris のシステム管理 (デバイスとファイルシステム)』の第 12 章「ディスクの管理 (手順)」を参照してください。
- シングルポイント障害を回避するために、複数のスライス、ドライブ、およびコントローラに状態データベースの複製を分散させてください。これは、単一のコンポーネントに障害が発生した場合でも、大半の複製を利用可能な状態に保つ必要があるからです。デバイス障害などによって複製が失われると、Solaris ボリュームマネージャの

動作やシステムのリブートに問題が生じることがあります。Solaris ボリュームマネージャが動作するためには、少なくとも半数の複製が有効でなければならず、システムをマルチユーザーモードでリブートするためには過半数 (半数+1) の複製が有効でなければなりません。

Solaris ボリュームマネージャではディスクセット当たり最低 3 つの複製を用意します。また最大 50 の複製を作成できます。次のガイドラインを推奨します。

- ドライブが 1 つだけのシステムでは、3 つの複製すべてを 1 つのスライスに置く
- ドライブの数が 2 から 4 のシステムでは、各ドライブに 2 つずつ複製を置く
- ドライブの数が 5 つ以上のシステムでは、各ドライブに 1 つずつ複製を置く

- 複数のコントローラが存在する場合は、複製をすべてのコントローラ上にできるだけ均一に分散するようにします。これによって、コントローラ障害に対する冗長性が確保できるだけでなく、負荷の分散も可能になります。同じコントローラ上に複数のディスクが存在する場合は、各コントローラで 2 つ以上のディスクに複製を配置します。
- 必要であれば、RAID-0、RAID-1、RAID-5 ボリューム、またはソフトパーティションの一部として使用されるスライス上に、状態データベースの複製を作成できます。ただし、その場合は、スライスをボリュームに追加する前に複製を作成する必要があります。Solaris ボリュームマネージャは、スライスの先頭部分を状態データベースの複製用に予約しています。

  ボリュームの一部となるスライス上に状態データベースの複製が置かれている場合、ボリュームの容量は、複製によって占有される領域分だけ少なくなります。複製が使用している領域は、次のシリンダ境界に丸められます。ボリュームはこの領域を飛び越します。
- RAID-1 ボリュームは、小さいランダム入出力 (データベースの場合など) に使用します。RAID-1 ボリュームごとに、その RAID-1 ボリュームに接続されていない複数のスライス (および、可能であれば別個のディスクとコントローラ) 上に 2 つ以上の複製を余分に作成します。これは、最適な性能を得るために必要な作業です。
- 状態データベースの複製を既存のファイルシステムや、ルート (/)、`/usr`、`swap` ファイルシステムに作成することはできません。必要に応じて、 `swap` から領域を割り当てることによって、(スライス名が使用できる場合) 新しいスライスを作成できます。その新しいスライスに状態データベースの複製を格納します。
- 未使用のスライス上に状態データベースの複製を作成できます。
- 状態データベースの複製は、いつでもシステムに追加できます。状態データベースの複製を追加すると、Solaris ボリュームマネージャの可用性が向上します。

注意 – Solstice DiskSuite 製品から Solaris ボリュームマネージャにアップグレードしていて、状態データベースの複製とファイルシステムまたは論理ボリュームの間でスライスが共有されている (それぞれが異なるスライス上に置かれていない) 場合は、既存の複製を削除して同じ場所に新しいデフォルトの複製を作成してはなりません。

Solaris ボリュームマネージャの状態データベースの複製のデフォルトサイズは 8192 ブロックですが、Solstice DiskSuite 製品のデフォルトサイズは 1034 ブロックです。Solstice DiskSuite 製品で作成されたデフォルトサイズの状態データベースの複製を削除し、Solaris ボリュームマネージャで新しくデフォルトサイズの複製を追加する場合は、注意してください。共有スライスの残りの部分を占有しているファイルシステムの先頭 7158 ブロックが上書きされるので、データが破壊されます。

状態データベースの複製のエラー処理

状態データベースの複製で障害が発生しても、残りの複製の半分以上が使用できる場合、システム動作は継続されます。使用可能な複製が半数を下回ると、パニックを起します。

システムは、過半数 (半数 + 1) が使用可能であれば、マルチユーザーモードでリブートできます。使用できる複製が過半数に満たない場合は、システムをシングルユーザーモードでリブートし、`metadb` コマンドを使って使用不能な複製を削除する必要があります。

たとえば、4 つの複製を使用しているとします。システムは、2 つの複製 (半数) が使用可能であれば動作を続けます。しかし、システムをマルチユーザーモードでリブートするためには、3 つの複製 (半数 + 1) が使用可能でなければなりません。

ディスクが 2 台の構成では、各ディスクに必ず 2 つ以上の複製を作成します。たとえば、ディスクが 2 台の構成で複製を 3 つしか作成しないとします (一方のディスクに 2 つの複製、他方のディスクに 1 つの複製を配置する)。2 つの複製からなるディスクで障害が発生すると、残りのディスクには複製が 1 つだけなので、システムはパニックを起こします。複製が 1 つということは、複製総数の半数に達しません。

注 – ディスクが 2 台の構成で各ディスクに 2 つずつ複製を作成すれば、一方のディスクに障害が発生しても、Solaris ボリュームマネージャは動作を続けます。しかし、システムのリブートには過半数の複製が必要なため、システムはリブートできません。

状態データベースの複製が格納されたスライスで障害が発生しても、構成の他の部分は動作し続けます。Solaris ボリュームマネージャは、ブート時に、過半数の状態データベースの複製が使用可能であれば、有効な状態データベースを探します。

状態データベースの複製を手動で修復して使用可能にすると、 Solaris ボリュームマネージャは有効なデータを使ってその複製を更新します。

シナリオ — 状態データベースの複製

状態データベースの複製は、Solaris ボリュームマネージャ構成全体に対してデータ冗長性を提供します。次の例は、第 5 章で示したシステムに基づいています。この例は、状態データベースの複製を分散させることによって十分な冗長性を確保する方法を示しています。

サンプルのシステムには、内蔵 IDE コントローラとドライブが 1 つずつ、SCSI コントローラが 2 つあります。各 SCSI コントローラには、6 つのディスクが接続されています。システムには 3 つのコントローラがあるため、システムを適切に構成することによって単一点障害を防止できます。Solaris ボリュームマネージャでは、2 つのコントローラしかないシステムで単一点障害を防止することはできません。複製を 3 つのコントローラすべてに (各コントローラの少なくとも 1 つ (可能であれば 2 つ) のディスクに) 均一に分散すれば、システムはどのようなハードウェアの単一点障害にも耐えることができます。

最小限の構成では、1 つの状態データベースの複製をルートディスクのスライス 7 に置き、追加の複製を他の 2 つの各コントローラの 1 つのディスク上のスライス 7 に置くことができます。媒体障害が発生する可能性は極めて低いですが、十分な安全性を求めるなら、ルートディスクに 複製をもう 1 つ追加し、各コントローラの 2 つのディスクにそれぞれ複製を置くようにします (合計で 6 つの複製)。

さらに安全性を高めるには、2 つのミラーの両側で、6 つのディスクに 12 の複製を追加して均一に分散させます。これによって、複製の数は、ルートディスクに 2 つ、各 SCSI コントローラに 8 つずつで合計 18 になります。これらの複製は、各コントローラのディスクに分散されています。

◆ ◆ ◆ 第 7 章

7 状態データベース (作業)

この章では、Solaris ボリュームマネージャの状態データベースの複製に関連する作業について説明します。これらの作業に関連する概念については、第 6 章を参照してください。

状態データベースの複製 (作業マップ)

次の表に、Solaris ボリュームマネージャの状態データベースの複製を管理するのに必要な作業を示します。

作業	説明	参照先
状態データベースの複製の作成	Solaris ボリュームマネージャの GUI か `metadb -a` コマンドを使って状態データベースの複製を作成します。	72 ページの「状態データベースの複製を作成するには」
状態データベースの複製の状態チェック	Solaris ボリュームマネージャの GUI か `metadb -a` コマンドを使って、既存の複製の状態をチェックします。	74 ページの「状態データベースの複製の状態をチェックするには」
状態データベースの複製の削除	Solaris ボリュームマネージャの GUI か `metadb -a` コマンドを使って状態データベースの複製を削除します。	75 ページの「状態データベースの複製を削除するには」

状態データベースの複製の作成

注意 – Solstice DiskSuite 製品から Solaris ボリュームマネージャにアップグレードしていて、状態データベースの複製とファイルシステムまたは論理ボリュームの間でスライスが共有されている (それぞれが異なるスライス上に置かれていない) 場合は、既存の複製を削除して同じ場所に新しいデフォルトの複製を作成してはなりません。

Solaris ボリュームマネージャの状態データベースの複製のデフォルトサイズは 8192 ブロックですが、Solstice DiskSuite 製品のデフォルトサイズは 1034 ブロックです。Solstice DiskSuite 製品で作成されたデフォルトサイズの状態データベースの複製を削除し、Solaris ボリュームマネージャで新しくデフォルトサイズの複製を追加する場合は、注意してください。共有スライスの残りの部分を占有しているファイルシステムの先頭 7158 ブロックが上書きされるので、データが破壊されます。

注意 – ファブリック接続型記憶領域 (FAS)、SAN などの、システムに直接接続されていない記憶領域に、状態データベースの複製を格納しないでください。Solaris ボリュームマネージャをブートできなくなる可能性があります。複製は、従来の SCSI または IDE ドライブにブートプロセス時に使用できる記憶デバイスに格納しなければなりません。

▼ 状態データベースの複製を作成するには

始める前に 49 ページの「Solaris ボリュームマネージャコンポーネントを作成するための前提条件」を確認します。

1. スーパーユーザーになります。
2. 次のどちらかの方法で状態データベースの複製を作成します。
   - Solaris 管理コンソール内の「拡張ストレージ」から「状態データベースの複製」ノードを開きます。「アクション (Action)」、「複製の作成 (Create Replicas)」の順に選択し、画面の指示に従います。詳細は、オンラインヘルプを参照してください。
   - 次の形式の `metadb` コマンドを実行します。`metadb(1M)` のマニュアルページを参照してください。

     `# metadb -a -c` *number* **`-l`** *length-of replica* **`-f`** *ctds-of-slice*

     `-a` 状態データベースの複製を追加または作成することを指定します。

     `-f` 複製が存在しなくても、強制的に操作を実行することを指定します。最初の複製を強制的に作成するには、`-f` フラグを指定します。

     `-c` *number* 特定のスライスに追加する複製の数を指定します。

-l *length-of-replica*	新しい複製のサイズをブロック数で指定します。デフォルトのサイズ (8192) は、数千の論理ボリュームを持つ構成を含め、事実上すべての構成に適しています。
ctds-of-slice	複製を格納するコンポーネントの名前を指定します。

注 – コマンド行にオプションを指定しないで metadb コマンドを入力すると、すべての状態データベースの複製の状況が示されます。

例 7–1 最初の状態データベースの複製を作成する

```
# metadb -a -f c0t0d0s7
# metadb
        flags           first blk       block count
...
     a      u           16              8192            /dev/dsk/c0t0d0s7
```

最初の状態データベースの複製を作成するには、-f オプションを -a オプションとともに使用する必要があります。-a オプションは、状態データベースの複製をシステムに追加します。- f オプションは、最初の複製を強制的に作成します (システムに補助的な複製を追加する場合は省略可能)。

例 7–2 2 つの状態データベースの複製を同じスライスに追加する

```
# metadb -a -c 2 c1t3d0s1
# metadb
        flags           first blk       block count
...
     a        u          16              8192            /dev/dsk/c1t3d0s1
     a        u          8208            8192            /dev/dsk/c1t3d0s1
```

-a オプションは、状態データベースの複製をシステムに追加します。-c 2 オプションは、指定したスライスに 2 つの複製を格納します。metadb コマンドは、複製が metadb コマンド出力の a フラグで示されたとおり、アクティブであるかどうかを調べます。

例 7–3 指定したサイズの状態データベースの複製を追加する

既存の状態データベースの複製を置き換える場合は、複製のサイズを指定しなければならない場合があります。特に、ファイルシステムとスライスを共有している状態データベースの複製がすでに存在している場合は (たとえば、Solstice DiskSuite 製品からアップグレードした場合など)、既存の複製を同じサイズの複製で置き換えるか、別の場所に新しい複製を作成する必要があります。

```
# metadb -a -c 3 -l 1034 c0t0d0s7
# metadb
```

```
        flags         first blk      block count
...
     a      u         16             1034           /dev/dsk/c0t0d0s7
     a      u         1050           1034           /dev/dsk/c0t0d0s7
     a      u         2084           1034           /dev/dsk/c0t0d0s7
```

-a オプションは、状態データベースの複製をシステムに追加します。-l オプションでは、追加する複製の長さをブロック数で指定します。

状態データベースの複製の保守

▼ 状態データベースの複製の状態をチェックするには

1. スーパーユーザーになります。
2. 次のどちらかの方法で状態データベースの複製の状態をチェックします。
   - Solaris 管理コンソール内の「拡張ストレージ」から「状態データベースの複製 (State Database Replicas)」ノードを開いて、存在するすべての状態データベースの複製の状態を表示します。詳細は、オンラインヘルプを参照してください。
   - metadb コマンドを実行して、状態データベースの複製の状態を表示します。-i オプションを指定すると、すべての状態フラグについての説明が表示されます (次の例を参照)。metadb(1M) のマニュアルページを参照してください。

例 7–4 すべての状態データベースの複製の状態をチェックする

```
# metadb -i
        flags           first blk       block count
     a m  p  luo        16              8192            /dev/dsk/c0t0d0s7
     a    p  luo        8208            8192            /dev/dsk/c0t0d0s7
     a    p  luo        16400           8192            /dev/dsk/c0t0d0s7
     a    p  luo        16              8192            /dev/dsk/c1t3d0s1
      W   p  l          16              8192            /dev/dsk/c2t3d0s1
     a    p  luo        16              8192            /dev/dsk/c1t1d0s3
     a    p  luo        8208            8192            /dev/dsk/c1t1d0s3
     a    p  luo        16400           8192            /dev/dsk/c1t1d0s3
 r - replica does not have device relocation information
 o - replica active prior to last mddb configuration change
 u - replica is up to date
 l - locator for this replica was read successfully
 c - replica's location was in /etc/lvm/mddb.cf
```

```
 p - replica's location was patched in kernel
 m - replica is master, this is replica selected as input
 W - replica has device write errors
 a - replica is active, commits are occurring to this replica
 M - replica had problem with master blocks
 D - replica had problem with data blocks
 F - replica had format problems
 S - replica is too small to hold current data base
 R - replica had device read errors
```

状態表示の後にすべてのフラグの説明が表示されます。デバイス名の前の文字はデバイスの状態を示します。大文字は障害状態を、小文字は「Okay」状態を示します。

▼ 状態データベースの複製を削除するには

Solaris ボリュームマネージャ構成を保守するために、状態データベースの複製を削除しなければならない場合があります。たとえば、ディスクドライブを交換する場合、ドライブを取り外す前に、状態データベースの複製を削除します。そうしないと、Solaris ボリュームマネージャからエラーが報告されます。

1. スーパーユーザーになります。
2. 次のどちらかの方法で状態データベースの複製を削除します。
   - Solaris 管理コンソール内の「拡張ストレージ」から「状態データベースの複製 (State Database Replicas)」ノードを開いて、存在するすべての状態データベースの複製の状態を表示します。次に、削除する複製を選択してから「編集 (Edit)」、「削除 (Delete)」の順に選択して複製を削除します。詳細は、オンラインヘルプを参照してください。
   - 次の形式の `metadb` コマンドを実行します。

     # metadb -d -f *ctds-of-slice*

     `-d` 状態データベースの複製の削除を指定します。

     `-f` 複製が存在しなくても、強制的に操作を実行することを指定します。

     ctds-of-slice 複製が格納されているコンポーネント名を指定します。

     削除したい状態データベースの複製が格納されているすべてのスライスを指定する必要があります。詳細は、`metadb(1M)` のマニュアルページを参照してください。

例 7–5 状態データベースの複製を削除する

```
# metadb -d -f c0t0d0s7
```

この例では、スライスから最後の複製を削除します。

システム上にある最後の複製を強制的に削除するには、-f オプションを指定する必要があります。

◆ ◆ ◆ 第 8 章

RAID-0 (ストライプ方式および連結方式) ボリューム (概要)

この章では、Solaris ボリュームマネージャで使用できる RAID-0 ボリューム (ストライプ方式と連結方式) について説明します。関連する作業については、第 9 章を参照してください。

この章では、次の内容について説明します。

RAID-0 ボリュームの概要

RAID-0 ボリュームは、スライスまたはソフトパーティションで構成されます。これらのボリュームによって、ディスク記憶容量を拡張できます。RAID-0 ボリュームは、そのまま使用することもできますが、RAID-1 (ミラー) ボリュームおよびソフトパーティションの構成要素として使用することもできます。RAID-0 ボリュームには、次の 3 種類があります。

- ストライプ方式ボリューム
- 連結方式ボリューム
- 連結ストライプ方式ボリューム

注 – コンポーネントとは、別の論理ボリュームの中で使用されるすべてのデバイス (スライスからソフトウェアパーティションまで) を意味します。

ストライプ方式ボリュームは、データをボリュームのすべてのコンポーネントに均一に分散させますが、連結方式ボリュームはデータを、使用可能な最初のコンポーネントに書き込み、そのコンポーネントが満杯になると、次の使用可能なコンポーネントに書き込みます。連結ストライプ方式ボリュームは、単に、コンポーネントの追加によって元の構成が拡張されたストライプ方式ボリュームです。

RAID-0 ボリュームでは、ディスクの記憶容量をすばやく簡単に拡張できます。ただし、RAID-1 や RAID-5 ボリュームとは異なり、RAID-0 ボリュームにはデータの冗長性はありません。したがって、RAID-0 ボリュームのコンポーネントに 1 つでも障害が発生すると、データは失われます。

ストライプ方式ボリュームで順次入出力操作を行うと、Solaris ボリュームマネージャは、先頭のコンポーネントからブロックセグメント (飛び越しと呼ぶ) 1 つ分のブロックを読み取り、次に 2 番目のコンポーネントのブロックセグメント 1 つ分のブロックを読み取るという処理を繰り返します。

連結方式ボリュームの順次入出力操作では、Solaris ボリュームマネージャは、先頭のコンポーネントからすべてのブロックを最初に読み取り、次に 2 番目のコンポーネントのすべてのブロックを読み取るという処理を繰り返します。

連結方式ボリュームとストライプ方式ボリュームはどちらも、すべての入出力操作が並行して行われます。

1 つのスライスを含む RAID-0 ボリュームは、任意のファイルシステムに使用できます。

複数のコンポーネントを含む RAID-0 ボリュームは、次のファイルシステムを除く、任意のファイシステムに使用できます。

- ルート (`/`)
- `/usr`
- `swap`
- `/var`
- `/opt`
- オペレーティングシステムのアップグレードやインストール時にアクセスされるファイルシステム

注 – ルート (`/`)、`/usr` 、`swap`、`/var`、 `/opt` のミラーを作成する場合は、このファイルシステムをサブミラーとして機能する、1 面の連結またはストライプ (1 つのスライスで構成される単純連結) に置きます。そして、この 1 面の連結 (サブミラー) を別のサブミラーでミラー化します。このサブミラーも連結でなければなりません。

RAID-0 (ストライプ方式) ボリューム

RAID-0 (ストライプ方式) ボリュームは、1 つ以上のコンポーネントにデータが配置されるボリュームです。ストライプ方式では、同じサイズのデータセグメントが 2 つ以上のコンポーネントに順に配置され、1 つの論理記憶ユニットが構成されます。これらのセグメントはラウンドロビン (巡回的な) 方式でインターリーブされ、領域は各コンポーネントからメタデバイスに交互に割り当てられます。

注 – ストライプ方式ボリュームの容量を拡張するためには、連結ストライプ方式ボリュームを作成する必要があります。81 ページの「RAID-0 (連結ストライプ方式) ボリューム」を参照してください。

ストライプ方式では、複数のコントローラがデータに同時にアクセスできます (並列アクセス)。並列アクセスではボリュームのほとんどのディスクが入出力要求の処理でビジーになるため、入出力スループットが向上します。

既存のファイルシステムをストライプに直接変換することはできません。既存のファイルシステムをストライプ方式ボリュームに置くには、ファイルシステムのバックアップを取り、ボリュームを作成してから、ファイルシステムをストライプ方式ボリュームに復元する必要があります。

RAID–0 (ストライプ方式) ボリュームの飛び越し値

飛び越し値は、ストライプ方式ボリューム上の論理データセグメントのサイズで、K バイト、M バイト、またはブロック数で表わします。アプリケーションによっては、飛び越し値を変えることによって性能が向上することがあります。性能の向上は、入出力要求をいくつかのディスクアームを使って管理することによって達成されます。性能の向上が期待できるのは、入出力要求が飛び越し値よりも大きい場合です。

注 – RAID-5 ボリュームも飛び越し値を使用します。詳細については、165 ページの「RAID-5 ボリュームの概要」を参照してください。

飛び越し値は、ストライプ方式ボリュームを作成するときに設定できます。あるいは、Solaris ボリュームマネージャのデフォルト値である 16K バイトを使用することもできます。ただし、ストライプ方式ボリュームを作成した後では飛び越し値を変更できません。飛び越し値を変更するときは、データのバックアップを取り、ストライプ方式ボリュームを削除し、新しい飛び越し値で新しいストライプ方式ボリュームを作成してから、データを復元します。

シナリオ — RAID-0 (ストライプ方式) ボリューム

図 8–1 に、3 つのコンポーネント (スライス) からなるストライプ方式ボリュームを示します。ラウンドロビン方式で、飛び越し値に基づいてボリュームコンポーネントにデータを書き込む方法も示します。

Solaris ボリュームマネージャは、ストライプ方式ボリュームのコンポーネントにデータを書き込むときに、飛び越し値の幅のデータブロックを ディスク A (飛び越し 1)、ディスク B (飛び越し 2)、および ディスク C (飛び越し 3) に書き込みます。Solaris ボリュームマネージャはこのパターンをさらに繰り返して、ディスク A (飛び越し 4)、ディスク B (飛び越し 5)、ディスク C (飛び越し 6) に書き込みます (以下同様)。

飛び越し値によって、スライスに 1 回に書き込むデータのサイズが設定されます。ストライプ方式ボリュームの合計容量は、最小コンポーネントのサイズにコンポーネント数を掛けた値です。(次の例に示す各スライスのサイズが 2G バイトであれば、ボリュームは 6G バイトです)。

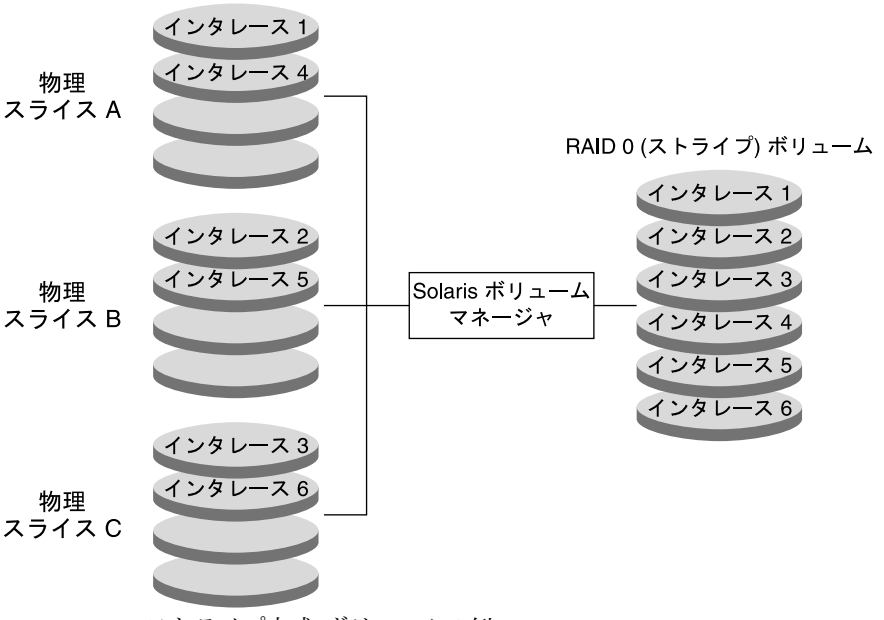

図 8–1 RAID-0 (ストライプ方式) ボリュームの例

RAID-0 (連結方式) ボリューム

RAID–0 (連結方式) ボリュームは、個々のコンポーネントにデータを隣接して順番に配置し、1 つの論理記憶ユニットを形成します。

連結方式ボリュームでは、いくつかのコンポーネントの容量を結合することによって記憶容量を拡張します。したがって、記憶容量の要件に応じてコンポーネントを連結方式ボリュームに追加できます。

連結方式ボリュームでは、記憶容量やファイルシステムのサイズをオンライン状態のまま動的に拡張できます。連結方式ボリュームにコンポーネントを追加するときは、他のコンポーネントがアクティブであってもかまいません。

また、連結方式ボリュームでは、システムを停止しなくても、マウントされている動作中の UFS ファイルシステムを拡張できます。通常、連結方式ボリュームの合計容量は、ボリューム内の全コンポーネントの合計サイズと同じです。ただし、連結方式ボリュームに状態データベースの複製を格納するスライスが含まれている場合には、ボリュームの合計容量は、コンポーネントの合計から複製に予約されている領域を差し引いたものです。

連結方式ボリュームは、1 つのコンポーネントから作成することもできます。記憶容量が必要になってから、ボリュームにコンポーネントを追加できます。

注 – ルート (/)、swap、/usr、 /opt、または /var ファイルシステムをミラー化する場合は、連結方式ボリュームを使ってこれらのファイルシステムをカプセル化する必要があります。

シナリオ — RAID-0 (連結方式) ボリューム

図 8–2 に、3 つのコンポーネント(スライス) からなる連結方式ボリュームを示します。飛び越し値に基づき、各スライスに順番に書き込むことによって、ボリュームコンポーネントにデータを書き込む方法も示します。

データブロックは、スライス A から各コンポーネントに順次書き込まれます。スライス A には論理データブロック 1 から 4 があり、ディスク B には論理データブロック 5 から 8 が格納され、ドライブ C には論理データブロック 9 から 12 が格納されます。ボリュームの総容量は、3 つのスライスの容量の合計です。したがって、各スライスの容量が 2G バイトであれば、ボリューム全体の容量は 6G バイトになります。

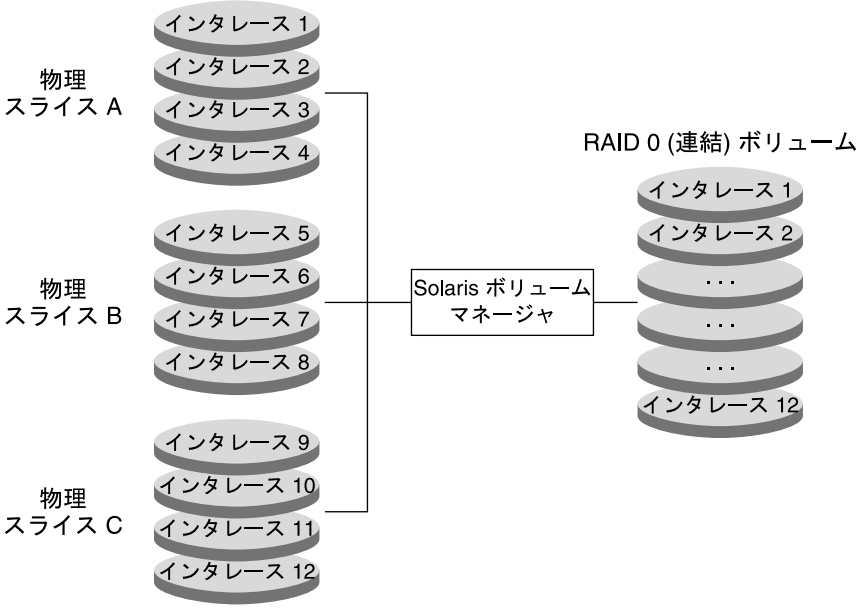

図 8–2 RAID-0 (連結方式) ボリュームの例

RAID-0 (連結ストライプ方式) ボリューム

RAID–0 (連結ストライプ方式) ボリュームは、コンポーネント (ストライプ方式) を追加することによって拡張されたストライプです。

連結ストライプ方式ボリュームの飛び越し値をストライプレベルで設定する場合は、Solaris 管理コンソール内の「拡張ストレージ」か `metattach -i` コマンドを使用します。連結ストライプ方式ボリュームの各ストライプには、別々の飛び越し値を設定することができます。連結ストライプ方式ボリュームを新たに作成する場合は、特定のストライプに飛び越し値を指定しないと、ボリュームに追加された直前のストライプの飛び越し値が使用されます。

例 —RAID-0 (連結ストライプ方式) ボリューム

図 8–3 に、3 つのストライプの連結である、連結ストライプ方式ボリュームを示します。

最初のストライプは 3 つのスライス (A から C) から構成され、飛び越し値は16K バイトです。2 つめのストライプは 2 つのスライス (D と E) から構成され、飛び越し値は 32K バイトです。最後のストライプは 2 つのスライス (F とG) から構成されています。このストライプには飛び越し値が指定されていないため、その前に追加されたストライプの飛び越し値 (32K バイト) が継承されます。データブロックはまず最初のストライプに順次追加されます。このストライプが満杯になると、2 つめのストライプにデータブロックが追加されます。さらに、このストライプが満杯になると、データブロックは 3 つめのストライプに追加されます。各ストライプでは、データブロックが、指定された飛び越し値に基づいてインタリーブされます。

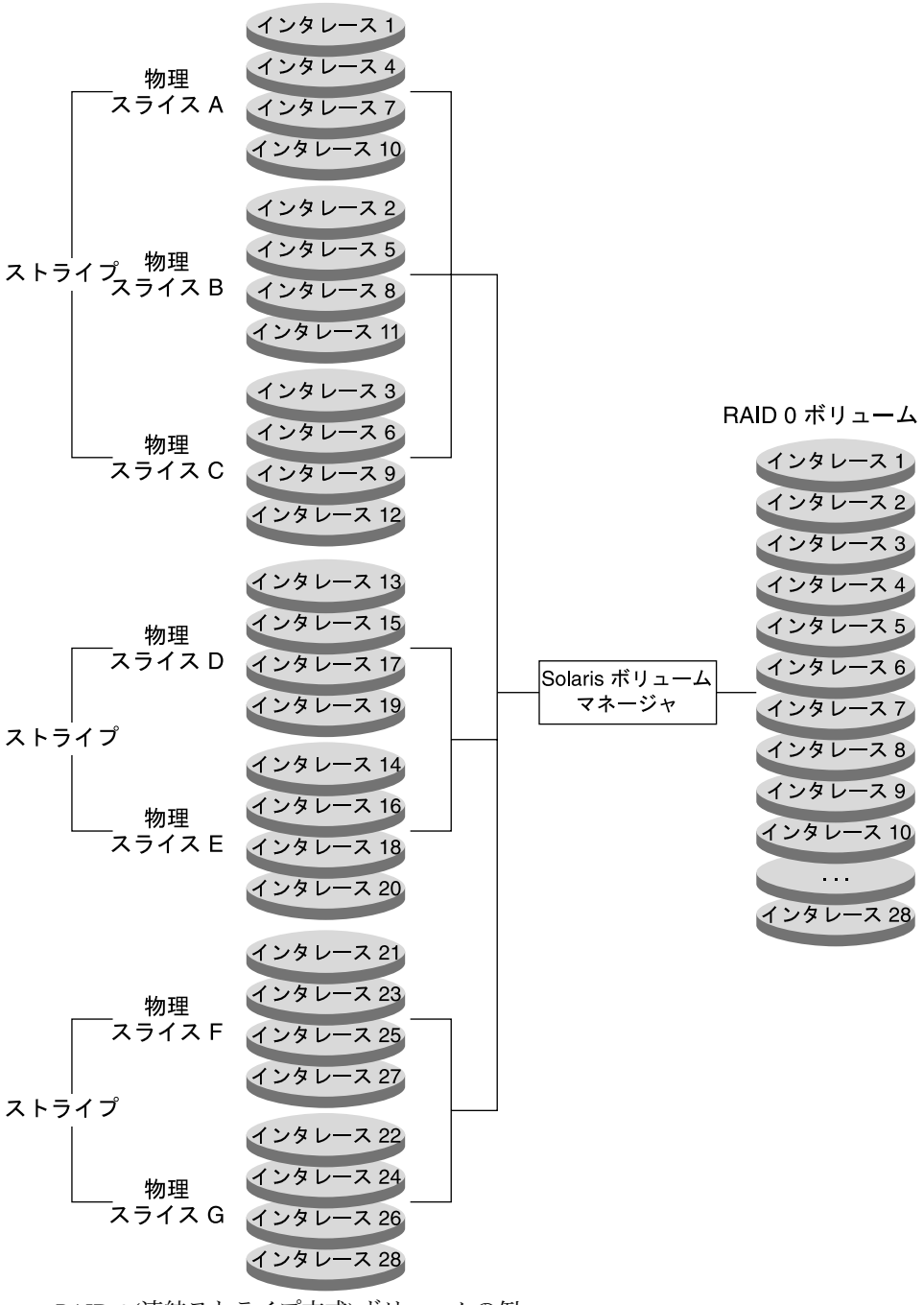

図 8–3 RAID-0 (連結ストライプ方式) ボリュームの例

RAID-0 ボリュームを作成するための背景情報

RAID-0 ボリュームの要件

RAID-0 ボリュームを使用するときは、次のことを考慮してください。

- コンポーネントを異なるコントローラに置くと、同時に実行できる読み取りや書き込みの数が増します。
- 既存のファイルシステムまたはデータからストライプを作成しないでください。作成するとデータが破壊されます。代わりに、連結方式を使用します。(既存のデータからストライプを作成することは可能ですが、その場合には、データのバックアップを取り、それをボリュームに復元する必要があります。)
- ストライプには、同じサイズのディスクコンポーネントを使用します。異なるサイズのコンポーネントをストライプに使用すると、ディスク領域が無駄になります。
- システムやアプリケーションの入出力要求に合わせてストライプの飛び越し値を設定します。
- ストライプ方式や連結方式には複製データが含まれないため、このようなボリュームのコンポーネントに障害が発生した場合には、そのコンポーネントを交換し、ストライプまたは連結を作成し直してから、バックアップのデータを復元する必要があります。
- ストライプや連結を再作成する場合には、障害が発生したコンポーネントと同じサイズのコンポーネントを新しいコンポーネントとして使用します。

RAID-0 ボリュームの指針

- 連結方式は、ストライプ方式ほど CPU サイクルを必要としません。連結方式は、小規模のランダム入出力や均一に分散された入出力に適しています。
- 可能であれば、ストライプ方式や連結方式のコンポーネントを異なるコントローラやバスに配置します。個々のストライプを異なるコントローラに置くと、同時に実行できる読み取りや書き込みの数が増えます。
- ストライプが置かれているコントローラに障害が発生した場合、別のコントローラが使用可能であれば、ディスクをコントローラに移動し、ストライプを再設定することによって、ストライプを新しいコントローラに「移動」できます。
- ディスクの数: ストライプの構成を性能要件に基づいて決める場合もあります。たとえば、特定のアプリケーションで 10.4M バイト/秒の性能を必要とする場合で、各ディスクの性能が約 4M バイト/秒であるとします。ストライプに必要なディスクスピンドルの数は、次の式で計算できます。

  ```
  10.4 Mbyte/sec / 4 Mbyte/sec = 2.6
  ```

  この例では、入出力操作を並行して行なうために 3 つのディスクが必要です。

シナリオ—RAID-0 ボリューム

RAID-0 ボリュームは、より複雑な記憶装置構成を実現する、またはミラーを構築するための基本構成ブロックになります。次の例では、第 5 章のシナリオを使用して、記憶領域の拡張や既存のファイルシステム (ルート (/) を含む) のミラー化に RAID-0 ボリュームをどのように使用できるかを示します。

シナリオのシステムには、比較的小さい (9G バイト) ディスクの集合が与えられていますが、アプリケーションによっては、それ以上の記憶領域が必要です。領域を拡張する (そして性能を高める) には、複数のディスクに渡るストライプを作成します。たとえば、`c1t1d0`、`c1t2d0`、`c1t3d0`、`c2t1d0`、`c2t2d0`、および `c2t3d0` を、これらのディスク全体にまたがるスライス `0` として構成できます。これによって、同じコントローラの 3 つのディスクからなるストライプはおよそ 27G バイトの記憶領域となり、アクセスも速くなります。2 つめのコントローラの 2 つめのストライプは、冗長性を確保する目的で使用できます。第 11 章と、具体的には 107 ページの「シナリオ—RAID-1 ボリューム (ミラー)」を参照してください。

◆ ◆ ◆ 第 9 章

RAID-0 (ストライプ方式および連結方式) ボリューム (作業)

この章では、RAID-0 ボリュームに関連する作業について説明します。関連する概念については、第 8 章を参照してください。

RAID-0 ボリューム (作業マップ)

次の表に、Solaris ボリュームマネージャの RAID-0 ボリュームを管理するために必要な作業を示します。

作業	説明	参照先
RAID−0 (ストライプ方式) ボリュームの作成	Solaris ボリュームマネージャの GUI か `metainit` コマンドを使って新しいボリュームを作成します。	88 ページの「RAID-0 (ストライプ方式) ボリュームを作成するには」
RAID−0 (連結方式) ボリュームの作成	Solaris ボリュームマネージャの GUI か `metainit` コマンドを使って新しいボリュームを作成します。	90 ページの「RAID-0 (連結方式) ボリュームを作成するには」
記憶領域の拡張	Solaris ボリュームマネージャの GUI か `metainit` コマンドを使って既存のファイルシステムを拡張します。	91 ページの「既存のデータの記憶容量を拡張するには」
既存の RAID-0 ボリュームの拡張	Solaris ボリュームマネージャの GUI か `metattach` コマンドを使って既存のボリュームを拡張します。	93 ページの「既存の RAID-0 ボリュームを拡張するには」
RAID−0 ボリュームの削除	Solaris ボリュームマネージャの GUI か `metainit` コマンドを使ってボリュームを削除します。	95 ページの「RAID-0 ボリュームを削除するには」

RAID-0 (ストライプ方式) ボリュームの作成

注意 – 既存のファイルシステムまたはデータからストライプを作成しないでください。作成するとデータが破壊されます。既存のデータからストライプを作成する場合は、データのバックアップを作成し、ストライプ方式ボリュームを作成して、そのボリュームにデータを復元する必要があります。

注意 – 32 ビットカーネルの Solaris ソフトウェアを使用する予定がある場合は、1T バイトを超えるボリュームを作成しないでください。また、Solaris 9 4/03 リリース以前の Solaris OS を使用する場合も、1T バイトを超えるボリュームを作成しないでください。Solaris ボリュームマネージャでサポートされる大容量ボリュームの詳細については、49 ページの「Solaris ボリュームマネージャにおけるマルチテラバイトサポートの概要」を参照してください。

▼ RAID-0 (ストライプ方式) ボリュームを作成するには

始める前に 49 ページの「Solaris ボリュームマネージャコンポーネントを作成するための前提条件」と 84 ページの「RAID-0 ボリュームを作成するための背景情報」を確認します。

◗ 次のどちらかの方法でストライプ方式ボリュームを作成します。

- Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開きます。「アクション (Action)」、「ボリュームを作成 (Create Volume)」の順に選択し、ウィザードの指示に従います。詳細は、オンラインヘルプを参照してください。
- 次の形式の `metainit` コマンドを使用します。

```
# metainit volume-name number-of-stripes
 components-per-stripe
component-names
[ -i interlace]
```

volume-name	作成するボリュームの名前を指定します。ボリュームの命名方式については、45 ページの「ボリューム名」を参照してください。
number-of-stripes	作成するストライプの数を指定します。
components-per-stripe	各ストライプに与えるコンポーネントの数を指定します。
component-names	使用するコンポーネントの名前を指定します。複数のコンポーネントを使用する場合は、スペースで各コンポーネントを区切ります。

`-i` *interlace*	ストライプに使用する飛び越し幅を指定します。飛び越し幅は、値の後ろに「k」(キロバイト)、「m」(メガバイト)、または「b」(ブロック) を加えて指定します。16 ブロック以上 100M バイト以下の飛び越し幅を指定する必要があります。デフォルトの飛び越し幅は 16K バイトです。

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

例 9–1 3 つのスライスからなる RAID–0 (ストライプ方式) ボリュームを作成する

```
# metainit d20 1 3 c0t1d0s2 c0t2d0s2 c0t3d0s2
d20: Concat/Stripe is setup
```

1 つのストライプ (数字の 1) からなるストライプ `d20` の例を示します。このストライプは 3 つのスライス (数字の 3) からなります。このストライプには飛び越し値が指定されていないため、デフォルト値の 16K バイトが使用されます。最後に、ボリュームが設定されたことを示すメッセージが出力されます。

例 9–2 32K バイトの飛び越し値を持つ 2 つのスライスからなる RAID-0 (ストライプ方式) ボリュームを作成する

```
# metainit d10 1 2 c0t1d0s2 c0t2d0s2 -i 32k
d10: Concat/Stripe is setup
```

1 つのストライプ (数字の 1) からなるストライプ `d10` の例を示します。このストライプは 2 つのスライス (数字の 2) からなります。`-i` オプションでは、飛び越し値として 32K バイトを設定します。飛び越し値は 8K バイト以上、100M バイト以下である必要があります。最後に、ボリュームが設定されたことを示すメッセージが出力されます。

参照 新たに作成したストライプにファイルシステムを作成する方法については、『Solaris のシステム管理 (デバイスとファイルシステム)』の第 18 章「UFS、TMPFS、LOFS ファイルシステムの作成 (手順)」を参照してください。データベースなど、アプリケーションによってはファイルシステムを使用しません。これらのアプリケーションでは、代わりに raw デバイスを使用します。アプリケーションは独自の方法で raw デバイスにアクセスできなければなりません。

RAID-0 (連結方式) ボリューム

▼ RAID-0 (連結方式) ボリュームを作成するには

注意 – 32 ビットカーネルの Solaris ソフトウェアを使用する予定がある場合は、1T バイトを超えるボリュームを作成しないでください。また、Solaris 9 4/03 リリース以前の Solaris OS を使用する場合も、1T バイトを超えるボリュームを作成しないでください。Solaris ボリュームマネージャのマルチテラバイトボリュームの詳細については、49 ページの「Solaris ボリュームマネージャにおけるマルチテラバイトサポートの概要」を参照してください。

始める前に 49 ページの「Solaris ボリュームマネージャコンポーネントを作成するための前提条件」と 84 ページの「RAID-0 ボリュームを作成するための背景情報」を確認します。

- 次のどちらかの方法で連結方式ボリュームを作成します。
  - Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開きます。「アクション (Action)」、「ボリュームを作成 (Create Volume)」の順に選択し、ウィザードの指示に従います。詳細は、オンラインヘルプを参照してください。
  - 次の形式の `metainit` コマンドを使用します。

    `# metainit` *volume-name number-of-stripes components-per-stripe component-names*

volume-name	作成するボリュームの名前を指定します。
number-of-stripes	作成するストライプの数を指定します。
components-per-concatenation	各連結に与えるコンポーネントの数を指定します。
component-names	使用するコンポーネントの名前を指定します。複数のコンポーネントを使用する場合は、スペースで各コンポーネントを区切ります。

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

例 9–3 1 つのスライスからなる連結を作成する

```
# metainit d25 1 1 c0t1d0s2
d25: Concat/Stripe is setup
```

連結 d25 を作成する例を示します。この連結は 1 つのストライプ (最初の数字 1) からなり、ストライプは 1 つのスライス (スライスの前の2 つめの数字 1) からなります。最後に、ボリュームが設定されたことを示すメッセージが出力されます。

既存のデータを安全にカプセル化できる連結の例を示します。

例 9–4 4 つのスライスからなる連結を作成する

```
# metainit d40 4 1 c0t1d0s2 1 c0t2d0s2 1 c0t2d0s3 1 c0t2d1s3
d40: Concat/Stripe is setup
```

連結 d40 を作成する例を示します。この連結は 4 つのストライプ (数字の 4) からなり、1 つのストライプは 1 つのスライス (各スライスの前の数字の 1) からなります。最後に、ボリュームが設定されたことを示すメッセージが出力されます。

参照 新たに作成した連結にファイルシステムを作成する方法については、『Solaris のシステム管理 (デバイスとファイルシステム)』の第 18 章「UFS、TMPFS、LOFS ファイルシステムの作成 (手順)」を参照してください。

記憶容量の拡張

ファイルシステムに記憶容量を追加するには、連結方式ボリュームを作成します。既存のストライプに記憶容量を追加する場合は、連結ストライプ方式ボリュームを作成します。

▼ 既存のデータの記憶容量を拡張するには

注意 – 32 ビットカーネルの Solaris ソフトウェアを使用する予定がある場合は、1T バイトを超えるボリュームを作成しないでください。また、Solaris 9 4/03 リリース以前の Solaris OS を使用する場合も、1T バイトを超えるボリュームを作成しないでください。Solaris ボリュームマネージャのマルチテラバイトボリュームサポートの詳細については、49 ページの「Solaris ボリュームマネージャにおけるマルチテラバイトサポートの概要」を参照してください。

始める前に 49 ページの「Solaris ボリュームマネージャコンポーネントを作成するための前提条件」と 84 ページの「RAID-0 ボリュームを作成するための背景情報」を確認します。

1 ファイルシステムをマウント解除します。

```
# umount /filesystem
```

2 次のどちらかの方法で連結を作成します。

- Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開きます。「アクション (Action)」、「ボリュームを作成 (Create Volume)」の順に選択し、ウィザードの指示に従います。詳細は、オンラインヘルプを参照してください。
- 次の形式の `metainit` コマンドを使用します。

  # **metainit** *volume-name number-of-stripes components-per-stripe component-names*

volume-name	作成するボリュームの名前を指定します。
number-of-stripes	作成するストライプの数を指定します。
components-per-stripe	各ストライプに与えるコンポーネントの数を指定します。
component-names	使用するコンポーネントの名前を指定します。複数のコンポーネントを使用する場合は、スペースで各コンポーネントを区切ります。

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

3 `/etc/vfstab` を編集して、このファイルシステムが連結の名前を参照するようにします。

4 ファイルシステムを再びマウントします。

**mount** */filesystem*

例 9–5 連結を作成してファイルシステムを拡張する

```
# umount /docs
# metainit d25 2 1 c0t1d0s2 1 c0t2d0s2
d25: Concat/Stripe is setup
    (/etc/vfstab ファイルを編集して、このファイルシステムがスライス c0t1d0s2 の代わりにボリューム d25 を参照するようにします。)
# mount /docs
```

2 つのスライス、`/dev/dsk/c0t1d0s2` (`/docs` にマウントされたファイルシステムが格納されている) と `/dev/dsk/c0t2d0s2` から連結 `d25` を作成する例を示します。ファイルシステムは、最初にマウント解除する必要があります。`metainit` コマンドに指定する最初のスライスは、ファイルシステムが格納されているスライスでなければなりません。そうでないと、データが破壊されます。

次に、`/etc/vfstab` ファイルにあるファイルシステムのエントリを、この連結を参照するように変更します (初めての場合は、入力します)。たとえば、最初は次のような行が `/etc/vfstab` ファイルにあります。

```
/dev/dsk/c0t1d0s2 /dev/rdsk/c0t1d0s2 /docs ufs 2 yes -
```

この行を次のように変更します。

```
/dev/md/dsk/d25 /dev/md/rdsk/d25 /docs ufs 2 yes -
```

最後に、ファイルシステムをマウントし直します。

参照　UFS ファイルシステムの場合は、連結に対して `growfs` コマンドを実行します。245 ページの「ファイルシステムを拡張するには」を参照してください。

データベースなど、アプリケーションによってはファイルシステムを使用しません。データベースなどのアプリケーションは、raw 連結を使用し、独自の方法でこの連結を認識するか、または領域を拡張できなければなりません。

▼ 既存の RAID-0 ボリュームを拡張するには

ストライプを連結することによって、既存のストライプを拡張できます。たとえば、ストライプの記憶容量が不足した場合は、連結ストライプ方式に変換します。これにより、データのバックアップや復元にわずらわされることなく、記憶容量を拡張できます。

この手順では、既存のストライプに別のストライプを追加するものとします。

注意 – 32 ビットカーネルの Solaris ソフトウェアを使用する予定がある場合は、1T バイトを超えるボリュームを作成しないでください。また、Solaris 9 4/03 リリース以前の Solaris OS を使用する場合も、1T バイトを超えるボリュームを作成しないでください。Solaris ボリュームマネージャのマルチテラバイトボリュームサポートの詳細については、49 ページの「Solaris ボリュームマネージャにおけるマルチテラバイトサポートの概要」を参照してください。

始める前に　49 ページの「Solaris ボリュームマネージャコンポーネントを作成するための前提条件」と 84 ページの「RAID-0 ボリュームを作成するための背景情報」を確認します。

◗ 次のどちらかの方法でストライプ連結を作成します。

- Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開きます。「アクション (Action)」、「ボリュームを作成 (Create Volume)」の順に選択し、ウィザードの指示に従います。詳細は、オンラインヘルプを参照してください。
- コマンド行から既存のストライプを連結する場合は、次の形式の `metattach` コマンドを使用します。

```
# metattach volume-name component-names
```

volume-name	作成するボリュームの名前を指定します。
component-names	使用するコンポーネントの名前を指定します。複数のコンポーネントを使用する場合は、スペースで各コンポーネントを区切ります。

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

例 9–6 1つのスライスを追加してストライプ連結を作成する

```
# metattach d2 c1t2d0s2
d2: components are attached
```

この例では、既存のストライプ `d2` にスライスを追加します。スライスが追加されたことを示すメッセージが表示されます。

例 9–7 複数のスライスを追加してストライプ連結を作成する

```
# metattach d25 c1t2d0s2 c1t2d1s2 c1t2d3s2
d25: components are attached
```

既存の 3 面ストライプ `d25` を使用し、それに別の 3 面ストライプを連結する例を示します。これら 3 つのスライスには飛び越し値が指定されていないので、ストライプは `d25` に設定された飛び越し値を引き継ぎます。最後に、ボリュームが設定されたことを示すメッセージが出力されます。

参照 UFS ファイルシステムの場合は、ボリュームに対して `growfs` コマンドを実行します。245 ページの「ファイルシステムを拡張するには」を参照してください。

データベースなど、アプリケーションによってはファイルシステムを使用しません。データベースなどのアプリケーションでは raw ボリュームを使用し、独自の方法でこのボリュームを認識したり、追加領域を拡張したりできなければなりません。

新たに作成したストライプ連結にファイルシステムを作成する方法については、『Solaris のシステム管理 (デバイスとファイルシステム)』の第 18 章「UFS、TMPFS、LOFS ファイルシステムの作成 (手順)」を参照してください。

RAID-0 ボリュームの削除

▼ RAID-0 ボリュームを削除するには

1. すべてのデータの最新のバックアップを取っているか確認します。また、この操作には root 権限が必要です。
2. このボリュームを本当に削除しても問題がないか確認します。
   ストライプまたは連結を削除し、そのボリュームの一部として使用されているスライスを再使用すると、ボリュームのすべてのデータがシステムから削除されます。
3. 必要であれば、ファイルシステムをマウント解除します。
   ```
   # umount /filesystem
   ```
4. 次のどちらかの方法でボリュームを削除します。
   - Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開きます。「編集」、「削除」の順に選択してから、画面上の指示に従います。詳細は、オンラインヘルプを参照してください。
   - 次の形式の metaclear コマンドを使ってボリュームを削除します。

     ```
     metaclear volume-name
     ```

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

例 9–8 連結を削除する

```
# umount d8
# metaclear d8
d8: Concat/Stripe is cleared
    (/etc/vfstab ファイルを編集します)
```

この例では、マウントされたファイルシステムが格納されている連結 d8 を削除する方法を示します。ボリュームを削除する場合は、あらかじめファイルシステムをマウント解除する必要があります。連結の削除が完了すると、そのことを示すメッセージが表示されます。 /etc/vfstab ファイルにこのボリュームのエントリがある場合は、このエントリを削除する必要があります。これによって、存在しないボリュームにファイルシステムをマウントするのを回避できます。

◆ ◆ ◆ 第 10 章

10 RAID-1 (ミラー) ボリューム (概要)

この章では、 Solaris ボリュームマネージャのミラーとサブミラーに関する基本的な概念について説明します。関連する作業の実行手順については、第 11 章を参照してください。

この章では、次の内容について説明します。

RAID-1 (ミラー) ボリュームの概要

RAID-1 ボリューム (またはミラー) とは、同じデータのコピーを複数の RAID-0 (ストライプ方式または連結方式) ボリュームで保持しているボリュームのことです。ミラー化された RAID-0 ボリュームをサブミラーと呼びます。ミラー化するためには、より多くのディスク容量が必要です。少なくとも、ミラー化するデータ量の 2 倍のディスク容量が必要になります。また、ミラー化ではデータがすべてのサブミラーに書き込まれるため、書き込み要求の処理時間が長くなります。

構成したミラーは、物理スライスと同じように使用できます。

既存のファイルシステムを含め、どのようなファイルシステムでもミラー化できます。これらのファイルシステムは、ルート (/)、swap、および /usr です。また、ミラーは、データベースなど、どのようなアプリケーションにも使用できます。

ヒント – データの安全性と可用性を確保するためには、Solaris ボリュームマネージャのホットスペア機能とミラーを併用します。ホットスペアについては、第 16 章と第 17 章を参照してください。

サブミラーの概要

ミラーは、サブミラーと呼ばれる 1 つ以上の RAID-0 ボリューム (ストライプ方式または連結方式) からなります。

ミラーには最大 4 つのサブミラーを使用できます。しかし通常、ほとんどのアプリケーションでは、2 面ミラーによって十分なデータ冗長性が得られますし、ディスクドライブのコストも低くなります。3 つめのサブミラーを構成すると、オンラインでバックアップを取ることができます。この場合、バックアップのために 1 つのサブミラーがオフラインになっていても、データの冗長性は失われません。

サブミラーを「オフライン」にすると、ミラーはそのサブミラーに対する読み書きを停止します。この時点で、このサブミラーへのアクセスが可能になり、バックアップを実行できます。ただし、オフライン状態のサブミラーは読み取り専用になります。サブミラーがオフライン状態の間、Solaris ボリュームマネージャはミラーに対するすべての書き込みを追跡管理します。サブミラーがオンライン状態に戻ると、サブミラーがオフラインの間に書き込まれた部分 (再同期領域) だけが再同期されます。また、サブミラーをオフラインにすると、エラーが発生した物理デバイスの問題を追跡したり修復したりすることが可能になります。

サブミラーは、いつでもミラーに接続したり、ミラーから切断できます。ただし、少なくとも 1 つのサブミラーが常時、接続されている必要があります。

通常は、サブミラーが 1 つだけのミラーを作成します。あとで 2 つめのサブミラーを追加します。

シナリオ—RAID-1 (ミラー) ボリューム

図 10–1に、ミラー `d20` を示します。このミラーは、2 つのボリューム (サブミラー) `d21` と `d22` からなります。

この例で Solaris ボリュームマネージャは、複数の物理ディスク上でデータを複製し、アプリケーションに 1 つの仮想ディスク `d20` を提供します。ディスクへの書き込みはすべて複製されます。ディスクからの読み取りはミラーを構成するサブミラーの 1 つから行われます。ミラー `d20` の容量は、もっとも小さいサブミラーのサイズと同じになります (サブミラーのサイズが異なる場合)。

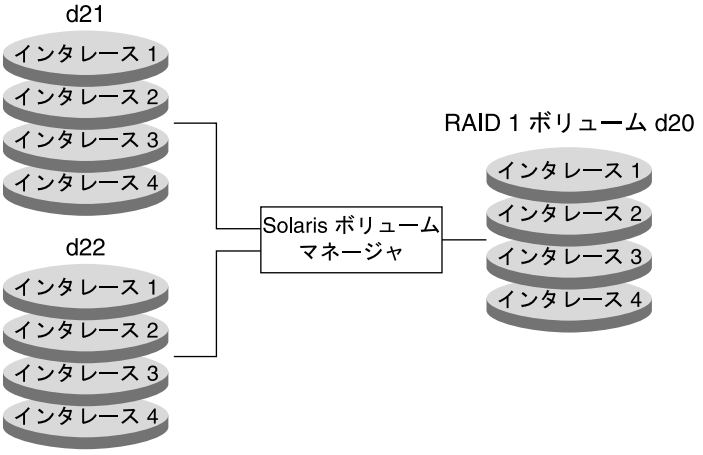

図 **10–1** RAID-1 (ミラー) の例

RAID-1+0 と RAID-0+1 の提供

Solaris ボリュームマネージャは、RAID-1+0 と RAID-0+1 の冗長性を両方ともサポートします。RAID-1+0 の冗長性は、ミラーの後にストライプ化される構成です。RAID-0+1 の冗長性は、ストライプの後にミラー化される構成です。Solaris ボリュームマネージャインタフェースは、すべての RAID-1 デバイスを RAID-0+1 として扱いますが、可能であれば、ボリュームを構成するコンポーネントやミラーを個別に認識します。

注 – Solaris ボリュームマネージャは、RAID-1+0 機能を常に提供できるわけではありません。しかし、両方のサブミラーが同じで、(ソフトパーティションではなく) ディスクスライスで構成されている場合、RAID-1+0 は可能です。

---

3 つのストライプ化されたスライスからなる 2 面ミラーでの RAID-0+1 の実装について考えてみます。Solaris ボリュームマネージャを使用しなかった場合、1 つのスライスで障害が発生すると、ミラーの片面が使用できなくなる可能性があります。ホットスペアが使用されていない場合、2 つめのスライスで障害が発生すると、このミラーは使用不能になります。Solaris ボリュームマネージャを使用すると、最大 3 つのスライスで障害が発生しても、ミラーが使用できなくなることはありません。ミラーが使用不能にならないのは、ストライプ化された 3 つのスライスのそれぞれが、ミラーのもう一方の側の対応するスライスに対してミラー化されているからです。

図 10–2 に、RAID-1 ボリュームではスライスが失われることがあるのに対して、RAID-1+0 の実装ではデータ損失が起きないことを示します。

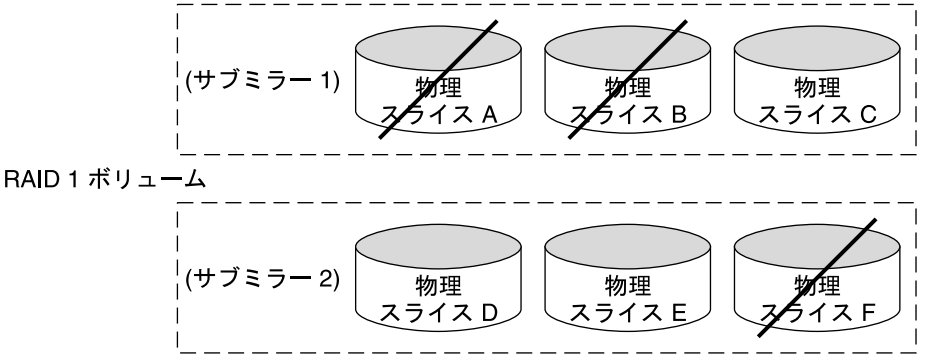

図 **10–2** RAID-1+0 の例

RAID-1 ボリュームは、2 つのサブミラーからなります。各サブミラーは、同じ飛び越し値が設定された3 つの同一物理ディスクからなります。3 つのディスク A、B、および F で障害が発生しても、支障はありません。ミラーの論理ブロック範囲全体が少なくとも 1 つの正常なディスクに格納されています。ボリュームの全データが使用可能です。

ただし、ディスク A と D で障害が発生した場合は、ミラーの一部分のデータがどのディスクでも使用できなくなります。これらの論理ブロックにはアクセスできません。しかし、データが使用可能なミラー部分へのアクセスは、引き続き可能です。この場合、ミラーは、不良ブロックを含む単一ディスクのように機能します。損傷部分は使用不能になりますが、残りの部分は使用可能です。

RAID-1 ボリューム (ミラー) の再同期

RAID-1 ボリューム (ミラー) の再同期は、次のいずれかの条件が発生したときに、あるサブミラーから別のサブミラーにデータをコピーするプロセスです。

- サブミラー障害
- システムクラッシュ
- オフラインであったサブミラーがオンラインに復帰
- 新規サブミラーの追加

再同期中も、ミラーの読み書きは実行できます。

ミラーの再同期は、すべてのサブミラーに同じデータを書き込むことによって (書き込みが進行中のデータは除く)、ミラーの有効性を保証します。

注 – ミラーの再同期を省略してはなりません。ただし、ミラーの再同期を手動で行う必要はありません。この処理は自動的に実行されます。

ミラー全体の再同期

ミラーに新しいサブミラーを接続 (追加) すると、そのミラー内の別のサブミラーのすべてのデータが新しいサブミラーに自動的に書き込まれます。ミラーの再同期が完了すると、新しいサブミラーは読み取り可能になります。サブミラーは切り離されないかぎり、ミラーに接続されたままになります。

再同期の実行中にシステムがクラッシュした場合は、システムがリブートしてから、再同期が再開されます。

再同期の最適化

Solaris ボリュームマネージャは、システム障害後のリブート時や、オフラインのサブミラーがオンラインに戻ったときに、最適化されたミラーの再同期を実行します。メタディスクドライバがサブミラーの領域を追跡します。この機能によって、メタディスクドライバは障害発生後に同期が外れたサブミラー領域を識別できます。最適化された再同期は、同期が取れていない領域に対してのみ行われます。リブート時にどの順序でミラーを再同期するかを指定できます。サブミラーパス番号をゼロに設定すると、ミラーの再同期を省略できます。パス番号の変更に関連する作業については、例 11–16 を参照してください。

注意 – パス番号ゼロは、読み取り専用としてマウントされているミラーに対してのみ設定します。

部分的な再同期

サブミラー内のスライスの交換後、Solaris ボリュームマネージャはデータの「部分的なミラー再同期」を実行します。Solaris ボリュームマネージャは、別のサブミラーの有効なスライスから新しいスライスにデータをコピーします。

RAID-1 ボリュームの作成と保守

ここでは、ミラーの作成に関する指針を示します。さらに、作成したミラーの性能に関する指針も示します。

RAID-1 ボリュームの構成指針

- ミラーを作成する前に、そのミラーを構成する RAID-0 (ストライプ方式または連結方式) ボリュームを作成する必要があります。

- ミラーを作成するときは、最初に 1 面ミラーを作成し、その後で 2 番目のサブミラーを接続します。この手順によって、再同期動作が開始されます。また、データが損傷しないという保証も得られます。1 面のミラーを作成しておけば、後でそれを 2 面または多面のミラーとして使用することもできます。
- 1 つのコマンドで、1 面ミラーから 2 面ミラー、3 面ミラー、または4 面ミラーを作成できます。1 つのコマンドですべてのサブミラーを作成することによって、作成プロセスを短時間で完了することもできます。この手順を使用するのは、既存データをミラー化しない場合で、なおかつすべてのサブミラー上のデータが壊れても問題がない場合に限られます。
- スライス上に構築された既存のファイルシステムから RAID-1 ボリュームを作成できます。基本 RAID-0 ボリューム (サブミラー) に含めることのできるスライスは 1 つだけです。ルートなど、システムにとって重要なファイルシステムをミラー化する場合は、すべてのサブミラーが単一のスライスで構成されていなければなりません。
- `swap -l` コマンドを使ってすべての `swap` デバイスを確認します。`swap` として指定されたスライスは、他のスライスとは別個にミラー化しなければなりません。
- Solaris 管理コンソール内の「拡張ストレージ」は、ルート (`/`)、`/opt`、`/usr`、または `swap`のミラー化解除をサポートしません。このツールは事実上、システムの稼働中にマウント解除できないあらゆるファイルシステムについて、ミラー化解除をサポートしません。これらのファイルシステムに対しては、コマンド行ユーティリティーを使用してください。
- 同じサイズのサブミラーを使用します。サイズが異なるサブミラーを使用すると、ディスク領域がむだになります。
- ミラーには、構成が同じサブミラーだけを使用します。たとえば、ディスクラベルのないサブミラーでミラーを作成すると、ラベルがあるサブミラーをミラーに追加できなくなります。
- 最初に追加したサブミラーがシリンダ 0 から始まらない形式でも、ファイルシステムをミラー化できます。この場合、それ以後追加するすべてのサブミラーについても、シリンダ 0 から始まらないようにしなければなりません。シリンダ 0 から始まるサブミラーを追加しようとすると、次のようなエラーメッセージが表示されます。

  ```
  can’t attach labeled submirror to an unlabeled mirror
  ```

  特定のミラーの中で使用するすべてのサブミラーがシリンダ 0 から始まるようにするか、すべてのサブミラーがシリンダ 0 から始まらないようにするか、どちらかでなければなりません。

  開始シリンダをすべてのサブミラーで一致させる必要はありません。しかし、すべてのサブミラーにシリンダ 0 を含めるか、すべてのサブミラーにシリンダ 0 を含めないか、どちらかにしなければなりません。
- ミラーを作成する前に状態データベースの複製を追加すると、ミラーの性能を向上させることができます。一般的な指針として、ミラーを追加するたびに 2 つの状態データベースの複製をシステムに追加する必要があります。Solaris ボリュームマネージャは、追加されたこれらの複製に、最適化された再同期を実行する際に使用するダーティリージョンログ (DRL) を格納します。十分な数の複製を用意することに

よって、RAID-1 ボリュームの性能に対する入出力の影響を最小限に抑えられます。複製がログを記録するミラーと同じディスクまたはコントローラに 2 つ以上の複製を使用すると、全体の性能も向上します。

- 直接マウントできるのはミラーデバイスだけです。オフライン状態のサブミラーを読み取り専用でマウントする場合を除き、サブミラーを直接マウントしてはなりません。また、サブミラーの一部であるスライスをマウントしてはなりません。データが壊され、システムが異常を起こすおそれがあります。

RAID-1 ボリュームの性能に関する指針

- サブミラーのスライスは、異なるディスクとコントローラに配置します。同じミラーの 2 つまたはそれ以上のサブミラーのスライスを同じディスクに置くと、データの保護機能が大幅に低下します。同じように、サブミラーは、別個のコントローラに配置します。これは、コントローラやそのケーブルでは、ディスクよりも障害が発生する確率が高いためです。これにより、ミラーの性能も向上します。
- 1 つのミラーでは、同じタイプのディスクとコントローラを使用します。特に、古いタイプの SCSI 記憶装置では、ディスクやコントローラの性能がモデルやブランドによって大幅に異なることがあります。1 つのミラーで性能レベルの異なるディスクやコントローラを使用すると、性能が大幅に低下する可能性があります。
- ミラー化によって読み取り性能が向上することはありますが、書き込み性能は常に低下します。ミラー化によって読み取り性能が向上するのは、スレッド化された入出力や非同期の入出力の場合だけです。ボリュームからの単一スレッド読み取りによって、性能が向上することはありません。
- ミラーの読み取りオプションの設定を変えてみると、性能が向上することがあります。たとえば、デフォルトの読み取りモードでは、各ディスクが巡回的に 1 つずつ読み取られます。ラウンドロビンは UFS マルチユーザー、マルチプロセッサ動作に最適なことが多いので、このポリシーがデフォルトです。

  場合によっては、`geometric` 読み取りオプションを使用すると、ヘッドの移動とアクセス時間が最小になり、性能が向上することがあります。このオプションは、次の場合にもっとも有効です。

  - 1 つのディスクでスライスを 1 つだけ使用する
  - 1 時点で 1 つのプロセスだけがスライスまたはファイルシステムを使用する
  - きわめて順次性の強い入出力パターンであるか、またはすべて読み取りアクセスである
- サービスの中断を伴わずにサブミラーをミラーに接続できます。サブミラーを接続することによって、2 面、3 面、4 面のミラーを作成できます。
- サブミラーをオフラインにすると、ミラーはそのサブミラーに対して読み書きできなくなります。しかし、サブミラーとミラー間の論理的な結びつきは維持されます。サブミラーがオフライン状態の間、Solaris ボリュームマネージャはミラーに対するすべての書き込みを追跡管理します。書き込みは、サブミラーがオンラインに戻ったときに、サブミラーに書き込まれます。Solaris ボリュームマネージャは、最適化された再同期を行うことによって、サブミラー全体ではなく、変更されたデータの再同期だけ

を行います。一方、サブミラーを切断すると、サブミラーとミラーの論理的な関連付けも断ち切られます。一般には、保守を行うときはサブミラーをオフラインにし、取り外すときはサブミラーを切断します。

RAID-1 ボリュームのオプション

ミラーの性能を最適化するには、次のオプションを使用します。

- ミラーからの読み取りポリシー
- ミラーへの書き込みポリシー
- ミラーを再同期する順序 (パス番号)

ミラーオプションは、ミラーを最初に作成するときに定義できます。ミラーが設定されて稼働したあとで、ミラーオプションを変更することもできます。これらのオプションの変更に関連する作業については、140 ページの「RAID-1 ボリュームオプションを変更するには」を参照してください。

RAID-1 ボリュームの読み取りおよび書き込みポリシー

Solaris ボリュームマネージャでは、RAID-1 ボリュームに対してさまざまな読み取りおよび書き込みポリシーを設定できます。構成に合わせて、読み取りおよび書き込みポリシーを適切に設定すると、性能が向上することがあります。

表 10–1 RAID-1 ボリュームの読み取りポリシー

読み取りポリシー	説明
ラウンドロビン (デフォルト)	すべてのサブミラーの負荷を均一にします。ミラーに属するすべてのサブミラーの読み取りは、ラウンドロビン方式で (1 つずつ順次に) 行われます。
ジオメトリック	読み取りを論理的なディスクブロックアドレスに基づいて個々のサブミラーに分割します。たとえば、2 面サブミラーの場合は、ミラーのディスク領域が、論理アドレスに基づいて同じサイズの 2 つの領域に分割されます。一方のサブミラーからの読み取りは、論理的な領域の半分に限定されます。他方のサブミラーからの読み取りは、残りの半分に限定されます。ジオメトリック読み取りポリシーでは、読み取りに必要なシーク時間が減少します。この読み取りポリシーによって得られる性能の向上は、システムの入出力負荷やアプリケーションのアクセスパターンによって異なります。
先頭のデバイスから読み取る	すべての読み取りを最初のサブミラーに送る。このポリシーは、先頭のサブミラーを構成するデバイスが 2 番目のサブミラーのデバイスよりも高速な場合にのみ使用します。

表 10–2 RAID-1 ボリュームの書き込みポリシー

書き込みポリシー	説明
並列 (デフォルト)	ミラーへの書き込みは複製され、すべてのサブミラーに対して同時に実行される。
順次	サブミラーへの書き込みは順次実行される (つまり、最初のサブミラーへの書き込みが終わってから、次のサブミラーへの書き込みが始まる)。このポリシーでは、1 つのサブミラーへの書き込みが終わらないと、次のサブミラーへの書き込みは開始されない。このポリシーは、電源の障害などによってサブミラーが読み取り不能になるのを防止するために使用する。

パス番号

パス番号 (0 から 9 の数字) は、システムのリブート時にミラーを再同期する順序を決定します。デフォルトのパス番号は 1 です。再同期は、パス番号の小さいミラーから行われます。ゼロを指定すると、ミラーの再同期はスキップされます。パス番号 0 は、読み取り専用としてマウントされているミラーに対してのみ設定します。同じパス番号をもつミラーの再同期は同時に実行されます。

保守作業を決定するサブミラーの状態

Solaris ボリュームマネージャの `metastat` コマンドによって、RAID-1 ボリュームとサブミラーの状態情報が報告されます。この状態情報に基づいて、RAID-1 ボリュームに対して保守作業が必要かどうかを判断できます。次の表で、RAID-1 ボリュームに対して metastat コマンドを実行したときに表示される、サブミラーの状態について説明します。

表 10–3 サブミラーの状態

状態	意味
正常 (Okay)	サブミラーにはエラーがなく正常に動作しています。
再同期中 (Resyncing)	サブミラーで再同期処理が実行されている。エラーが発生したが、すでに訂正され、オンラインに戻されたか、あるいは、新しいサブミラーが接続されています。
保守が必要 (Needs Maintenace)	サブミラー内のスライスに入出力エラーまたはオープンエラーが発生しました。スライスに対するすべての読み取りと書き込みはすでに停止されています。

`metastat` コマンドはさらに、サブミラーのスライスごとに次の情報を提供します。

Device　ストライプ内のスライスのデバイス名

Start Block　スライスの開始ブロック

Dbase　スライスに状態データベースの複製が含まれているかどうか

State　　スライスの状態

Hot Spare　　スライスは障害スライスのホットスペアとして使用中

サブミラーの状態が伝えるのは、サブミラーの総合的な状態情報だけです。ミラーのエラーに対処する場合は、通常、スライスの状態がもっとも重視しなければならない情報です。サブミラーが「保守が必要 (Needs Maintenance)」の状態になっている場合、スライスの状態を参照して詳細を調べる必要があります。

スライスの状態が「保守 (Maintenance)」の場合と「最後にエラー (Last Erred)」の場合では、障害から回復するための処置が異なります。「保守 (Maintenance)」状態のスライスだけが複数ある場合は、どのような順序でスライスを修理してもかまいません。しかし、「保守 (Maintenance)」状態のスライスと「最後にエラー (Last Erred)」状態のスライスの両方が混在する場合は、先に「保守 (Maintenance)」状態のスライスを修理する必要があります。「保守 (Maintenance)」状態のスライスを修理したあとで、「最後にエラー (Last Erred)」状態のスライスを修理します。詳細については、246 ページの「RAID-1 および RAID-5 ボリューム内のコンポーネントの交換と有効化の概要」を参照してください。

次の表に、サブミラーのスライスの状態と実行可能な処置を示します。

表 **10–4** サブミラーのスライスの状態

状態	意味	処置
正常 (Okay)	スライスはエラーがなく正常に動作しています。	必要ない
再同期中 (Resyncing)	スライスの再同期処理が進行しています。エラーが発生したが、すでに訂正され、オンラインに戻されたか、あるいは、新しいサブミラーが接続されています。	必要であれば、再同期処理が終了するまでサブミラーの状態を監視します。
保守 (Maintenance)	スライスで 入出力エラーかオープンエラーが発生しました。このコンポーネントに対するすべての読み取りと書き込みはすでに停止されています。	障害が発生したスライスを有効にするか、交換します。137 ページの「サブミラー内のスライスを有効にするには」または142 ページの「サブミラー内のスライスを交換するには」を参照してください。`metastat` コマンドを実行すると、`metareplace` コマンドを使って行うべき処置を示す `invoke` 回復メッセージが表示されます。`metareplace -e` コマンドを使用することもできます。

表 10–4 サブミラーのスライスの状態　(続き)

状態	意味	処置
最後にエラー (Last Erred)	スライスで 入出力エラーかオープンエラーが発生しました。しかし、別のスライスに障害があるため、データは他のスライスには複製されません。したがって、入出力は引き続きこのスライスに対して行われます。この入出力がエラーになると、ミラー入出力は失敗します。	まず、「保守 (Maintenance)」状態のスライスを有効にするか、交換します。137 ページの「サブミラー内のスライスを有効にするには」または 142 ページの「サブミラー内のスライスを交換するには」を参照してください。通常は、このエラーがあるとデータが失われるため、ミラーの修復後にミラーを検証する必要があります。ファイルシステムの場合は、`fsck` コマンドを実行してからデータをチェックします。アプリケーションやデータベースでは、独自の方法でデバイスを検証できなければなりません。

シングルユーザーモードでのブートが RAID-1 ボリュームに与える影響

ルート (/)、`/usr`、および `swap` という、いわゆる「ブート」ファイルシステム用のミラーを持つシステムをシングルユーザーモードで (つまり、`boot -s` コマンドを使用して) ブートしなければならない場合があります。この場合、`metastat` コマンドの出力では、これらのミラー (おそらくは、システム上のすべてのミラー) が「保守が必要 (Needing Maintenance)」状態と表示されることがあります。また、これらのスライスに書き込みがあった場合には、ミラーのダーティリージョンが増加していることが示されます。

この状況は潜在的な危険性を意味します。ただし、`metasync -r` コマンドは通常、ブート時にミラーの再同期のために実行されますが、 システムがシングルユーザーモードでブートされた場合には実行を中断されます。システムをリブートすると、`metasync -r` コマンドが実行され、すべてのミラーの再同期が取られます。

この状況が問題になる場合は、手動で `metasync -r` コマンドを実行してください。

シナリオ—RAID-1 ボリューム (ミラー)

RAID-1 ボリュームによって冗長ボリュームを作成できます。したがって、構成要素である RAID-0 ボリュームの 1 つで部分障害または完全障害が発生しても、データ損失は生じません。また、ファイルシステムにアクセスできなくなることもありません。次の例では、第 5 章と 85 ページの「シナリオ—RAID-0 ボリューム」で説明したシナリオを前提に、RAID-1 ボリュームで冗長記憶領域を提供する方法を示します。

85 ページの「シナリオ—RAID-0 ボリューム」で説明したとおり、このシステム例には 2 つの RAID-0 ボリュームがあります。1 つのボリュームは約 27G バイトの容量で、3 つの

ディスクにまたがっています。RAID-1 ボリュームを作成してこれら 2 つの RAID-0 ボリュームをミラー化すると、完全に冗長化された記憶領域によって、障害からの回復が可能なデータ記憶領域を構築できます。

この RAID-1 ボリュームでは、どちらのディスクコントローラに障害が発生しても、ボリュームへのアクセスは中断されません。さらに、最大 3 つのディスクに障害が発生しても、アクセスが中断されない場合もあります。

アクセス中断を引き起こす状況に対して保護を強化するには、ホットスペアを使用します。第 16 章を参照してください。特に 186 ページの「ホットスペアの仕組み」を参照してください。

◆ ◆ ◆ 第 11 章

RAID-1 (ミラー) ボリューム (作業)

この章では、 Solaris ボリュームマネージャの RAID-1 ボリュームに関連する作業について説明します。関連する概念については、第 10 章を参照してください。

RAID-1 ボリューム (作業マップ)

次の表に、Solaris ボリュームマネージャの RAID-1 ボリュームを管理するために必要な作業を示します。

作業	説明	参照先
未使用のスライスからミラーを作成する	Solaris ボリュームマネージャの GUI か `metainit` コマンドを使って、未使用のスライスからミラーを作成します。	111 ページの「未使用のスライスから RAID-1 ボリュームを作成するには」
既存のファイルシステムからミラーを作成する	Solaris ボリュームマネージャの GUI か `metainit` コマンドを使って、既存のファイルシステムからミラーを作成します。	113 ページの「ファイルシステムから RAID-1 ボリュームを作成するには」
ルート (/) ファイルシステムからミラーを作成する	Solaris ボリュームマネージャの GUI か `metainit` コマンドを使って、ルート (/) ファイルシステムからミラーを作成します。	118 ページの「SPARC: ルート (/) ファイルシステムから RAID-1 ボリュームを作成するには」 127 ページの「x86: DCA を使ってルート (/) ファイルシステムから RAID-1 ボリュームを作成するには」
サブミラーを接続する	Solaris ボリュームマネージャの GUI か `metattach` コマンドを使ってサブミラーを接続します。	134 ページの「サブミラーを接続するには」

作業	説明	参照先
サブミラーを切り離す	Solaris ボリュームマネージャの GUI か `metattach` コマンドを使ってサブミラーを切り離します。	135 ページの「サブミラーを切り離すには」
サブミラーをオンラインまたはオフラインにする	Solaris ボリュームマネージャの GUI か `metaonline` コマンドを使って、サブミラーをオンラインにします。Solaris ボリュームマネージャの GUI か `metaoffline` コマンドを使用して、サブミラーをオフラインにします。	136 ページの「サブミラーをオフラインまたはオンラインにするには」
サブミラー内のスライスを有効にする	Solaris ボリュームマネージャの GUI か `metareplace` コマンドを使って、サブミラー内のスライスを有効にします。	137 ページの「サブミラー内のスライスを有効にするには」
ミラーの状態をチェックする	Solaris ボリュームマネージャの GUI か `metastat` コマンドを使って、RAID-1 ボリュームの状態をチェックします。	138 ページの「ミラーとサブミラーの状態を表示するには」
ミラーオプションを変更する	Solaris ボリュームマネージャの GUI か `metaparam` コマンドを使って、特定の RAID-1 ボリュームのオプションを変更します。	140 ページの「RAID-1 ボリュームオプションを変更するには」
ミラーを拡張する	Solaris ボリュームマネージャの GUI か `metattach` コマンドを使ってミラーの容量を拡張します。	141 ページの「RAID-1 ボリュームを拡張するには」
サブミラー内のスライスを置き換える	Solaris ボリュームマネージャの GUI か `metareplace` コマンドを使って、サブミラーのスライスを交換します。	142 ページの「サブミラー内のスライスを交換するには」
サブミラーを置き換える	Solaris ボリュームマネージャの GUI か `metattach` コマンドを使ってサブミラーを置き換えます。	144 ページの「サブミラーを交換するには」
ミラーを削除する (ミラー化を解除する)	Solaris ボリュームマネージャの GUI、`metadetach` コマンド、または `metaclear` コマンドを使って、ファイルシステムのミラー化を解除します。	146 ページの「ファイルシステムのミラー化を解除するには」
マウント解除できないファイルシステムのミラーを削除する (ミラー化を解除する)	Solaris ボリュームマネージャの GUI、`metadetach` コマンド、または `metaclear` コマンドを使って、マウント解除できないファイルシステムのミラーを削除 (ミラー化を解除) します。	148 ページの「マウント解除できないファイルシステムのミラー化を解除するには」

作業	説明	参照先
ミラーを使ってバックアップを実行する	Solaris ボリュームマネージャの GUI、`metaonline` コマンド、または `metaoffline` コマンドを使って、ミラーのバックアップを行います。	151 ページの「RAID-1 ボリュームのオンラインバックアップを実行するには」

RAID-1 ボリュームの作成

▼ 未使用のスライスから RAID-1 ボリュームを作成するには

を作成する手順を示します。3 面ミラーまたは 4 面ミラーを作成する場合も、手順は同じです。

始める前に　49 ページの「Solaris ボリュームマネージャコンポーネントを作成するための前提条件」と 101 ページの「RAID-1 ボリュームの作成と保守」を確認します。

1. ストライプまたは連結を 2 つ作成します。これらのコンポーネントがサブミラーになります。

   88 ページの「RAID-0 (ストライプ方式) ボリュームを作成するには」または90 ページの「RAID-0 (連結方式) ボリュームを作成するには」を参照してください。

2. 次のどちらかの方法でミラーを作成します。

   - Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開き、「アクション (Action)」、「ボリュームを作成 (Create Volume)」の順に選択し、画面の指示に従います。詳細は、オンラインヘルプを参照してください。
   - 次の形式の `metainit` コマンドを使って 1 面ミラーを作成します。

     ```
     # metainit volume-name -m submirror-name
     ```

     volume-name　作成するボリュームの名前を指定します。

     `-m`　ミラーを作成することを指定します。

     submirror-name　ミラーの最初のサブミラーとして使用するコンポーネントの名前を指定します。

     詳細については、次の例と `metainit(1M)` のマニュアルページを参照してください。

3. 次のどちらかの方法で 2 つめのサブミラーを追加します。

- Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開き、変更するミラーを選択します。「アクション (Action)」、「プロパティ (Properties)」、「サブミラー (Submirrors)」の順に選択します。画面の指示に従って、サブミラーを追加します。詳細は、オンラインヘルプを参照してください。
- 次の形式の `metattach` コマンドを実行します。

  `# metattach` *volume-name submirror-name*

  volume-name サブミラーを追加する RAID-1 ボリュームの名前を指定します。

  submirror-name 次にミラーに追加するサブミラーとなるコンポーネントの名前を指定します。

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

例 11–1 2 面ミラーを作成する

```
# metainit d51 1 1 c0t0d0s2
d51: Concat/Stripe is setup
# metainit d52 1 1 c1t0d0s2
d52: Concat/Stripe is setup
# metainit d50 -m d51
d50: Mirror is setup
# metattach d50 d52
d50: Submirror d52 is attached
```

この例では、2 面ミラー `d50` を作成します。`metainit` コマンドは、 RAID-0 ボリュームである、2 つのサブミラー (`d51` と `d52`) を作成します。`metainit -m` コマンドは、RAID-0 ボリューム `d51` から 1 面ミラーを作成します。`metattach` コマンドは、 `d52` を接続して 2 面ミラーを作成し、同期を取り直します。接続されたサブミラー上のデータは、再同期の際に他のサブミラーによって上書きされます。

例 11–2 2 面ミラーを作成する (再同期なし)

```
# metainit d51 1 1 c0t0d0s2
d51: Concat/Stripe is setup
# metainit d52 1 1 c1t0d0s2
d52: Concat/Stripe is setup
# metainit d50 -m d51 d52
metainit: d50: WARNING: This form of metainit is not recommended.
The submirrors may not have the same data.
Please see ERRORS in metainit(1M) for additional information.
d50: Mirror is setup
```

この例では、2 面ミラー `d50` を作成します。`metainit` コマンドは、 RAID-0 ボリュームである、2 つのサブミラー (`d51` と `d52`) を作成します。`metainit -m` コマンドは、2 つのサブミラーからミラーを作成します。`metattach` コマンドではなく、`metainit` コマンドを

使ってミラーを作成した場合は、再同期操作が行われません。この場合、Solaris ボリュームマネージャはミラーの両側が同一であるとみなし、両方を区別なく使用するため、データが破壊されるおそれがあります。

参照　新たに作成したミラーにファイルシステムを作成する方法については、『Solaris のシステム管理 (デバイスとファイルシステム)』の第 18 章「UFS、TMPFS、LOFS ファイルシステムの作成 (手順)」を参照してください。データベースなど、アプリケーションによってはファイルシステムを使用しません。これらのアプリケーションでは、代わりに raw デバイスを使用します。アプリケーションは独自の方法で raw デバイスにアクセスできなければなりません。

▼ ファイルシステムから RAID-1 ボリュームを作成するには

この手順では、既存のファイルシステムをミラー化します。このファイルシステムがマウント解除できる場合は、システムをリブートしなくても、ここに示すすべての手順を完了することができます。`/usr` や `/swap` のようなマウント解除できないファイルシステムの場合、手順を完了するには、システムをリブートする必要があります。

スライス上に構築された既存のファイルシステムから RAID-1 ボリュームを作成する場合は、そのスライスだけを一次 RAID-0 ボリューム (サブミラー) に含めるようにします。システムにとって重要なファイルシステムをミラー化する場合は、すべてのサブミラーが単一のスライスだけで構成されている必要があります。

ルート (/) ファイルシステムのミラー化に関連する手順については、118 ページの「SPARC: ルート (/) ファイルシステムから RAID-1 ボリュームを作成するには」および 127 ページの「x86: DCA を使ってルート (/) ファイルシステムから RAID-1 ボリュームを作成するには」を参照してください。

この手順の例では、既存スライスは `c1t0d0s0` です。2 番目のスライス `c1t1d0s0` はミラーの 2 番目として使用します。サブミラーは `d1` と `d2`、ミラーは `d0` です。

注意 – まず `metainit` コマンドで 1 面ミラーを作成してから、`metattach` コマンドで追加のサブミラーを接続します。`metattach` コマンドを使用しないと、再同期は実行されません。この場合、Solaris ボリュームマネージャはミラーの両側が同一であるとみなし、両方を区別なく使用するため、データが破壊されるおそれがあります。

始める前に　49 ページの「Solaris ボリュームマネージャコンポーネントを作成するための前提条件」と 101 ページの「RAID-1 ボリュームの作成と保守」を確認します。

1. ミラー化する既存ファイルシステムが格納されているスライスを特定します。この例では、スライス `c1t0d0s0` を使用します。

2 次のどちらかの方法を使って、前の手順で特定したスライスに新しい **RAID-0** ボリュームを作成します。

- Solaris 管理コンソール内の「拡張ストレージ」から、「ボリューム (Volumes)」ノードを開き、「アクション (Action)」、「ボリュームの作成 (Create Volume)」の順に選択します。画面の指示に従います。詳細は、オンラインヘルプを参照してください。
- 次の形式の `metainit` コマンドを使用します。

```
# metainit -f  volume-name  number-of-stripes  components-per-stripe  component-name
```

`-f`	コマンド処理を強制的に続けます。スライスにマウントされたファイルシステムが含まれている場合は、このオプションを使用する必要があります。
volume-name	作成するボリュームの名前を指定します。ボリュームの命名方式については、45 ページの「ボリューム名」を参照してください。
number-of-stripes	作成するストライプの数を指定します。
components-per-stripe	各ストライプに与えるコンポーネントの数を指定します。
component-names	使用するコンポーネントの名前を指定します。この例では、ルートスライス `c0t0d0s0` を使用します。

3 未使用のスライス (この例では `c1t1d0s0`) に **2** 番目の **RAID-0** ボリューム (連結) を作成します。これは、後で **2** 番目のサブミラーとして使用します。**2** 番目のサブミラーのサイズは、最初のサブミラー以上でなければなりません。この手順では、次のどちらかの方法を使用します。

- Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開き、「アクション (Action)」、「ボリュームを作成 (Create Volume)」の順に選択し、画面の指示に従います。詳細は、オンラインヘルプを参照してください。
- 次の形式の `metainit` コマンドを使用します。

```
# metainit volume-name  number-of-stripes  components-per-stripe  component-name
```

注 – オプションの説明については、手順 2 を参照してください。

---

4 次のどちらかの方法で **1** 面ミラーを作成します。

- Solaris 管理コンソール内の「拡張ストレージ」から、「ボリューム (Volumes)」ノードを開き、「アクション (Action)」、「ボリュームの作成 (Create Volume)」の順に選択します。画面の指示に従います。詳細は、オンラインヘルプを参照してください。
- 次の形式の `metainit` コマンドを使用します。

```
# metainit volume-name -m submirror-name
```

volume-name 作成するボリュームの名前を指定します。

`-m` ミラーを作成することを指定します。

submirror-name ミラーの最初のサブミラーとして使用するコンポーネントの名前を指定します。この例では、ルートスライスが含まれる RAID-0 ボリュームです。

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

注意 – 既存のファイルシステムからミラーを作成する場合は、データが破壊されないように、次の 2 つの手順に忠実に従ってください。

5 ファイルシステムのマウント手順が (ブロックデバイスではなく) ミラーを参照するように、`/etc/vfstab` ファイルを編集します。`/etc/vfstab` ファイルについての詳細は、『**Solaris** のシステム管理 (デバイスとファイルシステム)』の「ファイルシステムのマウント」を参照してください。

たとえば、`/etc/vfstab` ファイルにファイルシステム用の次のエントリが指定されているとします。

`/dev/dsk/`*slice* `/dev/rdsk/`*slice* `/var ufs 2 yes -`

次のようにエントリを変更します。

`/dev/md/dsk/`*mirror-name* `/dev/md/rdsk/`*mirror-name* `/var ufs 2 yes -`

6 次のどちらかの方法で、新たにミラー化したファイルシステムをマウントし直します。

- マウント解除可能なファイルシステムをミラー化している場合は、ファイルシステムをマウント解除してから再びマウントします。

```
# umount /filesystem
# mount /filesystem
```

- マウント解除できないファイルシステムをミラー化している場合は、システムをリブートします。

```
# reboot
```

7 次の形式の `metattach` コマンドを使って、**2** 番目のサブミラーを追加します。

`# metattach` *volume-name submirror-name*

volume-name サブミラーを追加する RAID-1 ボリュームの名前を指定します。

submirror-name 次にミラーに追加するサブミラーとなるコンポーネントの名前を指定します。

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

例 11–3 マウント解除できるファイルシステムから 2 面ミラーを作成する

```
# metainit -f d1 1 1 c1t0d0s0
d1: Concat/Stripe is setup
# metainit d2 1 1 c1t1d0s0
d2: Concat/Stripe is setup
# metainit d0 -m d1
d0: Mirror is setup
# umount /master
    (ファイルシステムがミラーを参照するように /etc/vfstab ファイルを編集する)
# mount /master
# metattach d0 d2
d0: Submirror d2 is attached
```

この例では、-f オプションを使用して、最初の連結 d1 を強制的に作成します。これには、/dev/dsk/c1t0d0s0 にマウントされているファイルシステム /master が含まれています。 次に 2 番目の連結 d2 を /dev/dsk/c1t1d0s0 から作成します。このスライスは、d1と同じサイズか、またはそれより大きくなければなりません。そして、metainit コマンドに -m オプションを付けて実行し、d1 から1 面ミラー d0 を作成します。

次に、このファイルシステムのエントリがミラーを参照するように /etc/vfstab ファイルを編集します。/etc/vfstab ファイルの後続行は、最初、次のようになっています。

```
/dev/dsk/c1t0d0s0 /dev/rdsk/c1t0d0s0 /var ufs 2 yes -
```

このエントリを次のように変更します。

```
/dev/md/dsk/d0 /dev/md/rdsk/d0 /var ufs 2 yes -
```

最後に、ファイルシステムをマウントし直し、サブミラー d2 をミラーに接続して、ミラーの同期を取り直します。RAID-0 と RAID-1 ボリュームが設定され、サブミラー d2 が接続されたことを示すメッセージが表示されます。

例 11–4 マウント解除できないファイルシステムから 2 面ミラーを作成する

```
# metainit -f d12 1 1 c0t3d0s6
d12: Concat/Stripe is setup
# metainit d22 1 1 c1t0d0s6
d22: Concat/Stripe is setup
# metainit d2 -m d12
d2: Mirror is setup
    (/usr がミラーを参照するように /etc/vfstab ファイルを編集する)
# reboot
...
# metattach d2 d22
d2: Submirror d22 is attached
```

この例では、 /usr ファイルシステムが含まれるスライスを使用して、2 面ミラーを作成します。まず、-f オプションを使用して、最初の連結 d12 を強制的に作成します。これには、/dev/dsk/c0t3d0s6 にマウントされているファイルシステム /usr が含まれています。次に 2 番目の連結 d22 を /dev/dsk/c1t0d0s6 から作成します。このスライスは d12 と同じかそれ以上のサイズにする必要があります。そして、metainit コマンドに -m オプションを付けて実行し、/usr ファイルシステムを含む連結から1 面ミラー d2 を作成します。次に、/usr のエントリがミラーを参照するように /etc/vfstab ファイルを編集します。

/etc/vfstab ファイルには、/usr ファイルシステムに対応する次のエントリが指定されています。

```
/dev/dsk/c0t3d0s6 /dev/rdsk/c0t3d0s6 /usr ufs 1 yes -
```

次のようにエントリを変更します。

```
/dev/md/dsk/d2 /dev/md/rdsk/d2 /usr ufs 1 yes -
```

リブートが終わると、2 番目のサブミラー d22 がミラーに接続され、ミラーの再同期が実行されます。

例 11–5 /swap 領域からミラーを作成する

```
# metainit -f d11 1 1 c0t0d0s1
d11: Concat/Stripe is setup
# metainit d21 1 1 c1t0d0s1
d21: Concat/Stripe is setup
# metainit d1 -m d11
d1: Mirror is setup
    (swap がこのミラーを参照するように /etc/vfstab ファイルを編集する)
# reboot
...
# metattach d1 d21
d1: Submirror d21 is attached
```

この例ではまず、-f オプションを指定して最初の連結 d11 を強制的に作成します。これには、/dev/dsk/c0t0d0s1 にマウントされているファイルシステム swap が含まれています。次に 2 番目の連結 d21 を /dev/dsk/c1t0d0s1 から作成します。このスライスは d11 と同じかそれ以上のサイズにする必要があります。metainit コマンドに -m オプションを付けて実行し、swap を含む連結から 1 面ミラー d1 を作成します。次に、/etc/vfstab ファイルにswap のエントリがある場合は、このエントリがミラーを参照するようにファイルを編集する必要があります。

/etc/vfstab ファイルには、swap 領域に対応する次のエントリが指定されています。

```
/dev/dsk/c0t0d0s1 - - swap - no -
```

次のようにエントリを変更します。

```
/dev/md/dsk/d1 - - swap - no -
```

リブートが終わると、2 番目のサブミラー d21 がミラーに接続され、ミラーの再同期が実行されます。

swap 領域をミラー化した場合、クラッシュダンプを保存するには、dumpadm コマンドを使ってダンプデバイスをボリュームとして構成する必要があります。たとえば、swap デバイスの名前が /dev/md/dsk/d2 であれば、dumpadm コマンドを使ってこのデバイスをダンプデバイスとして設定します。

▼ SPARC: ルート (/) ファイルシステムから RAID-1 ボリュームを作成するには

SPARC プラットフォームでルート (/) ファイルシステムをミラー化する手順は、マウント解除できないルート以外のファイルシステムをミラー化する手順と同様です。異なるところは、/etc/vfstab ファイルを手動で編集するのではなく、metaroot コマンドを実行する点です。さらに、ルート (/) ファイルシステムをミラー化するには、代替ブートデバイスへのパスを記録する必要があります。サブミラーに障害が発生した場合、この代替ブートデバイスがシステムをリブートします。

この手順の例では、既存スライスは c1t0d0s0 です。2 番目のスライス c1t1d0s0 はミラーの 2 番目として使用します。サブミラーは d1 と d2、ミラーは d0 です。

注意 – まず metainit コマンドで 1 面ミラーを作成してから、metattach コマンドで追加のサブミラーを接続します。metattach コマンドを使用しないと、再同期は実行されません。この場合、Solaris ボリュームマネージャはミラーの両側が同一であるとみなし、両方を区別なく使用するため、データが破壊されるおそれがあります。

始める前に 49 ページの「Solaris ボリュームマネージャコンポーネントを作成するための前提条件」と 101 ページの「RAID-1 ボリュームの作成と保守」を確認します。

1. ミラー化する既存のルート (/) ファイルシステムが含まれているスライスを特定します。この例では、スライス c1t0d0s0 を使用します。
2. 次のどちらかの方法を使って、前の手順で特定したスライスに新しい **RAID-0** ボリュームを作成します。**RAID-0** ボリュームに含めることができるのは、単一のスライスだけです。
   - Solaris 管理コンソール内の「拡張ストレージ」から、「ボリューム (Volumes)」ノードを開き、「アクション (Action)」、「ボリュームの作成 (Create Volume)」の順に選択します。画面の指示に従います。詳細は、オンラインヘルプを参照してください。

- 次の形式の `metainit` コマンドを使用します。

  # metainit -f *volume-name number-of-stripes components-per-stripe component-name*

`-f`	コマンド処理を強制的に続けます。スライスにマウントされたファイルシステムが含まれている場合は、このオプションを使用する必要があります。
volume-name	作成するボリュームの名前を指定します。ボリュームの命名方式については、45 ページの「ボリューム名」を参照してください。
number-of-stripes	作成するストライプの数を指定します。
components-per-stripe	各ストライプに与えるコンポーネントの数を指定します。
component-names	使用するコンポーネントの名前を指定します。この例では、ルートスライス `c0t0d0s0` を使用します。

3 未使用のスライス (この例では `c1t1d0s0`) に **2** 番目の **RAID-0** ボリュームを作成します。このボリュームは **2** 番目のサブミラーとして使用します。**2** 番目のサブミラーのサイズは、最初のサブミラー以上でなければなりません。この手順では、次のどちらかの方法を使用します。

- Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開き、「アクション (Action)」、「ボリュームを作成 (Create Volume)」の順に選択し、画面の指示に従います。詳細は、オンラインヘルプを参照してください。
- 次の形式の `metainit` コマンドを使用します。

  # metainit *volume-name number-of-stripes components-per-stripe component-name*

  注 – オプションの説明については、ステップ 2 を参照してください。

4 次のどちらかの方法で **1** 面ミラーを作成します。

- Solaris 管理コンソール内の「拡張ストレージ」から、「ボリューム (Volumes)」ノードを開き、「アクション (Action)」、「ボリュームの作成 (Create Volume)」の順に選択します。画面の指示に従います。詳細は、オンラインヘルプを参照してください。
- 次の形式の `metainit` コマンドを使用します。

  # metainit *volume-name* **-m** *submirror-name*

volume-name	作成するボリュームの名前を指定します。
`-m`	ミラーを作成することを指定します。
submirror-name	ミラーの最初のサブミラーとして使用するコンポーネントの名前を指定します。この例では、ルートスライスが含まれる RAID-0 ボ

リュームです。

5 新たにミラー化したファイルシステムをマウントし直します。`metaroot` *volume-name* コマンドを実行します。*volume-name* のところには、作成したミラーの名前を入れます。さらに、システムをリブートします。

```
# metaroot volume-name
# reboot
```

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

6 次の形式の `metattach` コマンドを使って、**2** 番目のサブミラーを接続します。

```
# metattach volume-name submirror-name
```

volume-name サブミラーを追加する RAID-1 ボリュームの名前を指定します。

submirror-name 次にミラーに追加するサブミラーとなるコンポーネントの名前を指定します。

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

7 代替ブートパスを記録します。

a. 代替ルートデバイスへのパスを調べます。**2** 番目のサブミラーとしてルート **(/)** ファイルシステムのミラーに接続するスライスで、`ls -l` コマンドを使用します。

```
# ls -l /dev/dsk/c1t1d0s0
lrwxrwxrwx 1  root root  55 Mar 5 12:54  /dev/rdsk/c1t1d0s0 -> \
../../devices/sbus@1,f8000000/esp@1,200000/sd@3,0:a
```

b. ここで、`/devices` ディレクトリに続く次の文字列を記録しておきます。

```
/sbus@1,f8000000/esp@1,200000/sd@3,0:a.
```

注 – システムが利用できない場合に備えて、システム上以外の場所にこの情報を書き留めておくべきです。代替ブートデバイスからブートする方法についての詳細は、300 ページの「ブート障害からの回復」を参照してください。

---

c. この文字列を編集して、メジャー名 **(**この場合は `sd`**)** を `disk` に変更し、「`/sbus@1,f8000000/esp@1,200000/disk@3,0:a`」のようにします。システムが **IDE** バスを使用している場合、オリジナルのフルパスは次のようになります。

```
$ ls -l /dev/dsk/c1t1d0s0
lrwxrwxrwx 1 root root 38 Mar 13 15:03 /dev/dsk/c0t0d0s0 -> \
../../devices/pci@1f,0/ide@d/dad@0,0:a
```

メジャー名 `dad` を `disk` に変更すると、「`/pci@1f,0/ide@d/disk@0,0:a`」のようになります。

d. **OpenBoot™ PROM** `nvalias` コマンドを使って、二次ルート (/) ファイルシステムミラー用の「バックアップルート」デバイス別名を定義します。たとえば、次のように指定します。

```
ok  nvalias backup_root /sbus@1,f8000000/esp@1,200000/disk@3,0:a
```

e. 一次サブミラーと二次サブミラーの両方を参照するように `boot-device` 別名を定義し直して、構成を保存します。サブミラーは、指定された順に使用されます。

```
ok printenv boot-device
boot-device =         disk net
ok setenv boot-device disk backup_root net
boot-device =         disk backup_root net
ok nvstore
```

注 – 一次サブミラーに障害が発生すると、システムは 2 番目のサブミラーから自動的にブートします。自動ブートではなく、手動でブートする場合は、次のように入力します。

```
ok boot backup_root
```

例 11–6 SPARC: ルート (/) ファイルシステムからミラーを作成する

```
# metainit -f d1 1 1 c0t0d0s0
d1: Concat/Stripe is setup
# metainit d2 1 1 c0t1d0s0
d2: Concat/Stripe is setup
# metainit d0 -m d1
d0: Mirror is setup
# metaroot d0
# lockfs -fa
# reboot
...
# metattach d0 d2
d0: Submirror d2 is attached
# ls -l /dev/dsk/c0t1d0s0
lrwxrwxrwx   1 root     root          88 Feb  8 15:51 /dev/rdsk/c1t3d0s0 ->
../../devices/pci@1f,0/pci@1,1/ide@3/dad@0,0:a
# init 0
.
.
.
ok nvalias backup_root /pci@1f,0/pci@1,1/ide@3/disk@0,0:a
ok setenv boot-device disk backup_root net
ok nvstore
```

この例では、`-f` オプションによって、最初の RAID-0 ボリュームとして `d1` を作成します。これには、`/dev/dsk/c0t0d0s0` にマウントされているルート (/) ファイルシステムが

含まれています。次に 2 番目の連結 d2 を /dev/dsk/c0t1d0s0 から作成します。このスライスは d1 と同じかそれより大きいサイズにする必要があります。そして、metainit コマンドに -m オプションを付けて実行し、ルート (/) を含む連結から1 面ミラー d0 を作成します。

次に metaroot コマンドを使って /etc/vfstab ファイルと /etc/system ファイルを編集し、システムがボリューム上のルート (/) ファイルシステムからブートされるように指定します。リブートを行う前に lockfs -fa コマンドを実行することが推奨されます。詳細は、lockfs(1M) のマニュアルページを参照してください。

システムをリブートする前に、2 番目のサブミラーを接続しないでください。metaroot コマンドを実行した後、2 番目のサブミラーを接続する前に、システムをリブートする必要があります。

リブートが終わると、サブミラー d2 がミラーに接続され、ミラーの再同期が実行されます連結とミラーが設定され、サブミラー d2 が接続されたことを示すメッセージが表示されます。

最後に、ルート raw デバイスに対して ls -l コマンドを実行して、代替ルートデバイスへのパスを表示します。このパスは、このデバイスからシステムをブートしなければならない状況が発生したときに必要になります。

x86: ルート (/) ファイルシステムから RAID-1 ボリュームを作成する

Solaris 10 1/06 リリースから、x86 ベースシステムのブート処理とその構成に使用される Device Configuration Assistant (デバイス構成用補助、DCA) が GRand Unified Bootloader (GRUB) に置き換えられました。この機能とその拡張機能の概要については、『Solaris 10 の概要』の「GRUB ベースのブート」を参照してください。

この節では、ルート (/) ファイルシステムから RAID-1 ボリュームを作成する手順について説明します。Solaris 10 1/06 以降のリリースの OS を実行している場合は、GRUB を使用する最初の手順に従ってください。それ以外の場合は、DCA を使用する 2 番目の手順に従ってください。

x86 ベースシステムでルート (/) ファイルシステムをミラー化するプロセスは、SPARC システムでルートをミラー化するプロセスとほぼ同じです。ただし、x86 ベースシステムでは、BIOS と fdisk パーティションという複雑な階層があります。

この手順の例では、既存スライスは c1t0d0s0 です。2 番目のスライス c1t1d0s0 はミラーの 2 番目として使用します。サブミラーは d1 と d2、ミラーは d0 です。

注 – 手順を実行する前に、49 ページの「Solaris ボリュームマネージャコンポーネントを作成するための前提条件」と101 ページの「RAID-1 ボリュームの作成と保守」を確認してください。

▼ x86: GRUB を使ってルート (/) ファイルシステムから RAID-1 ボリュームを作成するには

1 ミラーの 2 番目のディスクからシステムがブートできるように、BIOS ブートデバイスの順番を構成できることを確認します。

カーネルが起動される前は、システムは、x86 ベースシステム上のファームウェアインタフェースである、読み取り専用メモリー (ROM) の Basic Input/Output System (BIOS) によって制御されます。この BIOS は、SPARC ベースシステムのブート PROM に似ています。BIOS に対して、以降の作業を行います。

- 起動関数を実行します。
- システムをブートする正しいデバイスを検出します。
- 前述のデバイスからマスターブートレコードをロードし、システムが自動的にブートするようにします。

通常は、BIOS を構成して、ブートレコードを探すデバイスの順番を選択できます。さらに、最近のほとんどの BIOS 実装では、二次サブミラーへのフェイルオーバーが自動的に行われるようにデバイスを構成できます。システムの BIOS にこの機能がなく、一次サブミラーで障害が発生した場合は、システムのブート中に BIOS にアクセスし、二次ルートスライスからブートするようにシステムを再構成する必要があります。BIOS の設定を構成する方法については、BIOS のユーザーガイドを参照してください。

ルートミラーを設定する前に、システム上の BIOS で、複数のディスクからブートできることを確認します。デバイスドライバによっては、システムの 1 つのディスクしか認識しないように構成されているものもあります。

2 ルートのミラー化をサポートするように `fdisk` パーティションが構成されていることを確認します。

別途 x86 ブートパーティションが存在する場合は、ルート (/) ファイルシステムのミラー化中に問題が発生します。x86 ブートパーティションは Solaris `fdisk` パーティションの外部に存在しているため、Solaris ボリュームマネージャではミラー化できません。さらに、x86 ブートパーティションのコピーは 1 つしかないため、シングルポイント障害にもなります。

Solaris 10 1/06 リリース以降に導入された GRUB ベースのインストールプログラムでは、x86 ブートパーティションは自動的には作成されません。しかし、システムに x86 がすでに存在する場合、インストールプログラムはデフォルトでそのパーティションを保存します。

システムに別途 x86 ブートパーティションがあるかどうかは、/etc/vfstab ファイルを確認すればわかります。このファイルに次のようなエントリが含まれている場合、x86 ブートパーティションは存在しています。

```
/dev/dsk/c2t1d0p0:boot  -       /boot   pcfs    -       no      -
```

Solaris ボリュームマネージャを使ってルート (/) ファイルシステムをミラー化する場合、このファイルシステムで単一の Solaris fdisk パーティションを使用する必要があります。したがって、システム内にすでに x86 ブートパーティションが存在する場合は、fdisk コマンドでこのパーティションを削除してから、Solaris ソフトウェアを再インストールしてください。再インストール時には、ブートパーティションは作成されません。

注 – Solaris ボリュームマネージャでミラー化できるのは、Solaris fdisk パーティション内のスライスだけです。複数の fdisk パーティションが存在する場合、Solaris fdisk パーティションの外にあるデータを保護するには、別の方法を使用する必要があります。

---

3 マスターブートプログラムを使って、二次サブミラーをブート可能にします。

a. マスターブートプログラムを指定します。

```
# fdisk -b /usr/lib/fs/ufs/mboot /dev/rdsk/c1t1d0p0
```

次の画面が表示されます。

```
            Total disk size is 31035 cylinders
            Cylinder size is 1146 (512 byte) blocks

                                               Cylinders
      Partition   Status    Type          Start   End   Length    %
      =========   ======    ============  =====   ===   ======   ===
          1        Active   Solaris           1  31034    31034    100


   SELECT ONE OF THE FOLLOWING:
      1. Create a partition
      2. Specify the active partition
      3. Delete a partition
      4. Change between Solaris and Solaris2 Partition IDs
      5. Exit (update disk configuration and exit)
      6. Cancel (exit without updating disk configuration)
   Enter Selection:
```

b. メニューから **5** 番を選択して、**Return** キーを押します。

4 二次ディスクをブート可能にします。

```
# /sbin/installgrub /boot/grub/stage1 /boot/grub/stage2 /dev/rdsk/c1t1d0s0
```

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

5 ミラー化する既存のルート (/) ファイルシステムが含まれているスライスを特定します。
この例では、スライス `c1t0d0s0` を使用します。

6 前の手順で指定したスライス上で新しい **RAID-0** ボリュームを作成します。
RAID-0 ボリュームに含めることができるのは、単一のスライスだけです。この手順では、次のどちらかの方法を使用します。

- Solaris 管理コンソール内の「拡張ストレージ」から、「ボリューム (Volumes)」ノードを開き、「アクション (Action)」、「ボリュームの作成 (Create Volume)」の順に選択します。画面の指示に従います。詳細は、オンラインヘルプを参照してください。
- 次の形式の `metainit` コマンドを使用します。

```
# metainit -f volume-name number-of-stripes components-per-stripe component-name
```

`-f`	コマンド処理を強制的に続けます。スライスにマウントされたファイルシステムが含まれている場合は、このオプションを使用する必要があります。
volume-name	作成するボリュームの名前を指定します。ボリュームの命名方式については、45 ページの「ボリューム名」を参照してください。
number-of-stripes	作成するストライプの数を指定します。
components-per-stripe	各ストライプに与えるコンポーネントの数を指定します。
component-names	使用するコンポーネントの名前を指定します。この例では、ルートスライス `c0t0d0s0` を使用します。

7 未使用のスライス上で、**2** 番目のサブミラーとして機能する **2** 番目の **RAID-0** ボリューム **(**この例では `c1t1d0s0`**)** を作成します。

注 – 2 番目のサブミラーのサイズは、最初のサブミラー以上でなければなりません。2 番目のサブミラーとして使用するスライスには「root」というスライスタグを付ける必要があります。また、ルートスライスはスライス 0 である必要があります。

スライスの構成方法については、`format(1M)` のマニュアルページを参照してください。

次のどちらかの方法を選択します。

- Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開き、「アクション (Action)」、「ボリュームを作成 (Create Volume)」の順に選択し、画面の指示に従います。詳細は、オンラインヘルプを参照してください。
- 次の形式の `metainit` コマンドを使用します。

```
# metainit volume-name number-of-stripes components-per-stripes component-names
```

注 – オプションの説明については、手順 6 を参照してください。

8 次のどちらかの方法で 1 面ミラーを作成します。

- Solaris 管理コンソール内の「拡張ストレージ」から、「ボリューム (Volumes)」ノードを開き、「アクション (Action)」、「ボリュームの作成 (Create Volume)」の順に選択します。画面の指示に従います。詳細は、オンラインヘルプを参照してください。
- 次の形式の `metainit` コマンドを使用します。

```
# metainit volume-name -m submirror-name
```

volume-name	作成するボリュームの名前を指定します。
`-m`	ミラーを作成することを指定します。
submirror-name	ミラーの最初のサブミラーとして使用するコンポーネントの名前を指定します。この例では、ルートスライスが含まれる RAID-0 ボリュームです。

9 新しくミラー化されたファイルシステムを再マウントし、システムをリブートします。

```
# metaroot volume-name
# reboot
```

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

10 2 番目のサブミラーを接続します。

```
# metattach volume-name submirror-name
```

volume-name	サブミラーを追加する RAID-1 ボリュームの名前を指定します。
submirror-name	次にミラーに追加するサブミラーとなるコンポーネントの名前を指定します。

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

11 `menu.lst` ファイルに代替ブートパスを定義します。

二次サブミラーを持つディスクからシステムがブートできるようにするには、そのディスクを代替ブートデバイスとして認識するようにシステムを構成します。現在の `c1t1d0s0` の例では、代替パスは、二次ディスクの最初の fdisk パーティションの、最初のスライス上にあります。したがって、`menu.lst` を次のように編集します。

```
title alternate boot
 root (hd1,0,a)
 kernel /boot/multiboot
 module /boot/x86.miniroot-safe
```

注 – menu.lst のエントリを適切に編集するには、GRUB のディスク命名規則について熟知している必要があります。詳細は、『Solaris のシステム管理 (基本編)』の第 11 章「GRUB ベースのブート (手順)」を参照してください。

menu.lst ファイルの編集後、システムは二次ディスクで処理を継続するようになります。一次ディスクで障害が発生した場合、システムが二次ディスクからブートするように、ディスク番号が変更されます。

注意 – 場合によっては、BIOS の自動ディスク番号再割り当て機能が、使用できなくなった一次ディスクからの回復に影響を及ぼすことがあります。ディスク番号再割り当て機能により、システムが強制的に二次ディスクからブートするようになると、一次ディスクのブートアーカイブは無効になります。同じ一次ディスクがあとで使用可能になって、システムをブートした場合、デフォルトの一次ディスクからシステムがブートするように、ディスク番号が再度切り替わります。しかし、この段階では、一次ディスクのブートアーカイブは無効なままです。したがって、システムはまったくブートしない可能性があります。このような場合は、GRUB メニューから正しいエントリを選択して、システムが有効なブートアーカイブからブートするようにします。システムの起動プロセスが完了したら、通常どおり metadevice による保守を実行して、一次ディスクと二次ディスクを同期化し、再度一次ディスクを有効なブートアーカイブにします。

▼ x86: DCA を使ってルート (/) ファイルシステムから RAID-1 ボリュームを作成するには

1 **ミラーの 2 番目のディスクからシステムがブートできるように、BIOS ブートデバイスの順番を構成できることを確認します。**

カーネルが起動される前は、システムは、x86 ベースシステム上のファームウェアインタフェースである、読み取り専用メモリー (ROM) の Basic Input/Output System (BIOS) によって制御されます。この BIOS は、SPARC ベースシステムのブート PROM に似ています。ほかの起動機能に加えて、BIOS は正しいブートデバイスを見つけ、そのデバイスからマスターブートレコードを読み込んで、システムがブートできるようにします。通常は、BIOS を構成して、ブートレコードを探すデバイスの順番を選択できます。さらに、最近のほとんどの BIOS 実装では、二次サブミラーへのフェイルオーバーが自動的に行われるようにデバイスを構成できます。使用中のシステムにこの機能がない場合で、一次サブミラーに障害が発生したときは、システムのブート中に BIOS にアクセスして、二次ルートスライスからブートするよう構成し直す必要があります。BIOS の設定を構成する方法については、BIOS のユーザーガイドを参照してください。

システム上の DCA を使って、複数のディスクからブート可能であることを確認できます。デバイスドライバによっては、システムの 1 つのディスクしか認識しないように構成されているものもあります。

2 ルートのミラー化をサポートするように fdisk パーティションが構成されていることを確認します。

x86 ベースシステムのもう一つの特長は、fdisk パーティションを使用することです。Solaris OS インストールプログラムのブートディスクパーティションのデフォルトのレイアウトでは、Solaris fdisk パーティションとは別に、「x86 ブートパーティション」という、10M バイトくらいの小さな fdisk パーティションが作成されます。

x86 ブートパーティションは、ルート (/) ファイルシステムをミラー化するときに問題になります。x86 ブートパーティションは、Solaris fdisk パーティションの外にあります。このため、x86 ブートパーティションは Solaris ボリュームマネージャでミラー化できません。さらに、x86 ブートパーティションのコピーは 1 つしかないため、シングルポイント障害にもなります。

Solaris OS に独立した x86 ブートパーティションがあるかどうかは、次のように識別できます。x86 ブートパーティションは、/etc/vfstab ファイル内に、次のようなエントリでマウントされます。

```
/dev/dsk/c2t1d0p0:boot  -       /boot   pcfs    -       no      -
```

別の x86 ブートパーティションが存在しない場合、このエントリは /etc/vfstab ファイルに存在しません。

ルート (/) ファイルシステムをミラー化するためには、fdisk パーティションをカスタマイズして、x86 ブートパーティションを削除し、単一の Solaris fdisk パーティションを使用するようにします。Solaris ボリュームマネージャのルートミラー化を使用する予定がある場合、システムのインストール時に、独立した x86 ブートパーティションを作成しないでください。すでにシステムがインストールされ、別の x86 ブートパーティションが作成されている場合、fdisk コマンドを使って、その fdisk パーティションを削除して、システムをインストールし直します。インストール中に別の x86 ブートパーティションを作成しないようにするには、インストールプロセス中にディスクパーティションをカスタマイズします。

注 – Solaris ボリュームマネージャでミラー化できるのは、Solaris fdisk パーティション内のスライスだけです。複数の fdisk パーティションが存在する場合、Solaris fdisk パーティションの外にあるデータを保護するには、別の方法を使用する必要があります。

---

3 マスターブートプログラムを使って、二次サブミラーをブート可能にします。

a. fdisk コマンドを使って、マスターブートプログラムを指定します。

```
# fdisk -b /usr/lib/fs/ufs/mboot /dev/rdsk/c1t1d0p0
```

次の画面が表示されます。

```
            Total disk size is 31035 cylinders
            Cylinder size is 1146 (512 byte) blocks

                                                 Cylinders
```

```
      Partition   Status    Type          Start   End   Length    %
      =========   ======    ============  =====   ===   ======   ===
          1        Active   Solaris           1  31034    31034    100


   SELECT ONE OF THE FOLLOWING:
      1. Create a partition
      2. Specify the active partition
      3. Delete a partition
      4. Change between Solaris and Solaris2 Partition IDs
      5. Exit (update disk configuration and exit)
      6. Cancel (exit without updating disk configuration)
   Enter Selection:
```

b. メニューから **5** 番を選択して、**Return** キーを押します。

4 二次サブミラーをブート可能にするため、二次サブミラーにブートブロックをインストールします。

この `fdisk` パーティションから Solaris OS をブートするには、二次サブミラーがあるディスクにスライス 8 が必要です。このスライスには、パーティションブートレコード (`pboot`)、当該ディスクの Solaris VTOC、およびブートブロックが含まれます。この情報はディスクに固有であり、Solaris ボリュームマネージャではミラー化されません。しかし、一次ディスクに障害が発生した場合に二次ディスクからブートできるようにするには、両方のディスクがブート可能であることを保証する必要があります。`installboot` コマンドを使って、2 番目のディスクを Solaris ブート可能ディスクと設定します。詳細は、`installboot(1M)` のマニュアルページを参照してください。

このディスクのスライス 2 をデバイスとして指定する必要があります。さらに、このスライス 2 はディスク全体から構成されている必要があります。

```
# installboot /usr/platform/i86pc/lib/fs/ufs/pboot \
/usr/platform/i86pc/lib/fs/ufs/bootblk /dev/rdsk/c1t1d0s2
```

5 ミラー化する既存のルート (/) ファイルシステムが含まれているスライスを特定します。この例では、スライス `c1t0d0s0` を使用します。

6 次のどちらかの方法を使って、前の手順で特定したスライスに新しい **RAID-0** ボリュームを作成します。**RAID-0** ボリュームに含めることができるのは、単一のスライスだけです。

- Solaris 管理コンソール内の「拡張ストレージ」から、「ボリューム (Volumes)」ノードを開き、「アクション (Action)」、「ボリュームの作成 (Create Volume)」の順に選択します。画面の指示に従います。詳細は、オンラインヘルプを参照してください。
- 次の形式の `metainit` コマンドを使用します。

```
# metainit -f volume-name number-of-stripes components-per-stripe component-name
```

`-f`	コマンド処理を強制的に続けます。スライスにマウントされたファイルシステムが含まれている場合は、このオプションを使用する必要があります。
volume-name	作成するボリュームの名前を指定します。ボリュームの命名方式については、45 ページの「ボリューム名」を参照してください。
number-of-stripes	作成するストライプの数を指定します。
components-per-stripe	各ストライプに与えるコンポーネントの数を指定します。
component-names	使用するコンポーネントの名前を指定します。この例では、ルートスライス `c0t0d0s0` を使用します。

7 未使用のスライス (この例では `c1t1d0s0`) に **2** 番目の **RAID-0** ボリュームを作成します。このボリュームは **2** 番目のサブミラーとして使用します。**2** 番目のサブミラーのサイズは、最初のサブミラー以上でなければなりません。この手順では、次のどちらかの方法を使用します。

注 – 2 番目のサブミラーとして使用するスライスは「root」というスライスタグを持っており、かつ、ルートスライスはスライス 0 である必要があります。スライスを構成する方法については、`format(1M)` のマニュアルページを参照してください。

- Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開き、「アクション (Action)」、「ボリュームを作成 (Create Volume)」の順に選択し、画面の指示に従います。詳細は、オンラインヘルプを参照してください。
- 次の形式の `metainit` コマンドを使用します。

  ```
  # metainit volume-name number-of-stripes components-per-stripes component-names
  ```

注 – オプションの説明については、手順 6 を参照してください。

---

8 次のどちらかの方法で **1** 面ミラーを作成します。

- Solaris 管理コンソール内の「拡張ストレージ」から、「ボリューム (Volumes)」ノードを開き、「アクション (Action)」、「ボリュームの作成 (Create Volume)」の順に選択します。画面の指示に従います。詳細は、オンラインヘルプを参照してください。
- 次の形式の `metainit` コマンドを使用します。

  ```
  # metainit volume-name -m submirror-name
  ```

volume-name	作成するボリュームの名前を指定します。
`-m`	ミラーを作成することを指定します。

submirror-name ミラーの最初のサブミラーとして使用するコンポーネントの名前を指定します。この例では、ルートスライスが含まれる RAID-0 ボリュームです。

9 新たにミラー化したファイルシステムをマウントし直します。`metaroot` *volume-name* コマンドを実行します。*volume-name* のところには、作成したミラーの名前を入れます。さらに、システムをリブートします。

```
# metaroot volume-name
# reboot
```

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

10 次の形式の `metattach` コマンドを使って、**2** 番目のサブミラーを接続します。

```
# metattach volume-name submirror-name
```

volume-name サブミラーを追加する RAID-1 ボリュームの名前を指定します。

submirror-name 次にミラーに追加するサブミラーとなるコンポーネントの名前を指定します。

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

11 代替ブートパスを記録します。

一次サブミラーに障害が発生した場合に二次サブミラーからブートするように、システムを構成する必要があります。二次サブミラーを持つディスクからシステムがブートできるようにするには、そのディスクを代替ブートデバイスとして認識するようにシステムを構成します。

a. 代替ブートデバイスへのパスを調べます。**2** 番目のサブミラーとしてルート **(/)** ファイルシステムのミラーに接続するスライスで、`ls -l` コマンドを使用します。

```
# ls -l /dev/dsk/c1t1d0s0
lrwxrwxrwx 1  root root  55 Mar 5 12:54  /dev/rdsk/c1t1d0s0 -> ../.
./devices/eisa/eha@1000,0/cmdk@1,0:a
```

b. ここで、`/devices` ディレクトリに続く文字列 `/eisa/eha@1000,0/cmdk@1,0:a` を記録しておきます。これがデバイスツリーパスです。

注 – システムが利用できない場合に備えて、システム上以外の場所にもこの情報を書き留めておくべきです。これによって、DCA を使ってシステムをブートする必要がある場合に、デバイスツリーパス情報の入力が簡単になります。

---

c. `eeprom` コマンドを使って、代替ブートパスを定義します。たとえば、次のように指定します。

```
# eeprom altbootpath=/eisa/eha@1000,0/cmdk@1,0:a
```

一次サブミラーに障害が発生した場合、システムは二次サブミラーからブートしようとします。自動的に二次ディスクにフェイルオーバーするように BIOS を構成できる場合、このブートプロセスは自動的に行われます。そうでない場合、BIOS に入って、二次ディスクからブートするように構成する必要があります。ブートを開始したあと、システムはまず、bootpath デバイスからブートしようとします。ルートミラーの一次ブートディスクは動作していないため、システムは次に、altbootpath デバイスからブートしようとします。BIOS の設定を構成する方法については、BIOS のユーザーガイドを参照してください。

システムが自動的にブートしない場合、DCA を使って、二次サブミラーを選択できます。システムによっては、ブートプロセス中に DCA に入ることも選択できます。この方法が使用できない場合、x86 ブートフロッピーディスクからブートしてから、DCA を使って、二次サブミラーを選択する必要があります。オペレーティングシステムがブートしたあと、eeprom bootpath の値を、代替ブートパスとして設定した値 (altbootpath の値) に変更します。こうすれば、システムが自動的にブートします。

eeprom コマンドを使用する方法についての詳細は、eeprom(1M) のマニュアルページを参照してください。

例 11–7 x86: DCA を使ってルート (/) ファイルシステムからミラーを作成する

```
# metainit -f d1 1 1 c0t0d0s0
d1: Concat/Stripe is setup
# metainit d2 1 1 c0t1d0s0
d2: Concat/Stripe is setup
# metainit d0 -m d1
d0: Mirror is setup
# metaroot d0
# lockfs -fa
# reboot
...
# metattach d0 d2
d0: Submirror d2 is attached
# ls -l /dev/dsk/c0t1d0s0
lrwxrwxrwx   1 root     root          88 Feb  8 15:51 /dev/dsk/c1t3d0s0 ->
../../devices/pci@1f,0/pci@1,1/ide@3/dad@0,0:a,raw
# eeprom altbootpath=/pci@1f,0/pci@1,1/ide@3/dad@0,0:a,raw
# fdisk -b /usr/lib/fs/ufs/mboot /dev/dsk/c0t1d0p0
             Total disk size is 31035 cylinders
             Cylinder size is 1146 (512 byte) blocks

                                               Cylinders
      Partition   Status    Type          Start   End   Length    %
      =========   ======    ============  =====   ===   ======   ===
          1        Active   Solaris           1  31034    31034    100
```

```
SELECT ONE OF THE FOLLOWING:
   1. Create a partition
   2. Specify the active partition
   3. Delete a partition
   4. Change between Solaris and Solaris2 Partition IDs
   5. Exit (update disk configuration and exit)
   6. Cancel (exit without updating disk configuration)
Enter Selection: 5
# installboot /usr/platform/i86pc/lib/fs/ufs/pboot \
/usr/platform/i86pc/lib/fs/ufs/bootblk /dev/rdsk/c0t1d0s2
```

ルート (/) ファイルシステムをミラー化した場合のブート時の警告について

ルート (/) ファイルシステムをミラー化すると、エラーメッセージがコンソールに表示され、/etc/syslog.conf ファイルの定義に従って システムログに記録されます。これらのエラーメッセージは、問題を意味するわけではありません。これらのエラーメッセージは、現在使用していないデバイスタイプごとに表示されます。未使用のモジュールを強制的にロードすることはできないからです。次のようなエラーメッセージが表示されます。

```
Jul 13 10:17:42 ifr genunix: [ID 370176 kern.warning] WARNING: forceload of
    misc/md_trans failed
Jul 13 10:17:42 ifr genunix: [ID 370176 kern.warning] WARNING: forceload of
    misc/md_raid failed
Jul 13 10:17:42 ifr genunix: [ID 370176 kern.warning] WARNING: forceload of
    misc/md_hotspares failed
```

このようなエラーメッセージは無視してかまいません。

サブミラーに関する作業

▼ サブミラーを接続するには

注 – 「`can't attach labeled submirror to an unlabeled mirror`」というエラーメッセージは、ミラーに RAID-0 ボリュームを接続できなかったことを意味します。ラベル付きボリューム (サブミラー) とは、その最初のコンポーネントがシリンダ 0 から始まるものをいいます。一方、ラベルなしボリュームの最初のコンポーネントはシリンダ 1 から始まります。Solaris ボリュームマネージャでは、ラベル付きサブミラーのラベルが壊れるのを防ぐため、ラベル付きサブミラーのラベルなしミラーへの接続を許可しません。

始める前に　101 ページの「RAID-1 ボリュームの作成と保守」を確認します。

1. サブミラーとして使用するコンポーネント (連結またはストライプ) を特定します。
   コンポーネントは、ミラー内の既存のサブミラーと同じかそれ以上のサイズにする必要があります。サブミラーとして使用するボリュームをまだ作成していない場合は、88 ページの「RAID-0 (ストライプ方式) ボリュームの作成」または90 ページの「RAID-0 (連結方式) ボリューム」を参照してください。

2. すべてのデータの最新のバックアップを取っているか確認します。また、この操作には **root** 権限が必要です。

3. `metastat` コマンドを使って、作業するミラーが「**Okay**」状態であることを確認します。

   ```
   # metastat mirror
   ```

4. 次のどちらかの方法でサブミラーを接続します。
   - Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開きます。ミラーを選択します。次に、「アクション (Action)」、「プロパティ (Properties)」の順に選択して、「サブミラー (Submirror)」タブをクリックします。画面の指示に従います。詳細は、オンラインヘルプを参照してください。
   - `metattach` *mirror submirror* コマンドを実行します。

     ```
     # metattach mirror submirror
     ```

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

5. `metastat` コマンドを使って、ミラーの状態を調べます。

   ```
   # metastat mirror
   ```

例 11–8 サブミラーを接続する

```
# metastat d30
d30: mirror
    Submirror 0: d60
      State: Okay
...
# metattach d30 d70
d30: submirror d70 is attached
# metastat d30
d30: mirror
    Submirror 0: d60
      State: Okay
    Submirror 1: d70
      State: Resyncing
    Resync in progress: 41 % done
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Size: 2006130 blocks
...
```

この例では、サブミラー `d70` を 1 面ミラー `d30` に接続します。ミラーにサブミラーを接続したときに、2 面ミラーを作成します。ミラー `d30` は、最初、サブミラー `d60` から構成されています。サブミラー `d70` は RAID-0 ボリュームです。まず、サブミラーを接続する前に、`metastat` コマンドでミラーが「正常 (Okay)」状態であることを確認します。`metattach` コマンドを実行すると、新しいサブミラーと既存のミラーの同期がとられます。ミラーに新しいサブミラーが接続されると、そのことを示すメッセージが表示されます。サブミラーとミラーの同期がとられていることを確認するために、`metastat` コマンドを実行します。

▼ サブミラーを切り離すには

始める前に 101 ページの「RAID-1 ボリュームの作成と保守」を確認します。

1. **root** 権限を持っていることを確認します。すべてのデータについて、最新のバックアップがあることを確認します。
2. `metastat` コマンドを使って、作業するミラーが「**Okay**」状態であることを確認します。
3. 次のどちらかの方法でサブミラーを切り離します。
   - Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開きます。ミラーを選択します。次に、「アクション (Action)」、「プロパティ (Properties)」の順に選択して、「サブミラー (Submirror)」タブをクリックします。画面の指示に従います。詳細は、オンラインヘルプを参照してください。

- metadetach コマンドを使ってミラーからサブミラーを切り離します。

  # metadetach *mirror submirror*

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

例 **11–9** サブミラーを切り離す

```
# metastat
d5: mirror
    Submirror 0: d50
...
# metadetach d5 d50
d5: submirror d50 is detached
```

この例では、ミラー d5 にサブミラー d50 があります。metadetach コマンドを使って、サブミラーを切り離します。d50 のスライスは他の場所で再使用できます。ミラーからサブミラーを切り離すと、確認メッセージが表示されます。

▼ サブミラーをオフラインまたはオンラインにするには

metaonline コマンドを実行できるのは、そのサブミラーが metaoffline コマンドによってオフラインにされている場合に限られます。metaonline コマンドを実行すると、サブミラーとミラーの再同期が自動的に開始します。

注 – metaoffline コマンドの機能は、metadetach コマンドの機能と同様です。ただし、metaoffline コマンドでは、サブミラーとミラーの論理的な関連付けは切り離されません。

始める前に 101 ページの「RAID-1 ボリュームの作成と保守」を確認します。

1. すべてのデータの最新のバックアップを取っているか確認します。また、この操作には **root** 権限が必要です。
2. 次のどちらかの方法でサブミラーをオンラインまたはオフラインにします。
   - Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開きます。ミラーを選択します。次に、「アクション (Action)」、「プロパティ (Properties)」の順に選択して、「サブミラー (Submirror)」タブをクリックします。画面の指示に従います。詳細は、オンラインヘルプを参照してください。
   - metaoffline コマンドでサブミラーをオフラインにします。

```
# metaoffline mirror submirror
```

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

- metaonline コマンドでサブミラーをオンラインにします。

```
# metaonline mirror submirror
```

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

例 11–10 サブミラーをオフラインにする

```
# metaoffline d10 d11
d10: submirror d11 is offlined
```

この例では、サブミラー d11 をミラー d10 からオフラインにします。読み取りは、他のサブミラーから引き続き行われます。最初の書き込みが行われた時点でミラーは同期していない状態になります。この不整合の状態は、オフラインにしたサブミラーをオンラインに戻すと訂正されます。

例 11–11 サブミラーをオンラインにする

```
# metaonline d10 d11d10: submirror d11 is onlined
```

この例では、サブミラー d11 をミラー d10 でオンラインに戻します。

▼ サブミラー内のスライスを有効にするには

始める前に 246 ページの「RAID-1 および RAID-5 ボリューム内のコンポーネントの交換と有効化の概要」および 101 ページの「RAID-1 ボリュームの作成と保守」を確認します。

1. すべてのデータの最新のバックアップを取っているか確認します。また、この操作には root 権限が必要です。
2. 次のどちらかの方法でサブミラー内のスライスを有効にします。
   - Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開きます。ミラーを選択します。次に、「アクション (Action)」、「プロパティ (Properties)」の順に選択して、「サブミラー (Submirror)」タブをクリックします。画面の指示に従います。詳細は、オンラインヘルプを参照してください。
   - metareplace コマンドを使用して、サブミラー内のエラーが発生したスライスを有効にします。

     ```
     # metareplace -e mirror failed-slice
     ```

metareplace コマンドを実行すると、修復または交換されたスライスとミラーのほかの部分との再同期が自動的に開始されます。

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

例 11–12 サブミラー内のスライスを有効にする

```
# metareplace -e d11 c1t4d0s7
d11: device c1t4d0s7 is enabled
```

この例では、ミラー d11 のサブミラーに含まれるスライス c1t4d0s7 にソフトエラーがあります。-e オプションを付けた metareplace コマンドを実行して、エラーの発生したスライスを有効にします。

物理ディスクに障害が発生した場合は、そのディスクをシステム上で利用可能なほかのディスク (およびそのスライス) と交換できます (142 ページの「サブミラー内のスライスを交換するには」を参照)。あるいは、ディスクを修復または交換し、フォーマットした上で、この例のように、 -e オプションを指定した metareplace コマンドを使用することもできます。

RAID-1 ボリュームの保守

▼ ミラーとサブミラーの状態を表示するには

始める前に RAID-1 ボリュームとサブミラーに関連する状態情報の概要については、105 ページの「保守作業を決定するサブミラーの状態」を参照してください。

◗ 次のどちらかの方法でミラーやサブミラーの状態をチェックします。

- Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開きます。ミラーを選択します。次に、「アクション (Action)」、「プロパティ (Properties)」の順に選択します。画面の指示に従います。詳細は、オンラインヘルプを参照してください。
- ミラーに metastat コマンドを実行し、各サブミラーの状態を表示します。

  ```
  # metastat mirror
  ```

  ミラーのパス番号、読み取りオプション、または書き込みオプションを変更する方法については、140 ページの「RAID-1 ボリュームオプションを変更するには」を参照してください。

  デバイス状態のチェックについては、metastat(1M) のマニュアルページを参照してください。

例 11–13 RAID-1 ボリュームの状態をチェックする

次に、metastat コマンドの出力例を示します。ミラー名を指定しないで metastat コマンドを使用すると、すべてのミラーのあらゆる状態が表示されます。

```
# metastatd70: Mirror
    Submirror 0: d71
      State: Okay
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Size: 12593637 blocks

d71: Submirror of d70
    State: Okay
    Size: 12593637 blocks
    Stripe 0:
        Device              Start Block  Dbase State        Reloc  Hot Spare
        c1t3d0s3                   0     No    Okay         Yes
    Stripe 1:
        Device              Start Block  Dbase State        Reloc  Hot Spare
        c1t3d0s4                   0     No    Okay         Yes
    Stripe 2:
        Device              Start Block  Dbase State        Reloc  Hot Spare
        c1t3d0s5                   0     No    Okay         Yes
d0: Mirror
    Submirror 0: d1
      State: Okay
    Submirror 1: d2
      State: Okay
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Size: 5600 blocks

d1: Submirror of d0
    State: Okay
    Size: 5600 blocks
    Stripe 0:
        Device              Start Block  Dbase State        Hot Spare
        c0t2d0s7                   0     No    Okay

...
```

ミラー名引数を指定して metastat コマンドを使用すると、そのミラーに対応する出力が表示されます。

```
metastat d70
d70: Mirror
    Submirror 0: d71
      State: Okay
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Size: 12593637 blocks

d71: Submirror of d70
    State: Okay
    Size: 12593637 blocks
    Stripe 0:
        Device              Start Block  Dbase State        Reloc  Hot Spare
        c1t3d0s3                  0     No    Okay         Yes
    Stripe 1:
        Device              Start Block  Dbase State        Reloc  Hot Spare
        c1t3d0s4                  0     No    Okay         Yes
    Stripe 2:
        Device              Start Block  Dbase State        Reloc  Hot Spare
        c1t3d0s5                  0     No    Okay         Yes
```

`metastat` コマンドは、ミラーのサブミラーごとに、その状態、「invoke」行 (エラーがある場合)、割り当てられたホットスペア集合 (ホットスペアがある場合)、ブロック数、サブミラーの各スライスの情報を表示します。

▼ RAID-1 ボリュームオプションを変更するには

始める前に　104 ページの「RAID-1 ボリュームのオプション」を確認します。

1. すべてのデータの最新のバックアップを取っているか確認します。また、この操作には **root** 権限が必要です。
2. 次のどちらかの方法で、**RAID-1** オプションを変更します。
   - Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開きます。ミラーを選択します。次に、「アクション (Action)」、「プロパティ (Properties)」の順に選択します。画面の指示に従います。詳細は、オンラインヘルプを参照してください。
   - `metaparam` コマンドを使ってミラーのオプションを表示および変更します。

     ```
     # metaparam [mirror options] mirror
     ```

     ミラーオプションについては、104 ページの「RAID-1 ボリュームのオプション」を参照してください。また、`metaparam(1M)` のマニュアルページも参照してください。

例 11–14 RAID-1 ボリュームの読み取りポリシーを変更する

```
# metaparam -r geometric d30
# metaparam d30
d30: mirror current parameters are:
    Pass: 1
    Read option: geometric (-g)
    Write option: parallel (default)
```

この例の -r オプションは、ミラーの読み取りポリシーを geometric に変更します。

例 11–15 RAID-1 ボリュームの書き込みポリシーを変更する

```
# metaparam -w serial d40
# metaparam d40
d40: mirror current parameters are:
    Pass: 1
    Read option: roundrobin (default)
    Write option: serial (-S)
```

この例の -w オプションは、ミラーの書き込みポリシーを serial に変更します。

例 11–16 RAID-1 ボリュームのパス番号を変更する

```
# metaparam -p 5 d50
# metaparam d50
d50: mirror current parameters are:
    Pass: 5
    Read option: roundrobin (default)
    Write option: parallel (default)
```

この例の -p オプションは、ミラーのパス番号を 5 に変更します。

▼ RAID-1 ボリュームを拡張するには

始める前に 101 ページの「RAID-1 ボリュームの作成と保守」を確認します。

1. すべてのデータの最新のバックアップを取っているか確認します。また、この操作には root 権限が必要です。
2. 次のどちらかの方法でミラーを拡張します。
   - Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開きます。ミラーを選択します。さらに「アクション (Action)」、「プロパティ (Properties)」の順に選択し、サブミラー (Submirror) タブをクリックします。画面の指示に従います。詳細は、オンラインヘルプを参照してください。

- metattach コマンドを使って各サブミラーにスライスを接続します。

  # metattach *submirror slice*

  ミラー内のすべてのサブミラーを拡張する必要があります。詳細は、metattach(1M) のマニュアルページを参照してください。

3 metattach コマンドを使って、ミラーのサイズをサブミラーのサイズに基づいて計算し直すように指示します。

 # metattach *mirror*

例 11–17 マウントしているファイルシステムを持つ 2 面ミラーを拡張する

```
# metastat
d8: Mirror
    Submirror 0: d9
      State: Okay
    Submirror 1: d10
      State: Okay
...
# metattach d9 c0t2d0s5
d9: component is attached
# metattach d10 c0t3d0s5
d10: component is attached
# metattach d8
```

この例では、2 つのディスクドライブをミラーの2 つのサブミラーに連結することによって、ミラー化したマウント済みのファイルシステムを拡張します。ミラー d8 は、2 つのサブミラー d9 と d10 から構成されています。

参照 UFS の場合は、ミラーボリュームに対して growfs(1M) コマンドを実行します。245 ページの「ファイルシステムを拡張するには」を参照してください。

データベースなど、raw ボリュームを使用するアプリケーションは、独自の方法で記憶領域を拡張できなければなりません。

RAID-1 ボリュームのコンポーネント障害に対する処置

▼ サブミラー内のスライスを交換するには

始める前に 246 ページの「RAID-1 および RAID-5 ボリューム内のコンポーネントの交換と有効化の概要」および 101 ページの「RAID-1 ボリュームの作成と保守」を確認します。

1. すべてのデータの最新のバックアップを取っているか確認します。また、この操作には **root** 権限が必要です。

2. `metastat` コマンドで **RAID-1** ボリュームとそのサブミラーの状態を調べます。

   ```
   # metastat mirror-name
   ```

3. 次のどちらかの方法でサブミラー内のスライスを交換します。

   - Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開きます。ミラーを選択します。次に、「アクション (Action)」、「プロパティ (Properties)」の順に選択して、「サブミラー (Submirror)」タブをクリックします。画面の指示に従います。詳細は、オンラインヘルプを参照してください。
   - 次の形式の `metareplace` コマンドを使用して、サブミラーのスライスを交換します。

     ```
     # metareplace mirror-name component-name
     ```

     - *mirror-name* は、作成するボリュームの名前です。
     - *component-name* は、置き換えられるコンポーネントの名前です。

mirror-name	作成するボリュームの名前を指定します。
component-name	交換するコンポーネントの名前を指定します。

     詳細については、次の例と `metainit(1M)` のマニュアルページを参照してください。

例 11–18 ミラー内の障害が発生したスライスを交換する

次の例では、障害が発生したスライスを交換します。ただし、システムは、ホットスペア集合を使って障害が発生したディスクを自動的に交換するようには構成されていないものとします。ホットスペア集合については、第 16 章を参照してください。

```
# metastat d6
d6: Mirror
    Submirror 0: d16
      State: Okay
    Submirror 1: d26
      State: Needs maintenance
...
d26: Submirror of d6
    State: Needs maintenance
    Invoke: metareplace d6 c0t2d0s2 <new device>
...
# metareplace d6 c0t2d0s2 c0t2d2s2
d6: device c0t2d0s2 is replaced with c0t2d2s2
```

`metastat` コマンドを使用して、ミラー `d6` にサブミラー `d26` があり、そのスライスの状態が「保守が必要 (Needs maintenance)」であることを確認します。`metareplace` コマンドに

より、metastat コマンドの出力中の「起動」行に従って、このスライスをシステム内の別のスライスで置き換えます。スライスが置き換えられ、サブミラーの再同期が開始されたことを示すメッセージが表示されます。

▼ サブミラーを交換するには

始める前に 246 ページの「RAID-1 および RAID-5 ボリューム内のコンポーネントの交換と有効化の概要」および 101 ページの「RAID-1 ボリュームの作成と保守」を確認します。

1. すべてのデータの最新のバックアップを取っているか確認します。また、この操作には **root** 権限が必要です。

2. metastat コマンドで **RAID-1** ボリュームとそのサブミラーの状態を調べます。

   ```
   # metastat mirror-name
   ```

3. 次のどちらかの方法でサブミラーを交換します。

   - Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開きます。ミラーを選択します。次に、「アクション (Action)」、「プロパティ (Properties)」の順に選択して、「サブミラー (Submirror)」タブをクリックします。画面の指示に従います。詳細は、オンラインヘルプを参照してください。
   - metadetach、metaclear、metatinit、および metattach コマンドを使って、サブミラー全体を交換します。

     a. metadetach コマンドを使ってミラーから障害のあるサブミラーを切り離します。

        ```
        # metadetach -f mirror-name submirror
        ```

-f	切り離しを強制的に実行します。
mirror-name	ミラー名を指定します。
submirror	切り離すサブミラーを指定します。

     b. metaclear コマンドを使用して、サブミラーを削除します。

        ```
        # metaclear -f submirror
        ```

-f	サブミラーの削除を強制的に実行します。
submirror	削除するサブミラーを指定します。

     c. metainit コマンドを使用して、新しいサブミラーを作成します。

        ```
        # metainit volume-name number-of-stripes components-per-stripe component-name
        ```

volume-name	作成するボリュームの名前を指定します。ボリュームの命名方式については、45 ページの「ボリューム名」を参照してください。

number-of-stripes 作成するストライプの数を指定します。

components-per-stripe 各ストライプに与えるコンポーネントの数を指定します。

component-names 使用するコンポーネントの名前を指定します。この例では、ルートスライス `c0t0d0s0` を使用します。

d. `metattach` コマンドを使って新しいサブミラーを接続します。

```
# metattach mirror submirror
```

例 11–19 ミラー内のサブミラーを交換する

次の例では、アクティブなミラー内のサブミラーを交換します。

```
# metastat d20
d20: Mirror
    Submirror 0: d21
      State: Okay
    Submirror 1: d22
      State: Needs maintenance
...
# metadetach -f d20 d22
d20: submirror d22 is detached
# metaclear -f d22
d22: Concat/Stripe is cleared
# metainit d22 2 1 c1t0d0s2 1 c1t0d1s2
d22: Concat/Stripe is setup
# metattach d20 d22
d20: components are attached
```

この例では、`metastat` コマンドを使って、2 面ミラー `d20` にサブミラー `d22` があり、その状態が「保守が必要 (Needs maintenance)」であることを確認します。この例では、サブミラー全体を削除し、作成し直す必要があります。`metadetach` コマンドに `-f` オプションを指定して、障害のあるサブミラーを強制的にミラーから切り離します。`metaclear` コマンドは、サブミラーを削除します。`metainit` コマンドは、新しいスライスからサブミラー `d22` を再作成します。最後に、`metattach` コマンドは、作成し直したサブミラーを接続します。ミラーの再同期が自動的に開始されます。

新しいボリューム `d22` の構成は、置き換えるコンポーネントによって異なります。この場合は連結で十分に連結を置き換えることができます。しかし、ストライプの場合は性能に影響が出るので、連結が最適な置換とはなりません。

ミラーが1 面ミラーとなっている間は、データの冗長性が一時的に失われます。

RAID-1 ボリュームの削除 (ミラー化の解除)

▼ ファイルシステムのミラー化を解除するには

この手順では、システムの動作中にマウント解除できるファイルシステムのミラー化を解除します。ルート (/)、 /var、/usr、swap、またはシステムの稼働中はマウントを解除できないその他のファイルシステムのミラー化を解除する場合は、148 ページの「マウント解除できないファイルシステムのミラー化を解除するには」を参照してください。

始める前に 101 ページの「RAID-1 ボリュームの作成と保守」を確認します。

1. すべてのデータの最新のバックアップを取っているか確認します。また、この操作には **root** 権限が必要です。
2. 少なくとも **1** つのサブミラーが「正常 **(Okay)**」状態であることを確認します。

   # metastat *mirror*
3. ファイルシステムをマウント解除します。

   # umount */file-system*
4. サブミラーを切り離します。このサブミラーは、この後もこのファイルシステムのために使用されます。

   # metadetach *mirror submirror*

   詳細は、metadetach(1M) のマニュアルページを参照してください。
5. ミラーと残りのサブコンポーネントを削除します。

   # metaclear -r *mirror*

   詳細は、metaclear(1M) のマニュアルページを参照してください。
6. 必要であれば、/etc/vfstab ファイルを編集して、手順 **4** で切り離したコンポーネントを使用するように指定します。
7. ファイルシステムを再びマウントします。

   # mount */file-system*

例 11–20 /opt ファイルシステムのミラー化を解除する

```
# metastat d4
d4: Mirror
    Submirror 0: d2
```

```
      State: Okay
    Submirror 1: d3
      State: Okay
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Size: 2100735 blocks (1.0 GB)

d2: Submirror of d4
    State: Okay
    Size: 2100735 blocks (1.0 GB)
    Stripe 0:
        Device     Start Block  Dbase        State Reloc Hot Spare
        c0t0d0s0          0     No            Okay   Yes


d3: Submirror of d4
    State: Okay
    Size: 2100735 blocks (1.0 GB)
    Stripe 0:
        Device     Start Block  Dbase        State Reloc Hot Spare
        c1t0d0s0          0     No            Okay   Yes

...
# umount /opt
# metadetach d4 d2
d4: submirror d2 is detached
# metaclear -r d4
d4: Mirror is cleared
d3: Concat/Stripe is cleared
    (Edit the /etc/vfstab file so that the entry for /opt is changed from d4 to the underlying slice or volume)
# mount /opt
```

この例の `/opt` ファイルシステムは 2 面ミラー `d4` から構成されています。このミラーのサブミラーは `d2` と `d3` です。サブミラーはスライス `/dev/dsk/c0t0d0s0` と `/dev/dsk/c1t0d0s0` からなります。`metastat` コマンドを使用して、少なくとも 1 つのサブミラーが「正常 (Okay)」状態であることを確認します。(「正常 (Okay)」状態のサブミラーが存在しないミラーは、最初に修復する必要があります)。ファイルシステムのマウントを解除します。次に、サブミラー `d2` を切り離します。`metaclear -r` コマンドで、ミラーともう 1 つのサブミラー `d3` を削除します。

次に、該当するスライスを参照するように `/etc/vfstab` ファイル内の `/opt` 用のエントリを編集します。

この例では、`/etc/vfstab` ファイルに `/opt` ファイルシステム用の次のエントリが指定されています。

```
/dev/md/dsk/d4  /dev/md/rdsk/d4  /opt ufs  2  yes -
```

次のようにエントリを変更します。

```
/dev/md/dsk/d2  /dev/md/rdsk/d2  /opt ufs  2  yes -
```

サブミラー名を使用することによって、ファイルシステムをボリュームにマウントしたままにできます。最後に、/opt ファイルシステムを再びマウントします。

/etc/vfstab ファイル内で d4 の代わりに d2 を使用したことによって、このミラーのミラー化を解除しています。d2 は 1 つのスライスから構成されているため、このデバイスでボリュームをサポートしたくなければ、ファイルシステムをスライス名 (/dev/dsk/c0t0d0s0) にマウントできます。

▼ マウント解除できないファイルシステムのミラー化を解除するには

次の作業で、ルート (/)、/usr、/opt、swap を含め、通常のシステム稼働時にはマウントを解除できないファイルシステムのミラー化を解除します。

1. すべてのデータの最新のバックアップを取っているか確認します。また、この操作には **root** 権限が必要です。

2. 少なくとも 1 つのサブミラーが「正常 (Okay)」状態であることを確認します。

   ```
   # metastat mirror
   ```

3. サブミラーを切り離します。このサブミラーは、引き続きファイルシステム用として使用します。

   ```
   # metadetach mirror submirror
   ```

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

4. ミラー化を解除するファイルシステムに応じて、次のどちらかのコマンドを使用します。

   - /usr、/opt、または swap ファイルシステムの場合は、/etc/vfstab ファイルに指定されているファイルシステムエントリを変更し、Solaris ボリュームマネージャ以外のデバイス (スライス) が使用されるようにします。
   - ルート (/) ファイルシステムの場合のみ:metaroot コマンドを実行します。

     ```
     # metaroot rootslice
     ```

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

5 システムをリブートします。

```
# reboot
```

6 残りのミラーとサブミラーを削除します。

```
# metaclear -r mirror
```

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

例 11–21 ルート (/) ファイルシステムのミラー化を解除する

```
# metastat d0
d0: Mirror
    Submirror 0: d10
      State: Okay
    Submirror 1: d20
      State: Okay
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Size: 2100735 blocks (1.0 GB)

d10: Submirror of d0
    State: Okay
    Size: 2100735 blocks (1.0 GB)
    Stripe 0:
        Device     Start Block  Dbase        State Reloc Hot Spare
        c0t3d0s0          0     No            Okay   Yes


d20: Submirror of d0
    State: Okay
    Size: 2100735 blocks (1.0 GB)
    Stripe 0:
        Device     Start Block  Dbase        State Reloc Hot Spare
        c1t3d0s0          0     No            Okay   Yes


# metadetach d0 d20
d0: submirror d20 is detached
# metaroot /dev/dsk/c0t3d0s0
# reboot
...
# metaclear -r d0
d0: Mirror is cleared
d10: Concat/Stripe is cleared
# metaclear d20
d20: Concat/Stripe is cleared
```

この例のルート (/) ファイルシステムは 2 面ミラー d0 です。このミラーのサブミラーは d10 と d20 です。サブミラーはスライス /dev/dsk/c0t3d0s0 と /dev/dsk/c1t3d0s0 からなります。metastat コマンドを使用して、少なくとも 1 つのサブミラーが「正常 (Okay)」状態であることを確認します。(「正常 (Okay)」状態のサブミラーが存在しないミラーは、最初に修復する必要があります)。次に、サブミラー d20 を切り離して、ミラー d0 を 1 面ミラーにします。

「ルートスライス」は、ルート (/) ファイルシステムが含まれているスライスです。metaroot コマンドには、システムのブートに使用する *rootslice* を指定します。このコマンドによって /etc/system ファイルと /etc/vfstab ファイルが編集されます。このコマンドは、ルート (/) ファイルシステムのミラー化を指定した情報を削除します。

システムのリブート後、metaclear -r コマンドがミラーともう 1 つのサブミラー d10 を削除します。最後の metaclear コマンドは、サブミラー d20 を削除します。

例 11–22 swap ファイルシステムのミラー化を解除する

```
# metastat d1
d1: Mirror
    Submirror 0: d11
      State: Okay
    Submirror 1: d21
      State: Okay
...
# metadetach d1 d21
d1: submirror d21 is detached
    (/etc/vfstab ファイルを編集して、swap のエントリをメタデバイスからスライス名に変更する)
# reboot
...
# metaclear -r d1
d1: Mirror is cleared
d11: Concat/Stripe is cleared
# metaclear d21
d21: Concat/stripe is cleared
```

この例の swap ファイルシステムは、2 面ミラー d1 からなります。このミラーのサブミラーは d11 と d21 です。サブミラーはスライス /dev/dsk/c0t3d0s1 と /dev/dsk/c1t3d0s1 からなります。metastat コマンドを使用して、少なくとも 1 つのサブミラーが「正常 (Okay)」状態であることを確認します。(「正常 (Okay)」状態のサブミラーが存在しないミラーは、最初に修復する必要があります)。次に、サブミラー d21 を切り離して、ミラー d1 を 1 面ミラーにします。次に、/etc/vfstab ファイルを編集して、swap のエントリが、サブミラー d21 のスライスを参照するように指定します。

この例では、/etc/vfstab ファイルに swap ファイルシステム用の次のエントリが指定されています。

```
/dev/md/dsk/d4  /dev/md/rdsk/d4  /opt ufs  2  yes -

/dev/md/dsk/d1 - - swap - no -
```

次のようにエントリを変更します。

```
/dev/dsk/c0t3d0s1 - - swap - no -
```

システムのリブート後、`metaclear -r` コマンドは、このミラーともう 1 つのサブミラー `d11` を削除します。最後の `metaclear` コマンドは、サブミラー `d21` を削除します。

RAID-1 ボリューム上でのデータのバックアップ

Solaris ボリュームマネージャは「バックアップ製品」を意図しているわけではありません。しかし、Solaris ボリュームマネージャは次のいずれも引き起こすことなく、ミラー化されたデータをバックアップする手段を提供します。

- ミラーのマウント解除
- ミラー全体のオフライン化
- システムの停止
- ユーザーに対するデータアクセス拒否

Solaris ボリュームマネージャは、最初にサブミラーの 1 つをオフラインにすることによって、ミラー化されたデータをバックアップします。バックアップの間、ミラー化は一時的に使用できなくなります。バックアップの完了と同時に、サブミラーがオンラインに戻り、再同期が実行されます。

注 – UFS のスナップショット機能で、ファイルシステムをオフラインにすることなく、システムをバックアップすることもできます。サブミラーの切り離しや、あとでミラーを再同期させることによる性能低下を伴わずに、バックアップを実行できます。UFS のスナップショット機能によるバックアップを実行する前に、UFS ファイルシステム上の使用可能領域が十分かどうかを確認してください。詳細は、『Solaris のシステム管理 (デバイスとファイルシステム)』の第 26 章「UFS スナップショットの使用 (手順)」を参照してください。

▼ RAID-1 ボリュームのオンラインバックアップを実行するには

この手順は、ルート (/) 以外のすべてのファイルシステムに使用できます。このタイプのバックアップは、動作中のファイルシステムのある時点での内容を保存することに注意

してください。ファイルシステムへの書き込みをロックしたときのファイルシステムの使用状況によっては、バックアップしたファイルがディスク上の実際のファイルに対応しないことがあります。

この手順には次の制約があります。

- この手順を 2 面ミラーに対して使用すると、1 つのサブミラーをバックアップのためにオフラインにしたときに、データの冗長性が失われます。多面ミラーにはこの問題はありません。
- バックアップの完了後に、再接続されたサブミラーを再同期するときに、システムにある程度のオーバーヘッドが生じます。

この手順の概要は次のとおりです。

- ファイルシステムへの書き込みをロックします (UFS のみ)。ルート (/) はロックしないようにします。
- キャッシュのすべてのデータをディスクにフラッシュします。
- `metadetach` コマンドを使って、このミラーの 1 つのサブミラーを切り離します。
- ファイルシステムのロックを解除します。
- `fsck` コマンドを使って、切り離されたサブミラーにファイルシステムがあることを確認します。
- 切り離されたサブミラーのデータをバックアップします。
- `metattach` コマンドを使って、切り離されたサブミラーをミラーに再び接続します。

注 – このような手順を定常的に使用する場合は、これをスクリプトにしておくと実行が容易になります。

ヒント – より安全な方法としては、ミラーに 3 番目または 4 番目のサブミラーを接続し、これを再同期し、バックアップに使用します。これによって、データの冗長性が常に保たれます。

---

1 ミラーが「正常 **(Okay)**」状態であることを確認します。

ミラーが「保守 (Maintenance)」状態の場合は、まずそれを修復する必要があります。

```
# metastat mirror
```

2 キャッシュのデータと **UFS** ロギングデータをディスクにフラッシュし、このファイルシステムの書き込みをロックします。

```
# /usr/sbin/lockfs -w mount-point
```

書き込みをロックする必要があるファイルシステムは UFS ボリュームだけです。このボリュームがデータベース管理ソフトウェアなどのアプリケーション用に raw デバイスとして設定されている場合は、`lockfs` コマンドを実行する必要はありません。ただし、ベ

ンダー提供の適切なユーティリティーを実行してバッファーをフラッシュしたり、アクセスをロックする必要がある場合もあります。

注意 – ルート (/) ファイルシステムは書き込みロックしてはなりません。ルート (/) ファイルシステムを書き込みロックすると、システムが停止します。ルート (/) ファイルシステムをバックアップしている場合は、この手順をスキップします。

3 ミラーから 1 つのサブミラーを切り離します。

```
# metadetach mirror submirror
```

mirror ミラーのボリューム名です。

submirror 切り離すサブミラー (ボリューム) のボリューム名です。

読み取りは、他のサブミラーから引き続き行われます。最初の書き込みが行われた時点でミラーは同期していない状態になります。この不整合の状態は、手順 7 でサブミラーが再び接続された時点で修復されます。

4 ファイルシステムのロックを解除し、書き込みを再開します。

```
# /usr/sbin/lockfs -u mount-point
```

手順 2 で使用したベンダー提供のユーティリティーを使用して、必要なロック解除手順を実行しなければならない場合があります。

5 `fsck` コマンドを使って、切り離されたサブミラーにファイルシステムがあることを確認します。この手順によって、クリーンなバックアップが保証されます。

```
# fsck /dev/md/rdsk/name
```

6 オフラインにしたサブミラーのバックアップを取ります。

これには、`ufsdump` コマンド、または通常使用しているバックアップユーティリティーを使用します。`ufsdump` コマンドを使ってバックアップを実行する方法については、321 ページの「`ufsdump` コマンドによるマウント済みファイルシステムのバックアップ」を参照してください。

注 – 適切なバックアップを確実に行うには、raw ボリューム名 (`/dev/md/rdsk/d4` など) を使用します。raw ボリューム名を使用すると、2G バイトを超える記憶領域にアクセスできます。

7 サブミラーを接続します。

```
# metattach mirror submirror
```

サブミラーとミラーの再同期が自動的に開始されます。

例 11–23 RAID-1 ボリュームのオンラインバックアップを実行する

この例では、ミラー d1 を使用します。このミラーはサブミラー d2、d3、および d4 からなります。サブミラー d3 を切り離して、このサブミラーのバックアップを取ります。この間、サブミラー d2 と d4 はオンラインのままです。このミラーにあるファイルシステムは /home1 です。

```
# metastat d1
d1: Mirror
    Submirror 0: d2
      State: Okay
    Submirror 1: d3
      State: Okay
    Submirror 1: d4
      State: Okay
...

# /usr/sbin/lockfs -w /home1
# metadetach d1 d3
# /usr/sbin/lockfs -u /home1
# /usr/sbin/fsck /dev/md/rdsk/d3
(Perform backup using /dev/md/rdsk/d3)
# metattach d1 d3
```

◆ ◆ ◆ 第 12 章

12 ソフトパーティション(概要)

この章では、Solaris ボリュームマネージャのソフトパーティションについて説明します。関連する作業については、第 13 章を参照してください。

この章では、次の内容について説明します。

ソフトパーティションの概要

ディスクの記憶容量が増えるにしたがって、ディスクアレイが Solaris システムに提供する論理デバイスも大きくなります。ファイルシステムやパーティションサイズを管理しやすくするために、ディスクや論理ボリュームを 8 より多くのパーティションに分割しなければならない場合があります。Solaris ボリュームマネージャのソフトパーティションは、この要求に応えるための機能です。

Solaris ボリュームマネージャは、ディスクセットあたり 8192 の論理ボリュームをサポートできます。この数にはローカルまたは無指定のディスクセットが含まれます。Solaris ボリュームマネージャは、必要に応じて動的にボリュームを構成します。

ソフトパーティションでは、ディスクスライスや論理ボリュームを任意の数のパーティションに分割できます。区画 (ソフトパーティション) には、名前を付ける必要があります。これは、ストライプやミラーなど、ほかの記憶ボリュームの場合と同じです。名前の付いているソフトパーティションには、このソフトパーティションがすでに別のボリュームに含まれていない限り、ファイルシステムなどのアプリケーションからアクセスできます。ボリュームにすでに含まれているソフトパーティションには、直接アクセスしないでください。

ソフトパーティションは、ディスクスライス上に直接置くことも、ミラー、ストライプ、または RAID-5 ボリューム上に置くこともできます。ただし、ソフトパーティションは他のボリュームの上下両方に置くことはできません。たとえば、ソフトパーティション上にミラー化したストライプを構築し、さらにこの上にソフトパーティションを構築することはできません。

ファイルシステムなどのアプリケーションからは、ソフトパーティションは連続した 1 つの論理ボリュームに見えます。しかし、実際には、ソフトパーティションを構成するメディアの任意の場所にある、一連のエクステントから構成されています。システムに重大な障害が発生した場合でも、その障害から回復できるように、ソフトパーティションに加えディスク上のエクステントヘッダー (システム回復データ域ともいう) にも、ソフトパーティションの情報が記録されます。

ソフトパーティション構成の指針

- ソフトパーティションとして使用されているスライスを他の目的で使用することはできません。
- ディスクをパーティション分割し、それらのスライスにファイルシステムをすでに構築している場合は、ディスクフォーマットを変更または破棄しなければ、スライスを拡張することはできません。ソフトパーティションの動作はさまざまです。ソフトパーティションを構成するデバイスの容量が許す限り、ソフトパーティションを拡張できます。他のソフトパーティションにあるデータを移動させたり、破棄したりする必要はありません。
- ソフトパーティションのエクステントをディスク上の任意の場所に手動で配置することは、技術的には可能ですが、システムで自動的に配置するようにしてください。手動で配置されたエクステントの例については、234 ページの「Solaris ボリュームマネージャ構成の表示」 にある `metastat` コマンドの出力を参照してください。
- ソフトパーティションは任意のスライス上に構築できます。ただし、ディスクレベルでソフトパーティションを使用するときは、ディスク全体を占有するスライスを作成し、そのスライスにソフトパーティションを作成するのが、もっとも効率的です。
- ソフトパーティションの最大サイズは、ソフトパーティションを構築するスライスまたはボリュームのサイズによって制限されます。この制限があるので、ディスクスライス上にボリュームを作成してから、そのボリューム上でソフトパーティションを作成してください。この方法では、ボリュームにコンポーネントを追加してから、必要に応じてソフトパーティションを拡張することができます。
- 柔軟性と可用性を最大限に高めるためには、ディスクスライスに RAID-1 (ミラー) か RAID-5 ボリュームを作成してから、そのミラーまたは RAID-5 ボリューム上にソフトパーティションを作成するようにします。

シナリオ—ソフトパーティション

ソフトパーティション用のツールを使えば、大きな記憶領域をそれより小さい管理しやすい領域に分割できます。たとえば、ほかのシナリオ (107 ページの「シナリオ—RAID-1 ボリューム (ミラー)」 または 172 ページの「シナリオ—RAID-5 ボリューム」) では、多数の記憶領域を 1 つにまとめることによって、G バイトクラスの冗長記憶領域を確保できます。しかし、考えられるシナリオの多くでは、最初からそれほど多くの記憶領域が必要になることはありません。ソフトウェアパーティションでは、この記憶領域を管理しやすいパーティションに分割できます。そして、パーティションのそれぞれに完全な

ファイルシステムを与えることができます。たとえば、RAID-1 または RAID-5 ボリュームに 1000 のソフトパーティションを作成すれば、ユーザーは個別のファイルシステムにホームディレクトリを持つことができます。ユーザーがより多くの領域を必要とする場合は、単にそのソフトパーティションを拡張するだけですみます。

◆ ◆ ◆ 第 13 章

13 ソフトパーティション(作業)

この章では、Solaris ボリュームマネージャのソフトパーティションに関連する作業について説明します。ソフトパーティションに関連する概念については、第 12 章を参照してください。

ソフトパーティション(作業マップ)

次の表に、Solaris ボリュームマネージャのソフトパーティションを管理するために必要な作業を示します。

作業	説明	参照先
ソフトパーティションの作成	Solaris ボリュームマネージャの GUI か `metainit` コマンドを使用して、ソフトパーティションを作成します。	160 ページの「ソフトパーティションを作成するには」
ソフトパーティションの状態チェック	Solaris ボリュームマネージャの GUI か `metastat` コマンドを使用して、ソフトパーティションの状態をチェックします。	161 ページの「ソフトパーティションの状態をチェックするには」
ソフトパーティションの拡張	Solaris ボリュームマネージャの GUI か `metattach` コマンドを使用して、ソフトパーティションを拡張します。	162 ページの「ソフトパーティションを拡張するには」
ソフトパーティションの削除	Solaris ボリュームマネージャの GUI か `metaclear` コマンドを使用して、ソフトパーティションを削除します。	163 ページの「ソフトパーティションを削除するには」

ソフトパーティションの作成

▼ ソフトパーティションを作成するには

始める前に　156 ページの「ソフトパーティション構成の指針」を確認します。

◗ 次のどちらかの方法でソフトパーティションを作成します。

- Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開きます。「アクション (Action)」、「ボリュームの作成 (Create Volume)」の順に選択します。さらに、ウィザードの指示に従います。詳細は、オンラインヘルプを参照してください。
- 次の形式の `metainit` コマンドを使用して、ソフトパーティションを作成します。

  `# metainit [-s diskset]` *soft-partition* **`-p [-e]`** *component size*

  `-s`*diskset*　使用するディスクセットを指定します。`-s` を指定しなかった場合、ローカル (デフォルト) のディスクセットが使用されます。

  `-p`　ソフトパーティションを構成することを指定します。

  `-e`　ディスク全体を再フォーマットすることを指定します。ディスクフォーマットによって、ディスクの大部分を占めるスライス 0 が得られます。さらに、4M バイト以上の容量のスライス 7 も得られます。スライス 7 には、状態データベースの複製が格納されます。

  soft-partition　ソフトパーティション名を指定します。名前の形式は d*nnn* で、*nnn* は 0 から 8192 の範囲内の数字です。

  component　ソフトパーティションの作成に使用するディスク、スライス、または論理ボリュームを指定します。ソフトパーティションのヘッダーがコンポーネントの先頭部分に書き込まれるため、コンポーネントにあるデータはすべて破壊されます。

  size　ソフトパーティションのサイズを指定します。数字に次のいずれか 1 つを加えてサイズを指定します。

  - `M` または `m` (メガバイト)
  - `G` または `g` (ギガバイト)
  - `T` または `t` (テラバイト)
  - `B` または `b` (ブロック数 (セクター数))

  詳細については、次の例と `metainit(1M)` のマニュアルページを参照してください。

例 13–1 ソフトパーティションを作成する

次の例では、d20 という名前の 4G バイトのソフトパーティションを c1t3d0s2 に作成します。

```
# metainit d20 -p c1t3d0s2 4g
```

例 13–2 ディスク全体をソフトパーティションに使用する

次の例では、ソフトパーティションを作成し、ディスク c1t2d0 をフォーマットします。この処置によって、そのディスク上のあらゆるデータが破壊され、スライス 0 に新しいソフトパーティションが作成されます。

```
# metainit d7 -p -e c1t2d0 1G
```

ソフトパーティションの保守

ソフトパーティションの保守は、他の論理ボリュームの保守と同じです。

▼ ソフトパーティションの状態をチェックするには

始める前に 156 ページの「ソフトパーティション構成の指針」を確認します。

◗ 次のどちらかの方法でソフトパーティションの状態をチェックします。

- Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開きます。監視するソフトパーティションを選択します。次に、「アクション (Action)」、「プロパティ (Properties)」の順に選択します。画面の指示に従います。詳細は、オンラインヘルプを参照してください。
- 次の形式の metastat コマンドを使って既存の構成を表示します。

  # metastat *soft-partition*

  soft-partition　　チェックするパーティションの名前を指定します。

例 13–3 ソフトパーティションの状態をチェックする

次の例では、ソフトパーティション d1 の状態をチェックします。このソフトパーティションはエクステントが 2 つあり、RAID-1 ボリューム d100 上に作成されています。

```
# metastat d1
d1: soft partition
    component:  d100
```

```
    state: OKAY
    size:  42674285 blocks
             Extent              Start Block               Block Count
             0                         10234                  40674285
             1                      89377263                   2000000
d100: Mirror
    Submirror 0: d10
    State: OKAY
    Read option: roundrobin (default)
    Write option: parallel (default)
    Size: 426742857 blocks

d10: Submirror of d100
    State:  OKAY
    Hot spare pool: hsp002
    Size: 426742857 blocks
    Stripe 0: (interlace: 32 blocks)
        Device              Start Block  Dbase State        Hot Spare
        c3t3d0s0                  0             No    Okay
```

▼ ソフトパーティションを拡張するには

ソフトパーティション上にほかの論理ボリュームが構築されていない場合は、そのソフトパーティションに領域を追加できます。空き領域を見つけ、パーティションの拡張に使用します。既存のデータは移動されません。

注 – ソフトパーティションを使用して別のボリュームを作成している場合 (RAID-0 ボリュームのコンポーネントの場合など)、そのソフトパーティションは拡張できません。ソフトパーティションを収容するデバイスの領域を増やすことが目的であれば、通常、収容デバイスに他のボリュームを連結することによって目的を達成できます。詳細は、91 ページの「記憶容量の拡張」を参照してください。

始める前に 156 ページの「ソフトパーティション構成の指針」を確認します。

◗ 次のどちらかの方法でソフトパーティションを拡張します。

- Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開きます。拡張したいソフトパーティションを選択して、「アクション (Action)」、「プロパティ (Properties)」の順に選択します。画面の指示に従います。詳細は、オンラインヘルプを参照してください。
- 次の形式の `metattach` コマンドを使用して、ソフトパーティションに領域を追加します。

```
# metattach [-s diskset] soft-partition size
```

diskset ソフトパーティションが含まれているディスクセットの名前を指定します。

soft-partition 既存のソフトパーティションの名前を指定します。

size 追加する記憶領域の大きさを指定します。

例 13–4 ソフトパーティションを拡張する

ソフトパーティションに領域を追加する例を示します。さらに、ソフトパーティションがオンライン状態であり、マウントされているときに、**growfs** コマンドを使ってファイルシステムを拡張します。

```
# mount /dev/md/dsk/d20 /home2
# metattach d20 10g
# growfs -M /home2 /dev/md/rdsk/d20
```

growfs コマンドの詳細については、244 ページの「growfs コマンドによるファイルシステムの拡張」を参照してください。

▼ ソフトパーティションを削除するには

始める前に 156 ページの「ソフトパーティション構成の指針」を確認します。

◗ 次のどちらかの方法でソフトパーティションを削除します。

- Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開きます。削除するソフトパーティションを選択します。次に、「アクション (Action)」、「プロパティ (Properties)」の順に選択します。画面の指示に従います。詳細は、オンラインヘルプを参照してください。
- 次のいずれかの形式の metaclear コマンドを使ってソフトパーティションを削除します。

```
# metaclear [-s diskset] component
# metaclear [-s diskset] -r soft-partition
# metaclear [-s diskset] -p component
```

diskset ソフトパーティションが含まれているディスクセットの名前を指定します。

soft-partition 削除するソフトパーティションを指定します。

-r 論理ボリュームを繰り返し削除することを指定します。ただし、他のボリュームに使用されているボリュームは削除されません。

`-p`	指定したコンポーネント上のソフトパーティションを削除することを指定します。ただし、開かれているソフトパーティションは除きます。
component	すべてのソフトパーティションを削除するコンポーネントを指定します。

例 13–5 ソフトパーティションを削除する

この例では、`c1t4d2s0` にあるすべてのソフトパーティションを削除します。

```
# metaclear -p c1t4d2s0
```

◆ ◆ ◆ 第 14 章

RAID-5 ボリューム (概要)

この章では、Solaris ボリュームマネージャの RAID-5 ボリュームに関連する概念について説明します。関連する作業の実行手順については、第 15 章を参照してください。

この章の内容は、次のとおりです。

RAID-5 ボリュームの概要

RAID レベル 5 はストライプ方式に似ていますが、パリティーデータがすべてのコンポーネント (ディスクまたは論理ボリューム) に分散されている点が異なります。コンポーネントに障害が発生した場合には、障害が発生したコンポーネント上のデータを、他のコンポーネント上に分散されているデータとパリティー情報から再構築することができます。Solaris ボリュームマネージャでは、「*RAID-5* ボリューム」は RAID レベル 5 をサポートするボリュームを意味します。

RAID-5 ボリュームでは、ボリューム内の 1 つのコンポーネントに相当する記憶容量を使用して、冗長情報 (パリティー) を格納します。このパリティー情報には、残りの RAID-5 ボリュームのコンポーネントに格納されているユーザーデータに関する情報が含まれます。つまり、3 つのコンポーネントがあれば、1 つのコンポーネントに相当する領域がパリティー情報に使用されます。同じように、5 つのコンポーネントがある場合にも、1 つのコンポーネントに相当する領域がパリティー情報に使用されます。パリティー情報は、ボリューム内のすべてのコンポーネントに分散されます。ミラーと同様に、RAID-5 ボリュームではデータの可用性が向上しますが、ハードウェアのコストは最小限に抑えることができます。書き込み性能に対する影響は中程度です。ただし、RAID-5 ボリュームを ルート (/)、 `/usr`、および `swap` ファイルシステム、あるいはその他のすでに存在するファイルシステムに対して使用することはできません。

Solaris ボリュームマネージャは、既存のコンポーネントが置き換えられると、RAID-5 ボリュームの再同期を自動的に実行します。また、Solaris ボリュームマネージャは、システム障害やパニックが発生した場合、リブート時に、RAID-5 ボリュームを再同期します。

例 — RAID-5 ボリューム

図 14–1 に、4 つのディスク (コンポーネント) からなる RAID-5 ボリュームを示します。

最初の 3 つのデータセグメントは、コンポーネント A (飛び越し値 1)、コンポーネント B (飛び越し値 2)、およびコンポーネント C (飛び越し値 3) に書き込まれます。次のデータセグメントは、パリティーセグメントに書き込まれます。このパリティーセグメントは、コンポーネント D (P 1–3) に書き込まれます。このセグメントは、最初の 3 つのデータセグメントの排他的論理和からなります。次の 3 つのデータセグメントは、コンポーネント A (飛び越し値 4)、コンポーネント B (飛び越し値 5)、およびコンポーネント D (飛び越し値 6) に書き込まれます。次に、別のパリティーセグメントが、コンポーネント C (P 4–6) に書き込まれます。

データセグメントとパリティーセグメントをこのように書き込むことによって、データとパリティーの両方が、RAID-5 ボリュームを構成するすべてのディスクに分散されます。各ドライブは個別に読み取ることができます。パリティー情報により、いずれか 1 つのディスクが故障しても、データの安全性が保証されます。この例のディスクがそれぞれ 2G バイトだとすると、RAID-5 ボリュームの総容量は 6G バイトになります。ディスク 1 つ分の領域がパリティー用に割り当てられます。

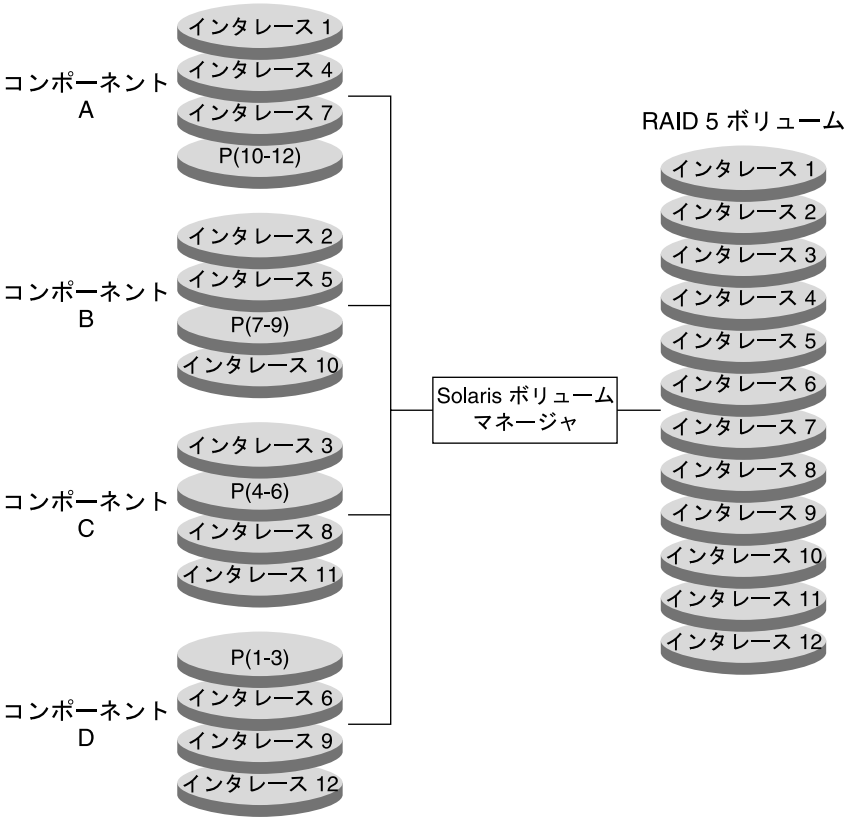

図 **14–1** RAID-5 ボリュームの例

例 — RAID-5 ボリュームの連結 (拡張)

次の図に、当初 4 つのディスク (コンポーネント) から構成されていた RAID-5 ボリュームの例を示します。5 つ目のディスクがボリュームに動的に連結され、RAID-5 ボリュームが拡張されています。

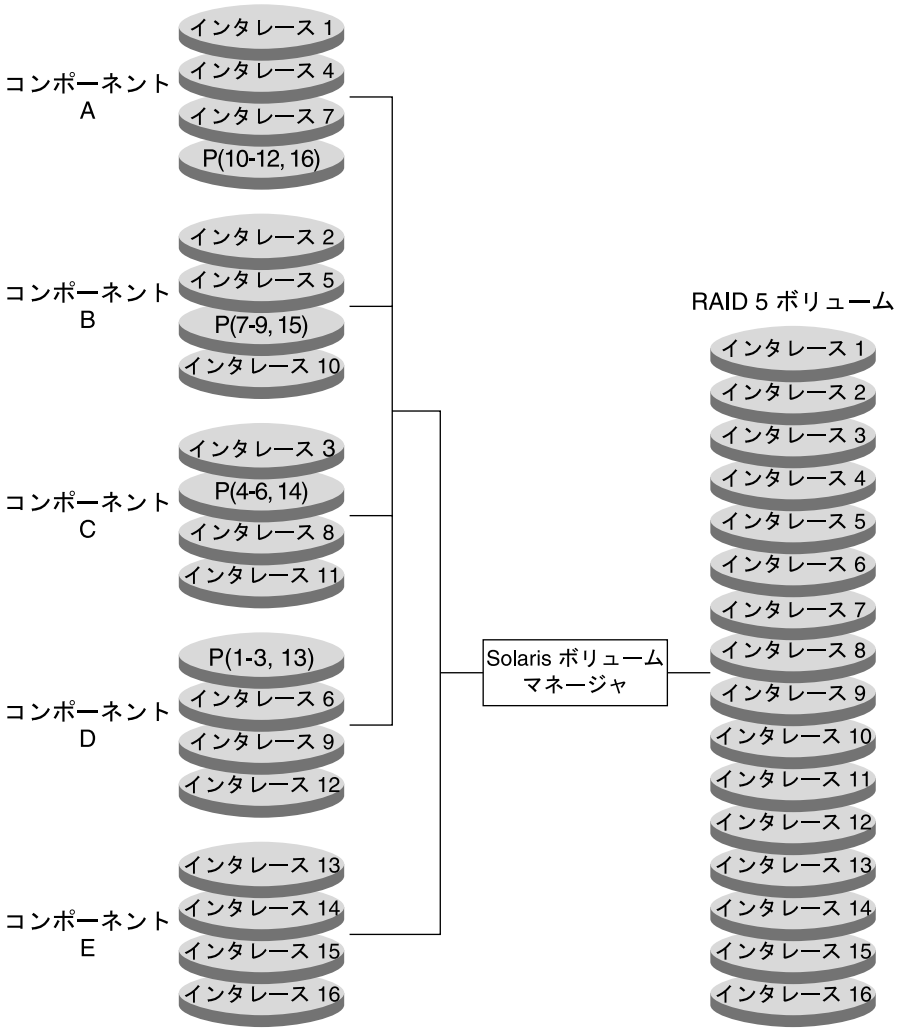

図 **14–2** RAID-5 ボリュームの拡張例

パリティー領域は、RAID 5-ボリュームの作成時に割り当てられます。パリティーには 1 つのコンポーネントに相当する領域が割り当てられますが、実際のパリティーブロックは、入出力を分散するためにすべてのオリジナルコンポーネントに分散されます。追加のコンポーネントを RAID-5 ボリュームに連結した場合、追加領域はすべてデータ用になります。新しいパリティーブロックは割り当てられません。ただし、連結されたコンポーネントのデータはパリティー計算の対象になるため、データは単一のデバイス障害から保護されます。

連結した RAID-5 ボリュームは長期間の使用には適しません。連結した RAID-5 ボリュームを使用するのは、大型の RAID-5 ボリュームを再構成できるようになるまでの間です。その後、データを大型ボリュームにコピーします。

注 – RAID-5 ボリュームに新しいコンポーネントを追加すると、Solaris ボリュームマネージャは、そのコンポーネントのすべてのデータブロックを「ゼロ」にします。この処理は、パリティー情報によって新しいデータを保護するために実行されます。つまり、データが新しい領域に書き込まれると、Solaris ボリュームマネージャはそのデータをパリティー計算の対象とします。

RAID-5 ボリュームを作成するための背景情報

RAID-5 ボリュームを使用するときは、169 ページの「RAID-5 ボリュームの要件」と 170 ページの「RAID-5 ボリュームの指針」を考慮してください。また、RAID-5 ボリュームの構成には、ストライプ化に関する指針の多くが適用されます。84 ページの「RAID-0 ボリュームの要件」を参照してください。

RAID-5 ボリュームの要件

- RAID-5 ボリュームは、3 つ以上のコンポーネントで構成されていなければなりません。ただし、RAID-5 ボリュームのコンポーネントの数が多くなればなるほど、いずれかのコンポーネントに障害が発生したときの読み取りおよび書き込み時間は長くなります。
- RAID-5 ボリュームをストライプ化、連結、ミラー化することはできません。
- 既存のファイルシステムが格納されているコンポーネントから RAID-5 ボリュームを作成しないようにします。作成すると、RAID-5 の初期化時にデータが消去されます。
- RAID-5 ボリュームを作成するときには飛び越し値を定義できます。飛び越し値を設定しないと、デフォルトとして 16K バイトが使用されます。この値は、ほとんどのアプリケーションにとって妥当な値です。
- RAID-5 ボリューム (ホットスペアなし) は、1 つのコンポーネントの障害にしか対応できません。
- RAID-5 ボリュームを作成するときには、別々のコントローラに分散されたコンポーネントを使用します。コントローラとそのケーブルは、ディスクより障害が発生しやすいからです。
- 同じサイズのコンポーネントを使用するようにします。サイズが異なるコンポーネントから RAID-5 ボリュームを作成すると、使用されないディスク容量が発生します。

RAID-5 ボリュームの指針

- パリティー計算は複雑なので、書き込み率が 20 パーセントを超えるボリュームは、通常、RAID-5 ボリュームにすべきではありません。書き込みが頻繁に発生するボリュームでデータの冗長性を確保したい場合は、ミラーの使用を検討してください。
- RAID-5 ボリューム内の各コンポーネントが異なるコントローラ上にあり、ボリュームへのアクセスが主に大容量の順次アクセスである場合は、飛び越し値を 32 K バイトに設定すると、性能が向上することがあります。
- RAID-5 ボリュームにコンポーネントを連結することによってボリュームを拡張できます。ただし、既存の RAID-5 ボリュームに新しいコンポーネントを連結すると、ボリュームの全体的な性能が低下します。これは、連結されたコンポーネント上のデータが順次処理されるためです。つまり、データは、すべてのコンポーネントにストライプ化されるわけではありません。ボリュームの元のコンポーネントのデータとパリティーは、すべてのコンポーネントについてストライプ化されますが、ストライプ化は連結されたコンポーネントでは失われます。ただし、コンポーネントの入出力時にパリティーが使用されるので、エラーが発生してもデータは回復可能です。新しいコンポーネントが連結された RAID-5 ボリュームも 1 つのコンポーネントの障害にのみ対応できます。

  連結されたコンポーネントは、それ自体のどの領域でもパリティーがストライプ化されないという点で元のコンポーネントとは違います。連結されたコンポーネントでは、全内容がデータに使用されます。

  コンポーネントを連結すると、大規模な書き込みや順次書き込みにおける性能上の利点は失われます。
- データブロックをゼロで初期化せずに、RAID-5 ボリュームを作成することもできます。そのためには、次のどちらかの方法を使用します。
  - `metainit` コマンドに `-k` オプションを指定します。`-k` オプションを指定すると、RAID-5 ボリュームが初期化なしで作成し直され、ディスクブロックが「正常 (Okay)」状態に設定されます。`-k` は、潜在的に危険なオプションです。ボリューム内のディスクブロックにエラーがあると、不正なデータの生成など、Solaris ボリュームマネージャが予期せぬ動作を起こすことがあります。
  - デバイスを初期化し、テープからデータを復元します。詳細は、`metainit(1M)` のマニュアルページを参照してください。

RAID-5 ボリュームの状態のチェック (概要)

RAID-5 ボリュームの状態をチェックするには、ボリュームの状態とスライスの状態を調べます。RAID-5 ボリュームのエラーに対処する場合は、スライスの状態からもっとも具体的な情報が得られます。RAID-5 ボリュームの状態からわかるのは、「正常 (Okay)」、「保守 (Maintenance)」といった総合的な状態情報だけです。

RAID-5 ボリュームの状態が「保守 (Maintenance)」になっている場合は、スライスの状態を確認します。スライスの状態は、スライスが「保守 (Maintenance)」状態なのか、それとも「最後にエラー (Last Erred)」状態なのかを具体的に示します。スライスが「保守

(Maintenance)」状態なのか「最後にエラー (Last Erred)」状態なのかによって、異なる回復処置を実行します。「保守 (Maintenance)」状態のスライスが 1 つだけ存在する場合は、データを失うことなくスライスを修理できます。「保守 (Maintenance)」状態のスライスと「最後にエラー (Last Erred)」状態のスライスが 1 つずつある場合は、おそらくデータは破壊されています。この場合には、「保守」状態のスライスを修理してから「最後にエラー」状態のスライスを修理する必要があります。

次の表に、RAID-5 ボリュームの状態を示します。

表 14–1 RAID-5 ボリュームの状態

状態	意味
初期化中 (Initializing)	スライスは、ディスクブロックをゼロで初期化しています。この処理は、データとパリティーを飛び越し方式でストライプ化する RAID-5 ボリュームの特性上、必要になるものです。 状態が「正常 (Okay)」になったら、初期化が完了しており、デバイスを開くことができます。この状態になるまで、アプリケーションはエラーメッセージを受け取ります。
正常 (Okay)	デバイスにはエラーはなく、使用可能な状態です。
保守 (Maintenance)	入出力エラーまたはオープンエラーが原因で、スライスが障害扱いになっています。このようなエラーが発生するのは、読み取り操作または書き込み操作時です。

次の表に、RAID-5 ボリュームのスライス状態と実行可能な処置を示します。

表 14–2 RAID-5 のスライスの状態

状態	意味	処置
初期化中 (Initializing)	スライスは、ディスクブロックをゼロで初期化しています。この処理は、データとパリティーを飛び越し方式でストライプ化する RAID-5 ボリュームの特性上、必要になるものです。	通常は必要ありません。この間に入出力エラーが発生すると、デバイスの状態は「保守」に変わります。初期化に失敗すると、このボリュームの状態は「初期化失敗」に、スライスの状態は「保守」になります。この場合には、ボリュームを削除してから作成し直す必要があります。
正常 (Okay)	デバイスにはエラーはなく、使用可能な状態です。	必要ない必要に応じて、スライスを追加および交換に使用できます。
再同期中 (Resyncing)	スライスの再同期処理が進行しています。エラーが発生したが、すでに訂正され、スライスが有効になっているか、あるいは、新しいスライスが追加された後です。	必要であれば、再同期が終了するまで RAID-5 ボリュームの状態を監視します。

表 14–2 RAID-5 のスライスの状態　(続き)

状態	意味	処置
保守 (Maintenance)	入出力エラーまたはオープンエラーが原因で、1 つのスライスが障害扱いになっています。このようなエラーが発生するのは、読み取り操作または書き込み操作時です。	障害が発生したスライスを有効にするか、交換します。179 ページの「RAID-5 ボリューム内のコンポーネントを有効にするには」または 180 ページの「RAID-5 ボリューム内のコンポーネントを置き換えるには」を参照してください。metastat コマンドを実行すると、metareplace コマンドを使用して行うべき処置を示す invoke 回復メッセージが表示されます。
保守/最後にエラー (Maintenance/Last Erred)	複数のスライスでエラーが発生しました。障害スライスの状態は「保守」または「最後にエラー」のどちらか一方です。この状態のとき、「保守」状態のスライスには入出力が行われません。しかし、「最後にエラー」とされたスライスには入出力が試行されます。その結果、入出力要求全体の状態が「最後にエラー」になります。	障害が発生したスライスを有効にするか、交換します。179 ページの「RAID-5 ボリューム内のコンポーネントを有効にするには」または 180 ページの「RAID-5 ボリューム内のコンポーネントを置き換えるには」を参照してください。metastat コマンドを実行すると、metareplace コマンドを使用して行うべき処置を示す invoke 回復メッセージが表示されます。このコマンドは、-f フラグを指定して実行する必要があります。この状態は、複数のスライスに障害が発生したため、不正なデータが生成された可能性があることを示しています。

RAID-5 ボリューム内のスライスの置き換えと有効化 (概要)

Solaris ボリュームマネージャには、ミラーおよび RAID-5 ボリューム内でコンポーネントの「置き換え」と「有効化」を行う機能があります。この機能についての問題点と要件は、ミラーおよび RAID-5 ボリュームに関するものと同じです。詳細については、246 ページの「RAID-1 および RAID-5 ボリューム内のコンポーネントの交換と有効化の概要」を参照してください。

シナリオ—RAID-5 ボリューム

RAID-5 ボリュームでは、RAID-1 ボリュームよりも少ないオーバーヘッドで記憶領域の冗長性を達成できます (RAID-1 ボリュームでデータの冗長性を得るには、合計記憶領域の 2 倍の領域が必要)。RAID-5 ボリュームを設定すると、同じ数のディスクコンポーネントでも、RAID-1 ボリュームで得られるより大容量の冗長記憶領域を確保できます。さらに、ホットスペア (第 16 章と特に186 ページの「ホットスペアの仕組み」を参照) を利用することによって、ほぼ同程度の安全性を確保できます。問題点は、コンポーネント障害が発生した場合に、書き込み時間が増大し、性能が相当低下することです。しかし、

このような問題点が目立つような状況はあまりありません。次の例では、第 5 章のサンプルシナリオを使用して、RAID-5 ボリュームで追加の記憶容量を提供する方法を示します。

RAID-0 と RAID-1 ボリュームに対応するシナリオでは、2 つのコントローラに分散された 6 つのディスク上の 6 つのスライス (`c1t1d0`、`c1t2d0`、`c1t3d0`、`c2t1d0`、`c2t2d0`、`c2t3d0`) によって、27G バイトの冗長記憶領域が得られます。RAID-5 構成で同じスライスを使用すると、45G バイトの記憶領域が使用できます。この構成の場合は、1 つのコンポーネントで障害が発生しても、データが失われたりアクセスが停止されたりすることはありません。さらに、RAID-5 ボリュームにホットスペアを追加すれば、複数のコンポーネントに障害が発生しても対応できます。この方式で最大の問題点は、コントローラ障害が発生すると、この RAID-5 ボリュームのデータが失われることです。RAID-1 ボリュームではこの問題は起こりません。107 ページの「シナリオ—RAID-1 ボリューム (ミラー)」を参照してください。

◆ ◆ ◆ 第 15 章

15 RAID-5 ボリューム (作業)

この章では、RAID-5 ボリュームに関連する Solaris ボリュームマネージャの作業について説明します。これらの作業に関連する概念については、第 14 章を参照してください。

RAID-5 ボリューム (作業マップ)

次の表に、Solaris ボリュームマネージャの RAID-5 ボリュームを管理するために必要な作業を示します。

作業	説明	参照先
RAID-5 ボリュームの作成	Solaris ボリュームマネージャの GUI か `metainit` コマンドを使って RAID-5 ボリュームを作成します。	176 ページの「RAID-5 ボリュームを作成するには」
RAID-5 ボリュームの状態チェック	Solaris ボリュームマネージャの GUI か `metastat` コマンドを使用して、RAID-5 ボリュームの状態をチェックします。	177 ページの「RAID-5 ボリュームの状態をチェックするには」
RAID-5 ボリュームの拡張	Solaris ボリュームマネージャの GUI か `metattach` コマンドを使って RAID-5 ボリュームを拡張します。	178 ページの「RAID-5 ボリュームを拡張するには」
RAID-5 ボリュームのスライスの有効化	Solaris ボリュームマネージャの GUI か `metareplace` コマンドを使って、RAID-5 ボリュームのスライスを有効にします。	179 ページの「RAID-5 ボリューム内のコンポーネントを有効にするには」
RAID-5 ボリュームのスライスの交換	Solaris ボリュームマネージャの GUI か `metareplace` コマンドを使用して、RAID-5 ボリュームのスライスを置き換えます。	180 ページの「RAID-5 ボリューム内のコンポーネントを置き換えるには」

RAID-5 ボリュームの作成

注意 – 32 ビットカーネルの Solaris ソフトウェアを実行する予定がある場合、または Solaris 9 4/03 リリースより前のバージョンの Solaris OS を使用する予定がある場合は、1T バイトを超えるボリュームを作成しないでください。Solaris ボリュームマネージャでサポートされる大容量ボリュームの詳細については、49 ページの「Solaris ボリュームマネージャにおけるマルチテラバイトサポートの概要」を参照してください。

▼ RAID-5 ボリュームを作成するには

始める前に 49 ページの「Solaris ボリュームマネージャコンポーネントを作成するための前提条件」と 169 ページの「RAID-5 ボリュームを作成するための背景情報」を確認します。

◗ 次のどちらかの方法で **RAID-5** ボリュームを作成します。

- Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開きます。「アクション (Action)」、「ボリュームの作成 (Create Volume)」の順に選択します。さらに、ウィザードの手順に従います。詳細は、オンラインヘルプを参照してください。
- 次の形式の `metainit` コマンドを使用します。

  # **metainit** *wolume-name* **-r** *component component component* **-i** *interlace-value*

volume-name	作成するボリュームの名前を指定します。
`-r`	RAID-5 ボリュームを作成することを指定します。
component	RAID-5 ボリュームに含めるスライスまたはソフトパーティションを指定します。3 つ以上のコンポーネントが必要です。
`-i`	飛び越し値を指定します。

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

例 15–1 3 つのスライスから成る RAID−5 ボリュームを作成する

この例では、`-r` オプションを使って 3 つのスライスから成る RAID-5 ボリューム `d45` を作成します。飛び越し値が指定されていないため、`d45` の飛び越し値にはデフォルトの 16K バイトを使用します。RAID-5 ボリュームが設定されて、ボリュームの初期化が開始されたことを示すメッセージが表示されます。

ボリュームの初期化が終わるまで RAID-5 ボリュームは使用できません。

```
# metainit d45 -r c2t3d0s2 c3t0d0s2 c4t0d0s2
d45: RAID is setup
```

参照　新たに作成した RAID-5 ボリュームにファイルシステムを作成する方法については、『Solaris のシステム管理 (デバイスとファイルシステム)』の第 18 章「UFS、TMPFS、LOFS ファイルシステムの作成 (手順)」を参照してください。データベースなど、アプリケーションによってはファイルシステムを使用しません。これらのアプリケーションでは、代わりに raw デバイスを使用します。アプリケーションは独自の方法で ボリュームを認識できなければなりません。

ホットスペア集合とRAID-5 ボリュームを関連付ける手順については、195 ページの「ホットスペア集合とボリュームを対応付けるには」を参照してください。

RAID-5 ボリュームの保守

▼ RAID-5 ボリュームの状態をチェックするには

RAID-5 ボリュームの状態をチェックする際、ボリュームの状態を完全に理解するには、RAID-5 とスライスの両方の状態をチェックする必要があります。さらに、ボリュームの状態が「正常 (Okay)」でない場合は、データが失われた可能性についても知る必要があります。詳細については、170 ページの「RAID-5 ボリュームの状態のチェック (概要)」を参照してください。

注 – RAID-5 ボリュームの初期化や再同期化を中断することはできません。

◗ 次のどちらかの方法で **RAID-5** ボリュームの状態をチェックします。

- Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開き、ボリュームの状態を表示します。ボリュームを選択します。次に、「アクション (Action)」、「プロパティ (Properties)」の順に選択して、詳細情報を確認します。詳細は、オンラインヘルプを参照してください。
- `metastat` コマンドを使って RAID-5 ボリュームの状態を表示します。

  `# metastat [-s` *diskset***`] [`***volume***`]**

  `-s` *diskset*　コマンドの実行対象となるディスクセットの名前を指定します。

  volume　表示するボリュームを指定します。

  `metastat` コマンドは、RAID-5 ボリュームのスライスごとに次の情報を表示します。

  Device　ストライプ内のスライスのデバイス名

  Start Block　スライスの開始ブロック

  Dbase　スライスに状態データベースの複製が含まれているかどうか

  State　スライスの状態。

Hot Spare　　スライスが障害スライスのホットスペアとして使用されているかどうか

例 15–2　RAID 5-ボリュームの状態を表示する

次の例は、RAID-5 ボリュームに関する metastat コマンドの出力です。

```
# metastat d10
d10: RAID
    State: Okay
    Interlace: 32 blocks
    Size: 10080 blocks
Original device:
    Size: 10496 blocks
        Device              Start Block  Dbase State        Hot Spare
        c0t0d0s1                 330     No    Okay
        c1t2d0s1                 330     No    Okay
        c2t3d0s1                 330     No    Okay
```

metastat コマンドの出力には、ボリュームが RAID-5 ボリュームであることが示されています。この情報は、ボリューム名の後ろの「RAID」で識別できます。RAID-5 ボリューム内のスライスごとに、次のような出力が表示されます。

- ストライプ内のスライスの名前。
- スライスが始まるブロック。
- どのスライスにも状態データベースの複製が含まれていないという表示。
- スライスの状態。この例では、すべてのスライスが「正常 (Okay)」状態です。
- スライスが障害スライスのホットスペアかどうか。

▼ RAID-5 ボリュームを拡張するには

一般に、コンポーネントの追加は、領域が不足している RAID-5 ボリュームに対する一時的な解決策です。性能上の理由から、「純粋な」RAID-5 ボリュームの使用をお勧めします。記憶領域を増やすために既存の RAID-5 ボリュームを拡張する必要がある場合は、この手順を使用してください。

注意 – 32 ビットカーネルの Solaris ソフトウェアを実行する予定がある場合、または Solaris 9 4/03 リリースより前のバージョンの Solaris OS を使用する予定がある場合は、1T バイトを超えるボリュームを作成しないでください。Solaris ボリュームマネージャのマルチテラバイトボリュームサポートの詳細については、49 ページの「Solaris ボリュームマネージャにおけるマルチテラバイトサポートの概要」を参照してください。

始める前に　169 ページの「RAID-5 ボリュームを作成するための背景情報」を確認します。

1. すべてのデータの最新のバックアップを取っているか確認します。また、この操作にはスーパーユーザー権限が必要です。

2. 次のどちらかの方法で **RAID-5** ボリュームに他のコンポーネントを追加します。

   - Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開き、次に RAID-5 ボリュームを開きます。「コンポーネント (Components)」区画を開きます。「コンポーネントの接続 (Attach Component)」を選択します。画面の指示に従います。詳細は、オンラインヘルプを参照してください。
   - 次の形式の `metattach` コマンドを実行します。

     `# metattach` *volume-name name-of-component-to-add*

volume-name	拡張する RAID-5 ボリュームの名前を指定します。
name-of-component-to-add	RAID-5 ボリュームを接続するコンポーネントの名前を指定します。

     詳細については、`metattach(1M)` のマニュアルページを参照してください。

例 15–3 RAID-5 ボリュームにコンポーネントを追加する

次の例では、スライス `c2t1d0s2` を既存の RAID-5 ボリューム `d2` に追加しています。

```
# metattach d2 c2t1d0s2
d2: column is attached
```

参照 UFS ファイルシステムの場合は、RAID-5 ボリュームに対して `growfs` コマンドを実行します。44 ページの「`growfs` コマンドによるボリュームとディスク領域の拡張」を参照してください。

データベースなど、アプリケーションによってはファイルシステムを使用しません。これらのアプリケーションでは、代わりに raw デバイスを使用します。この場合、アプリケーションは独自の方法で追加領域を拡張できなければなりません。

▼ RAID-5 ボリューム内のコンポーネントを有効にするには

ディスクドライブに障害が発生した場合は、そのドライブの代わりにシステム上のほかのディスク (およびそのスライス) を使用することができます (180 ページの「RAID-5 ボリューム内のコンポーネントを置き換えるには」を参照)。または、ディスクを修復し、ラベルを付け、`-e` オプションを指定して `metareplace` コマンドを実行することによって、ディスクをもう一度有効化することもできます。

1. すべてのデータの最新のバックアップを取っているか確認します。また、この操作にはスーパーユーザー権限が必要です。

2. 次のどちらかの方法で、**RAID−5** ボリューム内の、障害が発生したコンポーネントを有効にします。

   - Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開き、次に RAID-5 ボリュームを開きます。「コンポーネント (Components)」区画を開きます。障害の発生したコンポーネントを選択します。「コンポーネントの有効化 (Enable Component)」をクリックします。画面の指示に従います。詳細は、オンラインヘルプを参照してください。
   - 次の形式の `metareplace` コマンドを実行します。

     # metareplace -e *volume-name component-name*

`-e`	障害の発生したコンポーネントを使用可能な状態に戻し、そのコンポーネントに対して再同期を実行することを指定します。
volume-name	障害コンポーネントが含まれているボリュームの名前を指定します。
component-name	障害の発生したコンポーネントの名前を指定します。

     `metareplace` コマンドは、新しいコンポーネントと RAID-5 ボリュームの残りのコンポーネントとの再同期を自動的に開始します。

例 15–4 RAID-5 ボリューム内のコンポーネントを有効にする

次の例では、RAID-5 ボリューム `d20` 内のスライス `c2t0d0s2` にソフトエラーがあります。`metareplace` コマンドに `-e` オプションを指定して、このスライスを有効にします。

```
# metareplace -e d20 c2t0d0s2
```

▼ RAID-5 ボリューム内のコンポーネントを置き換えるには

この作業では、1 つのスライスに障害が発生した RAID-5 ボリュームでそのスライスを交換します。

注意 – 複数のスライスでエラーが発生している状態で、障害のあるスライスを 1 つだけ交換すると、不正なデータが生成されることがあります。この場合、データの整合性に疑問が生じます。

metareplace コマンドを障害が発生していないデバイス上で実行すれば、ディスクスライスなどのコンポーネントを交換できます。この方法は、RAID-5 ボリュームの性能を調整するときなどに便利です。

1 すべてのデータの最新のバックアップを取っているか確認します。また、この操作にはスーパーユーザー権限が必要です。

2 次のどちらかの方法で **RAID-5** ボリュームのどのスライスを交換するか判断します。

- Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開きます。さらに RAID-5 ボリュームを開きます。「コンポーネント (Components)」区画を開きます。個々のコンポーネントの状態を調べます。詳細は、オンラインヘルプを参照してください。
- metastat コマンドを使用します。

  ```
  # metastat volume
  ```

  volume　RAID-5 ボリュームの名前を指定します。

「保守が必要 (Needs Maintenance)」というキーワードを探して、RAID-5 ボリュームの状態を調べます。「保守 (Maintenance)」というキーワードを探して、障害のあるスライスを特定します。

3 次のどちらかの方法で、障害のあるスライスを別のスライスで置き換えます。

- Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開きます。さらに RAID-5 ボリュームを開きます。「コンポーネント (Components)」区画を開きます。障害の発生したコンポーネントを選択します。「コンポーネントを置換 (Replace Component)」をクリックし、画面の指示に従います。詳細は、オンラインヘルプを参照してください。
- 次の形式の metareplace コマンドを実行します。

  ```
  # metareplace volume-name failed-component new-component
  ```

  - *volume-name* は、障害コンポーネントが含まれている RAID-5 ボリュームの名前です。
  - *failed-component* は、置き換えられるコンポーネントの名前です。
  - *new-component* は、障害のあるコンポーネントの代わりにボリュームに追加するコンポーネントの名前です。

  詳細については、metareplace(1M) のマニュアルページを参照してください。

4 手順 2 のどちらかの方法で、新しいスライスの状態を確認します。

スライスは、「再同期中 (Resyncing)」状態か「正常 (Okay)」状態になるはずです。

例 15–5 RAID-5 コンポーネントを置き換える

```
# metastat d1
d1: RAID
State: Needs Maintenance
    Invoke: metareplace d1 c0t14d0s6 <new device>
    Interlace: 32 blocks
    Size: 8087040 blocks
Original device:
    Size: 8087520 blocks
    Device              Start Block  Dbase State        Hot Spare
    c0t9d0s6                 330     No    Okay
    c0t13d0s6                330     No    Okay
    c0t10d0s6                330     No    Okay
    c0t11d0s6                330     No    Okay
    c0t12d0s6                330     No    Okay
    c0t14d0s6                330     No    Maintenance

# metareplace d1 c0t14d0s6 c0t4d0s6
d1: device c0t14d0s6 is replaced with c0t4d0s6
# metastat d1
d1: RAID
    State: Resyncing
    Resync in progress: 98% done
    Interlace: 32 blocks
    Size: 8087040 blocks
Original device:
    Size: 8087520 blocks
    Device              Start Block  Dbase State        Hot Spare
    c0t9d0s6                 330     No    Okay
    c0t13d0s6                330     No    Okay
    c0t10d0s6                330     No    Okay
    c0t11d0s6                330     No    Okay
    c0t12d0s6                330     No    Okay
    c0t4d0s6                 330     No    Resyncing
```

この例では、`metastat` コマンドで RAID-5 ボリューム `d1` 内の障害の発生したスライスを表示します。交換用として使用可能なスライスを特定してから `metareplace` コマンドを実行します。このコマンドには、まず障害が発生したスライスを指定し、次に交換用のスライスを指定します。

使用可能なスライスがない場合は、`metareplace` コマンドに `-e` オプションを付けて実行し、障害のあるデバイスを再同期することによって、予想されるソフトエラーからの回復を試みます。この手順の詳細については、179 ページの「RAID-5 ボリューム内のコンポーネントを有効にするには」を参照してください。複数のエラーがある場合は、まず「保守(Maintenance)」状態のスライスを交換するか有効にする必要があります。そのあとで、「最後にエラー (Last Erred)」状態のスライスを修理します。`metareplace` コマン

ドの実行後、 metastat コマンドを使用すると、再同期の進捗状況を監視できます。交換中は、ボリュームの状態と新しいスライスは「再同期中 (Resyncing)」状態となります。この状態の間は、ボリュームを使い続けることが できます。

◆ ◆ ◆ 第 16 章

16 ホットスペア集合 (概要)

この章では、Solaris ボリュームマネージャでホットスペア集合をどのように使用するかについて説明します。関連する作業の実行手順については、第 17 章を参照してください。

この章では、次の内容について説明します。

ホットスペア集合とホットスペアの概要

ホットスペア集合 とは、RAID-1 (ミラー) や RAID-5 ボリュームのデータ可用度を高めるために Solaris ボリュームマネージャが使用するスライスの集合 (ホットスペア) です。サブミラーまたは RAID-5 ボリュームのスライスに障害が発生すると、Solaris ボリュームマネージャはそのスライスをホットスペアで自動的に置き換えます。

注 – ホットスペアは、RAID-0 ボリュームや 1 面ミラーには適用されません。交換を自動的に行うためには、冗長データが必要です。

アイドル状態のホットスペアにデータや状態データベースの複製を格納することはできません。ホットスペアは、対応付けられているボリューム内のスライスに障害が発生した場合には、ただちに使用できる状態でなければなりません。したがって、ホットスペアを使用するためには、システムが通常の動作を行うのに必要なディスクに加え、予備のディスクを用意しておく必要があります。

Solaris ボリュームマネージャでは、ホットスペア集合内のホットスペアを動的に追 加、削除、交換、および有効化することができます。ホットスペアやホットスペア集合の管理には、Solaris 管理コンソールまたはコマンド行ユーティリティーを使用します。これらの作業の詳細については、第 17 章を参照してください。

ホットスペア

ホットスペアは、動作可能、使用可能であり、なおかつ使用中ではないスライス (ボリュームではない) です。ホットスペアは、サブミラーや RAID-5 ボリュームのスライスに障害が発生したときに、ただちに交換できる予備のスライスとして用意されています。

ホットスペアによって、ハードウェア障害から保護されます。RAID-1 ボリュームと RAID-5 ボリュームのスライスは、障害が発生するとホットスペアで自動的に置き換えられます。ホットスペアは再同期化され、ボリューム内で使用できるようになります。ホットスペアは、サブミラーや RAID-5 ボリュームのスライスが修復されるか交換されるまで一時的に使用されるスライスです。

ホットスペアはホットスペア集合内に作成します。同じホットスペアを複数のホットスペア集合に登録することができます。たとえば、サブミラーとホットスペアが 2 つずつあるとします。この場合、これらのホットスペアを 2 つのホットスペア集合に登録し、それぞれの集合の中でホットスペアの優先順位を変えておくこともできます。この方式では、最初に使用するホットスペアを指定できます。この方式によって使用可能なホットスペアが増えるので、可用性も向上します。

サブミラーや RAID-5 ボリュームの不良スライスの代わりに使用できるホットスペアのサイズは、そのスライスと同じかそれ以上でなければなりません。たとえば、サブミラーの容量が 1G バイトであれば、サブミラー用のホットスペアの容量は 1G バイト以上でなければなりません。

ホットスペア集合

ホットスペア集合は、ホットスペアの順序付きリスト (集合) です。

ホットスペアを複数のホットスペア集合に登録することによって、最小数のスライスで最大限の柔軟性と安全性を達成できます。ホットスペアとして使用する 1 つのスライスを複数のホットスペア集合に割り当て、個々のホットスペア集合に異なるスライスや特性を持たせることができます。個々のホットスペア集合は、任意の数のサブミラーボリュームや RAID-5 ボリュームに割り当てることができます。

注 – 同じホットスペア集合を複数のサブミラーボリュームや RAID-5 ボリュームに割り当てることはできますが、個々のサブミラーボリュームや RAID-5 ボリュームは 1 つのホットスペア集合しか割り当てることはできません。

ホットスペアの仕組み

入出力エラーが発生した場合、Solaris ボリュームマネージャは、ホットスペア集合にホットスペアが追加された順序に基づいて、ホットスペア集合内でホットスペアを探します。Solaris ボリュームマネージャは、障害のあるスライスとサイズが同じかそれ以上

のサイズの最初のホットスペアをホットスペア集合から探します。Solaris ボリュームマネージャが最初に見つけた、適切なサイズのホットスペアが不良スライスの置き換えに使用されます。Solaris ボリュームマネージャはそのホットスペアを「使用中 (In-Use)」の状態にして、必要であれば、データの同期化を自動的に開始します。ホットスペア集合内でのホットスペアの順序は、置き換えの後も変わりません。

ホットスペアの同期化には、サブミラーの場合は有効なサブミラーのデータが使用され、RAID-5 ボリュームの場合は、同じボリュームの他のスライスが使用されます。十分な大きさのホットスペアがホットスペア集合にないと、障害が発生したサブミラーや RAID-5 ボリュームはエラー状態となり、ホットスペアは使用されません。この場合、サブミラーではデータを完全に複製することができなくなり、RAID-5 ボリュームではデータの冗長性が失われます。

ヒント – ホットスペア集合にホットスペアを追加するときは、小さいものから順に追加してください。これによって、小さいスライスの置き換えのために「大きい」スライスがむだに使用されることを防げます。

スライスで入出力エラーが発生すると、そのスライスは「障害 (Broken)」状態になります。この状態から回復するためには、まず、不良スライスを修理または交換します。次に、Solaris 管理コンソール内の「拡張ストレージ」を使用して、スライスを「使用可能 (Available)」状態に戻します。または、`metahs -e` コマンドを使用します。

サブミラーまたは RAID-5 ボリュームは、障害の発生したスライスが有効になるか、または交換されるまで、そのスライスの代わりにホットスペアを使用します。使用が終わると、そのホットスペアはホットスペア集合内で「使用可能 (Available)」になります。このホットスペアは再び使用可能になります。

ホットスペア集合の状態

次の表に、ホットスペア集合の状態と実行可能な処置を示します。

表 16–1 ホットスペア集合の状態 (コマンド行)

状態	意味	処置
使用可能 (Available)	ホットスペア集合内のホットスペアが動作していて、データを受け付けることのできる状態です。ホットスペアは現在のところ、書き込みも読み取りも行っていません。	必要ない
使用中 (In-Use)	このホットスペア集合には、冗長ボリュームの不良スライスの代わりに使用されているホットスペアが含まれています。	ホットスペアの使用状況を調べます。次に、このホットスペア集合によって置き換えられたボリュームのスライスを修理します。

表 16–1 ホットスペア集合の状態 (コマンド行)　(続き)

状態	意味	処置
障害 (Broken)	ホットスペアまたはホットスペア集合に問題があります。ただし、ただちにデータが失われる危険はありません。この状態は、すべてのホットスペアが使用中の場合にも表示されます。	ホットスペアの使用状況または故障の原因を調べます。必要であれば、別のホットスペアをホットスペア集合に追加することができます。

例 — ホットスペア集合

図 16–1に、ミラー d1 のサブミラー d11 と d12 に対応付けられているホットスペア集合を示します。どちらかのサブミラーのスライスに障害が発生すると、そのスライスは自動的にホットスペアで置き換えられます。ホットスペア集合自体はミラーではなく、個々のサブミラーボリュームに対応付けられています。必要であれば、このホットスペア集合を他のサブミラーや RAID-5 ボリュームに対応付けることもできます。

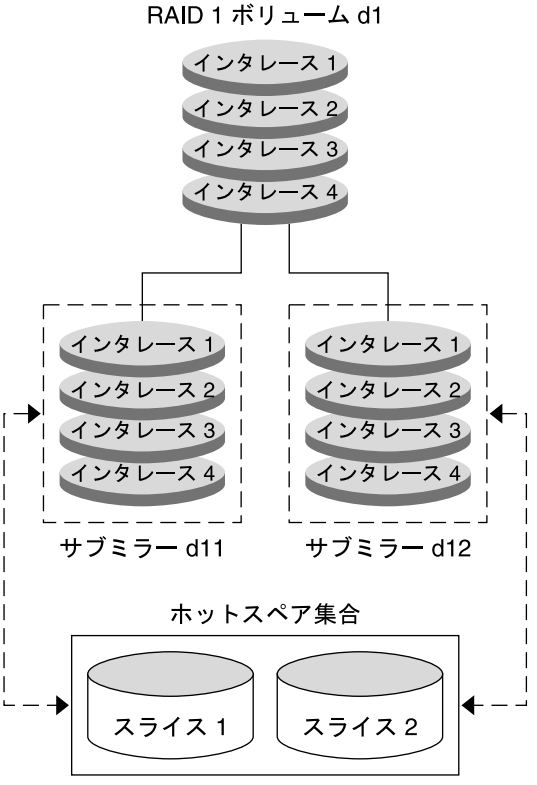

図 16–1 ホットスペア集合の例

シナリオ — ホットスペア

ホットスペアは、冗長ボリューム (RAID-1 および RAID-5) の保護機能を強化し、データの安全性をさらに向上します。ホットスペアと、RAID-0 サブミラーや RAID-5 ボリュームを構成するスライスを対応付けることによって、障害が発生したスライスは、ホットスペア集合内の有効なスライスで自動的に置き換えられます。使用するために置き換えられたスライスは更新され、本来あるべき情報が与えられます。したがって、置き換えられたスライスは引き続き、元のスライスと同じように動作します。不良スライスは、いつでも交換できます。

◆ ◆ ◆ 第 17 章

17 ホットスペア集合 (作業)

この章では、Solaris ボリュームマネージャのホットスペアとホットスペア集合の使用方法について説明します。関連する概念については、第 16 章を参照してください。

ホットスペア集合 (作業マップ)

次の表に、Solaris ボリュームマネージャのホットスペアを管理するために必要な作業を示します。

作業	説明	参照先
ホットスペア集合を作成する	Solaris ボリュームマネージャの GUI か `metainit` コマンドを使ってホットスペア集合を作成します。	192 ページの「ホットスペア集合を作成するには」
ホットスペア集合にスライスを追加する	Solaris ボリュームマネージャの GUI か `metahs` コマンドを使って、ホットスペア集合にスライスを追加します。	193 ページの「ホットスペア集合にホットスペアを追加するには」
ホットスペア集合とボリュームを対応付ける	Solaris ボリュームマネージャの GUI か `metaparam` コマンドを使って、ホットスペア集合とボリュームを対応付けます。	195 ページの「ホットスペア集合とボリュームを対応付けるには」
ホットスペア集合とボリュームの対応付けを変更する	Solaris ボリュームマネージャの GUI か `metaparam` コマンドを使ってホットスペア集合とボリュームの対応付けを変更します。	196 ページの「ホットスペア集合の対応付けを変更するには」

作業	説明	参照先
ホットスペアとホットスペア集合の状態をチェックする	Solaris ボリュームマネージャの GUI か、`metastat` コマンドまたは `metahs -i` コマンドを使って、ホットスペアやホットスペア集合の状態をチェックします。	198 ページの「ホットスペア集合とホットスペアの状態を確認するには」
ホットスペア集合内のホットスペアを交換する	Solaris ボリュームマネージャの GUI か `metahs` コマンドを使ってホットスペア集合内のホットスペアを交換します。	198 ページの「ホットスペア集合内のホットスペアを置き換えるには」
ホットスペア集合からホットスペアを削除する	Solaris ボリュームマネージャの GUI か `metahs` コマンドを使ってホットスペア集合からホットスペアを削除します。	200 ページの「ホットスペア集合からホットスペアを削除するには」
ホットスペアを有効にする	Solaris ボリュームマネージャの GUI か `metahs` コマンドを使ってホットスペア集合内のホットスペアを有効にします。	201 ページの「ホットスペアを有効にするには」

ホットスペア集合の作成

▼ ホットスペア集合を作成するには

注意 – 32 ビットカーネルの Solaris ソフトウェアを実行する予定がある場合、または Solaris 9 4/03 リリースより前のバージョンの Solaris OS を使用する予定がある場合は、1T バイトを超えるボリュームまたはホットスペアを作成しないでください。Solaris ボリュームマネージャのマルチテラバイトボリュームサポートの詳細については、49 ページの「Solaris ボリュームマネージャにおけるマルチテラバイトサポートの概要」を参照してください。

注意 – 作成するホットスペアの大きさが十分であるかどうかを示すメッセージは出力されません。ホットスペアのサイズが、対応付けられているボリュームのサイズ以上でないと、ホットスペアは使用されません。

始める前に 49 ページの「Solaris ボリュームマネージャコンポーネントを作成するための前提条件」を確認します。

1. スーパーユーザーになります。
2. 次のどちらかの方法でホットスペア集合を作成します。

- Solaris 管理コンソール内の「拡張ストレージ」から「ホットスペアプール (Hot Spare Pools)」ノードを開きます。次に、「アクション (Action)」、「ホットスペアプールの作成 (Create Hot Spare Pool)」の順に選択します。詳細は、オンラインヘルプを参照してください。
- 次の形式の `metainit` コマンドを使用します。

  `#` `metainit` *hot-spare-pool-name ctds-for-slice*

  hot-spare-pool-name ホットスペア集合の名前を指定します。

  ctds-for-slice ホットスペア集合に追加するスライスを指定します。ホットスペア集合に追加するスライスごとにこのオプションを繰り返します。

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

注 – ホットスペア集合の作成には、`metahs` コマンドを使用することもできます。

例 17–1 ホットスペア集合の作成

```
# metainit hsp001 c2t2d0s2 c3t2d0s2
hsp001: Hotspare pool is setup
```

この例では、ホットスペア集合 `hsp001` にホットスペアとして 2 つのディスクを割り当てます。ホットスペア集合が設定されたことを示すメッセージが出力されます。

参照 ホットスペア集合にホットスペアを追加する方法については、193 ページの「ホットスペア集合にホットスペアを追加するには」を参照してください。ホットスペア集合を作成したら、それをサブミラーまたは RAID-5 ボリュームと対応付ける必要があります。195 ページの「ホットスペア集合とボリュームを対応付けるには」を参照してください。

▼ ホットスペア集合にホットスペアを追加するには

始める前に 49 ページの「Solaris ボリュームマネージャコンポーネントを作成するための前提条件」を確認します。

1. スーパーユーザーになります。
2. 次のどちらかの方法で既存のホットスペア集合にホットスペアを追加します。
   - Solaris 管理コンソール内の「拡張ストレージ」から「ホットスペアプール (Hot Spare Pools)」ノードを開きます。変更するホットスペア集合を選択します。「アクション (Action)」、「プロパティ (Properties)」の順に選択します。さらに「ホットスペア (Hot Spares)」パネルを選択します。詳細は、オンラインヘルプを参照してください。

- 次のどちらかの形式の metahs コマンドを使用します。

  # metahs -a *hot-spare-pool-name slice-to-add*

  # metahs -a -all *hot-spare-pool-name slice-to-add*

-a *hot-spare-pool-name*	特定のホットスペア集合にスライスを追加することを指定します。
-a all	すべてのホットスペア集合にスライスを追加することを指定します。
slice-to-add	ホットスペア集合に追加するスライスを指定します。

  詳細は、metahs(1M) のマニュアルページを参照してください。

注 – 同じホットスペアを複数のホットスペア集合に追加することができます。ホットスペアをホットスペア集合に追加すると、ホットスペア集合のスライスリストの最後にそのホットスペアが追加されます。

例 17–2 ホットスペアスライスを 1 つのホットスペア集合に追加する

この例では、-a オプションを使ってスライス /dev/dsk/c3t0d0s2 をホットスペア集合 hsp001 に追加します。スライスがホットスペア集合に追加されたことを示すメッセージが表示されます。

```
# metahs -a hsp001 /dev/dsk/c3t0d0s2
hsp001: Hotspare is added
```

例 17–3 ホットスペアスライスをすべてのホットスペア集合に追加する

この例では、- a オプションを all と組み合わせて使用し、システム上のすべてのホットスペア集合にスライス /dev/dsk/c3t0d0s2 を追加します。スライスがすべてのホットスペア集合に追加されたことを示すメッセージが表示されます。

```
# metahs -a -all /dev/dsk/c3t0d0s2
hsp001: Hotspare is added
hsp002: Hotspare is added
hsp003: Hotspare is added
```

ホットスペア集合とボリュームの対応付け

▼ ホットスペア集合とボリュームを対応付けるには

始める前に 49 ページの「Solaris ボリュームマネージャコンポーネントを作成するための前提条件」を確認します。

1. スーパーユーザーになります。
2. 次のどちらかの方法でホットスペア集合と **RAID-5** ボリュームまたはサブミラーを対応付けます。
   - Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」を開き、ボリュームを選択します。「アクション (Action)」、「プロパティ (Properties)」の順に選択します。さらに「ホットスペアプール (Hot Spare Pool)」パネルを選択します。最後に「HSP の接続 (Attach HSP)」を選択します。詳細は、オンラインヘルプを参照してください。
   - 次の形式の `metaparam` コマンドを使用します。

     # metaparam -h *hot-spare-pool component*

`-h`	特定のホットスペア集合を変更することを指定します。
hot-spare-pool	ホットスペア集合の名前を指定します。
component	ホットスペア集合に対応付けるサブミラーまたは RAID-5 ボリュームの名前を指定します。

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

例 17–4 ホットスペア集合とサブミラーを対応付ける

次の例では、`-h` オプションを使ってホットスペア集合 `hsp100` をミラー `d0` の 2 つのサブミラー `d10` と `d11` に対応付けます。`metastat` コマンドを実行して、ホットスペア集合と 2 つのサブミラーが対応付けられていることを確認します。

```
# metaparam -h hsp100 d10
# metaparam -h hsp100 d11
# metastat d0
d0: Mirror
    Submirror 0: d10
      State: Okay
    Submirror 1: d11
      State: Okay
...
```

```
d10: Submirror of d0
    State: Okay
    Hot spare pool: hsp100
...


d11: Submirror of d0
    State: Okay
    Hot spare pool: hsp100
...
```

例 17–5 ホットスペア集合と RAID-5 ボリュームを対応付ける

次の例では、-h オプションを使ってホットスペア hsp001 と RAID-5 ボリューム d10 を対応付けます。metastat コマンドを実行して、ホットスペア集合と RAID-5 ボリュームが対応付けられていることを確認します。

```
# metaparam -h hsp001 d10
# metastat d10
d10: RAID
    State: Okay
    Hot spare pool: hsp001
...
```

▼ ホットスペア集合の対応付けを変更するには

始める前に 49 ページの「Solaris ボリュームマネージャコンポーネントを作成するための前提条件」を確認します。

1. スーパーユーザーになります。
2. 次のどちらかの方法でホットスペア集合とボリュームの対応付けを変更します。
   - Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開き、ボリュームを選択します。「アクション (Action)」、「プロパティ (Properties)」の順に選択します。さらに「ホットスペアプール (Hot Spare Pool)」パネルを選択します。画面の指示に従って不要なホットスペア集合を切り離してから、新しいホットスペア集合を接続します。詳細は、オンラインヘルプを参照してください。
   - 次の形式の metaparam コマンドを使用します。

     # metaparam -h *hot-spare-pool-name RAID5-volume-or-submirror-name*

     -h ホットスペア集合の対応付けを変更することを意味します。

hot-spare-pool 新しいホットスペア集合の名前を指定します。ホットスペア集合の対応付けを削除する場合は、特別なキーワード none を指定します。

component ホットスペア集合に対応付けるサブミラーまたは RAID-5 ボリュームの名前を指定します。

詳細は、metaparam(1M) のマニュアルページを参照してください。

例 17–6 ホットスペア集合の対応付けを変更する

次の例では、ホットスペア集合 hsp001 と RAID-5 ボリューム d4 がすでに対応付けられているものとします。ボリュームに対応するホットスペア集合を hsp002 に変更します。metastat コマンドを実行して、ホットスペア集合の対応付けが削除されていることを確認します。

```
# metastat d4
d4: RAID
    State: Okay
    Hot spare pool: hsp001
...
# metaparam -h hsp002 d4
# metastat d4
d4: RAID
    State: Okay
    Hot spare pool: hsp002
...
```

例 17–7 ボリュームとホットスペア集合の対応付けを削除する

次の例では、ホットスペア集合 hsp001 と RAID-5 ボリューム d4 がすでに対応付けられているものとします。ホットスペア集合の対応付けを none に変更します。これによって、このボリュームは、ホットスペア集合がまったく対応付けられていない状態になります。metastat コマンドを実行して、ホットスペア集合の対応付けが削除されていることを確認します。

```
# metastat d4
d4: RAID
    State: Okay
    Hot spare pool: hsp001
...
# metaparam -h none d4
# metastat d4
d4: RAID
    State: Okay
    Hot spare pool:
...
```

ホットスペア集合の保守

以下の各節では、ホットスペア集合の保守作業を実行する方法を説明します。

▼ ホットスペア集合とホットスペアの状態を確認するには

◗ 次のどちらかの方法でホットスペア集合とそのホットスペアの状態を表示します。

- Solaris 管理コンソール内の「拡張ストレージ」から「ホットスペアプール (Spare Pools)」ノードを開き、ホットスペア集合を選択します。「アクション (Action)」、「プロパティ (Properties)」の順に選択して、詳細な状態情報を表示します。詳細は、オンラインヘルプを参照してください。
- 次の形式の `metastat` コマンドを実行します。

  ```
  # metastat hot-spare-pool-name
  ```

例 17–8 ホットスペア集合の状態を表示する

次に、ホットスペア集合に対する `metastat` コマンドの出力例を示します。

```
# metastat hsp001
hsp001: 1 hot spare
        c1t3d0s2                Available       16800 blocks
```

ホットスペア集合の状態を確認するには、`metahs` コマンドを使用することもできます。

ホットスペア集合の状態と考えられる処置については、187 ページの「ホットスペア集合の状態」を参照してください。

▼ ホットスペア集合内のホットスペアを置き換えるには

1 スーパーユーザーになります。

2 次のどちらかの方法で、ホットスペアがすでに使用されているかどうかを確認します。

- Solaris 管理コンソール内の「拡張ストレージ」から「ホットスペアプール (Spare Pools)」ノードを開き、ホットスペア集合を選択します。「アクション (Action)」、「プロパティ (Properties)」の順に選択します。さらに「ホットスペア (Hot Spares)」パネルを選択します。画面の指示に従います。詳細は、オンラインヘルプを参照してください。
- 次の形式の `metastat` コマンドを使って、ホットスペア集合の状態を表示します。

  ```
  # metastat hot-spare-pool-name
  ```

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

3 次のどちらかの方法でホットスペアを置き換えます。

- Solaris 管理コンソール内の「拡張ストレージ」から「ホットスペアプール (Spare Pools)」ノードを開き、ホットスペア集合を選択します。「アクション (Action)」、「プロパティ (Properties)」の順に選択します。さらに「ホットスペア (Hot Spares)」パネルを選択します。画面の指示に従います。詳細は、オンラインヘルプを参照してください。
- 次の形式の `metahs` コマンドを使用します。

  ```
  # metahs -r hot-spare-pool-name current-hot-spare replacement-hot-spare
  ```

`-r`	特定のホットスペア集合のディスクを交換することを意味します。
hot-spare-pool-name	ホットスペア集合の名前を指定します。特殊なキーワード `all` を使って、すべてのホットスペア集合の対応付けを変更することもできます。
current-hot-spare	置き換えの対象となる現在のホットスペアの名前を指定します。
replacement-hot-spare	特定のホットスペア集合に含まれる現在のホットスペアと置き換えるスライスの名前を指定します。

  詳細については、`metahs(1M)`のマニュアルページを参照してください。

例 17–9 1 つのホットスペア集合内のホットスペアを置き換える

次の例では、`metastat` コマンドを使って、ホットスペアが使用中でないことを確認します。`metahs -r` コマンドは、ホットスペア集合 `hsp003` の ホットスペア `/dev/dsk/c0t2d0s2` を `/dev/dsk/c3t1d0s2` で置き換えます。

```
# metastat hsp003
hsp003: 1 hot spare
        c0t2d0s2                Broken       5600 blocks
# metahs -r hsp003 c0t2d0s2 c3t1d0s2
```

```
hsp003: Hotspare c0t2d0s2 is replaced with c3t1d0s2
```

例 17–10 対応付けられているすべてのホットスペア集合内のホットスペアを置き換える

次の例では、キーワード all を使って、対応付けられているすべてのホットスペア集合のホットスペア /dev/dsk/c1t0d0s2 を /dev/dsk/c3t1d0s2 で置き換えます。

```
# metahs -r all c1t0d0s2 c3t1d0s2
hsp001: Hotspare c1t0d0s2 is replaced with c3t1d0s2
hsp002: Hotspare c1t0d0s2 is replaced with c3t1d0s2
hsp003: Hotspare c1t0d0s2 is replaced with c3t1d0s2
```

▼ ホットスペア集合からホットスペアを削除するには

1. スーパーユーザーになります。
2. 次のどちらかの方法で、ホットスペアがすでに使用されているかどうかを確認します。
   - Solaris 管理コンソール内の「拡張ストレージ」から「ホットスペアプール (Spare Pools)」ノードを開き、ホットスペア集合を選択します。「アクション (Action)」、「プロパティ (Properties)」の順に選択します。さらに「ホットスペア (Hot Spares)」パネルを選択します。画面の指示に従います。詳細は、オンラインヘルプを参照してください。
   - 次の形式の metastat コマンドを使って、ホットスペア集合の状態を表示します。

     ```
     # metastat hot-spare-pool-name
     ```

     詳細は、metastat(1M) のマニュアルページを参照してください。
3. 次のどちらかの方法でホットスペアを削除します。
   - Solaris 管理コンソール内の「拡張ストレージ」から「ホットスペアプール (Spare Pools)」ノードを開き、ホットスペア集合を選択します。「アクション (Action)」、「プロパティ (Properties)」の順に選択します。さらに「ホットスペア (Hot Spares)」パネルを選択します。画面の指示に従います。詳細は、オンラインヘルプを参照してください。
   - 次の形式の metahs コマンドを使用します。

     ```
     # metahs -d hot-spare-pool-name current-hot-spare
     ```

`-d`	特定のホットスペア集合からホットスペアを削除することを指定します。
hot-spare-pool	ホットスペア集合の名前を指定します。特殊なキーワード `all` を使用して、すべてのホットスペア集合からホットスペアを削除することもできます。
current-hot-spare	削除する現在のホットスペアの名前を指定します。

詳細については、`metahs(1M)`のマニュアルページを参照してください。

例 17–11 1つのホットスペア集合からホットスペアを削除する

次の例では、`metastat` コマンドを使って、ホットスペアが使用中でないことを確認します。次に、`metahs -d` コマンドを使用して、ホットスペア集合 `hsp003` からホットスペア `/dev/dsk/c0t2d0s2` を削除します。

```
# metastat hsp003
hsp003: 1 hot spare
        c0t2d0s2                Broken       5600 blocks
# metahs -d hsp003 c0t2d0s2
```

▼ ホットスペアを有効にするには

1. スーパーユーザーになります。
2. 次のどちらかの方法でホットスペアを「使用可能 **(available)**」状態に戻します。
   - Solaris 管理コンソール内の「拡張ストレージ」から「ホットスペアプール (Spare Pools)」ノードを開き、ホットスペア集合を選択します。「アクション (Action)」、「プロパティ (Properties)」の順に選択します。さらに「ホットスペア (Hot Spares)」パネルを選択します。画面の指示に従います。詳細は、オンラインヘルプを参照してください。
   - 次の形式の `metahs` コマンドを使用します。

     # metahs -e *hot-spare-slice*

`-e`	ホットスペアを有効にすることを意味します。
hot-spare-slice	有効にするスライスの名前を指定します。

     詳細については、`metahs(1M)`のマニュアルページを参照してください。

例 17–12 ホットスペアを有効にする

この例では、metahs コマンドで、修理の完了したホットスペア /dev/dsk/c0t0d0s2 を「使用可能 (Available)」状態にします。ホットスペア集合を指定する必要はありません。

```
# metahs -e c0t0d0s2
```

◆ ◆ ◆ 第 18 章

18 ディスクセット (概要)

この章では、ディスクセットの概念について説明します。関連する作業の実行手順については、第 19 章を参照してください。

この章では、次の内容について説明します。

ディスクセットの新機能

この節では、今回の Solaris リリースで導入された、ディスクセットの新機能について説明します。

Solaris の新機能の一覧と Solaris のリリースの説明については、『Solaris 10 の概要』を参照してください。

ディスクセットの紹介

ディスクセットとは、論理ボリュームとホットスペアを含む物理的な記憶領域ボリュームの集まりのことです。ボリュームやホットスペア集合は、そのディスクセット内のドライブだけで構成される必要があります。ディスクセット内で作成したボリュームは、物理スライスと同じように使用できます。このボリュームを使用して、ファイルシステムの作成やマウント、およびデータの格納が可能です。

注 – ディスクセットは、SPARC ベースのプラットフォームでも x86 ベースのプラットフォームでもサポートされています。

ディスクセットのタイプ

この節では、 Solaris ボリュームマネージャで利用できるディスクセットのタイプについて説明します。

ローカルディスクセット

どのホストにも 1 つのローカルディスクセットがあります。ローカルディスクセットは、ホスト上のすべてのディスクのうち、名前付きディスクセットに含まれないディスクで構成されます。ローカルディスクセットは特定のホストだけに属します。ローカルディスクセットには、そのホストの構成を記録した状態データベースが格納されています。ローカルディスクセット内のボリュームやホットスペア集合は、ローカルディスクセット内のドライブだけで構成されます。

名前付きディスクセット

ローカルディスクセットに加えて、ホストは名前付きディスクセットにも参加できます。名前付きディスクセットは、ローカルディスクセットではない任意のディスクセットです。以下にあげるタイプの名前付きディスクセットを実装して、ボリュームを管理できます。どのタイプを実装するかは、システムの構成によって異なります。

共有ディスクセット

「共有ディスクセット」は、複数のホストで共有できるディスクセットです。共有ディスクセットは参加しているすべてのホストから認識できますが、アクセスできるのはそのディスクセットの所有者だけです。各ホストは共有ディスクセットを制御できますが、1 度に 1 つのホストだけが制御できます。さらに、共有ディスクセットは独自の名前空間を提供します。ボリュームはこの名前空間内で管理されます。

共有ディスクセットを使用すると、データの冗長性と可用性が向上します。あるホストで障害が発生すると、そのホストのディスクセットを別のホストで引き継ぐことができます (この種の構成を「フェイルオーバー構成」と呼ぶ)。

注 – 共有ディスクセットは、Sun Cluster、Solstice HA (High Availability)、またはサポートされる他社製の HA フレームワークと組み合わせて使用することを想定しています。Solaris ボリュームマネージャ単体では、フェイルオーバー構成を実装するのに必要なすべての機能が提供されるとは限りません。

各ホストでディスクセットを制御できますが、1 つのホストが 1 時点で制御できるディスクセットは 1 つだけです。

自動取得ディスクセット

Solaris 9 4/04 リリースで自動取得機能が利用できるようになるまで、Solaris ボリュームマネージャは、`/etc/vfstab` ファイル経由による、ディスクセット上のファイルシステムの自動マウントをサポートしていませんでした。Solaris ボリュームマネージャでディスクセット上のファイルシステムにアクセスするには、このディスクセットを取得するコマンド、つまり、`metaset -s` *setname* `-t` コマンドを、システム管理者が手動で実行する必要がありました。

自動取得機能を使用すると、ブート時にディスクセットが自動的にアクセスされるように設定できます。この設定には、`metaset -s` *setname* `-A enable` コマンドを使用します。自動取得機能によって、ブート時に `/etc/vfstab` ファイル内のファイルシステムに対して、マウントオプションを定義できます。この機能を使用すると、有効なディスクセット内のボリューム上にあるファイルシステム用のマウントオプションを、`/etc/vfstab` ファイルに定義できます。

自動取得機能をサポートするのは、シングルホストのディスクセットだけです。自動取得機能を使用するには、ディスクセットがほかのシステムによって共有されていないことが条件となります。共有ディスクセットは、自動取得機能を使用するように設定できません。 `metaset -A` コマンドは失敗します。ただし、ディスクセットから他のホストを削除すれば、自動取得するように設定できます。同様に、自動取得ディスクセットにはほかのホストを追加できません。しかし、自動取得機能を無効にすると、そのディスクセットにほかのホストを追加できるようになります。

注 – Sun Cluster 環境では、自動取得機能は無効になります。Sun Cluster 自身がディスクセットを取得または解放するためです。

自動取得機能の詳細については、`metaset(1M)` の `-A` オプションを参照してください。

複数所有者ディスクセット

Sun Cluster 環境で作成された名前付きディスクセットのことを複数所有者ディスクセットといいます。複数所有者ディスクセットを使用すると、複数のノードがディスクセットの所有権を共有したり、共有ディスクに同時にアクセスしたりできます。複数所有者ディスクセット内のすべてのディスクとボリュームは、クラスタ内のすべてのノードから直接アクセスできます。各複数所有者ディスクセットは、そのディスクセットに

追加されたホストリストを持っています。結果として、同じクラスタ構成内の各複数所有者ディスクセットは、異なる (ときには、重複する) ホストセットを持つことができます。

各複数所有者ディスクセットはマスターノードを持っています。マスターノードの機能は、状態データベースの複製の変更を管理および更新することです。ディスクセットごとにマスターノードが存在するため、複数のマスターが共存することもあります。マスターの選ばれ方には、2 とおりあります。まず、ディスクセットに最初のディスクを追加したときには、そのノードがマスターになります。もうひとつは、マスターノードがパニックになり、失敗したときです。この場合は、最も低いノード ID を持つノードがマスターノードになります。

複数所有者ディスクセットの機能が有効なのは、複数所有者ディスクセット記憶領域を管理する Sun Cluster 環境に限られます。Solaris Volume Manager for Sun Cluster機能は、Sun Cluster 10/04 ソフトウェアコレクション以降の Sun Cluster リリースと Oracle Real Applications Clusters などのアプリケーションで使用できます。Solaris Volume Manager for Sun Clusterの詳細については、第 4 章「Solaris Volume Manager for Sun Cluster (概要)」を参照してください。

複数所有者ディスクセットを構成するには、Solaris OS とともに次のソフトウェアをインストールしておく必要があります。

- Sun Cluster 初期クラスタフレームワーク
- Sun Cluster Support for Oracle Real Application Clusters ソフトウェア
- Oracle Real Application Clusters ソフトウェア

注 – Sun Cluster と Oracle Real Application Clusters ソフトウェアの設定については、『Sun Cluster ソフトウェアのインストールガイド (Solaris OS 版)』、および『Sun Cluster Data Service for Oracle Real Application Clusters ガイド (Solaris OS 版)』を参照してください。

Solaris ボリュームマネージャにおけるディスクセットの管理

ローカルディスクセットを管理する場合とは異なり、ディスクセットの状態データベースを手動で作成したり削除したりする必要はありません。なぜなら、Solaris ボリュームマネージャが自動的に、ディスクセット内の各ディスク (のスライス 7) に状態データベースの複製を 1 つずつ配置するためです。 1 つのディスクセットで合計 50 までの複製を作成できます。

ディスクセットにディスクを追加すると、Solaris ボリュームマネージャはディスクセット上に状態データベースの複製を自動的に作成します。ディスクセットがディスクを受け入れると、Solaris ボリュームマネージャは、ディスクセットの状態データベースの複製をそのディスクに配置できるように、ディスクのパーティションを再分割することがあります (209 ページの「ディスクの自動パーティション分割」を参照)。

ディスクセット内のボリューム上にあるファイルシステムが、ブート時に /etc/vfstab ファイルを介して自動的にマウントされることは通常ありません。これは、必要とされる、Solaris ボリュームマネージャ RPC デーモン (rpc.metad と rpc.metamhd) が、ブート時にはまだ実行されていないためです。また、ディスクセットの所有権はリブート時に失われます。/etc/inetd.conf ファイル内で Solaris ボリュームマネージャ RPC デーモンを無効にしないでください。これらのデーモンはデフォルトでブートするように構成されています。これらのデーモンは、Solaris ボリュームマネージャがその機能を完全に使用できるように、有効にしておく必要があります。

metaset コマンドの -A オプションによって、自動取得機能が有効になっている場合は、ブート時にディスクセットが自動的に取得されます。この場合、ディスクセット内のボリューム上にあるファイルシステムは、/etc/vfstab ファイルを使用して自動的にマウントできます。ブート時に自動取得を有効にするには、ディスクセットが 1 つのホストにだけ対応付けられていて、なおかつ自動取得機能が有効になっていなければなりません。ディスクセットは、ディスクセットの作成時でも作成後でも有効にできます。自動取得機能の詳細については、205 ページの「自動取得ディスクセット」を参照してください。

注 – ディスクセットはシングルホスト構成でもサポートされますが、通常、「ローカル」な (二重に接続されていない) 使用形態には適していません。例外的な使用形態として、ディスクセットを使用して論理ボリュームの名前空間を管理しやすくする場合と、Storage Area Network (SAN) 構成においての記憶領域の管理を容易にする場合が挙げられます (214 ページの「シナリオ — ディスクセット」)を参照)。

ディスクセットの作成と構成は、Solaris ボリュームマネージャのコマンド行インタフェース (metaset コマンド) または Solaris 管理コンソール内の「拡張ストレージ」を使って行います。

ディスクセットにディスクを追加すると、そのディスクセットを共有するホストは、ディスクセットを「予約」(または「取得」) したり、「解放」したりできます。ディスクセットが任意のホストに予約されている場合、そのディスクセットを共有するほかのホストは、そのディスクセットのディスクのデータにアクセスできません。ディスクセットの保守を行うためには、そのホストがディスクセットを所有しているか予約していなければなりません。ディスクセットに最初のディスクを設定したホストは、暗黙的にディスクセットの所有者になります。

metaimport コマンドを使用すると、ディスクセット (ほかのシステム上で作成されたディスクセットを含む) を既存の Solaris ボリュームマネージャ構成にインポートできます。

ディスクセットの取得

ホストがディスクセット内のディスクを使用するには、先にディスクセットを取得する必要があります。ディスクセットを取得するには、次の 2 とおりの方法があります。

- 安全取得 - この方法でディスクセットを取得すると、Solaris ボリュームマネージャはこのディスクセットを確保しようとし、他のホストはこのディスクセットを解放しようとします。この解放 (つまり、取得) は失敗することもあります。
- 強制取得 - この方法でディスクセットを取得すると、Solaris ボリュームマネージャは、別のホストがこのディスクセットを取得しているかどうかに関係なく、ディスクセットを取得します。この方法は、通常、ディスクセットを共有するホストの 1 つが停止したり、他のホストと通信していないときに使用されます。これによって、そのディスクセットのすべてのディスクが新しいホストに引き継がれます。そして、この取得を行なったホストが状態データベースを読み込み、このディスクセットの共有ボリュームにアクセスできるようになります。この時点で他のホストがこのディスクセットをすでに取得していると、そのホストは取得が失われたためにパニックを起こします。

  正常な状態においては、ディスクセットを共有している 2 つのホストが協調関係にあるため、ディスクセットのディスクを両者が同時に取得することはありません。正常な状態とは、両方のホストが動作し、相互に通信している状態のことです。

注 – 予想に反してディスクが取得されていない場合 (このディスクセットを共有する別のホストがこのディスクを強制的に確保したことが原因の可能性がある)、そのホストはパニックを起こします。これによって、2 つのホストが同時に同じディスクにアクセスした場合に起こり得るデータの消失が最小限に抑えられます。

ディスクセットの取得または予約の詳細については、224 ページの「ディスクセットを取得するには」を参照してください。

ディスクセットの解放

ディスクセットの物理ディスクを保守する場合は、ディスクセットをあらかじめ解放しておくと便利です。ディスクセットを解放すると、そのディスクセットはホストからアクセスできなくなります。ディスクセットを共有しているホストが両方ともディスクセットを解放すると、どちらのホストもそのディスクセットのディスクにアクセスできなくなります。

ディスクセットの解放についての詳細は、226 ページの「ディスクセットを解放するには」を参照してください。

ディスクセットのインポート

Solaris 9 9/04 リリースから、`metaimport` コマンドを使用して、ディスクセットのデバイス ID サポートが設定されている Solaris ボリュームマネージャ構成に、ディスクセット (複製されたディスクセットを含む) をインポートできるようになりました。また、`metaimport` コマンドを使用すると、インポートに利用できるディスクセットについて報告できます。

複製されたディスクセットを作成するには、リモート複製ソフトウェアを使用します。複製されたディスクセットを metaimport コマンドでインポートするには、ディスクセットの各ディスクの状態データベースの複製を含むスライスも、複製されたディスクセットの同じスライスに複製する必要があります。これは、EFI 以外のディスクではスライス 7 に相当し、EFI ディスクではスライス 6 に相当します。ディスクセットを複製する前に、複製するデータのディスク構成とリモートサイトのディスク構成が一致することを確認します。この手順は、状態データベースの複製とデータの両方が正確に複製されることを保証します。

ディスクセット内のあるディスクにボリュームまたは状態データベースの複製が含まれていない場合、metaimport コマンドもそのディスクをインポートしません。このような状況は、ボリュームまたは状態データベースの複製がディスクに追加されていなかった (あるいは、ディスクからすでに削除されていた) 場合に発生します。この場合、あるディスクセットを別のシステムにインポートすると、そのディスクセットからあるディスクがなくなったように見えることがあります。たとえば、Solaris ボリュームマネージャディスクセットに許可される状態データベースの複製の最大数は 50 です。あるディスクセットに 60 台のディスクがある場合、10 台のディスクには状態データベースの複製が含まれないため、それらのディスクをインポートするには、それらのディスクにボリュームが含まれている必要があります。

ディスクセットのインポートに関連する作業については、229 ページの「ディスクセットのインポート」を参照してください。

注 – Sun Cluster 環境では、metaimport コマンドは使用できません。

ディスクの自動パーティション分割

ディスクセットに新しいディスクが追加されると、Solaris ボリュームマネージャがディスクのフォーマットをチェックします。必要に応じて、Solaris ボリュームマネージャはディスクのパーティションを再分割し、状態データベースの複製が納まるだけの領域を備え、適切に構成されたスライス 7 がディスクに必ずあるようにします。スライス 7 の正確な容量は、ディスクのジオメトリによって決まります。ただし、4M バイトよりは小さくなることはなく、通常は 6M バイト近くになります (シリンダ境界がどこにあるかによる)。

デフォルトでは、Solaris ボリュームマネージャはスライス 7 に状態データベースの複製を 1 つ格納します。複数の状態データベースの複製をスライスに収めるために、スライス 7 のデフォルト容量を増やしたり、状態データベースの複製を小さくしたりできます。

注 – スライス 7 の最小限のサイズは、状態データベースの複製のサイズ、状態データベースの複製に保管する情報など、さまざまな要因によって、将来変わってくる可能性があります。複数所有者ディスクセットの場合、状態データベースの複製のデフォルトサイズは 16M バイトです。

ディスクセットで使用するディスクには、次の条件を満たしたスライス 7 を与える必要があります。

- セクター 0 から始まる
- ディスクラベルと状態データベースの複製とを格納できる領域がある
- マウントできない
- スライス 2 などの他のスライスと重なり合わない

既存のパーティションテーブルがこれらの条件を満たしていない場合、Solaris ボリュームマネージャはディスクを再分割します。各ドライブ上で、Solaris ボリュームマネージャによって使用される小さい領域が、スライス 7 として確保されます。各ドライブの残りの領域は、スライス 0 に割り当てられます。パーティションが再分割されると、ディスク上のすべてのデータが失われます。

ヒント–ディスクセットにドライブを追加した後、ディスクのパーティションは必要に応じて再分割できますが、スライス 7 は変更できません。

以下に、prtvtoc コマンドで表示した、ディスクセットに追加する前のディスク情報の出力例を示します。

```
[root@lexicon:apps]$ prtvtoc /dev/rdsk/c1t6d0s0
* /dev/rdsk/c1t6d0s0 partition map
*
* Dimensions:
*     512 bytes/sector
*     133 sectors/track
*      27 tracks/cylinder
*    3591 sectors/cylinder
*    4926 cylinders
*    4924 accessible cylinders
*
* Flags:
*   1: unmountable
*  10: read-only
*
*                          First     Sector    Last
* Partition  Tag  Flags    Sector     Count    Sector  Mount Directory
       0      2    00          0   4111695   4111694
       1      3    01    4111695   1235304   5346998
       2      5    01          0  17682084  17682083
       3      0    00    5346999   4197879   9544877
       4      0    00    9544878   4197879  13742756
       5      0    00   13742757   3939327  17682083
```

この出力から、ディスクにスライス 7 がないことがわかります。したがって、ディスクセットにディスクを追加すると、Solaris ボリュームマネージャがディスクのパーティションを再分割します。以下に、prtvtoc コマンドで表示した、ディスクセットに追加した後のディスク情報の出力例を示します。

```
[root@lexicon:apps]$ prtvtoc /dev/rdsk/c1t6d0s0
* /dev/rdsk/c1t6d0s0 partition map
*
* Dimensions:
*     512 bytes/sector
*     133 sectors/track
*      27 tracks/cylinder
*    3591 sectors/cylinder
*    4926 cylinders
*    4924 accessible cylinders
*
* Flags:
*   1: unmountable
*  10: read-only
*
*                          First     Sector    Last
* Partition  Tag  Flags    Sector     Count    Sector  Mount Directory
       0      0    00      10773  17671311  17682083
   7   0   01     0   10773   10772
[root@lexicon:apps]$
```

この出力から、ディスクが再分割されてシリンダ 0 から始まるスライス 7 が含まれ、状態データベースの複製を格納できるだけの領域が確保されていることがわかります。ディスクセットに追加した各ディスクに適切なスライス 7 がある場合は、再フォーマットは行なわれません。

注 – Solstice DiskSuite ソフトウェアで使用していたディスクセットが存在する場合、それらのディスクセット上の状態データベースの複製のデフォルトサイズは 1034 ブロックです。一方、Solaris ボリュームマネージャで使用されるデフォルトサイズは 8192 ブロックです。そのため、Solstice DiskSuite ソフトウェアで追加されたディスクのスライス 7 は、Solaris ボリュームマネージャで追加されたディスクのスライス 7 よりも小さくなっています。

ディスクセットの命名規則

ディスクセットのボリューム名は、Solaris ボリュームマネージャの他のコンポーネント名と同様です。ただし、ディスクセット名が名前の中に含まれます。たとえば、ボリュームパス名では、`/dev/md/` とパス内の実際のボリューム名の間にディスクセット名が入ります。

次の表に、ディスクセットボリューム名の例を示します。

表 18–1 ディスクセットボリューム名の例

/dev/md/blue/dsk/d0	ディスクセット blue 内のブロック型ボリューム d0
/dev/md/blue/dsk/d1	ディスクセット blue 内のブロック型ボリューム d1
/dev/md/blue/rdsk/d126	ディスクセット blue 内の raw ボリューム d126
/dev/md/blue/rdsk/d127	ディスクセット blue 内の raw ボリューム d127

同様に、ホットスペア集合の場合も、その名前の一部にディスクセット名が含まれます。

例 — 2 つの共有ディスクセット

図 18–1 に、2 つのディスクセットを使用する構成例を示します。

この構成では、ホスト A とホスト B はディスクセット red と blue を共有します。各ホストは独自のローカルディスクセットを持っており、これらは共有されません。ホスト A に障害が発生すると、ホスト B がホスト A の共有ディスクセットであるディスクセット red の制御を引き継ぎます。同様に、ホスト B に障害が発生すると、ホスト A がホスト B の共有ディスクセットであるディスクセット blue の制御を引き継ぎます。

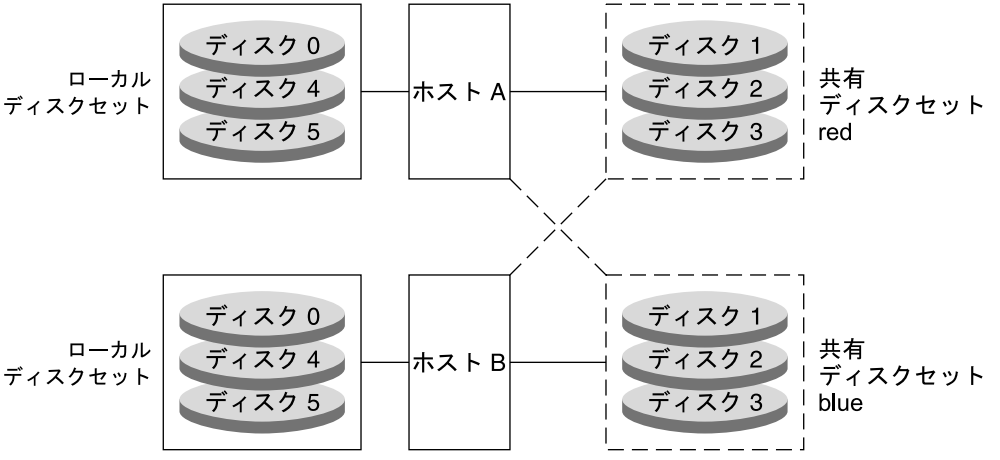

図 18–1 ディスクセットの例

ディスクセットを使用するときの指針

ディスクセットを使用するときには、次の指針を考慮してください。

- ディスクセットに接続されるホストごとに Solaris ボリュームマネージャを構成する必要があります。
- ディスクセットを作成するためには、各ホスト上にローカルの状態データベースが設定されていなければなりません。
- ディスクセットを作成して、そのディスクセットのボリュームを作成する手順では、まず最初に、ディスクセットを作成します。次に、そのディスクセットにディスクを追加します。最後に、そのディスクセットにボリュームを作成します。
- クラスタ環境内でディスクセットを作成および使用する場合、`root` はシステム管理グループ (Group 14) のメンバーである必要があります。あるいは、各ホスト上にある `/.rhosts` ファイルに、そのディスクセットに関連付けられたほかのホスト名のエントリがなければなりません。

注 – この手順は、SunCluster 3.x 環境では不要です。

- ディスクセットの保守を行うためには、そのホストがディスクセットを所有しているか予約していなければなりません。ディスクセットに最初のドライブを設定したホストは、暗黙的にディスクセットの所有者になります。
- ファイルシステム、データベース、またはほかのアプリケーションに使用されているドライブは、ディスクセットに追加できません。ドライブを追加する前に、そのディスクが使用中でないことを確認してください。
- 保存しておく必要のある既存のデータが入っているドライブをディスクセットに追加してはなりません。ディスクセットにディスクを追加すると、パーティションが再分割され、データが破壊されます。
- ローカルボリュームの管理と異なり、ディスクセット上の状態データベースの複製を手動で作成したり削除したりする必要はありません。Solaris ボリュームマネージャは、ディスクセットの各ドライブに適切な数の状態データベースの複製を配置しようとします。
- ディスクセットにドライブを追加すると、Solaris ボリュームマネージャは、既存のドライブにある状態データベースの複製の配置を再調整します。必要であれば、後で `metadb` コマンドを使って複製の配置を変更できます。

ディスクセット内で非同期的に共有される記憶領域

Solaris ボリュームマネージャの以前のバージョンでは、ディスクセット内のホスト間で共有しようとしているすべてのディスクは、各ホストに接続する必要がありました。また、このようなディスクは各ホストに対して、まったく同じパス、ドライバ、および名前を持つ必要がありました。特に、共有ディスクドライブは、同じ場所 (`/dev/rdsk/c#t#d#`) にある両方のホストから見ることができる必要がありました。さらに、共有されるディスクは同じドライバ名 (`ssd`) を使用する必要がありました。

現在の Solaris OS リリースでは、共通にアクセスできる記憶領域に対して異なるパスを持っているシステム同士は、あるディスクセットに対して同時にアクセスできません。ディスクセットのデバイス ID サポートの導入により、Solaris ボリュームマネージャは、名前付きディスクセット内におけるディスクの移動を自動的に追跡できます。

注 – Sun Cluster 環境では、ディスクセットのデバイス ID サポートはありません。

最新の Solaris OS にアップグレードしたとき、ディスクの追跡を有効にするには、一度、ディスクセットを取得する必要があります。ディスクセットの取得の詳細については、224 ページの「ディスクセットを取得するには」を参照してください。

自動取得機能が有効でない場合、各ディスクセットを手動で取得する必要があります。自動取得機能が有効である場合、各ディスクセットはシステムのリブート時に自動的に取得されます。自動取得機能の詳細については、205 ページの「自動取得ディスクセット」を参照してください。

この拡張されたデバイス ID サポートを使用すると、ディスクセット (異なるシステム上で作成されたディスクセットを含む) をインポートできます。ディスクセットのインポートの詳細については、208 ページの「ディスクセットのインポート」を参照してください。

シナリオ—ディスクセット

次の例では、第 5 章のサンプルシステムを使用して、SAN (Storage Area Network) ファブリック上にある記憶領域をディスクセットを使用して管理する方法を示します。

サンプルシステムには、ファイバスイッチと SAN 記憶領域に接続されたもう 1 つのコントローラがあるものとします。ブートプロセスの段階では、システムは SCSI ディスクや IDE ディスクの場合と同様、SAN ファブリック上の記憶領域を使用できません。Solaris ボリュームマネージャは、ブート時にファブリック上の論理ボリュームについて、使用不可であると伝えます。しかし、ディスクセットに記憶領域を追加し、さらにディスクセットツールで記憶領域を管理することによって、ブート時の可用性に関する問題を回避できます。また、ファブリックに接続された記憶領域は、ローカルな記憶領域からディスクセットが制御する独立した名前空間内で容易に管理できます。

◆ ◆ ◆ 第 19 章

19 ディスクセット (作業)

この章では、ディスクセットに関連する作業について説明します。各作業に関連する概念については、第 18 章を参照してください。

ディスクセット (作業マップ)

次の作業マップに、Solaris ボリュームマネージャのディスクセットと Solaris Volume Manager for Sun Cluster 複数所有者ディスクセットを管理するために必要な作業を示します。特に注記していない限り、すべてのコマンドは両方のディスクセットに使用できます。複数所有者ディスクセットに関連する作業には、Solaris ボリュームマネージャ GUI は使用できません。

作業	説明	参照先
ディスクセットを作成する	Solaris ボリュームマネージャの GUI か `metaset` コマンドを使ってディスクセットを作成します。 `metaset -M` コマンドを使用して、複数所有者ディスクセットを作成します。	216 ページの「ディスクセットを作成するには」
ディスクセットにディスクを追加する	Solaris ボリュームマネージャの GUI か `metaset` コマンドを使用して、ディスクセットにディスクを追加します。	218 ページの「ディスクセットにディスクを追加するには」
ディスクセットにホストを追加する	Solaris ボリュームマネージャの GUI か `metaset` コマンドを使ってディスクセットにホストを追加します。	219 ページの「ディスクセットに別のホストを追加するには」
ディスクセットに Solaris ボリュームマネージャのボリュームを作成する	Solaris ボリュームマネージャの GUI か `metainit` コマンドを使ってディスクセットにボリュームを作成します。	221 ページの「ディスクセットに Solaris ボリュームマネージャのコンポーネントを作成するには」

作業	説明	参照先
ディスクセットの状態をチェックする	Solaris ボリュームマネージャの GUI か `metastat` コマンドを使ってディスクセットの状態をチェックします。	222 ページの「ディスクセットの状態をチェックするには」
ディスクセットからディスクを削除する	Solaris ボリュームマネージャの GUI か `metaset` コマンドを使用して、ディスクセットからディスクを削除します。	223 ページの「ディスクセットからディスクを削除するには」
ディスクセットを取得する	Solaris ボリュームマネージャの GUI か `metaset` コマンドを使ってディスクセットを取得します。	224 ページの「ディスクセットを取得するには」
ディスクセットを解放する	Solaris ボリュームマネージャの GUI か `metaset` コマンドを使ってディスクセットを解放します。	226 ページの「ディスクセットを解放するには」
ディスクセットからホストを削除する	Solaris ボリュームマネージャの GUI か `metaset` コマンドを使ってディスクセットからホストを削除します。	227 ページの「ホストまたはディスクセットを削除するには」
ディスクセットを削除する	Solaris ボリュームマネージャの GUI か `metaset` コマンドを使ってディスクセットから最後のホストを削除し、ディスクセットを削除します。	227 ページの「ホストまたはディスクセットを削除するには」
ディスクセットをインポートする	`metaimport` コマンドを使用して、ディスクセットについてのレポートを取得することによって、どのディスクセットをインポートできるかを判断したり、あるシステムから別のシステムにディスクセットをインポートしたりします。	229 ページの「ディスクセットのインポート」

ディスクセットの作成

▼ ディスクセットを作成するには

始める前に　213 ページの「ディスクセットを使用するときの指針」を確認します。

1 次のどちらかの方法でディスクセットを作成します。

- Solaris 管理コンソール内の「拡張ストレージ」から「ディスクセット (Disk Sets)」ノードを開きます。「アクション (Action)」、「ディスクセットの作成 (Create Disk Set)」の順に選択します。さらに、ウィザードの指示に従います。詳細は、オンラインヘルプを参照してください。

- コマンド行から次の形式の metaset コマンドを実行して、ディスクセットを最初から作成します。

  # metaset -s *diskset-name* **-a -h -M** *hostname*

  -s *diskset-name* — metaset コマンドの実行対象となるディスクセットの名前を指定します。

  -a — ディスクセットにホストを追加することを意味します。Solaris ボリュームマネージャは、ディスクセット当たり最大で 4 つのホストをサポートします。

  -M — 複数所有者ディスクセットを作成することを意味します。

  -h *hostname* — ディスクセットに追加する 1 つまたは複数のホストを指定します。最初のホストを追加すると、ディスクセットが作成されます。2 番目のホストはあとで追加できます。ただし、指定した *hostname* にディスクセット内のディスクが全部見つからなかった場合、2 番目のホストは受け付けられません。*hostname* は、/etc/nodename ファイルに指定されている名前と同じでなければなりません。

  詳細は、metaset(1M) のマニュアルページを参照してください。

2 新しいディスクセットの状態をチェックする

```
# metaset
```

例 19–1 ディスクセットを作成する

次の例では、ホスト host1 から共有ディスクセット blue を作成します。metaset コマンドは状態を表示します。この時点ではディスクセットの所有者はいません。ディスクセットにディスクを追加するホストがデフォルトで所有者になります。

```
# metaset -s blue -a -h host1
# metaset
Set name = blue, Set number = 1

Host                Owner
  host1
```

例 19–2 複数所有者ディスクセットを作成する

次の例では、複数所有者ディスクセット red を作成します。metaset コマンド出力の先頭行に表示されている「Multi-owner」は、このディスクセットが複数所有者ディスクセットであることを示しています。

```
# metaset -s red -a -M -h nodeone
# metaset -s red
```

```
Multi-owner Set name = red, Set number = 1, Master =

Host                Owner          Member
  nodeone                          Yes
```

ディスクセットの拡張

▼ ディスクセットにディスクを追加するには

注意 – 32 ビットカーネルの Solaris ソフトウェアを実行する予定がある場合、または Solaris 9 4/03 リリースより前のバージョンの Solaris OS を使用する予定がある場合は、1T バイトを超えるディスクをディスクセットに追加しないでください。Solaris ボリュームマネージャのマルチテラバイトボリュームサポートの詳細については、49 ページの「Solaris ボリュームマネージャにおけるマルチテラバイトサポートの概要」を参照してください。

ディスクセットに追加されるディスクは、次の条件を満たしている必要があります。

- ディスクがボリュームまたはホットスペア集合で使用中であってはなりません。
- ディスクに状態データベースの複製が含まれていてはなりません。
- ディスクがマウントされていたり、スワップされていたり、アプリケーションによって開かれていたりしてはなりません。

始める前に　213 ページの「ディスクセットを使用するときの指針」を確認します。

1 次のどちらかの方法でディスクセットにディスクを追加します。

- Solaris 管理コンソール内の「拡張ストレージ」から「ディスクセット (Disk Sets)」ノードを開きます。変更するディスクを選択します。右マウスボタンをクリックして、「プロパティ (Properties)」を選択します。ディスク (Disks) タブを選択します。「ディスクを追加 (Add Disk)」をクリックします。さらに、ウィザードの指示に従います。詳細は、オンラインヘルプを参照してください。
- コマンド行から次の形式の `metaset` コマンドを実行して、ディスクセットにディスクを追加します。

```
# metaset -s diskset-name -a disk-name
```

`-s` *diskset-name*　`metaset` コマンドの実行対象となるディスクセットの名前を指定します。

`-a`　ディスクセットにディスクを追加することを意味します。

disk-name ディスクセットに追加するディスクの名前です。ディスク名の形式は *cxtxdx* です。ディスクセットにディスクを追加する場合は、スライス識別子「*sx*」を指定しません。

詳細は、`metaset(1M)` のマニュアルページを参照してください。

ディスクセットに最初にディスクを追加するホストがディスクセットの所有者になります。

注意 – データの含まれているディスクをデータセットに追加してはなりません。ディスクセットにデータが含まれているディスクを追加すると、ディスクのパーティションが再分割され、データが破壊されることがあります。

2 ディスクセットとディスクの状態を検査します。

```
# metaset
```

例 **19–3** ディスクセットにディスクを追加する

```
# metaset -s blue -a c1t6d0
# metaset
Set name = blue, Set number = 1

Host                Owner
  host1            Yes

Drive               Dbase
  c1t6d0             Yes
```

この例では、ホスト名は `host1` で、共有ディスクセットの名前は `blue` です。ディスクセット `blue` に追加されたディスクは `c1t6d0` だけです。

コマンド行から個々のディスクを指定すると、一度に複数のディスクを追加できます。たとえば、次のコマンドを使用すると、ディスクセットに 2 つのディスクを同時に追加できます。

```
# metaset -s blue -a c1t6d0 c2t6d0
```

▼ ディスクセットに別のホストを追加するには

この手順では、1 つのホストがすでに接続されているディスクセットに別のホストを追加する手順について説明します。Solaris ボリュームマネージャは、ディスクセット当たり最大で 4 つのホストをサポートします。

始める前に 213 ページの「ディスクセットを使用するときの指針」を確認します。

1 次のどちらかの方法でディスクセットにホストを追加します。

- Solaris 管理コンソール内の「拡張ストレージ」から「ディスクセット (Disk Sets)」ノードを開き、変更するディスクセットを選択します。変更するディスクを選択します。右マウスボタンをクリックして、「プロパティ (Properties)」を選択します。ホスト (Hosts) タブを選択します。「ホストを追加 (Add Host)」をクリックします。さらに、ウィザードの指示に従います。詳細は、オンラインヘルプを参照してください。
- コマンド行から次の形式の `metaset` コマンドを実行して、ディスクセットにホストを追加します。

  # metaset -s *diskset-name* **-a -h** *hostname*

`-s` *diskset-name*	ホストを追加するディスクセットの名前を指定します。
`-a`	指定されたディスクセットにホストを追加します。
`-h` *hostname*	ディスクセットに追加する 1 つまたは複数のホスト名を指定します。最初のホストを追加すると、ディスクセットが作成されます。ホスト名は、`/etc/nodename` ファイルに指定されている名前と同じである必要があります。

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

2 ディスクセットにホストが追加されたかどうかを確認します。

```
# metaset
```

例 19–4 ディスクセットに別のホストを追加する

```
# metaset -s blue -a -h host2
# metaset
Set name = blue, Set number = 1

Host                Owner
  host1                Yes
  host2

Drive               Dbase
  c1t6d0             Yes
  c2t6d0             Yes
```

この例では、ディスクセット `blue` にホスト `host2` を追加します。

▼ ディスクセットに Solaris ボリュームマネージャのコンポーネントを作成するには

ディスクセットを作成したら、ディスクセットに追加したディスクを使ってボリュームやホットスペア集合を作成できます。この操作には、Solaris 管理コンソール内の「拡張ストレージ」かコマンド行ユーティリティーを使用します。

- 次のどちらかの方法で、ボリュームなどの **Solaris** ボリュームマネージャコンポーネントをディスクセット内に作成します。
  - Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」、「状態データベースの複製 (State Database Replicas)」、または「ホットスペアプール (Hot Spare Pools)」ノードを選択します。「アクション (Action)」、「作成 (Create)」の順に選択します。さらに、ウィザードの指示に従います。詳細は、オンラインヘルプを参照してください。
  - 同じコマンド行ユーティリティーと同じ基本構文を使用して、ボリューム、状態データベースの複製、またはホットスペア集合を作成します。ただし、コマンドごとに、コマンドのすぐ後ろに - s *disk-set* を指定します。

    # *command* **-s** *disk-set*

例 19–5 ディスクセットに Solaris ボリュームマネージャのボリュームを作成する

次の例では、ディスクセット `blue` にミラー `d10` を作成します。ミラーはサブミラー (RAID-0 ボリューム) `d11` と `d12` からなります。

```
# metainit -s blue d11 1 1 c1t6d0s0
blue/d11: Concat/Stripe is setup
# metainit -s blue d12 1 1 c2t6d0s0
blue/d12: Concat/Stripe is setup
# metainit -s blue d10 -m d11
blue/d10: Mirror is setup
# metattach -s blue d10 d12
blue/d10: submirror blue/d12 is attached

# metastat -s blue
blue/d10: Mirror
    Submirror 0: blue/d11
      State: Okay
    Submirror 1: blue/d12
      State: Resyncing
    Resync in progress: 0 % done
    Pass: 1
    Read option: roundrobin (default)
```

```
    Write option: parallel (default)
    Size: 17674902 blocks

blue/d11: Submirror of blue/d10
    State: Okay
    Size: 17674902 blocks
    Stripe 0:
        Device              Start Block  Dbase State        Reloc  Hot Spare
        c1t6d0s0                  0     No    Okay


blue/d12: Submirror of blue/d10
    State: Resyncing
    Size: 17674902 blocks
    Stripe 0:
        Device              Start Block  Dbase State        Reloc  Hot Spare
        c2t6d0s0                  0     No    Okay
```

ディスクセットの保守

▼ ディスクセットの状態をチェックするには

- 次のどちらかの方法でディスクセットの状態をチェックします。
  - Solaris 管理コンソール内の「拡張ストレージ」から「ディスクセット (Disk Sets)」ノードを開きます。監視するディスクセットで右マウスボタンをクリックします。メニューから「プロパティ (Properties) を選択します。詳細は、オンラインヘルプを参照してください。
  - 次の形式の `metaset` コマンドを使用して、ディスクセットの状態を表示します。

    ```
    # metaset -s diskset-name
    ```

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

    注 – ディスクセットの所有権は、ディスクセットを所有するホストだけに表示されます。

例 19–6 ディスクセットの状態をチェックする

次の例では、metaset コマンドに -s オプションとディスクセット名 blue を指定します。このコマンドの出力は、指定したディスクセットの状態情報です。出力から、host1 がディスクセットの所有者であることがわかります。metaset コマンドでは、ディスクセットに属するディスクも表示されます。

```
red# metaset -s blue


Set name = blue, Set number = 1

Host                Owner
  host1             Yes

Drive               Dbase
  c1t6d0             Yes
  c2t6d0             Yes
```

オプションを指定しないで metaset コマンドを実行すると、すべてのディスクセットの状態が表示されます。

▼ ディスクセットからディスクを削除するには

1. 次のどちらかの方法でディスクセットからディスクを削除します。
   - Solaris 管理コンソール内の「拡張ストレージ」から「ディスクセット (Disk Sets)」ノードを開きます。解放するディスクセットで右マウスボタンをクリックします。メニューから「プロパティ (Properties) を選択します。ディスク (Disks) タブをクリックします。詳細は、オンラインヘルプを参照してください。
   - 次の形式の metaset コマンドを使用して、ディスクセットからディスクを削除します。

     # metaset -s *diskset-name* **-d** *disk-name*

     -s *diskset-name* ディスクを削除するディスクセットの名前を指定します。

     -d *disk-name* ディスクセットから削除するディスクの名前です。ディスク名の形式は *cxtxdx* です。ディスクセットからディスクを削除する場合は、スライス識別子「s*x*」を指定しません。

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

2. ディスクセットからディスクが削除されたかどうかを確認します。

   # metaset -s *diskset-name*

注 – ディスクセットを削除するには、先にディスクセットからすべてのディスクを削除する必要があります。

例 19–7 ディスクセットからディスクを削除する

次の例では、ディスクセット blue からディスク c1t6d0 を削除します。

```
host1# metaset -s blue -d c1t6d0
host1# metaset -s blue

Set name = blue, Set number = 1

Host                Owner
  host1
    host2

Drive               Dbase
  c2t6d0             Yes
```

▼ ディスクセットを取得するには

注 – このオプションは複数所有者ディスクセット には使用できません。

- 次のどちらかの方法でディスクセットを取得します。
  - Solaris 管理コンソール内の「拡張ストレージ」から「ディスクセット (Disk Sets)」ノードを開きます。取得するディスクセットで右マウスボタンをクリックします。メニューから「所有権取得 (Take Ownership)」を選択します。詳細は、オンラインヘルプを参照してください。
  - 次の形式の metaset コマンドを実行します。

    ```
    # metaset -s diskset-name -t -f
    ```

    -s *diskset-name*　取得するディスクセットの名前を指定します。

    -t　ディスクセットを取得することを意味します。

    -f　ディスクセットを強制的に取得することを意味します。

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

1 時点で 1 つのホストだけがディスクセットを所有できます。ディスクセットのあるホストがそのディスクセットを取得すると、このディスクセットのほかのホストは、このディスクセットのディスクのデータにアクセスできません。

metaset コマンドはデフォルトの動作で、ディスクセットの所有権が与えられているホスト上で解放が可能な場合にかぎり、ホストにディスクセットを取得させます。ディスクセットを強制的に取得する場合は、-f オプションを指定します。このオプションを指定すると、ディスクセットは、別のホストが所有しているか否かにかかわらず、このホストによって取得されます。この方法は、通常、ディスクセットを共有するホストの 1 つが停止したり、通信していないときに使用されます。ディスクセットを強制的に取得させられた他のホストは、このディスクセットの入出力操作を試みたときにパニック状態になります。

注 – ディスクセットの所有権は、ディスクセットを所有するホストだけに表示されます。

例 19–8 ディスクセットを取得する

次の例では、ホスト host1 がホスト host2 と通信します。この通信によって、ホスト host1 がディスクセットの取得を試みる前に、ホスト host2 がディスクセットを解放することが保証されます。

```
host1# metaset
...
Set name = blue, Set number = 1

Host                Owner
  host1
  host2
...
host1# metaset -s blue -t
host2# metaset
...
Set name = blue, Set number = 1

Host                Owner
  host1                Yes
  host2
...
```

host2 がディスクセット blue を所有していた場合、前の出力の「Owner」欄はそのまま空白になります。metaset コマンドは、ホストがディスクセット所有コマンドを発行したかどうかを示すだけです。

例 19–9 ディスクセットを強制的に取得する

次の例では、ホストがほかのホストと通信しないで、ディスクセットを取得します。-f オプションによって、警告を出さないまま、ディスクセット内のディスクが強制的に取得されます。他のホストがディスクセットを所有していた場合には、そのホストが、ディスクセットの入出力操作を試みたときにパニックを引き起こします。

```
# metaset -s blue -t -f
```

▼ ディスクセットを解放するには

ディスクセットの物理ディスクを保守する場合は、ディスクセットをあらかじめ解放しておくと便利です。ディスクセットを解放すると、そのディスクセットはホストからアクセスできなくなります。ディスクセットに含まれる両方のホストがディスクセットを解放した場合、ディスクセット内のホストはどちらも、そのディスクセットで定義されているボリュームにもホットスペア集合にも直接アクセスできません。しかし、両方のホストがディスクセットを解放しても、対応する c*t*d* 名を使用することによって、ディスクには直接アクセスできます。

注 – このオプションは複数所有者ディスクセット には使用できません。

始める前に 213 ページの「ディスクセットを使用するときの指針」を確認します。

1 次のどちらかの方法でディスクセットを解放します。

- Solaris 管理コンソール内の「拡張ストレージ」から「ディスクセット (Disk Sets)」ノードを開きます。解放するディスクセットで右マウスボタンをクリックします。メニューから「所有権解放 (Release Ownership)」を選択します。詳細は、オンラインヘルプを参照してください。
- ディスクセットの所有権を解放するには、次の形式の metaset コマンドを使用します。

```
# metaset -s diskset-name -r
```

-s *diskset-name* metaset コマンドの実行対象となるディスクセットの名前を指定します。

-r ディスクセットを解放することを意味します。ディスクセット内のすべてのディスクの予約が取り消されます。ディスクセット内のボリュームにはアクセスできなくなります。

詳細は、metaset(1M) のマニュアルページを参照してください。

注–ディスクセットの所有権は、ディスクセットを所有するホストだけに表示されます。

2 このホストでディスクセットが解放されたかどうかを確認します。

```
# metaset
```

例 19–10 ディスクセットの解放

次の例では、ディスクセット blue を解放します。この時点では、ディスクセットの所有者がいないことに注意してください。ホスト host1 から状態を調べると、誤解を招くおそれがあります。これは、ホストが調査できるのは、そのホストがディスクセットを所有しているかどうかだけだからです。たとえば、ホスト host2 がディスクセットの所有権を得ようとしていた場合、ホスト host1 からは所有権を確認できません。host2 にディスクセットの所有権があることを示すのは、ホスト host2 だけです。

```
host1# metaset -s blue -r
host1# metaset -s blue

Set name = blue, Set number = 1

Host                Owner
  host1
    host2

Drive               Dbase
  c1t6d0             Yes
  c2t6d0             Yes
```

▼ ホストまたはディスクセットを削除するには

ディスクセットを削除する場合は、ディスクセットにディスクが含まれていたり、ディスクセットにほかのホストが接続していたりしてはなりません。最後のホストを削除すると、ディスクセットは削除されます。

1 次のどちらかの方法でディスクセットからホストを削除し、そのホストが最後のホストである場合は、ディスクセットも削除します。

- Solaris 管理コンソール内の「拡張ストレージ」から「ディスクセット (Disk Sets)」ノードを開きます。解放するディスクセットでマウスの右ボタンをクリックし、メニューから「削除 (Delete)」を選択します。詳細は、オンラインヘルプを参照してください。
- ホストを削除するには、次の形式の metaset コマンドを使用します。

```
metaset -s diskset-name -d -h hostname
```

`-s` *diskset-name* `metaset` コマンドの実行対象となるディスクセットの名前を指定します。

`-d` ディスクセットからホストを削除することを意味します。

`-h` *hostname* 削除するホストの名前を指定します。

同じ形式の `metaset` コマンドを使用して、ディスクセットを削除します。ディスクセットを削除する場合は、ディスクセットにディスクが含まれていたり、ほかのホストがそのディスクセットを所有していたりしてはなりません。最後のホストを削除すると、ディスクセットは削除されます。

詳細は、`metaset(1M)` のマニュアルページを参照してください。

2 `metaset` コマンドを実行して、このホストがディスクセットから削除されていることを確認します。出力には、現在の (所有者側) ホストだけが表示され、他のホストがすでに削除されていることがわかります。

```
# metaset -s disk-set
```

例 19–11 ディスクセットからホストを削除する

次の例では、ディスクセット `blue` からホスト `host2` を削除します。

```
# metaset -s blue
Set name = blue, Set number = 1

Host                Owner
  host1            Yes
..host2

Drive               Dbase
  c1t2d0             Yes
  c1t3d0             Yes
  c1t4d0             Yes
  c1t5d0             Yes
  c1t6d0             Yes
  c2t1d0             Yes

# metaset -s blue -d -h host2

# metaset -s blue
Set name = blue, Set number = 1

Host                Owner
  host1            Yes
```

```
Drive              Dbase
  c1t2d0            Yes
  c1t3d0            Yes
  c1t4d0            Yes
  c1t5d0            Yes
  c1t6d0            Yes
  c2t1d0            Yes
```

例 19–12 ディスクセットから最後のホストを削除する

次の例では、ディスクセット blue から最後のホストを削除します。

```
host1# metaset -s blue -d -h host1
host1# metaset -s blue

metaset: host: setname "blue": no such set
```

ディスクセットのインポート

metaimport 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ディスクセットをインポートできます。

▼ インポートに利用できるディスクセットのレポートを出力するには

1. スーパーユーザーになります。
2. インポートに利用できるディスクセットについてのレポートを取得します。

   ```
   # metaimport -r -v
   ```

   -r システム上にあってインポートに利用できる未構成のディスクセットのレポートを提供します。

   -v 状態データベース (metadb) の複製の場所についての詳細な情報と、システム上にあってインポートに利用できる未構成のディスクセットのディスクの状態についての詳細な情報を提供します。

例 19–13 インポートに利用できるディスクセットを報告する

次の例では、インポートに利用できるディスクセットについてのレポートを出力する方法を示します。 `metaimport` コマンドの出力では、標準ディスクセットと複製のディスクセットが区別されます。

```
# metaimport -r
# metaimport -r
Drives in regular diskset including disk c1t2d0:
   c1t2d0
   c1t3d0
 More info:
   metaimport -r -v c1t2d0
Import:   metaimport -s <newsetname> c1t2d0
 Drives in replicated diskset including disk c1t4d0:
   c1t4d0
   c1t5d0
 More info:
   metaimport -r -v c1t4d0
Import:   metaimport -s <newsetname> c1t4d0
# metaimport -r -v c1t2d0
Import: metaimport -s <newsetname> c1t2d0
Last update: Mon Dec 29 14:13:35 2003
Device       offset       length replica flags
c1t2d0           16         8192      a        u
c1t3d0           16         8192      a        u
```

▼ あるシステムから別のシステムにディスクセットをインポートするには

1. スーパーユーザーになります。
2. ディスクセットがインポートに利用できることを確認します。

   ```
   # metaimport -r -v
   ```

3. 利用できるディスクセットをインポートします。

   ```
   # metaimport -s diskset-name disk-name
   ```

   - `-s` *diskset-name* 作成するディスクセットの名前です。
   - *disk-name* インポートするディスクセットの状態データベースの複製を含むディスク (`c#t#d#`) を指定します。

4 ディスクセットがインポートされたことを確認します。

```
# metaset -s diskset-name
```

例 19–14 ディスクセットのインポート

次の例では、ディスクセットをインポートする方法を示します。

```
# metaimport -s red c1t2d0
Drives in diskset including disk c1t2d0:
  c1t2d0
  c1t3d0
  c1t8d0
More info:
  metaimport -r -v c1t2d0
# metaset -s red


Set name = red, Set number = 1

Host                Owner
  host1            Yes

Drive    Dbase

c1t2d0   Yes

c1t3d0   Yes

c1t8d0   Yes
```

◆ ◆ ◆ 第 20 章

Solaris ボリュームマネージャの保守 (作業)

この章では、Solaris ボリュームマネージャの一般的な記憶領域管理に関連する保守作業について説明します。

この章の内容は次のとおりです。

Solaris ボリュームマネージャの保守 (作業マップ)

次の表に、Solaris ボリュームマネージャの保守に必要な作業を示します。

作業	説明	参照先
Solaris ボリュームマネージャ構成の表示	Solaris ボリュームマネージャの GUI か `metastat` コマンドを使ってシステム構成を表示します。	234 ページの「Solaris ボリュームマネージャのボリューム構成を表示する」
ボリューム名の変更	Solaris ボリュームマネージャの GUI か `metarename` コマンドを使ってボリューム名を変更します。	240 ページの「ボリューム名を変更するには」
構成ファイルの作成	`metastat -p` コマンドと `metadb` コマンドを使って構成ファイルを作成します。	241 ページの「構成ファイルを作成するには」

作業	説明	参照先
構成ファイルを使用した Solaris ボリュームマネージャの初期化	metainit コマンドを使って構成ファイルから Solaris ボリュームマネージャを初期化します。	242 ページの「構成ファイルを使って Solaris ボリュームマネージャを初期化するには」
ファイルシステムの拡張	growfs コマンドを使ってファイルシステムを拡張します。	245 ページの「ファイルシステムを拡張するには」
コンポーネントの有効化	Solaris ボリュームマネージャの GUI か metareplace コマンドを使ってコンポーネントを有効にします。	247 ページの「コンポーネントの有効化」
コンポーネントの交換 (置き換え)	Solaris ボリュームマネージャの GUI か metareplace コマンドを使ってコンポーネントを交換します。(置き換える)	248 ページの「コンポーネントを他の使用可能なコンポーネントで置き換える」

Solaris ボリュームマネージャ構成の表示

ヒント – metastat の出力はソートされていません。構成リストを編集したい場合は、metastat -p コマンドの出力を入力して sort または grep コマンドを実行してください。

▼ Solaris ボリュームマネージャのボリューム構成を表示する

◗ 次のどちらかの方法でボリューム構成を表示します。

- Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」ノードを開きます。詳細は、オンラインヘルプを参照してください。
- 次の形式の metastat コマンドを使用します。

```
# metastat -p -i component-name
```

-p	要約形式で出力を表示することを指定します。この出力は、md.tab ファイルを作成するときに使用するのに適しています。
-i	RAID-1 (ミラー) ボリューム、RAID-5 ボリューム、およびホットスペアにアクセスできるかどうかを確認することを指定します。
component-name	表示するボリュームの名前を指定します。ボリューム名を指定しないと、すべてのコンポーネントが表示されます。

例 20–1 Solaris ボリュームマネージャのボリューム構成を表示する

次に、metastat コマンドの出力例を示します。

```
# metastat
d50: RAID
    State: Okay
    Interlace: 32 blocks
    Size: 20985804 blocks
Original device:
    Size: 20987680 blocks
        Device              Start Block  Dbase State        Reloc  Hot Spare
        c1t4d0s5                 330     No    Okay         Yes
        c1t5d0s5                 330     No    Okay         Yes
        c2t4d0s5                 330     No    Okay         Yes
        c2t5d0s5                 330     No    Okay         Yes
        c1t1d0s5                 330     No    Okay         Yes
        c2t1d0s5                 330     No    Okay         Yes

d1: Concat/Stripe
    Size: 4197879 blocks
    Stripe 0:
        Device              Start Block  Dbase  Reloc
        c1t2d0s3                   0     No     Yes

d2: Concat/Stripe
    Size: 4197879 blocks
    Stripe 0:
        Device              Start Block  Dbase  Reloc
        c2t2d0s3                   0     No     Yes


d80: Soft Partition
    Device: d70
    State: Okay
    Size: 2097152 blocks
        Extent              Start Block              Block count
             0                      1                  2097152

d81: Soft Partition
    Device: d70
    State: Okay
    Size: 2097152 blocks
        Extent              Start Block              Block count
             0                2097154                  2097152

d70: Mirror
```

```
    Submirror 0: d71
      State: Okay
    Submirror 1: d72
      State: Okay
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Size: 12593637 blocks

d71: Submirror of d70
    State: Okay
    Size: 12593637 blocks
    Stripe 0:
        Device              Start Block  Dbase State        Reloc  Hot Spare
        c1t3d0s3                   0     No    Okay         Yes
    Stripe 1:
        Device              Start Block  Dbase State        Reloc  Hot Spare
        c1t3d0s4                   0     No    Okay         Yes
    Stripe 2:
        Device              Start Block  Dbase State        Reloc  Hot Spare
        c1t3d0s5                   0     No    Okay         Yes


d72: Submirror of d70
    State: Okay
    Size: 12593637 blocks
    Stripe 0:
        Device              Start Block  Dbase State        Reloc  Hot Spare
        c2t3d0s3                   0     No    Okay         Yes
    Stripe 1:
        Device              Start Block  Dbase State        Reloc  Hot Spare
        c2t3d0s4                   0     No    Okay         Yes
    Stripe 2:
        Device              Start Block  Dbase State        Reloc  Hot Spare
        c2t3d0s5                   0     No    Okay         Yes

hsp010: is empty

hsp014: 2 hot spares
        Device              Status      Length          Reloc
        c1t2d0s1            Available    617652 blocks  Yes
        c2t2d0s1            Available    617652 blocks  Yes

hsp050: 2 hot spares
        Device              Status      Length          Reloc
        c1t2d0s5            Available    4197879 blocks Yes
        c2t2d0s5            Available    4197879 blocks Yes
```

```
hsp070: 2 hot spares
        Device              Status      Length          Reloc
        c1t2d0s4            Available    4197879 blocks Yes
        c2t2d0s4            Available    4197879 blocks Yes

Device Relocation Information:
Device              Reloc       Device ID
c1t2d0              Yes         id1,sd@SSEAGATE_ST39204LCSUN9.0G3BV0N1S200002103AF29
c2t2d0              Yes         id1,sd@SSEAGATE_ST39204LCSUN9.0G3BV0P64Z00002105Q6J7
c1t1d0              Yes         id1,sd@SSEAGATE_ST39204LCSUN9.0G3BV0N1EM00002104NP2J
c2t1d0              Yes         id1,sd@SSEAGATE_ST39204LCSUN9.0G3BV0N93J000071040L3S
c0t0d0              Yes         id1,dad@s53554e575f4154415f5f535433393134304125254159333
```

例 20–2 マルチテラバイトのSolaris ボリュームマネージャボリュームを表示する

次に、マルチテラバイトの記憶ボリューム (11T バイト) に対する metastat コマンドの出力例を示します。

```
# metastat d0
 d0: Concat/Stripe
    Size: 25074708480 blocks (11 TB)
    Stripe 0: (interlace: 32 blocks)
        Device      Start Block  Dbase  Reloc
        c27t8d3s0           0     No     Yes
        c4t7d0s0        12288     No     Yes
    Stripe 1: (interlace: 32 blocks)
        Device      Start Block  Dbase  Reloc
        c13t2d1s0       16384     No     Yes
        c13t4d1s0       16384     No     Yes
        c13t6d1s0       16384     No     Yes
        c13t8d1s0       16384     No     Yes
        c16t3d0s0       16384     No     Yes
        c16t5d0s0       16384     No     Yes
        c16t7d0s0       16384     No     Yes
        c20t4d1s0       16384     No     Yes
        c20t6d1s0       16384     No     Yes
        c20t8d1s0       16384     No     Yes
        c9t1d0s0        16384     No     Yes
        c9t3d0s0        16384     No     Yes
        c9t5d0s0        16384     No     Yes
        c9t7d0s0        16384     No     Yes
    Stripe 2: (interlace: 32 blocks)
        Device      Start Block  Dbase  Reloc
        c27t8d2s0       16384     No     Yes
```

```
    c4t7d1s0       16384     No     Yes
Stripe 3: (interlace: 32 blocks)
    Device      Start Block  Dbase  Reloc
    c10t7d0s0      32768     No     Yes
    c11t5d0s0      32768     No     Yes
    c12t2d1s0      32768     No     Yes
    c14t1d0s0      32768     No     Yes
    c15t8d1s0      32768     No     Yes
    c17t3d0s0      32768     No     Yes
    c18t6d1s0      32768     No     Yes
    c19t4d1s0      32768     No     Yes
    c1t5d0s0       32768     No     Yes
    c2t6d1s0       32768     No     Yes
    c3t4d1s0       32768     No     Yes
    c5t2d1s0       32768     No     Yes
    c6t1d0s0       32768     No     Yes
    c8t3d0s0       32768     No     Yes
```

次の作業

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

ボリューム名の変更

ボリューム名を変更するための背景情報

Solaris ボリュームマネージャでは、多少の制約はありますが、ほとんどのタイプのボリューム名をいつでも変更できます。 ボリューム名の変更は、Solaris 管理コンソール内の「拡張ストレージ」かコマンド行 (`metarename(1M)` コマンド) から行うことができます。

ボリューム名の変更や交換は、ボリューム名を管理する上で便利な機能です。たとえば、ファイルシステムのすべてのマウントポイントをある範囲の数値内に収めることができます。あるいは、論理ボリュームの命名規則に従ってボリューム名を変更したり、トランザクションボリュームの名前にそのボリュームを構成しているボリュームと同じ名前を使用したりすることができます。

注 – トランザクションボリュームは、Solaris ボリュームマネージャでは使用できなくなりました。トランザクションボリュームの名前を変更して置き換えることができます。

ボリューム名を変更するときは、ボリュームが使用中でないことを確認してください。ファイルシステムとして使用されている場合は、マウントされていたり、swap として使用されていないことを確認してください。データベースなど、raw デバイスを使用するその他のアプリケーションは、独自の方法でデータへのアクセスを停止できる必要があります。

ボリューム名の変更に伴う特別な考慮事項は、次のとおりです。

- 次のものを除く任意のボリュームの名前を変更できます。
  - ソフトパーティション
  - ソフトパーティションが直接作成されているボリューム
  - ログデバイスとして使用されているボリューム
  - ホットスペア集合
- ディスクセット内のボリュームの名前は変更できます。しかし、そのボリュームを別のディスクセットに移動させることはできません。

ボリューム名の交換

metarename コマンドに -x オプションを指定すると、親子関係のあるボリュームの名前が交換されます。詳細は、240 ページの「ボリューム名を変更するには」と metarename(1M) のマニュアルページを参照してください。既存のボリュームの名前がサブコンポーネントの 1 つと交換されます。たとえば、ミラーとそのサブミラーの 1 つで、このタイプの交換を実行できます。metarename -x コマンドを使用すると、既存のボリュームのミラー化またはミラー化解除が簡単にできます。

注 – ボリューム名の交換には、コマンド行を使用する必要があります。この機能は、現在のところ Solaris ボリュームマネージャの GUI にはありません。ただし、ボリューム名の変更はコマンド行からでも GUI からでも行えます。

ボリューム変更するときには、次の指針を考慮してください。

- 使用されているボリュームの名前は変更できません。この制約は、マウントされているファイルシステム、 swap として使用されているボリューム、アプリケーションやデータベースのアクティブ記憶領域として使用されているボリュームなどに当てはまります。したがって、metarename コマンドを使用する前に、名前を変更するボリュームへのすべてのアクセスを停止する必要があります。たとえば、マウントされているファイルシステムのマウントを解除します。
- 障害が発生している状態のボリュームは交換できません。
- ホットスペア交換を使用しているボリュームは交換できません。
- 交換するボリュームは、親子関係のあるものでなければなりません。
- ログデバイスの交換 (または名前の変更) を行うことはできません。この問題を回避するには、ログデバイスを切り離し、適切な名前の別のログデバイスを接続します。

- 交換できるのはボリュームだけです。スライスやホットスペアは交換できません。

▼ ボリューム名を変更するには

始める前に　ボリューム名の要件 (45 ページの「ボリューム名」) と 238 ページの「ボリューム名を変更するための背景情報」 を確認します。

1. このボリュームを使用するファイルシステムのマウントを解除します。

   ```
   # umount /filesystem
   ```

2. 次のどちらかの方法でボリューム名を変更します。

   - Solaris 管理コンソール内の「拡張ストレージ」から「ボリューム (Volumes)」を開きます。名前を変更するボリュームを選択します。アイコンを右マウスクリックします。「プロパティ (Properties)」オプションを選択します。画面の指示に従います。詳細は、オンラインヘルプを参照してください。
   - 次の形式の `metarename` コマンドを実行します。

     ```
     # metarename old-volume-name new-volume-name
     ```

     old-volume-name　既存ボリュームの名前を指定します。

     new-volume-name　既存のボリュームに新しい名前を指定します。

     詳細については、`metarename(1M)` コマンドのマニュアルページを参照してください。

3. 必要であれば、エントリが新しいボリューム名を参照するように `/etc/vfstab` ファイルを編集します。

4. ファイルシステムを再びマウントします。

   ```
   # mount /filesystem
   ```

例 20–3　ファイルシステムとして使用されているボリュームの名前を変更する

次の例では、ボリュームの名前 `d10` を `d100` に変更します。

```
# umount /home
# metarename d10 d100
d10: has been renamed to d100
    (ファイルシステムが新しいボリュームを参照するように /etc/vfstab ファイルを編集する)
# mount /home
```

d10 にはマウントされたファイルシステムがあるので、ファイルシステムをマウント解除してからでなければ、ボリューム名を変更できません。このファイルシステムのエントリが /etc/vfstab ファイル内に存在している場合は、そのエントリが新しいボリューム名を参照するように変更します。

たとえば、/etc/vfstab ファイルにファイルシステム用の次のエントリが指定されているとします。

```
/dev/md/dsk/d10 /dev/md/rdsk/d10 /docs home 2 yes -
```

次のようにエントリを変更します。

```
/dev/md/dsk/d100 /dev/md/rdsk/d100 /docs home 2 yes -
```

その後、ファイルシステムを再びマウントします。

既存のミラーやトランザクションボリュームが存在する場合、metarename -x コマンドでミラーやトランザクションボリュームを削除しても、そのボリュームを構成するボリュームのデータは保持できます。たとえば、トランザクションボリュームの場合には、マスターデバイスが ボリューム (RAID 0、RAID 1、または RAID 5 ボリュームのいずれか) である限り、そのボリュームのデータは保持されます。

構成ファイルの使用

Solaris ボリュームマネージャの構成ファイルには、Solaris ボリュームマネージャの基本情報の他に、構成を再設定するのに必要なほとんどのデータが含まれています。次の手順で、構成ファイルに関連する作業について説明します。

▼ 構成ファイルを作成するには

◗ **Solaris ボリュームマネージャ環境用のすべてのパラメータを適切に設定したら、metastat -p コマンドで /etc/lvm/md.tab ファイルを作成します。**

```
# metastat -p > /etc/lvm/md.tab
```

このファイルには、metainit コマンドと metahs コマンドが使用するすべてのパラメータが含まれています。このファイルは、類似した環境をいくつも用意しなければならない場合、またはシステム障害後に構成を再作成する場合に使用します。

md.tab ファイルの詳細については、326 ページの「md.tab ファイルの概要」 と md.tab(4) のマニュアルページを参照してください。

▼ 構成ファイルを使って Solaris ボリュームマネージャを初期化するには

注意 – この手順は、次の状況で使用します。

- Solaris ボリュームマネージャの構成がすべて失われた場合
- 構成がまだなくて、保存されている構成ファイルから構成を作成しなければならない場合

状態データベースで維持していた情報が失われることがあります。たとえば、状態データベースの複製をすべて削除したあとで、システムをリブートすると、この状況が生じる場合があります。状態データベースの消失後、ボリュームがまったく作成されていない場合は、`md.cf` ファイルまたは `md.tab` ファイルを使用して、Solaris ボリュームマネージャの構成を回復できます。

注 – `md.cf` ファイルには、アクティブなホットスペアの情報は格納されません。そのため、Solaris ボリュームマネージャ構成が失われたときにホットスペアが使用されていると、アクティブなホットスペアを使用していたボリュームの内容は破壊されていることがあります。

これらのファイルの詳細については、`md.cf(4)` と `md.tab(4)` のマニュアルページを参照してください。

1. 状態データベースの複製を作成します。
   詳細は、72 ページの「状態データベースの複製の作成」を参照してください。

2. `/etc/lvm/md.tab` ファイルを作成するか更新します。
   - 最後の状態でSolaris ボリュームマネージャの構成を回復する場合は、`md.cf` ファイルを `/etc/lvm/md.tab` ファイルにコピーします。
   - 保存されている `md.tab` ファイルのコピーを使って新しい Solaris ボリュームマネージャ構成を作成する場合は、保存されているファイルを `/etc/lvm/md.tab` ファイルにコピーします。

3. 「新しい」 `/etc/lvm/md.tab` ファイルを編集して、次の作業を行います。
   - 新しい構成を作成している場合や、クラッシュの後で構成を回復している場合には、ミラーを 1 面ミラーとして構成します。たとえば、次のように指定します。

```
d80 -m d81 1
d81 1 1 c1t6d0s3
```

ミラーの各サブミラーのサイズが同じでない場合は、もっとも小さいサブミラーがこの 1 面ミラーに使用されるようにします。そうしないと、データが失われるおそれがあります。

- 既存の構成を回復している場合、Solaris ボリュームマネージャが正常に終了しているのであれば、ミラー構成を多面ミラーのままにして回復します。たとえば、次のように指定します。

```
d70 -m d71 d72 1
d71 1 1 c1t6d0s2
d72 1 1 c1t5d0s0
```

- RAID-5 ボリュームの場合は、デバイスの再初期化を防止するために -k オプションを指定します。たとえば、次のように指定します。

```
d45 -r c1t3d0s5 c1t3d0s3 c1t3d0s4 -k -i 32b
```

詳細は、metainit(1M) のマニュアルページを参照してください。

4 次の形式の metainit コマンドを使用して、変更をコミットすることなく、/etc/lvm/md.tab ファイルエントリの構文を調べます。

```
# metainit -n md.tab-entry
```

```
# metainit -n -a
```

metainit コマンドに -n オプションを付けて実行した場合、同じ実行中にすでに仮想的に作成されているデバイスを記憶しません。このため、md.tab 内に記述された別のボリュームに依存するボリュームを作成しようとすると、-n オプションを指定しなければ正常に実行される場合でも -n オプションを指定するとエラーになることがあります。

-n	デバイスを実際には作成しないことを指定します。このオプションは、予期した結果が得られるかどうかを確認する場合に使用します。
md.tab-entry	初期化するコンポーネントの名前を指定します。
-a	すべてのコンポーネントをチェックすることを指定します。

5 前の手順で特に問題がなければ、md.tab ファイルを使ってボリュームとホットスペア集合を作成し直します。

```
# metainit -a
```

-a /etc/lvm/md.tab file のエントリをアクティブにすることを指定します。

6 必要であれば、metattach コマンドを使用して、**1** 面ミラーを多面ミラーに変更します。

```
# mettach mirror submirror
```

7 ボリューム上のデータを検証し、構成が正しく再作成されたかどうかを確認します。

```
# metastat
```

Solaris ボリュームマネージャのデフォルト値の変更

Solaris 10 リリースで、Solaris ボリュームマネージャはボリュームを動的に構成するように拡張されました。/kernel/drv/md.conf ファイルの nmd パラメータと md_nsets パラメータを編集しなくてすみます。新しいボリュームは必要に応じて作成されます。

Solaris ボリュームマネージャ構成の最大値は同じです。

- サポートされるボリュームの最大数は 8192 です。
- サポートされるディスクセットの最大数は 32 です。

growfs コマンドによるファイルシステムの拡張

UFS ファイルシステムが含まれているボリュームを拡張 (つまり、領域を追加) した場合は、追加した領域が認識されるように、ファイルシステムを拡張することも必要です。ファイルシステムの拡張は、growfs コマンドを使って手動で行う必要があります。growfs コマンドは、ファイルシステムがマウントされている場合でも拡張できます。ただし、growfs コマンドの実行中は、ファイルシステムに書き込みアクセスを行うことはできません。

データベースなど、raw デバイスを使用するアプリケーションは、独自の方法で追加された領域を組み込むことができなければなりません。Solaris ボリュームマネージャには、この機能はありません。

growfs コマンドはファイルシステムを拡張するとき、マウントされているファイルシステムを「書き込みロック」します。ファイルシステムが書き込みロックされている時間を短縮する必要がある場合は、ファイルシステムを段階的に拡張できます。たとえば、1G バイトのファイルシステムを 2G バイトに拡張する場合、-s オプションを使用することによって、16M バイト単位でファイルシステムを拡張できます。このオプションでは、ステージごとの新しいファイルシステムの総容量を指定します。

拡張処理の間は書き込みロックが機能するので、このファイルシステムへの書き込みは実行できません。書き込みアクセスは growfs コマンドがファイルシステムのロックを解除するまで自動的に中断され、ロックが解除されると再開されます。読み取りアクセスは影響を受けません。しかし、ロックが有効な間、アクセス時間は保証されません。

スライスやボリュームを拡張するための背景情報

注 – Solaris ボリュームマネージャのボリュームは拡張可能です。ただし、ボリュームを縮小することはできません。

- ボリュームはファイルシステム用であるか、アプリケーション用であるか、それともデータベース用であるかに関係なく、拡張できます。RAID-0 (ストライプ方式と連結方式) ボリューム、RAID-1 (ミラー) ボリューム、RAID-5 ボリュームだけではなく、ソフトパーティションも拡張できます。
- ファイルシステムが格納されているボリュームを連結できます。UFS ファイルシステムの場合、(`growfs` コマンドを使用して) より大きい領域を満たすように拡張できます。データに対する読み取りアクセスを妨げることなく、ファイルシステムを拡張できます。
- 拡張したファイルシステムは、UFS ファイルシステムの制約により縮小できません。
- raw デバイスを使用するアプリケーションやデータベースは、独自の方法で領域を拡張し、それを認識できなければなりません。Solaris ボリュームマネージャには、この機能はありません。
- RAID-5 ボリュームにコンポーネントを追加すると、コンポーネントはボリュームへの連結になります。新しいボリュームにはパリティー情報は格納されませんが、このコンポーネントのデータは、ボリュームに対して行われる全体的なパリティー計算によって保護されます。
- コンポーネントを追加することによってログデバイスを拡張できます。リブート時に Solaris ボリュームマネージャは新しい領域を自動的に認識するため、`growfs` コマンドを実行する必要はありません。
- ソフトパーティションを拡張するには、そのパーティションを構成するボリュームまたはスライスから領域を追加します。その他のボリュームはすべて、スライスを追加することによって拡張できます。

▼ ファイルシステムを拡張するには

始める前に 49 ページの「Solaris ボリュームマネージャコンポーネントを作成するための前提条件」を確認します。

1 ファイルシステムに対応付けられたディスク領域を確認します。

```
# df -k
```

詳細については、`df(1M)` のマニュアルページを参照してください。

2 **UFS ファイルシステムを論理ボリュームで拡張します。**

```
# growfs -M /mount-point /dev/md/rdsk/volume-name
```

`-M` *`/mount-point`*	拡張するファイルシステムのマウントポイントを指定します。
`/dev/md/rdsk/`*`volume-name`*	拡張するボリュームの名前を指定します。

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

例 20–4 ファイルシステムを拡張する

次の例では、新しいスライスがボリューム d10 にすでに追加されています。このボリュームには、マウントされているファイルシステム /home2 があります。growfs コマンドでは、-M オプションを使ってマウントポイントに /home2 を指定し、これを raw ボリューム /dev/md/rdsk/d10 上に拡張します。growfs コマンドが終了すると、このファイルシステムはボリューム全体を占めます。ファイルシステムを拡張する前後に df -k コマンドを使用すると、ディスクの総容量を確認できます。

```
# df -k
Filesystem            kbytes    used   avail capacity  Mounted on
...
/dev/md/dsk/d10        69047   65426       0   100%    /home2
...
# growfs -M /home2 /dev/md/rdsk/d10
/dev/md/rdsk/d10:       295200 sectors in 240 cylinders of 15 tracks, 82 sectors
        144.1MB in 15 cyl groups (16 c/g, 9.61MB/g, 4608 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 19808, 39584, 59360, 79136, 98912, 118688, 138464, 158240, 178016, 197792,
 217568, 237344, 257120, 276896,
# df -k
Filesystem            kbytes    used   avail capacity  Mounted on
...
/dev/md/dsk/d10       138703   65426   59407    53%    /home2
...
```

ミラーボリュームの場合、growfs コマンドは常に、最上位のボリュームで実行します。サブミラーまたはマスターデバイスに領域を追加する場合であっても、サブミラーまたはマスターデバイスに対してコマンドを実行してはなりません。

RAID-1 および RAID-5 ボリューム内のコンポーネントの交換と有効化の概要

Solaris ボリュームマネージャは、RAID-1 (ミラー) および RAID-5 ボリューム内のコンポーネントを 交換したり、有効にしたりできます。

Solaris ボリュームマネージャでは、コンポーネントの交換は、サブミラーまたは RAID-5 ボリューム内のコンポーネントをシステム上で使用可能な他のコンポーネントと交換することを意味します。このプロセスは、コンポーネントの物理的な交換ではなく論理的な交換です。詳細については、248 ページの「コンポーネントを他の使用可能なコンポーネントで置き換える」を参照してください。

コンポーネントの有効化とは、コンポーネントをアクティブにする (または、コンポーネントをそれ自身で置き換える) ことを意味します。したがって、コンポーネント名は変わりません。詳細については、247 ページの「コンポーネントの有効化」を参照してください。

注 – ディスクエラーから回復するときは、`/var/adm/messages` を調べ、どのような種類のエラーが発生したかを確認してください。エラーが一時的なものであり、ディスク自体に問題がない場合は、コンポーネントを有効にしてみます。また、`format` コマンドを使ってディスクをテストすることもできます。

コンポーネントの有効化

コンポーネントを有効にする必要があるのは次のような場合です。

- Solaris ボリュームマネージャが物理ディスクにアクセスできない場合。この問題は、たとえば、電源が切断されていたり、ドライブケーブルが外れていることが原因で発生します。この場合、Solaris ボリュームマネージャは、コンポーネントを「保守 (Maintenance)」状態に変更します。この場合には、電源を復旧したり、ケーブルを接続し直したりしてドライブを使用可能にしてから、ボリュームのコンポーネントを有効にする必要があります。
- 物理ディスクに、ディスクに関連しない一時的な障害があると考えられる場合。コンポーネントが「保守 (Maintenance)」状態であれば、そのコンポーネントを有効にするだけでコンポーネントを修復できる場合があります。コンポーネントを有効にしても問題が解決しない場合は、次のどちらかの作業が必要です。
  - ディスクドライブを物理的に交換してコンポーネントを有効にする
  - コンポーネントをシステム上で使用可能な別のコンポーネントと交換する

  ディスクを物理的に置き換える場合は、新しいディスクを前のものと同じようにパーティション分割し、各コンポーネント上に十分な領域を確保する必要があります。

注 – 交換対象のディスクに状態データベースの複製やホットスペアがないか必ずチェックしてください。「エラー (erred)」状態の状態データベースの複製を削除してから、ディスクを交換する必要があります。さらに、コンポーネントを有効にしてから、同じサイズで状態データベースの複製を作成し直します。ホットスペアについても同じように処理してください。

コンポーネントを他の使用可能なコンポーネントで置き換える

既存のコンポーネントを他の未使用のコンポーネントで置き換える (または交換する) には、`metareplace` コマンドを使用します。

このコマンドを使用できるのは、次のような場合です。

- ディスクドライブに問題があるが、交換用ドライブがない場合で、なおかつ、システム上に使用可能なコンポーネントがある場合。

  どうしても交換する必要があるが、システムを停止したくない場合には、この方法を使用します。
- 物理ディスク上でソフトエラーが発生した場合。

  ミラー、サブミラー、または RAID-5 ボリュームの状態が、Solaris ボリュームマネージャによれば「正常 (Okay)」であるにもかかわらず、物理ディスク上でソフトエラーが報告される場合があります。問題のコンポーネントを他の使用可能なコンポーネントで置き換えると、ハードウェアエラーの発生を防止するための予防的保守を実行できます。
- 性能の調整を行いたい場合。

  コンポーネントを評価する 1 つの方法は、Solaris 管理コンソール内の「拡張ストレージ」から性能監視機能を使用することです。たとえば、RAID-5 ボリュームの特定のコンポーネントが「正常 (Okay)」状態であっても、平均負荷が上昇しているといったことがわかります。その場合には、ボリュームの負荷を均一化するために、このコンポーネントを、これより使用率の低いディスクのコンポーネントで置き換えることができます。このような処理は、ボリュームへのサービスを中断せずにオンラインで行うことができます。

「保守 (Maintenance)」状態と「最後にエラー (Last Erred)」状態

RAID-1 または RAID-5 ボリュームのコンポーネントでエラーが発生すると、Solaris ボリュームマネージャはそのコンポーネントを「保守 (Maintenance)」状態にします。これ以降、「保守 (Maintenance)」状態のコンポーネントには読み書きは実行されません。

コンポーネントは「最後にエラー (Last Erred)」状態になることもあります。RAID-1 ボリュームの場合、このエラーは通常、1 面ミラーで発生します。ボリュームにエラーが発生したとします。しかし、読み取り対象となる冗長コンポーネントはありません。RAID-5 ボリュームでは、あるコンポーネントが「保守 (Maintenance)」状態になり、かつ、別のコンポーネントで障害が発生した場合に、この状態になります。障害の発生した 2 番目のコンポーネントが、「最後にエラー (Last Erred)」状態になります。

RAID-1 ボリュームまたは RAID-5 ボリュームのどちらかに「最後にエラー (LastErred)」状態のコンポーネントがあっても、入出力は引き続き、その「最後にエラー (LastErred)」状態のコンポーネントに対して試行されます。この入出力が試行されるのは、「最後にエラー (Last Erred)」状態のコンポーネントにデータの最後の正常なコピーが含まれていると、Solaris ボリュームマネージャが判断するためです。「最後にエラー (Last Erred)」状態のコンポーネントを含むボリュームは正常なデバイス (ディスク) のように動作し、アプリケーションに入出力エラーを返します。通常、この時点では、データの一部がすでに失われています。

同じボリューム内の他のコンポーネントで次のエラーが発生した場合、そのエラーの処理方法は、ボリュームのタイプによって異なります。

RAID-1 ボリューム
: RAID-1 ボリュームでは、多数のコンポーネントが「保守 (Maintenance)」状態になっても、読み取りや書き込みを継続できることがあります。コンポーネントが「保守 (Maintenance)」状態である場合、データは失われていません。コンポーネントをどのような順序で置き換え、有効にしてもかまいません。コンポーネントが「最後にエラー (Last Erred)」状態の場合は、「保守 (Maintenance)」状態のコンポーネントを置き換えてから、このコンポーネントを置き換えます。「最後にエラー (Last Erred)」状態のコンポーネントを交換したり有効にすることは、通常、一部のデータがすでに失われていることを意味します。ミラーを修復したら、必ずデータを検証してください。

RAID-5 ボリューム
: RAID-5 ボリュームは、「保守 (Maintenance)」状態のコンポーネントが 1 つであれば許容できます。「保守 (Maintenance)」状態のコンポーネントが 1 つの場合、そのコンポーネントはデータを失うことなく安全に交換できます。ほかのコンポーネントに障害が発生すると、そのコンポーネントは「最後にエラー (Last Erred)」状態になります。この時点で RAID-5 ボリュームは読み取り専用になります。必要な回復処置を行なって、RAID-5 ボリュームを安定状態にし、データ損失の可能性を減らします。RAID-5 ボリュームが「最後にエラー (Last Erred)」状態の場合には、データがすでに失われている可能性が高くなります。RAID-5 ボリュームを修復したあと、必ずデータを検証してください。

必ず「保守 (Maintenance)」状態のコンポーネントを先に交換してから、「最後にエラー (Last Erred)」状態のコンポーネントを交換します。コンポーネントの交換と再同期の実行後、`metastat` コマンドを使用してコンポーネントの状態を確認します。さらに、データを検証します。

RAID-1 および RAID-5 ボリューム内のコンポーネントを交換または有効にするための背景情報

RAID-1 ボリュームまたは RAID-5 ボリュームのコンポーネントを交換する場合は、次の指針に従ってください。

- 必ず「保守 (Maintenance)」状態のコンポーネントを先に交換してから、「最後にエラー (Last Erred)」状態のコンポーネントを交換します。
- コンポーネントの交換と再同期の実行後、`metastat` コマンドを使用してボリュームの状態を確認します。さらに、データを検証します。「最後にエラー (Last Erred)」状態のコンポーネントを交換したり有効にすることは、通常、一部のデータがすでに失われていることを意味します。ボリュームの修復後に、そのデータが正しいことを必ず確認してください。UFS の場合は、`fsck` コマンドを実行して、「メタデータ (ファイルシステムの構造)」を検証します。さらに、実際のユーザーデータを確認します(実際には、ユーザーが各自のファイルを確認する必要があります)。データベースなどのアプリケーションは、独自の方法で内部のデータ構造を検証できなければなりません。
- コンポーネントを交換する場合は、状態データベースの複製やホットスペアが存在していないか必ずチェックします。「エラー (erred)」状態の状態データベースの複製を削除してから、物理ディスクを交換する必要があります。そして、コンポーネントを有効にする前に、この状態データベースの複製を追加してください。ホットスペアの場合も同じ処理が必要です。
- RAID-5 ボリュームでは、コンポーネントの交換時に次のどちらかの方法でデータが回復されます。現在使用中のホットスペアから、またはホットスペアが使用されていない場合は RAID-5 パリティーを使用して、データが回復されます。
- RAID-1 ボリュームでコンポーネントを交換した場合は、Solaris ボリュームマネージャがボリュームの他の部分に対して新しいコンポーネントの再同期を自動的に開始します。再同期が終了すると、新しいコンポーネントは読み書きが可能になります。障害の発生したコンポーネントがホットスペアのデータで置き換えられると、そのホットスペアが「使用可能 (Available)」状態になり、ほかのホットスペア交換に使用できるようになります。
- 新しいコンポーネントのサイズは、古いコンポーネントを置き換えられるだけの大きさでなければなりません。
- 「最後にエラー (Last Erred)」状態のデバイスを交換する場合は、用心のためにすべてのデータのバックアップを取ってください。

◆ ◆ ◆ 第 21 章

Solaris ボリュームマネージャで構築可能な最善の記憶装置構成

この章では、Solaris ボリュームマネージャを使用して構築できる最善の記憶装置構成について、現実的な構成例を示して説明します。まず一般的な構成について説明し、次にその構成の分析を行い、最後に要件にもっとも合った推奨 (最善の) 構成を示します。

この章では、次の内容について説明します。

小規模なサーバーを運用する場合の構成例

分散コンピューティング環境では、複数の場所に類似または同一のサーバーを配置しなければならないことがよくあります。このような環境としては ISP、地理的に分散している営業所、電気通信のサービスプロバイダなどが挙げられます。分散コンピューティング環境に配置されたサーバーは、次のようなサービスを提供します。

- ルーターまたはファイアウォールサービス
- 電子メールサービス
- DNS キャッシュ
- Usenet (ネットワークニュース) サーバー
- DHCP サービス
- その他、さまざまな場所で最適な形で提供されるサービス

これらの小規模なサーバーは、次のような共通する要件を満たす必要があります。

- 高い信頼性
- 高い可用性
- 汎用性と性能に優れたハードウェア

最初の例として、SCSI バスを 1 つ、内蔵ディスクを 2 つ備えた Netra™ サーバーを取り上げます。これはそのまま使用できる構成なので、分散型サーバーの出発点に適しています。Solaris ボリュームマネージャを使えば、一部またはすべてのスライスをミラー化

し、冗長記憶領域を構成することにより、ディスク障害に対する保護機能を簡単に強化できます。次の図に、この小規模サーバー構成の例を示します。

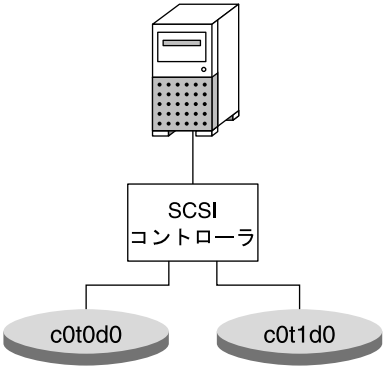

図 21–1 小規模なシステム構成

この構成では、ルート (/)、/usr、swap、/var および、/export ファイルシステムに加え、状態データベースの複製 (ディスクごとに 1 つ) をミラー化できます。したがって、ミラーの片側で障害が発生しても、必ずしもシステム障害にはつながりません。また、最大 5 つの個別の障害に耐えられます。しかし、このシステムは、ディスクやスライスの障害に対して十分に保護されているとはいえません。さまざまな潜在的な障害によって致命的なシステム障害が引き起こされ、オペレータの介入が必要になる場合があります。

この構成は致命的なディスク障害に対してある程度の保護機能を備えていますが、次のような重大な単一点障害が存在します。

- 1 つの SCSI コントローラが単一点障害の原因となり得ます。コントローラに障害が発生すると、そのコントローラが交換されるまでシステムは停止します。
- 2 つのディスクでは、状態データベースの複製の分散という観点からは十分とはいえません。多数決アルゴリズムでは、状態データベースの複製の半数が使用可能であることがシステム動作継続の条件になります。このアルゴリズムではさらに、過半数 (半数プラス 1) の複製がリブートの条件です。したがって、各ディスクに 1 つずつ状態データベースの複製があり、1 つのディスクまたは複製が格納されているスライスで障害が発生した場合は、システムをリブートできません。その結果、ミラー化されたルート (/) ファイルシステムが無効になります。各ディスクに状態データベースの複製が 2 つ以上ある場合は、1 つのスライスで障害が発生しても、問題になることはほとんどありません。ただし、ディスク障害の場合はやはり、リブートできません。各ディスクに異なる数の複製がある場合には、一方のディスクに半数を超える数の複製が、他方のディスクには半数未満の複製が存在します。複製の数が少ない方のディスクに障害が発生しても、システムはリブートし、動作を続けられますが、複製の数が多い方のディスクに障害が発生すると、システムはただちにパニック状態になります。

結論として、このシステムについては、1つのコントローラと1つのハードドライブを追加した構成が「最善の構成」となります。このように構成を修正すると、耐障害性が大幅に向上します。

ネットワーク接続された記憶装置に対するSolarisボリュームマネージャの使用方法

Solarisボリュームマネージャは、ネットワーク接続された記憶装置、特に、構成可能なRAIDレベルと柔軟なオプションを備えた記憶装置に対しても使用できます。通常、Solarisボリュームマネージャとこのような装置を組み合わせると、一方だけの場合よりも優れた性能と柔軟性が得られます。

通常、ハードウェアでRAID-1、RAID-5ボリュームなどの冗長性を提供する記憶デバイス上には、Solarisボリュームマネージャの RAID-5ボリュームを設定しないでください。よほど特殊な状況でもないかぎり、性能が低下します。また、冗長性や可用性向上の面でも、メリットはほとんどありません。

一方、ハードウェアでRAID-5ボリュームを構成する記憶デバイスは、きわめて有効です。Solarisボリュームマネージャボリュームに最適な土台が得られます。ハードウェアRAID-5によって、Solarisボリュームマネージャの RAID-1ボリューム、ソフトパーティション、またはその他のボリュームの冗長性が強化されます。

注 – 類似したソフトウェアデバイスとハードウェアデバイスをいっしょに構成しないようにします。たとえば、ハードウェアRAID-1デバイス上にソフトウェアRAID-1ボリュームを作成しないでください。類似したデバイスを使用すると、性能が低下し、信頼性が向上することもありません。

構成要素であるハードウェア記憶デバイスで構築されたSolarisボリュームマネージャのRAID-1ボリュームは、RAID-1+0ではありません。Solarisボリュームマネージャ は構成要素の記憶装置を十分に認識できないので、RAID-1+0の機能は提供しません。

ハードウェアRAID-5デバイスにSolarisボリュームマネージャのRAID-1ボリュームを作成し、さらにその上にソフトパーティションを作成すると、柔軟で耐障害性に優れた構成になります。

◆ ◆ ◆ 第 22 章

ボリュームのトップダウン作成 (概要)

この章では、Solaris ボリュームマネージャにおけるボリュームのトップダウン作成に関連する概念について説明します。

この章では、次の内容について説明します。

関連作業の実行手順については、第 23 章を参照してください。

ボリュームのトップダウン作成の概要

ボリュームのトップダウン作成では、`metassist` コマンドを使用して Solaris ボリュームマネージャのボリューム構成を自動的に作成できます。ディスクのパーティション分割、RAID-0 ボリュームの (サブミラーとしての) 作成、ホットスペア集合とホットスペアの作成からミラーの作成に至るまでの作業を手動で行わなくてすむようになります。代わりに、`metassist` コマンドを使用すればボリュームを作成できます。Solaris ボリュームマネージャがユーザーの代わりに残りの作業を実行します。

`metassist` コマンドを使用すれば、1 つのコマンドで Solaris ボリュームマネージャのボリューム構成を作成できます。サービス品質の面からボリューム特性を指定できます。サービス品質特性によって、ボリュームで使用するハードウェアコンポーネントを指定することなく、`metassist` コマンドへの入力を使用して、以下の指定が可能です。

- ボリュームのサイズ
- 冗長性のレベル (データコピーの数)
- ボリュームに対するデータパスの数
- 障害からの回復 (ボリュームをホットスペア集合と対応付けるかどうか)

ボリュームの指定は、コマンド行オプションや、コマンド行に指定する入力ファイルを使って、サービス品質に基づいて行うことができます。

状況によっては、ボリューム特性やボリューム作成時の制約を具体的に定義することが重要になります。このような場合、次の特性も指定できます。

- ボリュームのタイプ (たとえば、RAID-0 (連結方式) または RAID-0 (ストライプ方式))
- 特定のボリュームで使用するコンポーネント
- 使用できるコンポーネントと使用できないコンポーネント
- 使用するコンポーネントの数
- 作成するボリュームタイプ固有の詳細。この詳細には、ストライプ、ミラーの読み取りポリシー、類似特性が含まれます。

ボリュームの名前、サイズ、およびコンポーネントを詳細に指定したい場合は、入力ファイルを使用します。入力ファイルには、ボリューム要求ファイルとボリューム仕様ファイルが含まれます。入力ファイルの使い方の詳細については、256 ページの「ボリュームのトップダウン作成処理」を参照してください。

最後に、`metassist` コマンドで、特定のディスクやパスが使用される (または使用されない) ように指定します。

ディスクセットを使用するボリュームのトップダウン作成の実装

`metassist` コマンドは、Solaris ボリュームマネージャのディスクセットを使って、ボリュームのトップダウン作成に使用するボリュームや使用可能なディスクを管理します。ボリュームのトップダウン作成では、構成要素となるすべてのディスクが、ディスクセットに属しているか、ディスクセットに追加可能であることが必要です。トップダウン作成手順を使用すると、さまざまなディスクセットでボリュームを作成できます。ただし、使用できるディスクやコンポーネントは、ディスクセットの機能によって制限されます。

ボリュームのトップダウン作成処理

ボリュームのトップダウン作成処理では、次の 2 つの処理方法を提供することで柔軟性を実現しています。

- 必要な制約を指定でき、コマンドが完了した時点で必要なボリュームが作成される、完全に自動化されたエンドツーエンドの処理が可能です。
- ブレークポイントごとに個々の XML ベースのファイルに書き込むといった、きめ細やかな処理が可能です。

次の図に、コマンド行入力と入力ファイルに基づいて、`metassist` コマンドがどのようにエンドツーエンド処理をサポートするかを示します。さらに、`metassist` コマンドによる部分処理サポートも示します。この場合、ファイルベースのデータが得られ、ボリュームの特性を確認できます。

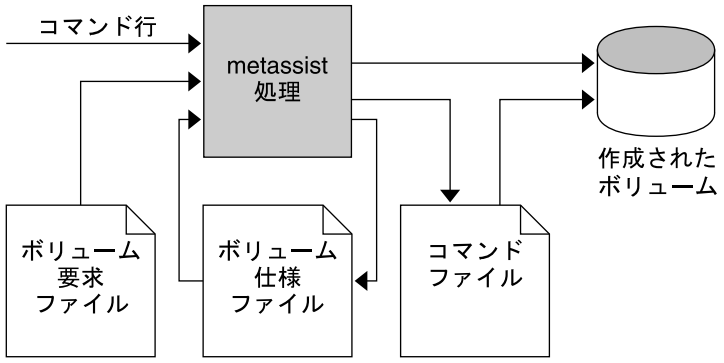

図 22–1 ボリュームのトップダウン作成処理のオプション

介入が不要なボリューム自動作成処理では、必要なサービス品質特性をコマンド行から指定すれば、`metassist` コマンドが要求されたボリュームを自動的に作成してくれます。たとえば、次のように指定します。

```
# metassist create -s storagepool  -S 10Gb
```

このコマンドによって、容量 10G バイトのストライプ方式ボリュームがディスクセット `storagepool` に作成されます。このコマンドでは、ディスクセット `storagepool` に存在している使用可能な記憶領域が使用されます。

または、「ボリューム要求ファイル」を使用して、ボリュームの特性を定義することもできます。この場合、`metassist -F` *request-file* コマンドを使用して、指定した特性を備えたボリュームを作成します。

`metassist -d` コマンドを使用すると、ボリューム仕様ファイルを作成できます。このファイルを使用すると、意図どおりの実装かどうかを査定し、必要に応じてファイルを編集できます。このボリューム仕様ファイルは以後、`metassist` コマンドへの入力として使用できます。

最後に、`metassist -c` コマンドを使用して、コマンドファイルを作成します。「コマンドファイル」は、`metassist` コマンドで指定された Solaris ボリュームマネージャの装置構成を実装するシェルスクリプトです。繰り返しボリュームを作成する場合、このファイルを使用し、適切に編集します。

`metassist` コマンドを使用してこれらのファイルを作成すると、`metassist` コマンドが何を行うか、どのように決定を下すかを理解できます。この情報は、以下のような問題の解決に役立ちます。

- ボリュームがなぜある方法で作成されたのか。
- ボリュームがなぜ作成されなかったのか。
- `metassist` コマンドがどのようなボリュームを作成するのか (ボリュームを実際に作成するわけではない)。

ボリュームのトップダウン作成に使用できるディスクの判別

`metassist` コマンドはディスクを調べ、未使用と思われるディスクを判別します。このコマンドは、使用できるディスクを慎重に判別します。使用中のディスクやスライスは、`metassist` コマンドの使用対象にはなりません。`metassist` コマンドがチェックする内容は、次のとおりです。

- ほかのディスクセットで使用されているディスク
- マウントされているスライス
- ファイルシステムスーパーブロックをもつスライス (マウント可能なファイルシステム)
- ほかの Solaris ボリュームマネージャのボリュームで使用されているスライス

これらの条件を 1 つでも満たしているスライスは、ボリュームのトップダウン作成に使用できません。

◆ ◆ ◆ 第 23 章

ボリュームのトップダウン作成 (作業)

この章では、metassist コマンドを使用して行う、Solaris ボリュームマネージャのボリュームトップダウン作成に関連する作業について説明します。

この章の内容は次のとおりです。

ボリュームのトップダウン作成に関連する概念については、第 22 章を参照してください。

ボリュームのトップダウン作成 (作業マップ)

次の表にmetassist コマンドを使用してSolaris ボリュームマネージャのボリュームをトップダウンで作成するために必要な手順を示します。このコマンドでは、QoS (サービス品質) 特性に基づいてボリュームを指定できます。また、1 つのコマンドで、レイヤー型のボリュームセットを作成できます。

作業	説明	参照先
ボリュームを自動作成します	metassist コマンドを使って、1 つまたは複数の Solaris ボリュームマネージャのボリュームを作成できます。 また、ボリュームの作成過程で metassist コマンドから出力される情報量を制御します。この情報はトラブルシューティングや診断のために使用されます。	261 ページの「ボリュームの自動作成」 261 ページの「出力の詳細度指定によるボリューム作成分析」
コマンドファイルを作成します	metassist コマンドを使って、ボリュームを生成するためのシェルスクリプトを作成します。	270 ページの「metassist コマンドによるボリューム構成ファイルの作成」
シェルスクリプトによってボリュームを作成します	前の手順において metassist コマンドで生成したシェルスクリプトを使用して、Solaris ボリュームマネージャのボリュームを作成します。	269 ページの「metassist コマンドで作成されたシェルスクリプトによるボリュームの作成」
ボリューム構成ファイルを作成します	作成するボリュームの特性を定義したボリューム構成ファイルを作成します。	270 ページの「metassist コマンドによるボリューム構成ファイルの作成」
ボリュームデフォルトファイルを変更します	デフォルトのボリューム特性を設定して、 metassist コマンドの動作をカスタマイズします。	273 ページの「ボリュームデフォルトファイルの変更」

ボリュームをトップダウン作成するための前提条件

metassist コマンドを使ってボリュームやボリューム構成を自動的に作成するためには、Solaris ボリュームマネージャ構成が正常に動作していなければなりません。作業を始める前に、次の内容が必要です。

- スーパーユーザーのアクセス権または役割に基づくアクセス制御 (RBAC) の同等の役割。詳細については、『Solaris のシステム管理 (基本編)』の「スーパーユーザー (root) になるか役割を引き受ける」を参照してください。
- 使用するシステム上に適切に分散された状態データベースの複製。状態データベースの複製の詳細については、65 ページの「Solaris ボリュームマネージャの状態データベースと状態データベースの複製について」を参照してください。
- ボリュームの作成に使用できるディスク。metassist コマンドではディスクセットを使用して、記憶領域を管理します。metassist コマンドで新しいボリュームを作成するには、まったく未使用のディスク (または既存のディスクセット) を使用する必要があります。ディスクの可用性については、258 ページの「ボリュームのトップダウン作成に使用できるディスクの判別」を参照してください。

これらの最小要件に加えて、/etc/inetd.conf ファイルにおいて、Solaris ボリュームマネージャの RPC デーモン(rpc.metad、rpc.metamhd、および rpc.metamedd) を無効にしてはなりません。デフォルトでは、これらのデーモンは起動するように構成されています。Solaris ボリュームマネージャが共有ディスクセットを使用できるように、これらのデーモンを引き続き有効にしておく必要があります。

ボリュームの自動作成

metassist コマンドを使用すると、QoS (サービス品質) の条件に基づいて、Solaris ボリュームマネージャのボリュームやボリュームセットを作成できます。Solaris ボリュームマネージャではこれまで、ボリュームを作成するために一連のコマンドが必要でしたが、metassist コマンドによって、1 つのコマンドでボリュームを作成できるようになりました。

metassist コマンドを使用すると、RAID-1 (ミラー) ボリュームを直接作成できます。したがって、RAID-1 (ミラー) ボリュームのコンポーネントとして使用するサブミラー (連結方式またはストライプ方式) を先に作成する必要はありません。

出力の詳細度指定によるボリューム作成分析

metassist コマンドを実行する際には、出力の詳細度を指定できます。出力が詳細になれば、それだけ問題の診断に役立ちます。たとえば、あるディスクがボリュームの作成になぜ選択されたのか、あるいは選択されなかったのかを判別したり、特定のコマンドがなぜ失敗したのかを判別したりすることが容易になります。出力の詳細度を下げれば、ユーザーに不必要な情報の出力を減らすことができます。

出力の詳細度を指定すると、metassist コマンドが何を行い、どのように決定を下すかを理解できます。この情報は、以下のような問題の解決に役立ちます。

- ボリュームがなぜある方法で作成されたのか。
- ボリュームがなぜ作成されなかったのか。
- metassist コマンドがどのようなボリュームを作成するのか (ボリュームを実際に作成するわけではない)。

▼ metassist コマンドを使用して RAID-1 (ミラー) ボリュームを作成するには

始める前に 260 ページの「ボリュームをトップダウン作成するための前提条件」を確認します。

1 ボリュームの作成に使用する記憶領域を特定します。

記憶領域を明示的に指定しなかった場合、システム上の未使用の記憶領域を Solaris ボリュームマネージャが特定し、必要に応じて使用します。記憶領域を指定すると、この

記憶領域を Solaris ボリュームマネージャが使用します。記憶領域の指定は、広義に (たとえば、コントローラ 1 のすべての記憶領域) 行う場合もあれば、狭義に (たとえば、`c1t4d2` は使用し、`c1t4d1` は使用しない) 行う場合もあります。

2 作業に応じて、`metassist` コマンドと適切なオプションを使用します。

- コマンド行からボリュームを作成するには、次の書式で `metassist` コマンドを実行します。

 # metassist create -s *diskset-name* **-f -r** *redundancy* **-a** *device1, device2...* **-S** *size* **-v** *verbosity*

`create`	ボリュームを作成するために使用するサブコマンド。
`-s` *diskset-name*	ボリュームに使用するディスクセットの名前を指定します。
`-f`	ボリュームとホットスペアを対応付けることを指定します。
`-r` *redundancy*	作成する冗長レベル (データコピー数) を指定します。
`-a` *device1, device2...*	ボリュームの作成用に使用できるデバイスを指定します。
`-S` *size*	作成するボリュームのサイズを `KB` (キロバイト)、`MB` (メガバイト)、`GB` (ギガバイト)、または `TB` (テラバイト) 単位で指定します。
`-v` *verbosity*	出力の詳細度を指定します。指定できる値の範囲は `0` (出力がほとんどない) から `2` (出力が多い) です。デフォルトレベルは `1` (中程度の出力) です。

- ボリュームの特性を指定する入力ファイルを使ってボリュームを作成するには、次のいずれかの書式を使って `metassist` コマンドを実行します。

 # metassist create [-v n] [-c] -F *config_file*
 # metassist create [-v n] [-c | -d] -F *request_file*

`-c`	指定のボリューム構成または生成されたボリューム構成を実装するコマンドスクリプトを出力するように指定します。コマンドスクリプトは実行されず、処理はこの段階で中断されます。
`-d`	指定のボリューム要求または生成されたボリューム要求を満たすボリューム構成を出力するように指定します。コマンドスクリプトは生成も実行もされません。処理はこの段階で中断されます。
`-F` *config_file* | *request_file*	処理対象のボリューム要求ファイルまたはボリューム構成ファイルを指定します。*config_file* または *request_file* の位置にダッシュ (-) を指定した場合、ファイルは標準入力から読み込まれます。入力ファイルがボリューム構成ファイルである場合、`-d` オプションは指定できません。

ボリューム構成ファイルには、作成するボリュームの詳細な構成情報が記述されています。一方、ボリューム要求ファイルには、作成するボリュームの特性が記載されています。詳細は、`volume-config(4)` および `volume-request(4)` のマニュアルページを参照してください。

`-v` *verbosity* — 出力の詳細度を指定します。指定できる値の範囲は **0** (出力がほとんどない) から **2** (出力が多い) です。デフォルトレベルは **1** (中程度の出力) です。

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

3 ボリュームの作成後、新しいボリュームを表示します。

```
# metastat -s diskset-name
```

例 23–1 metassist コマンドを使って 2 面ミラーを作成する

次に、容量が 10M バイトの 2 面ミラーを作成する例を示します。`metassist` コマンドは、未使用のディスクを特定し、これらのディスクを使ってできるだけ条件の良いミラーを作成します。`-s myset` 引数で、`myset` ディスクセットにボリュームを作成することを指定します。ディスクセットは必要に応じて作成されます。

```
# metassist create -s myset -r 2 -S 10mb
```

例 23–2 metassist コマンドを使って 2 面ミラーとホットスペアを作成する

次に、`metassist` コマンドを使用して、容量が 10M バイトの 2 面ミラーとホットスペアを作成し、障害耐性を強化する例を示します。`-f` オプションで障害耐性を指定します。

```
# metassist create -s myset -f -r 2 -S 10mb
```

例 23–3 metassist コマンドを使って特定のコントローラにストライプを作成する

次に、`metassist` コマンドを使用し、コントローラ 1 上の使用可能なディスクでストライプを作成する例を示します。 `-a` オプションで、使用可能なコントローラを指定します。

```
# metassist create -s myset -a c1 -S 10mb
```

例 23–4 metassist コマンドによる出力詳細度の指定

次に、`metassist` コマンドを使用して、容量が 10M バイトの 2 面ミラーとホットスペアを作成し、障害耐性を強化する例を示します。`-f` オプションで障害耐性を指定します。最

後の引数 (-v 2) では、出力の詳細度として最も大きな値である 2 を指定しています。これによって、metassist コマンドの実行結果が最も詳細に出力されます。

```
# metassist create -s myset -f -r 2 -S 10mb -v 2
Scanning system physical device configuration...

  These HBA/Controllers are known:.
    c0                      /pci@1f,0/pci@1,1/ide@3
    c1                      /pci@1f,0/pci@1/pci@2/SUNW,isptwo@4

  These disks are known:

    c0t0d0                  id1,dad@AST34342A=_____________VGD97101
    c1t1d0                  id1,sd@SSEAGATE_ST39204LCSUN9.0G3BV0L88P000021097XNL
    c1t2d0                  id1,sd@SSEAGATE_ST39102LCSUN9.0GLJW22867000019171JDF
    c1t3d0                  id1,sd@SSEAGATE_ST39204LCSUN9.0G3BV0L7RV00007108TG0H
    c1t4d0                  id1,sd@SSEAGATE_ST39204LCSUN9.0G3BV0LDFR000021087R1T
    c1t5d0                  id1,sd@SSEAGATE_ST39204LCSUN9.0G3BV0L0M200002109812L
    c1t6d0                  id1,sd@SSEAGATE_ST39204LCSUN9.0G3BV0L8K8000021087R0Z
.
.
.
(output truncated)
```

次に、metassist コマンドを使用して、容量が 10M バイトの 2 面ミラーとホットスペアを作成し、障害耐性を強化する例を示します。-f オプションで障害耐性を指定します。最後の引数 (-v 0) では、出力の詳細度として最も小さな値である 0 を指定しています。コマンドを実行しても、出力はほとんどありません。

```
# metassist create -s myset -f -r 2 -S 10mb -v 0
myset/hsp000: Hotspare pool is setup
myset/hsp000: Hotspare is added
myset/d2: Concat/Stripe is setup
myset/d1: Concat/Stripe is setup
myset/d0: Mirror is setup
myset/d0: submirror myset/d1 is attached
```

例 23–5 入力ファイルを使ってボリュームを作成する

次に、metassist コマンドで、入力ファイルを使用してボリュームを作成する例を示します。

```
# metassist create -F request.xml
```

metassist コマンドで入力ファイルを使用する方法については、265 ページの「metassist コマンドによるファイルベースのデータ処理」を参照してください。

metassist コマンドによるファイルベースのデータ処理

metassist コマンドを使用すると、ボリューム特性の評価に使用するファイル、または実際にボリュームを作成するために使用するファイルを作成できます。

metassist コマンドを使ってコマンドファイル (シェルスクリプト) を作成する

metassist コマンドに -c 引数を指定すると、ボリューム構成の作成に使用できるコマンドを含んだ Bourne シェルスクリプトが生成されます。この方法を使用すれば、ボリュームを実際に作成する前にコマンドを確認したり、場合によっては、必要に応じてスクリプトを微調整することができます。

▼ metassist コマンドを使ってコマンドファイル (シェルスクリプト) を作成するには

始める前に 260 ページの「ボリュームをトップダウン作成するための前提条件」を確認します。

1 ボリュームの作成に使用する記憶領域を特定します。

記憶領域を明示的に指定しなかった場合、システム上の未使用の記憶領域を Solaris ボリュームマネージャが特定し、必要に応じて使用します。記憶領域を指定すると、この記憶領域を Solaris ボリュームマネージャが使用します。記憶領域の指定は、広義に (たとえば、コントローラ 1 のすべての記憶領域) 行う場合もあれば、狭義に (たとえば、c1t4d2 は使用し、c1t4d1 は使用しない) 行う場合もあります。

2 作業に応じて、metassist コマンドと適切なオプションを使用します。

-c オプションを使用して、ボリュームを実際には作成しないことを指定します。

```
# metassist create -s diskset-name -f -r redundancy -a device1, device2... \
 -S size -v verbosity [-c]
```

create	ボリュームを作成するために使用するサブコマンド。
-s *diskset-name*	ボリュームに使用するディスクセットの名前を指定します。
-f	ボリュームとホットスペアを対応付けることを指定します。
-r *redundancy*	作成する冗長レベル (データコピー数) を指定します。
-a *device1, device2...*	ボリュームの作成用に使用できるデバイスを指定します。
-S *size*	作成するボリュームのサイズを KB (キロバイト)、MB (メガバイト)、GB (ギガバイト)、または TB (テラバイト) 単位で指定します。

-v *verbosity*	出力の詳細度を指定します。指定できる値の範囲は 0 (出力がほとんどない) から 2 (出力が多い) です。デフォルトレベルは 1 (中程度の出力) です。
-c	ボリュームを実際には作成しないことを意味します。その代わりに、指定した構成を作成するためのシェルスクリプトが、標準出力に送信されます。

注 – -c 引数によって、シェルスクリプトを標準出力に送信することが必須になりますが、metassist コマンドの他の出力は標準エラーに送信されます。出力ストリームは自由にリダイレクトできます。

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

例 23–6 metassist コマンドを使ってコマンドファイル (シェルスクリプト) を作成する

次に、metassist コマンドを使用して、容量が 10M バイトの 2 面ミラーとホットスペアを作成し、障害耐性を強化する例を示します。-f オプションで障害耐性を指定します。最後の引数 (-c) で、ボリュームを実際には作成しないことを指定します。その代わりに、指定した構成を作成するためのシェルスクリプトが、標準出力に送信されます。

```
# metassist create -s myset -f -r 2 -S 10mb -c
(output truncated)
.
.
.
Volume request completed successfully.
#!/bin/sh

#
# Environment
#

# Amend PATH
PATH="/usr/sbin:/usr/bin:$PATH"
export PATH

# Disk set name
diskset=’myset’

#
# Functions
#
```

```
# Echo (verbose) and exec given command, exit on error
execho () {
    test -n "$verbose" && echo "$@"
    "$@" || exit
}

# Get full /dev/rdsk path of given slice
fullpath () {
    case "$1" in
        /dev/dsk/*|/dev/did/dsk/*) echo "$1" | sed 's/dsk/rdsk/' ;;
        /*) echo "$1" ;;
        *) echo /dev/rdsk/"$1" ;;
    esac
}

# Run fmthard, ignore partboot error, error if output
fmthard_special () {
    ignore='Error writing partboot'
    out=`fmthard "$@" 2>&1`
    result=$?
    echo "$out" |
    case "$out" in
        *"$ignore"*) grep -v "$ignore"; return 0 ;;
        '') return "$result" ;;
        *) cat; return 1 ;;
    esac >&2
}

#
# Main
#

# Verify root
if [ "`id | sed 's/^[^(]*(\([^)]*\).*/\1/'`" != root ]
then
    echo "This script must be run as root." >&2
    exit 1;
fi

# Check for verbose option
case "$1" in
    -v) verbose=1 ;;
    *) verbose= ;;
esac

# Does the disk set exist?
```

```
if metaset -s "$diskset" >/dev/null 2>&1
then
    # Take control of disk set
    execho metaset -s "$diskset" -t
else
    # Create the disk set
    autotakeargs=
    /usr/sbin/clinfo || autotakeargs='-A enable'
    execho metaset -s "$diskset" $autotakeargs -a -h `uname -n | cut -f1 -d.`
fi

# Format slices
execho fmthard_special -d 7:0:0:0:0 `fullpath c1t3d0s7`
execho fmthard_special -d 7:0:0:0:0 `fullpath c1t6d0s7`
execho fmthard_special -d 7:0:0:0:0 `fullpath c1t4d0s7`

# Add disks to set
execho metaset -s "$diskset" -a c1t3d0
execho metaset -s "$diskset" -a c1t6d0
execho metaset -s "$diskset" -a c1t4d0

# Format slices
execho fmthard_special -d 0:4:0:10773:17649765 `fullpath c1t3d0s0`
execho fmthard_special -d 0:4:0:10773:17649765 `fullpath c1t6d0s0`
execho fmthard_special -d 0:4:0:10773:17649765 `fullpath c1t4d0s0`
execho fmthard_special -d 1:4:0:17660538:21546 `fullpath c1t3d0s1`
execho fmthard_special -d 1:4:0:17660538:21546 `fullpath c1t4d0s1`
execho fmthard_special -d 1:4:0:17660538:21546 `fullpath c1t6d0s1`

# Does hsp000 exist?
metahs -s "$diskset" -i hsp000 >/dev/null 2>&1 || {
    # Create hsp hsp000
    execho metainit -s "$diskset" hsp000
}

# Add slices to hsp000
execho metahs -s "$diskset" -a hsp000 c1t3d0s1

# Create concat d2
execho metainit -s "$diskset" d2 1 1 c1t4d0s1

# Associate concat d2 with hot spare pool hsp000
execho metaparam -s "$diskset" -h hsp000 d2

# Create concat d1
execho metainit -s "$diskset" d1 1 1 c1t6d0s1
```

```
# Associate concat d1 with hot spare pool hsp000
execho metaparam -s "$diskset" -h hsp000 d1

# Create mirror d0
execho metainit -s "$diskset" d0 -m d2 1
execho metattach -s "$diskset" d0 d1
#
```

例 **23–7** metassist コマンドを使ってコマンドファイル (シェルスクリプト) を保存する

次に、metassist コマンドを使用して、容量が 10M バイトの 2 面ミラーとホットスペアを作成し、障害耐性を強化する例を示します。-f オプションで障害耐性を指定します。最後の引数 (-c) で、ボリュームを実際には作成しないことを指定します。その代わりに、指定した構成を作成するためのシェルスクリプトが、標準出力に送信されます。コマンドの最後の部分では、標準出力をリダイレクトして /tmp/metassist-shell-script.sh シェルスクリプトを作成することを指定します。このファイルはあとで、指定したボリュームを作成するために使用できます。

```
# metassist create -s myset -f -r 2 -S 10mb -c > \
 /tmp/metassist-shell-script.sh
```

metassist コマンドで作成されたシェルスクリプトによるボリュームの作成

metassist コマンドでシェルスクリプトを作成したら、このスクリプトを使ってボリュームを作成できます。ボリュームは、シェルスクリプトの作成時に指定したとおりに作成されます。

注意 – metassist コマンドで作成したコマンドスクリプトは、スクリプトを作成したシステムのその時点の構成に大きく依存しています。したがって、このスクリプトを別のシステムで使用したり、このスクリプトをシステム構成の変更後に使用したりすると、データが壊れたり失われたりすることがあります。

▼ 保存された metassist コマンドのシェルスクリプトを実行するには

始める前に 260 ページの「ボリュームをトップダウン作成するための前提条件」を確認します。

1 シェルスクリプトの作成後にシステム構成が変更されていないことを確認します。さらに、スクリプトを実行するシステムが、このスクリプトを作成したシステムであることを確認します。

2 保存されたシェルスクリプトを実行します。

```
# sh ./metassist-shell-script-name
```

3 新しいボリュームを表示します。

```
# metastat -s diskset-name
```

例 23–8 保存された metassist コマンドのシェルスクリプトを実行する

次に、metassist コマンドでシェルスクリプトを使用してボリュームを作成する例を示します。

```
# sh ./tmp/metassist-shell-script.sh
myset/hsp000: Hotspare pool is setup
myset/hsp000: Hotspare is added
myset/d2: Concat/Stripe is setup
myset/d1: Concat/Stripe is setup
myset/d0: Mirror is setup
myset/d0: submirror myset/d1 is attached
```

metassist コマンドによるボリューム構成ファイルの作成

metassist コマンドに -d 引数を指定すると、XML ベースのボリューム構成ファイルが生成されます。このファイルには、ボリュームに関連するすべてのオプションや情報など、ボリュームとそのコンポーネントの詳細が含まれています。このファイルを調べることによって、metassist コマンドが推奨する構成を知ることができます。さらに、このボリューム構成ファイルを慎重に変更して構成を微調整したあと、実際のボリューム作成の際に metassist コマンドへの入力として使用することもできます。

▼ metassist コマンドを使ってボリューム構成ファイルを作成するには

始める前に 260 ページの「ボリュームをトップダウン作成するための前提条件」を確認します。

1 ボリュームの作成に使用する記憶領域を特定します。

記憶領域を明示的に指定しなかった場合、システム上の未使用の記憶領域を Solaris ボリュームマネージャが特定し、必要に応じて使用します。記憶領域を指定すると、この記憶領域を Solaris ボリュームマネージャが使用します。記憶領域の指定は、広義に (たとえば、コントローラ 1 のすべての記憶領域) 行う場合もあれば、狭義に (たとえば、c1t4d2 は使用し、c1t4d1 は使用しない) 行う場合もあります。

2 作業に応じて、metassist コマンドと適切なオプションを使用します。

-d オプションを使用して、ボリュームを実際には作成しないことを指定します。代わりに、XML ベースのボリューム構成ファイルが標準出力に送信されます。

```
# metassist create -s diskset-name -f -r redundancy -a device1, device2... \
 -S size -v verbosity [-d]
```

create	ボリュームを作成するために使用するサブコマンド。
-s *diskset-name*	ボリュームに使用するディスクセットの名前を指定します。
-f	ボリュームとホットスペアを対応付けることを指定します。
-r *redundancy*	作成する冗長レベル (データコピー数) を指定します。
-a *device1, device2...*	ボリュームの作成用に使用できるデバイスを指定します。
-S *size*	作成するボリュームのサイズを KB (キロバイト)、MB (メガバイト)、GB (ギガバイト)、または TB (テラバイト) 単位で指定します。
-v *verbosity*	出力の詳細度を指定します。指定できる値の範囲は 0 (出力がほとんどない) から 2 (出力が多い) です。デフォルトレベルは 1 (中程度の出力) です。
-d	ボリュームを実際には作成しないことを意味します。

注 – -d 引数によって、XML ベースのボリューム構成ファイルを標準出力に送信することが必須になります。しかし、metassist コマンドの他の出力は標準エラーに送信されます。出力ストリームは自由にリダイレクトできます。

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

例 23–9 metassist コマンドを使ってボリューム構成ファイルを作成する

次に、metassist コマンドを使用して、容量が 10M バイトの 2 面ミラーとホットスペアを作成し、障害耐性を強化する例を示します。-f オプションで障害耐性を指定します。最後の引数 (-c) で、ボリュームを実際には作成しないことを指定します。その代わりに、指定した構成を作成するために使用できるボリューム構成ファイルが、標準出力に送信されます。

```
# metassist create -s myset -f -r 2 -S 10mb -d


.(output truncated)
.
.
Volume request completed successfully.
```

```
<?xml version="1.0"?>
<!DOCTYPE volume-config SYSTEM "/usr/share/lib/xml/dtd/volume-config.dtd">
 <volume-config>
<diskset name="myset"/>
<disk name="c1t3d0"/>
<disk name="c1t6d0"/>
<disk name="c1t4d0"/>
<slice name="c1t3d0s7" sizeinblocks="0"/>
<slice name="c1t3d0s0" sizeinblocks="17649765" startsector="10773"/>
<slice name="c1t6d0s7" sizeinblocks="0"/>
<slice name="c1t6d0s0" sizeinblocks="17649765" startsector="10773"/>
<slice name="c1t4d0s7" sizeinblocks="0"/>
<slice name="c1t4d0s0" sizeinblocks="17649765" startsector="10773"/>
<hsp name="hsp000">
<slice name="c1t3d0s1" sizeinblocks="21546" startsector="17660538"/>
</hsp>
<mirror name="d0" read="ROUNDROBIN" write="PARALLEL" passnum="1">
<concat name="d2">
<slice name="c1t4d0s1" sizeinblocks="21546" startsector="17660538"/>
<hsp name="hsp000"/>
</concat>
<concat name="d1>
<slice name="c1t6d0s1"sizeinblocks="21546" startsector="17660538"/>
<hsp name="hsp000"/>
</concat>
</mirror>
</volume-config>
#
```

例 23–10 metassist コマンドを使ってボリューム構成ファイルを保存する

次に、metassist コマンドを使用して、容量が 10M バイトの 2 面ミラーとホットスペアを作成し、障害耐性を強化する例を示します。-f オプションで障害耐性を指定します。最後の引数 (-c) で、ボリュームを実際には作成しないことを指定します。その代わりに、指定した構成を作成するために使用できるボリューム構成ファイルが、標準出力に送信されます。コマンドの最後の部分では、標準出力をリダイレクトして /tmp/metassist-volume-config.xml ボリューム構成ファイルを作成することを指定します。このファイルはあとで、指定したボリュームを作成するために使用できます。

```
# metassist create -s myset -f -r 2 -S 10mb -d >  \
/tmp/metassist-volume-config.xml
```

metassist コマンドのデフォルト動作の変更

ボリュームデフォルトファイル (/etc/defaults/metassist.xml) を使用すると、metassist コマンドのデフォルト動作を変更できます。デフォルトファイルを変更することによって、特定のディスクやコントローラを明示的に考慮から除外したり、考慮に含めたりできます。metassist コマンドで使用する大部分のボリューム設定値に要件を指定することもできます。

/etc/defaults/metassist.xml の形式は、/usr/share/lib/xml/dtd/volume-defaults.dtd の文書型定義 (DTD) で規定されています。この形式については、volume-defaults(4) のマニュアルページを参照してください。

ボリュームデフォルトファイルの変更

ボリュームデフォルトファイル (/etc/defaults/metassist.xml) を編集して、metassist コマンドの動作を指定します。

注 – ファイルを編集する際には、ファイルが文書型定義 (DTD) /usr/share/lib/xml/dtd/volume-defaults.dtd に準拠するようにしてください。XML ファイルが DTD に準拠していないと、metassist コマンドはエラーメッセージを出して異常終了します。

例 23–11 metassist コマンドを使ってデフォルト設定を変更したボリュームを作成する

ボリュームを作成する前に、/etc/default/metassist.xml ファイルを編集してデフォルト設定を指定します。この設定は、metassist コマンドを使って作成するすべてのボリュームに適用されます。この例に示す metassist コマンドでは、ボリュームをコントローラ c1 上にだけ作成します。さらに、ストライプを作成する際には必ず、4 つのコンポーネントと飛び越し値 512KB からなるストライプを作成します。/etc/default/metassist.xml ファイルが再び変更されない限り、これらの制約はすべての metassist コマンドに適用されます。

```
#  cat /etc/default/metassist.xml
<!DOCTYPE volume-defaults SYSTEM \
"/usr/share/lib/xml/dtd/volume-defaults.dtd">

<volume-defaults>
<available name="c1" />
<stripe mincomp="4" maxcomp="4" interlace="512KB" ></stripe>
</volume-defaults>

# metassist create -s myset -S 10Gb
```

例 23–11 metassist コマンドを使ってデフォルト設定を変更したボリュームを作成する　(続き)

このmetassist コマンドによって、/etc/default/metassist.xml ファイルで指定されたとおりに、4 つのスライスと 512K バイトの飛び越し値を使用する、10G バイトのストライプが作成されます。

◆ ◆ ◆ 第 24 章

監視とエラーレポート (作業)

Solaris ボリュームマネージャで、スライスレベルの物理エラーが原因でボリュームに書き込みできないというような問題が起きることがあります。問題が発生すると、Solaris ボリュームマネージャがボリュームの状態を変更するので、システム管理者は常に情報を把握できます。ただし、Solaris 管理コンソールを通じて Solaris ボリュームマネージャの GUI を使用したり、`metastat` コマンドを実行してボリューム状態の変化を定期的にチェックしないと、状態の変化をタイムリーに把握することはできません。

この章では、Solaris ボリュームマネージャ SNMP エージェント (Solstice Enterprise Agents™ 監視ソフトウェアのサブエージェント) など、Solaris ボリュームマネージャのさまざまな監視ツールについて説明します。このツールを設定して SNMP トラップを報告する以外に、シェルスクリプトを作成して Solaris ボリュームマネージャのさまざまな機能を能動的に監視することもできます。このシェルスクリプトは `cron` ジョブとして動作し、潜在的な問題が顕在化する前にそれらを検出する上で役立ちます。

この章の内容は次のとおりです。

Solaris ボリュームマネージャの監視機能と報告機能 (作業マップ)

次の表に、Solaris ボリュームマネージャのエラーレポートを管理するために必要な作業を示します。

作業	説明	参照先
mdmonitord デーモンを設定してエラーを周期的にチェックする	/lib/svc/method/svc-mdmonitor スクリプトを編集して、mdmonitord デーモンに使用させるエラーチェック間隔を設定します。	276 ページの「エラーを周期的にチェックするための mdmonitord デーモンの構成」
Solaris ボリュームマネージャ SNMP エージェントを構成する	/etc/snmp/conf ディレクトリの構成ファイルを編集して、Solaris ボリュームマネージャが適切なシステムにトラップを送信できるようにします。	277 ページの「Solaris ボリュームマネージャ SNMP エージェントの構成」
cron コマンドでスクリプトを実行して Solaris ボリュームマネージャを監視する	エラーをチェックするスクリプトを作成または変更し、cron コマンドでスクリプトを実行します。	281 ページの「cron ジョブによる Solaris ボリュームマネージャの監視」

エラーを周期的にチェックするための mdmonitord デーモンの構成

Solaris ボリュームマネージャには /usr/sbin/mdmonitord デーモンが組み込まれています。このデーモンは、Solaris ボリュームマネージャのボリュームにエラーがないかどうかを調べるプログラムです。このプログラムはデフォルトで、ボリューム上で (書き込みエラーなどの) エラーが検出された場合に限り、RAID-1 (ミラー) ボリューム、RAID-5 ボリューム、およびホットスペアのエラーの有無を調べます。ただし、指定の間隔でエラーを能動的にチェックするようにこのプログラムを設定することもできます。

▼ mdmonitord コマンドを設定してエラーを周期的にチェックするには

/lib/svc/method/svc-mdmonitor スクリプトを編集して、周期的に検査する時間間隔を追加します。

1. スーパーユーザーになります。
2. 任意のエディタで /lib/svc/method/svc-mdmonitor スクリプトを開きます。スクリプトから次の部分を検索します。

```
$MDMONITORD
error=$?
case $error in
0)      exit 0
        ;;

*)      echo "Could not start $MDMONITORD. Error $error."
        exit 0
```

3 mdmonitord コマンドから始まる行に、-t フラグとチェック間隔の秒数を追加します。

```
$MDMONITORD -t 3600
error=$?
case $error in
0)      exit 0
        ;;

*)      echo "Could not start $MDMONITORD. Error $error."
        exit 0
        ;;
esac
```

4 mdmonitord コマンドを再起動して変更を有効にします。

```
# svcadm restart system/mdmonitor
```

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

Solaris ボリュームマネージャ SNMP エージェントの概要

Solaris ボリュームマネージャ SNMP トラップエージェントには、コアパッケージ SUNWlvmr と SUNWlvma のパッケージのほかに Solstice Enterprise Agents が必要です。必要なコアパッケージは、次のとおりです。

- SUNWmibii
- SUNWsacom
- SUNWsadmi
- SUNWsasnm

これらのパッケージは Solaris オペレーティングシステムに組み込まれています。これらのパッケージは、インストール時にパッケージ選択を変更した場合、または最小限のパッケージセットをインストールした場合を除き、通常はデフォルトでインストールされます。各パッケージが使用可能かどうかを確認するには、pkginfo SUNWsasnm のように、pkginfo *pkgname* コマンドを使用します。5 つのパッケージがすべて使用できることを確認してから、以下の説明に従って Solaris ボリュームマネージャ SNMP エージェントを構成する必要があります。

Solaris ボリュームマネージャ SNMP エージェントの構成

Solaris ボリュームマネージャ SNMP エージェントはデフォルトでは有効にされていません。SNMP トラップを有効にするには、次の手順を実行します。

おそらく、Solaris オペレーティングシステムをアップグレードするたびに、/etc/snmp/conf/enterprises.oid ファイルを編集し、その終わりに 手順 6 の行を追加してから、Solaris Enterprise Agents サーバーの停止と再起動を行う必要があります。

この手順が終わると、SNMP トラップが、指定されたホストに送信されるようになります。送信されたトラップを表示するためには、Solstice Enterprise Agents ソフトウェアなど、適切な SNMP モニターを使用する必要があります。

問題が発生したときにトラップを受信するためには、mdmonitord コマンドを設定してシステムを定期的にチェックする必要があります。276 ページの「エラーを周期的にチェックするための mdmonitord デーモンの構成」を参照してください。また、その他のエラーチェックオプションについては、281 ページの「cron ジョブによる Solaris ボリュームマネージャの監視」を参照してください。

▼ Solaris ボリュームマネージャ SNMP エージェントを構成するには

1 スーパーユーザーになります。

2 /etc/snmp/conf/mdlogd.rsrc– 構成ファイルを /etc/snmp/conf/mdlogd.rsrc に移動します。

```
# mv /etc/snmp/conf/mdlogd.rsrc- /etc/snmp/conf/mdlogd.rsrc
```

3 /etc/snmp/conf/mdlogd.acl ファイルを編集して、SNMP トラップをどのホストに送信するかを指定します。次の部分を探してください。

```
        trap = {
             {
                trap-community = SNMP-trap
                hosts = corsair
                {
                    enterprise = "Solaris Volume Manager"
                    trap-num = 1, 2, 3
                }
```

hosts = corsair の行に、Solaris ボリュームマネージャ SNMP トラップの送信先ホストを指定します。たとえば、SNMP トラップを lexicon に送信する場合は、この行を hosts = lexicon に変更します。複数のホストを指定する場合は、hosts = lexicon, idiom のように、ホスト名をコンマで区切って指定します。

4 次に、/etc/snmp/conf/snmpdx.acl ファイルを編集して、SNMP トラップをどのホストに送信するかを指定します。

trap = で始まるブロックを探し、ここに前の手順で指定したのと同じホスト名のリストを指定します。このセクションは # でコメント文にされていることがあります。その場合は、必要な行の始めにある # を取り除いてください。トラップセクションにはコメントになっている行がほかにもあります。これらの行はそのまま残しておいても、わかり

やすくするために削除してもかまいません。必要な行のコメントを解除し、ホスト名の行を変更した後のセクションは次のようになります。

```
###################
# trap parameters #
###################

trap = {
  {
        trap-community = SNMP-trap
        hosts =lexicon
        {
          enterprise = "sun"
          trap-num = 0, 1, 2-5, 6-16
        }
#       {
#         enterprise = "3Com"
#         trap-num = 4
#       }
#       {
#         enterprise = "snmp"
#         trap-num = 0, 2, 5
#       }
#  }
#  {
#       trap-community = jerry-trap
#       hosts = jerry, nanak, hubble
#       {
#         enterprise = "sun"
#         trap-num = 1, 3
#       }
#       {
#         enterprise = "snmp"
#         trap-num = 1-3
#       }
  }
}
```

注 – `/etc/snmp/conf/snmpdx.acl` ファイルに同じ数の左括弧と右括弧があることを確認してください。

---

5 前の手順でコメントを解除した、`/etc/snmp/conf/snmpdx.acl` ファイルのセクションの中に新しい **Solaris** ボリュームマネージャセクションを追加します。

```
        trap-community = SNMP-trap
        hosts = lexicon
        {
          enterprise = "sun"
          trap-num = 0, 1, 2-5, 6-16
```

```
}
{
    enterprise = "Solaris Volume Manager"
    trap-num = 1, 2, 3
}
```

追加する 4 行は、enterprise = “sun” ブロックのすぐ後に挿入する必要があります。

6 /etc/snmp/conf/enterprises.oid ファイルの最後に次の行を追加します。

```
"Solaris Volume Manager"                          "1.3.6.1.4.1.42.104"
```

7 **Solstice Enterprise Agents** サーバーを停止し、再起動します。

```
# /etc/init.d/init.snmpdx stop
# /etc/init.d/init.snmpdx start
```

Solaris ボリュームマネージャ SNMP エージェントの制約

Solaris ボリュームマネージャ SNMP エージェントは、システム管理者が把握すべき Solaris ボリュームマネージャのあらゆる問題についてトラップを送信するわけではありません。具体的には、このエージェントは、次の場合にのみトラップを送信します。

- RAID-1 または RAID-5 のサブコンポーネントが「保守が必要 (Needs Maintenance)」状態に移行した場合
- ホットスペアが、障害のあるディスクに代わって使用されるようになった場合
- ホットスペアが再同期処理を開始した場合
- ホットスペアが再同期処理を完了した場合
- ミラーがオフライン状態になった場合
- ディスクセットが別のホストに予約されたため、現在のホストがパニック状態になった場合

RAID-0 ボリュームやソフトパーティションが定義されているディスクが使用不能な場合など、問題が発生しても、SNMP トラップが送信されない状況は少なくありません。これは、このデバイスに対する読み取りや書き込みが行われた場合でも同様です。このような状況では通常、SCSI または IDE エラーは報告されます。しかし、これらのエラーが監視コンソールに伝えられるようにするには、他の SNMP エージェントからトラップを発行する必要があります。

cron ジョブによる **Solaris** ボリュームマネージャの監視

▼ ボリュームのエラーを自動的にチェックするには

◗ **Solaris** ボリュームマネージャ構成のエラーを自動的にチェックするために、cron ユーティリティーで定期的に実行できるスクリプトを作成します。

次のスクリプト例は、必要に応じて変更することができます。

注 – このスクリプトは、Solaris ボリュームマネージャのエラーチェックを自動化するための基本的なスクリプトです。各自の構成に合わせて変更する必要があります。

```
#
#!/bin/ksh
#ident "@(#)metacheck.sh   1.3     96/06/21 SMI"
# ident='%Z%%M%   %I%     %E% SMI'
#
# Copyright (c) 1999 by Sun Microsystems, Inc.
#
# metacheck
#
# Check on the status of the metadevice configuration.  If there is a problem
# return a non zero exit code.  Depending on options, send email notification.
#
# -h
#    help
# -s setname
#    Specify the set to check.  By default, the 'local' set will be checked.
# -m recipient [recipient...]
#    Send email notification to the specified recipients.  This
#    must be the last argument. The notification shows up as a short
#    email message with a subject of
#        "Solaris Volume Manager Problem: metacheck.who.nodename.setname"
#    which summarizes the problem(s) and tells how to obtain detailed
#    information. The "setname" is from the -s option, "who" is from
#    the -w option, and "nodename" is reported by uname(1).
#    Email notification is further affected by the following options:
#        -f    to suppress additional messages after a problem
#            has been found.
#        -d    to control the supression.
#        -w    to identify who generated the email.
#        -t    to force email even when there is no problem.
# -w who
#    indicate who is running the command. By default, this is the
#    user-name as reported by id(1M). This is used when sending
```

```
#    email notification (-m).
# -f
#    Enable filtering.  Filtering applies to email notification (-m).
#    Filtering requires root permission.  When sending email notification
#    the file /etc/lvm/metacheck.setname.pending is used to
#    controll the filter.  The following matrix specifies the behavior
#    of the filter:
#
#    problem_found    file_exists
#      yes          no        Create file, send notification
#      yes          yes        Resend notification if the current date
#                     (as specified by -d datefmt) is
#                     different than the file date.
#      no          yes        Delete file, send notification
#                     that the problem is resolved.
#      no          no        Send notification if -t specified.
#
# -d datefmt
#    Specify the format of the date for filtering (-f).  This option
#    controls the how often re-notification via email occurs. If the
#    current date according to the specified format (strftime(3C)) is
#    identical to the date contained in the
#    /etc/lvm/metacheck.setname.pending file then the message is
#    suppressed. The default date format is "%D", which will send one
#    re-notification per day.
# -t
#    Test mode.  Enable email generation even when there is no problem.
#    Used for end-to-end verification of the mechanism and email addresses.
#
#
# These options are designed to allow integration of metacheck
# into crontab.  For example, a root crontab entry of:
#
# 0,15,30,45 * * * * /usr/sbin/metacheck -f -w SVMcron \
#   -d '\%D \%h' -m notice@example.com 2148357243.8333033@pager.example.com
#
# would check for problems every 15 minutes, and generate an email to
# notice@example.com (and send to an email pager service) every hour when
# there is a problem.  Note the \ prior to the '%' characters for a
# crontab entry.  Bounced email would come back to root@nodename.
# The subject line for email generated by the above line would be
# Solaris Volume Manager Problem: metacheck.SVMcron.nodename.local
#

# display a debug line to controlling terminal (works in pipes)
decho()
{
    if [ "$debug" = "yes" ] ; then
```

```
    echo "DEBUG: $*"    < /dev/null > /dev/tty 2>&1
    fi
}

# if string $1 is in $2-* then return $1, else return ""
strstr()
{
    typeset    look="$1"
    typeset    ret=""

    shift
#   decho "strstr LOOK .$look. FIRST .$1."
    while [ $# -ne 0 ] ; do
    if [ "$look" = "$1" ] ; then
        ret="$look"
    fi
    shift
    done
    echo "$ret"
}

# if string $1 is in $2-* then delete it. return result
strdstr()
{
    typeset    look="$1"
    typeset    ret=""

    shift
#   decho "strdstr LOOK .$look. FIRST .$1."
    while [ $# -ne 0 ] ; do
    if [ "$look" != "$1" ] ; then
        ret="$ret $1"
    fi
    shift
    done
    echo "$ret"
}

merge_continued_lines()
{
    awk -e '\
    BEGIN { line = "";} \
    $NF == "\\" { \
        $NF = ""; \
        line = line $0; \
        next; \
    } \
    $NF != "\\" { \
```

```
        if ( line != "" ) { \
        print line $0; \
        line = ""; \
        } else { \
        print $0; \
        } \
    }'
}

# trim out stuff not associated with metadevices
find_meta_devices()
{
    typeset    devices=""

#   decho "find_meta_devices .$*."
    while [ $# -ne 0 ] ; do
    case $1 in
    d+([0-9]) )    # metadevice name
        devices="$devices $1"
        ;;
    esac
    shift
    done
    echo "$devices"
}

# return the list of top level metadevices
toplevel()
{
    typeset    comp_meta_devices=""
    typeset    top_meta_devices=""
    typeset    devices=""
    typeset    device=""
    typeset    comp=""

    metastat$setarg -p | merge_continued_lines | while read line ; do
    echo "$line"
    devices=‘find_meta_devices $line‘
    set -- $devices
    if [ $# -ne 0 ] ; then
        device=$1
        shift
        # check to see if device already refered to as component
        comp=‘strstr $device $comp_meta_devices‘
        if [ -z $comp ] ; then
        top_meta_devices="$top_meta_devices $device"
        fi
        # add components to component list, remove from top list
```

```
        while [ $# -ne 0 ] ; do
        comp=$1
        comp_meta_devices="$comp_meta_devices $comp"
        top_meta_devices=‘strdstr $comp $top_meta_devices‘
        shift
        done
    fi
    done > /dev/null 2>&1
    echo $top_meta_devices
}

#
# - MAIN
#
METAPATH=/usr/sbin
PATH=//usr/bin:$METAPATH
USAGE="usage: metacheck [-s setname] [-h] [[-t] [-f [-d datefmt]] \
    [-w who] -m recipient [recipient...]]"

datefmt="%D"
debug="no"
filter="no"
mflag="no"
set="local"
setarg=""
testarg="no"
who=‘id | sed -e ’s/^uid=[0-9][0-9]*(//’ -e ’s/).*//’‘

while getopts d:Dfms:tw: flag
do
    case $flag in
    d)    datefmt=$OPTARG;
    ;;
    D)    debug="yes"
    ;;
    f)    filter="yes"
    ;;
    m)    mflag="yes"
    ;;
    s)    set=$OPTARG;
    if [ "$set" != "local" ] ; then
        setarg=" -s $set";
    fi
    ;;
    t)    testarg="yes";
    ;;
    w)    who=$OPTARG;
    ;;
```

```
    \?)    echo $USAGE
    exit 1
    ;;
    esac
done

# if mflag specified then everything else part of recipient
shift ‘expr $OPTIND - 1‘
if [ $mflag = "no" ] ; then
    if [ $# -ne 0 ] ; then
    echo $USAGE
    exit 1
    fi
else
    if [ $# -eq 0 ] ; then
    echo $USAGE
    exit 1
    fi
fi
recipients="$*"

curdate_filter=‘date +$datefmt‘
curdate=‘date‘
node=‘uname -n‘

# establish files
msg_f=/tmp/metacheck.msg.$$
msgs_f=/tmp/metacheck.msgs.$$
metastat_f=/tmp/metacheck.metastat.$$
metadb_f=/tmp/metacheck.metadb.$$
metahs_f=/tmp/metacheck.metahs.$$
pending_f=/etc/lvm/metacheck.$set.pending
files="$metastat_f $metadb_f $metahs_f $msg_f $msgs_f"

rm -f $files                              > /dev/null 2>&1
trap "rm -f $files > /dev/null 2>&1; exit 1" 1 2 3 15

# Check to see if metadb is capable of running
have_metadb="yes"
metadb$setarg                               > $metadb_f 2>&1
if [ $? -ne 0 ] ; then
    have_metadb="no"
fi
grep "there are no existing databases"      < $metadb_f    > /dev/null 2>&1
if [ $? -eq 0 ] ; then
    have_metadb="no"
fi
grep "/dev/md/admin"                < $metadb_f    > /dev/null 2>&1
```

```
if [ $? -eq 0 ] ; then
    have_metadb="no"
fi

# check for problems accessing metadbs
retval=0
if [ "$have_metadb" = "no" ] ; then
    retval=1
    echo "metacheck: metadb problem, can't run '$METAPATH/metadb$setarg'" \
                                >> $msgs_f
else
    # snapshot the state
    metadb$setarg 2>&1 | sed -e '1d' | merge_continued_lines    > $metadb_f
    metastat$setarg 2>&1 | merge_continued_lines        > $metastat_f
    metahs$setarg -i 2>&1 | merge_continued_lines        > $metahs_f

    #
    # Check replicas for problems, capital letters in the flags
    # indicate an error, fields are seperated by tabs.
    #
    problem=`awk < $metadb_f -F\t '{if ($1 ~ /[A-Z]/) print $1;}'`
    if [ -n "$problem" ] ; then
    retval=`expr $retval + 64`
    echo "\
metacheck: metadb problem, for more detail run:\n\t$METAPATH/metadb$setarg -i" \
                                >> $msgs_f
    fi

    #
    # Check the metadevice state
    #
    problem=`awk < $metastat_f -e \
        '/State:/ {if ($2 != "Okay" && $2 != "Resyncing") print $0;}'`
    if [ -n "$problem" ] ; then
    retval=`expr $retval + 128`
    echo "\
metacheck: metadevice problem, for more detail run:" \
                                >> $msgs_f

    # refine the message to toplevel metadevices that have a problem
    top=`toplevel`
    set -- $top
    while [ $# -ne 0 ] ; do
        device=$1
        problem=`metastat $device | awk -e \
        '/State:/ {if ($2 != "Okay" && $2 != "Resyncing") print $0;}'`
        if [ -n "$problem" ] ; then
        echo "\t$METAPATH/metastat$setarg $device"    >> $msgs_f
```

```
        # find out what is mounted on the device
        mp='mount|awk -e '/\/dev\/md\/dsk\/'$device'[ \t]/{print $1;}''
        if [ -n "$mp" ] ; then
            echo "\t\t$mp mounted on $device"        >> $msgs_f
        fi
        fi
        shift
    done
    fi

    #
    # Check the hotspares to see if any have been used.
    #
    problem=""
    grep "no hotspare pools found"    < $metahs_f        > /dev/null 2>&1
    if [ $? -ne 0 ] ; then
    problem='awk < $metahs_f -e \
        '/blocks/ { if ( $2 != "Available" ) print $0;}''
    fi
    if [ -n "$problem" ] ; then
    retval='expr $retval + 256'
    echo "\
metacheck: hot spare in use, for more detail run:\n\t$METAPATH/metahs$setarg -i" \
                                >> $msgs_f
    fi
fi

# If any errors occurred, then mail the report
if [ $retval -ne 0 ] ; then
    if [ -n "$recipients" ] ; then
    re=""
    if [ -f $pending_f ] && [ "$filter" = "yes" ] ; then
        re="Re: "
        # we have a pending notification, check date to see if we resend
        penddate_filter='cat $pending_f | head -1'
        if [ "$curdate_filter" != "$penddate_filter" ] ; then
        rm -f $pending_f                > /dev/null 2>&1
        else
         if [ "$debug" = "yes" ] ; then
            echo "metacheck: email problem notification still pending"
            cat $pending_f
        fi
        fi
    fi
    if [ ! -f $pending_f ] ; then
        if [ "$filter" = "yes" ] ; then
        echo "$curdate_filter\n\tDate:$curdate\n\tTo:$recipients" \
                                > $pending_f
```

```
        fi
        echo "\
Solaris Volume Manager: $node: metacheck$setarg: Report: $curdate"        >> $msg_f
        echo "\
------------------------------------------------------------" >> $msg_f
        cat $msg_f $msgs_f | mailx -s \
        "${re}Solaris Volume Manager Problem: metacheck.$who.$set.$node" $recipients
    fi
    else
    cat $msgs_f
    fi
else
    # no problems detected,
    if [ -n "$recipients" ] ; then
    # default is to not send any mail, or print anything.
    echo "\
Solaris Volume Manager: $node: metacheck$setarg: Report: $curdate"        >> $msg_f
    echo "\
------------------------------------------------------------" >> $msg_f
    if [ -f $pending_f ] && [ "$filter" = "yes" ] ; then
        # pending filter exista, remove it and send OK
        rm -f $pending_f                    > /dev/null 2>&1
        echo "Problem resolved"                >> $msg_f
        cat $msg_f | mailx -s \
        "Re: Solaris Volume Manager Problem: metacheck.$who.$node.$set" $recipients
    elif [ "$testarg" = "yes" ] ; then
        # for testing, send mail every time even thought there is no problem
        echo "Messaging test, no problems detected"        >> $msg_f
        cat $msg_f | mailx -s \
        "Solaris Volume Manager Problem: metacheck.$who.$node.$set" $recipients
    fi
    else
    echo "metacheck: Okay"
    fi
fi

rm -f $files                            > /dev/null 2>&1
exit $retval
```

cron ユーティリティーを使ってスクリプトを起動する手順については、cron(1M) のマニュアルページを参照してください。

◆ ◆ ◆ 第 25 章

Solaris ボリュームマネージャのトラブルシューティング (作業)

この章では、Solaris ボリュームマネージャに関連する問題の解決方法について説明します。また、この章では、トラブルシューティングの一般的な指針と、既知の問題を解決するための具体的な手順も示します。

この章では、次の内容について説明します。

この章では、Solaris ボリュームマネージャの問題とその解決方法について説明しますが、すべてを扱うわけではなく、一般的なシナリオと回復手順を紹介します。

Solaris ボリュームマネージャのトラブルシューティング (作業マップ)

次の表に、Solaris ボリュームマネージャのトラブルシューティングに必要な作業を示します。

作業	説明	参照先
不良ディスクを交換する	ディスクを交換してから、新しいディスク上の状態データベースの複製と論理ボリュームを更新します。	294 ページの「不良ディスクを交換するには」
ディスク移動の問題から回復する	ディスクを元の場所に戻すか、製品サポートに連絡します。	296 ページの「ディスク移動の問題からの回復」
/etc/vfstab 内の不適切なエントリを修正する	ミラーに対して fsck コマンドを実行してから、システムが正しくブートするように /etc/vfstab ファイルを編集します。	301 ページの「/etc/vfstab 内の不適切なエントリを修正するには」
ブートデバイスの障害から回復する	ほかのサブミラーからブートします。	303 ページの「ブートデバイスの障害から回復するには」
状態データベースの複製数の不足から回復する	metadb コマンドを使って、使用不能な複製を削除します。	307 ページの「状態データベースの複製数の不足から回復するには」
ソフトパーティションの失われた構成データを復元する	metarecover コマンドを使って、ソフトパーティションの構成データを復元します。	310 ページの「ソフトパーティションの構成データを復元するには」
別のディスクから Solaris ボリュームマネージャ構成を復元する	新しいシステムにディスクを追加し、既存の状態データベースの複製から Solaris ボリュームマネージャ構成を再構築します。	313 ページの「ローカルディスクセットから記憶領域を回復するには」
別のシステムから記憶領域を回復する	既知のディスクセットから別のシステムへ記憶領域をインポートします。	313 ページの「別のシステムからの記憶領域の回復」
アクセスできないディスクセットを削除する	metaset コマンドを使用して、取得または使用できないディスクセットのレコードを削除します。	320 ページの「ディスクセットの問題からの回復」
Solaris ボリュームマネージャボリュームに格納されたシステム構成を復元する	Solaris OS インストールメディアを使用して、Solaris ボリュームマネージャボリュームに格納されたシステム構成を復元します。	323 ページの「システムの復元」

トラブルシューティングの概要

トラブルシューティングの前提条件

Solaris ボリュームマネージャの記憶領域管理に関連する問題を解決するには、次の条件を満たしている必要があります。

- root 権限を持っている
- すべてのデータの最新バックアップを取っている

Solaris ボリュームマネージャによるトラブルシューティングの一般的な指針

Solaris ボリュームマネージャでトラブルシューティングを行うときは、次の情報を用意してください。

- `metadb` コマンドの出力
- `metastat` コマンドの出力
- `metastat -p` コマンドの出力
- `/etc/vfstab` ファイルのバックアップコピー
- `/etc/lvm/mddb.cf` ファイルのバックアップコピー
- `prtvtoc` コマンド (SPARC® システム) または `fdisk` コマンド (x86 ベースのシステム) で出力されるディスクパーティションの情報
- システムで稼働している Solaris のバージョン
- Solaris のインストール済みパッチを記したリスト
- Solaris ボリュームマネージャのインストール済みパッチを記したリスト

ヒント – Solaris ボリュームマネージャ構成を更新したり、記憶領域やオペレーティングシステムに関連するその他の変更をシステムに適用した場合は、その構成情報の最新コピーを生成してください。`cron` ジョブを使えば、この情報を自動的に生成できます。

一般的なトラブルシューティング方法

1 つの手順で Solaris ボリュームマネージャに関連するすべての問題を検証できるわけではありませんが、一般には次の手順に従って障害を追跡します。

1. 現在の構成に関する情報を収集します。
2. `metastat` や `metadb` コマンドの出力など、最新の状態情報を調べます。この情報から、問題のあるコンポーネントがわかります。
3. 障害が起こりそうなハードウェア部分をチェックします
   - すべてのハードウェアが適切に接続されているか、
   - 最近、停電がなかったか
   - 機器を変更または追加しなかったか

ディスクの交換

この節では、Solaris ボリュームマネージャ環境でのディスク交換の方法について説明します。

注意 – 不良ディスク上または不良ディスク上に構築したボリューム上でソフトパーティションを使用していた場合は、新しいディスクを交換対象ディスクと同じ物理位置に配置し、交換対象ディスクと同じ *cntndn* 番号を使用する必要があります。

▼ 不良ディスクを交換するには

1. `/var/adm/messages` ファイルと `metastat` コマンドの出力を調べて、交換すべき障害ディスクを特定します。

2. 不良ディスクに状態データベースの複製がないかチェックします。

   `metadb` コマンドを使って複製を表示します。

   `metadb` コマンドの出力を見ると、不良ディスクの状態データベースの複製にエラーが表示されていることがわかります。この例では、`c0t1d0` のデバイスに問題があります。

   ```
   # metadb
      flags       first blk        block count
     a m     u         16               1034            /dev/dsk/c0t0d0s4
     a       u         1050             1034            /dev/dsk/c0t0d0s4
     a       u         2084             1034            /dev/dsk/c0t0d0s4
     W   pc luo        16               1034            /dev/dsk/c0t1d0s4
     W   pc luo        1050             1034            /dev/dsk/c0t1d0s4
     W   pc luo        2084             1034            /dev/dsk/c0t1d0s4
   ```

   この出力から、ローカルディスク `c0t0d0` と `c0t1d0` の各スライス 4 に、状態データベースの複製が 3 つあることがわかります。`c0t1d0s4` スライスのフラグフィールドの `W` は、このデバイスに書き込みエラーがあることを示しています。`c0t0d0s4` スライスの 3 つの複製は正常です。

3. 状態データベースの複製が配置されているスライスの名前と状態データベースの複製数を記録します。その後、状態データベースの複製を削除します。

   状態データベースの複製の数は、`metadb` コマンド出力に同じスライス名が表示される行数と同じです。この例では、`c0t1d0s4` にある 3 つの状態データベースの複製を削除します。

   ```
   # metadb -d c0t1d0s4
   ```

注意 – 不良の状態データベースの複製を削除すると、複製の数が 3 以下になることがあります。その場合は、状態データベースの複製を追加してから先に進みます。これによって、前と同じ構成情報が保たれます。

4 不良ディスクにホットスペアがないか探して、あれば削除します。

ホットスペアの検索には、`metastat` コマンドを使用します。この例では、`c0t1d0s6` がホットスペア集合 `hsp000` に含まれていたので、集合から削除します。

```
# metahs -d hsp000 c0t1d0s6
hsp000: Hotspare is deleted
```

5 不良ディスクを交換します。

この手順では、使用しているハードウェアと環境によって、`cfgadm` コマンド、`luxadm` コマンド、またはその他のコマンドを使用しなければならないことがあります。この手順の実行時には、使用しているハードウェアのマニュアルを参照しながら、Solaris 上でのこのディスクの状態を適切に操作してください。

6 新しいディスクのパーティションを再分割します。

`format` か `fmthard` コマンドを使い、不良ディスクと同じスライス情報に基づいてディスクをパーティション分割します。不良ディスクに対する `prtvtoc` の出力がある場合は、`fmthard -s` */tmp/failed-disk-prtvtoc-output* コマンドで新しいディスクをフォーマットできます。

7 状態データベースの複製を削除した場合は、同じ数の複製を適切なスライスに追加します。

この例では、`/dev/dsk/c0t1d0s4` を使用します。

```
# metadb -a -c 3 c0t1d0s4
```

8 ディスク上のスライスが、**RAID-5** ボリュームのコンポーネントである場合、あるいは **RAID-0** ボリュームのコンポーネントで、それが **RAID-1** ボリュームのサブミラーになっている場合は、各スライスに `metareplace -e` コマンドを実行します。

この例では、`/dev/dsk/c0t1d0s4` およびミラー `d10` を使用します。

```
# metareplace -e d10 c0t1d0s4
```

9 交換したディスクのスライス上にソフトパーティションが直接構築されている場合は、ソフトパーティションが含まれているスライスごとに `metarecover -m -p` コマンドを実行します。このコマンドによって、ディスク上にエクステントヘッダーが再作成されます。

この例では、 `/dev/dsk/c0t1d0s4` の再作成されたディスクにソフトパーティションのマーキングが必要です。そこでスライスをスキャンし、状態データベースの複製の情報に基づいてマーキングを再適用します。

```
# metarecover c0t1d0s4 -m -p
```

10 ディスク上のソフトパーティションが、**RAID-5** ボリュームのコンポーネントである場合、あるいは **RAID-0** ボリュームのコンポーネントで、それが **RAID-1** ボリュームのサブミラーになっている場合は、各スライスに `metareplace -e` コマンドを実行します。

この例では、`/dev/dsk/c0t1d0s4` およびミラー `d10` を使用します。

```
# metareplace -e d10 c0t1d0s4
```

11 **RAID-0** ボリューム上にソフトパーティションが構築されている場合は、各 **RAID-0** ボリュームに `metarecover` コマンドを実行します。

この例では、RAID-0 ボリューム `d17` にソフトパーティションが構築されています。

```
# metarecover d17 -m -p
```

12 削除されたホットスペアを置き換え、それを **1** つまたは複数の適切なホットスペア集合に追加します。

この例では、ホットスペア集合 `hsp000` に `c0t1d0s6` が含まれていました。このスライスをホットスペア集合に追加します。

```
# metahs -a hsp000 c0t1d0s6
hsp000: Hotspare is added
```

13 ソフトパーティションまたは非冗長ボリュームが障害の影響を受けた場合は、バックアップからデータを復元します。冗長ボリュームだけが影響を受けた場合は、データを検証します。

すべてのボリュームのユーザーデータとアプリケーションデータを調べます。必要であれば、アプリケーションレベルの整合性チェックプログラムなどのツールを使ってデータをチェックします。

ディスク移動の問題からの回復

ここでは、Solaris ボリュームマネージャ環境でディスクを移動させたあとで発生する予想外の問題から回復する方法について説明します。

ディスク移動とデバイス ID の概要

Solaris ボリュームマネージャでは、特定のディスクと対応づけられたデバイス ID を使用して、Solaris ボリュームマネージャ構成で使用されているあらゆるディスクを追跡します。ディスクを別のコントローラに移した場合、または SCSI ターゲット番号が変更された場合、通常は Solaris ボリュームマネージャが移動を正しく認識して、関連するすべての Solaris ボリュームマネージャレコードを相応に更新します。システム管理者の介入は不要です。とはいえ、まれに、Solaris ボリュームマネージャがレコードを正しく更新できず、ブート時にエラーが通知されることがあります。

名前のないデバイスに関するエラーメッセージを解決するには

新しいハードウェアを追加したり、ハードウェアを移動させると (あるコントローラから別のコントローラに一連のディスクを移動させた場合など)、Solaris ボリュームマネージャが、移動されたディスクに対応するデバイス ID を調べ、内部 Solaris ボリュームマネージャレコードの *cnt ndn* 名を適切に更新します。レコードを更新できなかった場合は、`svc:/system/mdmonitor` サービスによって生成されたブートプロセスがブート時にコンソールに対してエラーを伝えます。

```
Unable to resolve unnamed devices for volume management.
Please refer to the Solaris Volume Manager documentation,
Troubleshooting section, at http://docs.sun.com or from
your local copy.
```

この問題によってデータが消失したわけでも、特に何かが起きるわけでもありません。このエラーメッセージは、Solaris ボリュームマネージャの名前レコードが部分的に更新されたことを意味します。`metastat` コマンドの出力に、前に使用されていた *cntndn* 名の一部が示されます。この出力には、移動後の状態が反映された *cntndn* 名の一部も示されます。

この条件下で Solaris ボリュームマネージャ構成の更新が必要になった場合は、`meta*` コマンドを実行する際に、必ず metastat コマンドの出力どおりの *cntndn* 名を使用してください。

このエラー条件が発生した場合、次の方法のどちらかで解決できます。

- すべてのディスクを元の場所に戻す。次に、再構成時のリブートを実行するか、(単一コマンドとして) 次のコマンドを実行する

  ```
  /usr/sbin/devfsadm && /usr/sbin/metadevadm -r
  ```

  これらのコマンドが完了すると、エラー条件が解消されます。
- サポートの担当者に連絡し、指示を受ける

注 – このエラー条件はめったに発生しません。万一発生した場合は、高い確率で、ファイバチャネルに接続された記憶装置が影響を受けます。

Solaris 10 リリースにアップグレードした場合のデバイス ID の不一致

Solaris 10 リリースから、デバイス ID 出力の表示形式が新しくなりました。Solaris ボリュームマネージャによるデバイス ID 出力の表示が新旧どちらの形式であるかは、デバイス ID 情報が状態データベースの複製に追加された時期によって決まります。

デバイス ID はこれまで 16 進値で表示されていました。新しい形式では、ASCII 文字列でデバイス ID が表示されます。次の例からわかるように、通常、変化はわずかです。

旧形式　`id1,ssd@w 600c0ff00000000007ecd255a9336d00`

新形式　`id1,ssd@n 600c0ff00000000007ecd255a9336d00`

次の例のように、変化が大きい場合もないわけではありません。

旧形式
```
id1,sd@w484954414348492044 4b3332454a2d33364 \
e432020202020343333423938 3939
```

新形式　`id1,ssd@n600c0ff00000000007ecd255a9336d00`

Solaris 10 リリースにアップグレードした場合、既存のディスクセットに対応するデバイス IDは、Solaris ボリュームマネージャ構成で更新されません。以前の Solaris リリースに戻さなければならなくなった場合、アップグレード後にディスクセットに対して行った構成の変更は、以前のリリースでは使用できないことがあります。該当する構成変更は、次のとおりです。

- アップグレード以前に存在していたディスクセットへの新しいディスクの追加
- 新規ディスクセットの作成
- 状態データベースの複製の作成

このような構成変更は、ローカルディスクセットを含め、Solaris ボリュームマネージャで作成可能なあらゆるディスクセットに影響を与えます。たとえば、Solaris 10 リリースで作成したディスクセットに対してこのような変更を実行した場合、そのディスクセットは以前のリリースにインポートできません。もう 1 つ例を挙げます。ミラー化されたルートの片側を Solaris 10 リリースにアップグレードし、さらにローカルディスクセットに対して構成変更を行うことがあります。その後、サブミラーを以前のリリースに戻した場合、このような変更は認識されません。

Solaris 10 OS の構成には、アップグレードの場合も含めて、新しい形式のデバイス ID が示されます。`prtconf -v` コマンドを使用すると、この情報を表示できます。一方、Solaris ボリュームマネージャは、旧形式と新形式のどちらでも表示します。Solaris ボリュームマネージャでどちらの形式が表示されるかは、ディスクを使い始めたときに稼働していた Solaris OS のバージョンによって決まります。Solaris ボリュームマネージャが Solaris 10 リリースのデバイス ID と形式は異なるが同じ内容を示しているかどうかを判断するには、`metastat` コマンドの出力と `prtconf -v` コマンドの出力を比較します。

次の例では、metastat コマンドの出力は、同じディスクに対する prtconf -v コマンドの出力に含まれている c1t6d0 のデバイス ID と形式は異なりますが、内容は同じです。

```
# metastat
d127: Concat/Stripe
    Size: 17629184 blocks (8.4 GB)
    Stripe 0:
        Device     Start Block  Dbase   Reloc
        c1t6d0s2      32768     Yes     Yes

Device Relocation Information:
Device   Reloc  Device ID
 c1t6d0   Yes   id1,sd@w48495441434848920444b3332454a2d33364e43202020202034333334239383939
```

```
# prtconf -v
.(output truncated)

.
.
sd, instance #6
         System properties:
              name='lun' type=int items=1
                 value=00000000
              name='target' type=int items=1
                 value=00000006
              name='class' type=string items=1
                 value='scsi'
         Driver properties:
              name='pm-components' type=string items=3 dev=none
                 value='NAME=spindle-motor' + '0=off' + '1=on'
              name='pm-hardware-state' type=string items=1 dev=none
                 value='needs-suspend-resume'
              name='ddi-failfast-supported' type=boolean dev=none
              name='ddi-kernel-ioctl' type=boolean dev=none
              Hardware properties:
              name='devid' type=string items=1
                 value='id1,@THITACHI_DK32EJ-36NC_____433B9899'
.
.
.
(output truncated)
```

prtconf -v コマンドの出力に含まれている「instance #6」の行は、metastat コマンドの出力に含まれているディスク c1t6d0 と対応しています。prtconf -v コマンドの出力に含まれているデバイス ID id1,@THITACHI_DK32EJ-36NC_____433B9899 は、デバイス ID id1,sd@w48495441434848920444b3332454a2d33364e43202020202034333334239383939 という、metastat コマンドの出力と対応しています。この出力の相違は、Solaris ボリュームマネー

ジャが metastat コマンドの出力では 16 進形式でデバイス ID を表示し、Solaris 10 OS 構成では prtconf コマンドの出力として ASCII 文字列を表示することを示します。

ブート障害からの回復

Solaris ボリュームマネージャではルート (/)、 swap、および /usr ディレクトリをミラー化できるので、システムのブート時に特殊な問題が発生することがあります。原因はハードウェア障害またはオペレータエラーのどちらかです。この節では、このような障害に対処するための手順について説明します。

次の表に、そのような障害の原因と適切な解決方法を示します。

表 25–1 Solaris ボリュームマネージャで一般的なブート障害

障害の原因	参照先
/etc/vfstab ファイルの情報が正しくない	301 ページの「/etc/vfstab 内の不適切なエントリを修正するには」
十分な数の状態データベースの複製が定義されていない	307 ページの「状態データベースの複製数の不足から回復するには」
ブートデバイス (ディスク) に障害が発生した	303 ページの「ブートデバイスの障害から回復するには」

ブート障害の背景情報

- エラーのためにボリュームが Solaris ボリュームマネージャによってオフラインにされた場合は、障害が発生したディスクにあるすべてのファイルシステムのマウントを解除する必要があります。

  個々のディスクスライスは独立しているため、 同じディスクに複数のファイルシステムがマウントされていることがあります。ドライバソフトウェアに障害が発生した場合には、同じディスクの他のスライスでもまもなく障害が発生するはずです。ディスクスライスに直接マウントされているファイルシステムは、Solaris ボリュームマネージャによるエラー処理の対象になりません。このようなファイルシステムをマウントしたままにしておくと、システムクラッシュによってデータを失う可能性があります。
- サブミラーを無効な状態やオフラインのままにしておく時間を最小限に抑えます。再同期やオンラインバックアップの処理中は、ミラー化による保護は不完全になります。

/etc/vfstab 内の不適切なエントリを修正するには

ルート (/) ファイルシステムをミラー化するときなどに、/etc/vfstab ファイルに不適切なエントリを作成した場合、システムは一見、正常にブートしているように見えますが、あとからシステム障害が発生します。この問題を解決するためには、シングルユーザーモードで /etc/vfstab ファイルを編集する必要があります。

/etc/vfstab ファイル内の不適切なエントリを修正するおおよその手順は次のとおりです。

1. シングルユーザーモードでシステムをブートします。
2. ミラーボリュームに対して fsck コマンドを実行します。
3. 読み取りオプションと書き込みオプションを有効にして、ファイルシステムをマウントし直します。
4. (省略可能)ルート (/) ミラーに対して metaroot コマンドを実行します。
5. /etc/vfstab ファイル内のファイルシステムエントリがこのボリュームを参照していることを確認します。
6. システムのリブート

▼ ルート (/) RAID-1 (ミラー) ボリュームを回復する

次の例では、ルート (/) ファイルシステムが 2 面ミラー d0 でミラー化されています。/etc/vfstab ファイルのルート (/) エントリは、ファイルシステムの元のスライスに戻っています。しかし、/etc/system ファイルの情報は、ミラー d0 からブートすることを示しています。この状況は通常、metaroot コマンドを使用しないで /etc/system ファイルと /etc/vfstab ファイルを保守した場合に発生します。または、/etc/vfstab ファイルの古いコピーを現在の /etc/vfstab ファイルにコピーしたことが原因になる場合もあります。

不適切な /etc/vfstab ファイルの例を示します。

```
#device        device          mount           FS      fsck   mount    mount
#to mount      to fsck         point           type    pass   at boot  options
#
/dev/dsk/c0t3d0s0 /dev/rdsk/c0t3d0s0  /       ufs      1     no       -
/dev/dsk/c0t3d0s1 -                   -       swap     -     no       -
/dev/dsk/c0t3d0s6 /dev/rdsk/c0t3d0s6  /usr    ufs      2     no       -
#
/proc             -                  /proc    proc     -     no       -
swap              -                  /tmp     tmpfs    -     yes      -
```

エラーがあるために、システムは、ブート時に自動的にシングルユーザーモードになります。

```
ok boot
...
configuring network interfaces: hme0.
Hostname: host1
mount: /dev/dsk/c0t3d0s0 is not this fstype.
setmnt: Cannot open /etc/mnttab for writing

INIT: Cannot create /var/adm/utmp or /var/adm/utmpx

INIT: failed write of utmpx entry:"  "

INIT: failed write of utmpx entry:"  "

INIT: SINGLE USER MODE

Type Ctrl-d to proceed with normal startup,
(or give root password for system maintenance): <root-password>
```

この時点で、ルート (/) ファイルシステムと /usr ファイルシステムは、読み取り専用としてマウントされています。次の手順を実行します。

1 ルート (/) ミラーに対して fsck コマンドを実行します。

注 – ルート (/) ミラーに正しいボリュームを使用するように注意してください。

```
# fsck /dev/md/rdsk/d0
** /dev/md/rdsk/d0
** Currently Mounted on /
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Cyl groups
2274 files, 11815 used, 10302 free (158 frags, 1268 blocks,
0.7% fragmentation)
```

2 ルート (/) ファイルシステムを読み取り/書き込みファイルシステムとしてマウントし直し、 /etc/vfstab ファイルを編集できるようにします。

```
# mount -o rw,remount /dev/md/dsk/d0 /
mount: warning: cannot lock temp file </etc/.mnt.lock>
```

3 metaroot コマンドを実行します。

```
# metaroot d0
```

このコマンドを実行すると、/etc/system ファイルと /etc/vfstab ファイルが編集され、ルート (/) ファイルシステムのエントリがボリューム d0 を参照します。

4 /etc/vfstab ファイルに正しいボリュームのエントリが含まれていることを確認します。

/etc/vfstab ファイルのルート (/) エントリは次のようになっているはずです。これによって、ファイルシステムのエントリは RAID-1 ボリュームを正しく参照します。

```
#device           device              mount    FS      fsck   mount   mount
#to mount         to fsck             point    type    pass   at boot options
#
/dev/md/dsk/d0    /dev/md/rdsk/d0     /        ufs     1      no      -
/dev/dsk/c0t3d0s1 -                   -        swap    -      no      -
/dev/dsk/c0t3d0s6 /dev/rdsk/c0t3d0s6  /usr     ufs     2      no      -
#
/proc             -                  /proc     proc    -      no      -
swap              -                  /tmp      tmpfs   -      yes     -
```

5 システムをリブートします。

システムは正常な動作に戻ります。

▼ ブートデバイスの障害から回復するには

ルート (/) がミラー化されているシステムのブートデバイスに障害がある場合は、代替ブートデバイスを設定する必要があります。

この手順の概要は次のとおりです。

- 代替ルート (/) サブミラーからシステムをブートします。
- エラーのある状態データベースの複製とボリュームを特定します。
- 不良ディスクを修復します。
- 状態データベースの複製とボリュームを元の状態に復元します。

ブートデバイスに障害が発生すると、最初に次のようなメッセージが表示されます。このメッセージは、アーキテクチャーによってこれとは異なる場合があります。

```
Rebooting with command:
Boot device: /iommu/sbus/dma@f,81000/esp@f,80000/sd@3,0
The selected SCSI device is not responding
Can't open boot device
...
```

このメッセージが表示された場合は、このデバイス名を書き留めておき、次の手順を実行します。

1 ルート (/) の別のサブミラーからシステムをブートします。

この例では 2 つの状態データベースの複製だけが異常であるため、システムをブートすることができます。起動できない場合は、アクセスできない状態データベースの複製をシングルユーザーモードで削除する必要があります。この手順については、307 ページの「状態データベースの複製数の不足から回復するには」を参照してください。

ルート (/) ファイルシステムのミラーを作成したときに、その手順の中で代替ブートデバイスを書き留めてあるはずです。この例では、disk2 が代替ブートデバイスになります。

```
ok boot disk2
SunOS Release 5.9 Version s81_51 64-bit
Copyright 1983-2001 Sun Microsystems, Inc.  All rights reserved.
Hostname: demo
...
demo console login: root
Password: <root-password>
Dec 16 12:22:09 host1 login: ROOT LOGIN /dev/console
Last login: Wed Dec 12 10:55:16 on console
Sun Microsystems Inc.   SunOS 5.9       s81_51  May 2002
...
```

2 metadb コマンドを使って、状態データベースの使用不能な複製数を判別します。

```
# metadb
       flags         first blk    block count
    M     p          unknown      unknown      /dev/dsk/c0t3d0s3
    M     p          unknown      unknown      /dev/dsk/c0t3d0s3
    a m  p  luo      16           1034         /dev/dsk/c0t2d0s3
    a    p  luo      1050         1034         /dev/dsk/c0t2d0s3
    a    p  luo      16           1034         /dev/dsk/c0t1d0s3
    a    p  luo      1050         1034         /dev/dsk/c0t1d0s3
```

この例では、不良ディスクの一部であるスライス /dev/dsk/c0t3d0s3 上の状態データベースの複製は、システムから認識されません。

3 metastat コマンドを使って、ルート (/)、swap、および /usr の各ミラーの半分が使用不能であることを確認します。

```
# metastat
d0: Mirror
    Submirror 0: d10
      State: Needs maintenance
    Submirror 1: d20
      State: Okay
...

d10: Submirror of d0
    State: Needs maintenance
    Invoke: "metareplace d0 /dev/dsk/c0t3d0s0 <new device>"
    Size: 47628 blocks
    Stripe 0:
    Device              Start Block  Dbase State        Hot Spare
    /dev/dsk/c0t3d0s0          0     No    Maintenance

d20: Submirror of d0
    State: Okay
    Size: 47628 blocks
```

```
    Stripe 0:
    Device              Start Block  Dbase State        Hot Spare
    /dev/dsk/c0t2d0s0          0     No    Okay

d1: Mirror
    Submirror 0: d11
      State: Needs maintenance
    Submirror 1: d21
      State: Okay
...

d11: Submirror of d1
    State: Needs maintenance
    Invoke: "metareplace d1 /dev/dsk/c0t3d0s1 <new device>"
    Size: 69660 blocks
    Stripe 0:
    Device              Start Block  Dbase State        Hot Spare
    /dev/dsk/c0t3d0s1          0     No    Maintenance

d21: Submirror of d1
    State: Okay
    Size: 69660 blocks
    Stripe 0:
    Device              Start Block  Dbase State        Hot Spare
    /dev/dsk/c0t2d0s1          0     No    Okay

d2: Mirror
    Submirror 0: d12
      State: Needs maintenance
    Submirror 1: d22
      State: Okay
...

d12: Submirror of d2
    State: Needs maintenance
    Invoke: "metareplace d2 /dev/dsk/c0t3d0s6 <new device>"
    Size: 286740 blocks
    Stripe 0:
    Device              Start Block  Dbase State        Hot Spare
    /dev/dsk/c0t3d0s6          0     No    Maintenance


d22: Submirror of d2
    State: Okay
    Size: 286740 blocks
    Stripe 0:
    Device              Start Block  Dbase State        Hot Spare
    /dev/dsk/c0t2d0s6          0     No    Okay
```

この例では、次のサブミラーの保守が必要であることがわかります。

- サブミラー d10、デバイス c0t3d0s0
- サブミラー d11、デバイス c0t3d0s1
- サブミラー d12、デバイス c0t3d0s6

4 システムを停止して、ディスクを交換します。format か fmthard コマンドを使って、障害の発生前と同じようにディスクをパーティション分割します。

ヒント–新しいディスクが既存ディスクとまったく同じ場合 (この例では、ミラーの変更のない側)、ただちに新しいディスクをフォーマットします。フォーマットするには、prtvtoc /dev/rdsk/c0t2d0s2 | fmthard -s - /dev/rdsk/c0t3d0s2 コマンドを使用します (この例では c0t3d0)。

```
# halt
...
Halted
...
ok boot
...
# format /dev/rdsk/c0t3d0s0
```

5 システムをリブートします。

ここでは、ルート (/) ミラーの他方の半分からリブートする必要があります。この代替ブートデバイスは、ミラーを作成したときに書き留めておいたものです。

```
# halt
...
ok boot disk2
```

6 metadb コマンドを使って、使用不能な状態データベースの複製を削除してから、新しい複製を追加します。

```
# metadb
       flags         first blk    block count
    M     p          unknown      unknown      /dev/dsk/c0t3d0s3
    M     p          unknown      unknown      /dev/dsk/c0t3d0s3
    a m  p  luo      16           1034         /dev/dsk/c0t2d0s3
    a    p  luo      1050         1034         /dev/dsk/c0t2d0s3
    a    p  luo      16           1034         /dev/dsk/c0t1d0s3
    a    p  luo      1050         1034         /dev/dsk/c0t1d0s3
# metadb -d c0t3d0s3
# metadb -c 2 -a c0t3d0s3
# metadb
       flags         first blk    block count
     a m  p  luo     16           1034         /dev/dsk/c0t2d0s3
     a    p  luo     1050         1034         /dev/dsk/c0t2d0s3
     a    p  luo     16           1034         /dev/dsk/c0t1d0s3
```

```
     a    p  luo     1050         1034         /dev/dsk/c0t1d0s3
     a       u       16           1034         /dev/dsk/c0t3d0s3
     a       u       1050         1034         /dev/dsk/c0t3d0s3
```

7 metareplace コマンドを使って、サブミラーを再び有効にします。

```
# metareplace -e d0 c0t3d0s0
Device /dev/dsk/c0t3d0s0 is enabled

# metareplace -e d1 c0t3d0s1
Device /dev/dsk/c0t3d0s1 is enabled

# metareplace -e d2 c0t3d0s6
Device /dev/dsk/c0t3d0s6 is enabled
```

しばらくすると、再同期処理が終了します。これで、また元のデバイスからブートできるようになります。

状態データベースの複製の障害からの回復

ドライブ障害などのために、状態データベースの複製の数が規定数に達していないと、システムをマルチユーザーモードでリブートできなくなります。この状況は、Solaris ボリュームマネージャが状態データベースの使用可能な複製が半数に満たないことを検出したときに、パニックに続いて発生することがあります。状態データベースの使用可能な複製が半数かそれより少ない場合にシステムをリブートしたときにも、この状況が発生することがあります。Solaris ボリュームマネージャでは、これを、状態データベースの複製が「無効」状態になったといいます。この問題から回復するための手順を次に示します。

▼ 状態データベースの複製数の不足から回復するには

1 システムをブートします。

2 使用不能な状態データベースの複製を特定します。

```
# metadb -i
```

3 1 つまたは複数のディスクが使用不能であることがわかっている場合は、それらのディスク上の状態データベースの複製をすべて削除します。そうでない場合は、障害のある状態データベースの複製 (metadb の出力のステータスフラグ **W**、**M**、**D**、**F**、**R** で示される) を削除して、状態データベースの複製の過半数が正常なものであるようにします。

```
# metadb -d disk-slice
```

ヒント–大文字の状態フラグが設定されている状態データベースの複製は、エラー状態です。小文字の状態フラグが設定されている状態データベースの複製は、正常に動作しています。

4 複製が削除されたことを確認します。

```
# metadb
```

5 システムをリブートします。

```
# reboot
```

6 必要であれば、ディスクを交換し、適切にフォーマットしてから、必要な状態データベースの複製をディスクに追加します。

72 ページの「状態データベースの複製の作成」の指示に従います。

交換用のディスクが用意できたら、システムを停止し、不良ディスクを交換してから、システムをリブートします。`format` か `fmthard` コマンドを使って、障害の発生前と同じようにディスクをパーティション分割します。

例 25–1 状態データベースの複製数の不足から回復する

この例では、7 つの状態データベースの複製を含むディスクに障害が発生したものとします。その結果、システムに与えられている正常な複製は 3 つだけです。システムパニックが発生し、マルチユーザーモードでリブートできなくなります。

```
panic[cpu0]/thread=70a41e00: md: state database problem

403238a8 md:mddb_commitrec_wrapper+6c (2, 1, 70a66ca0, 40323964, 70a66ca0, 3c)
  %l0-7: 0000000a 00000000 00000001 70bbcce0 70bbcd04 70995400 00000002 00000000
40323908 md:alloc_entry+c4 (70b00844, 1, 9, 0, 403239e4, ff00)
  %l0-7: 70b796a4 00000001 00000000 705064cc 70a66ca0 00000002 00000024 00000000
40323968 md:md_setdevname+2d4 (7003b988, 6, 0, 63, 70a71618, 10)
  %l0-7: 70a71620 00000000 705064cc 70b00844 00000010 00000000 00000000 00000000
403239f8 md:setnm_ioctl+134 (7003b968, 100003, 64, 0, 0, ffbffc00)
  %l0-7: 7003b988 00000000 70a71618 00000000 00000000 000225f0 00000000 00000000
40323a58 md:md_base_ioctl+9b4 (157ffff, 5605, ffbffa3c, 100003, 40323ba8, ff1b5470)
  %l0-7: ff3f2208 ff3f2138 ff3f26a0 00000000 00000000 00000064 ff1396e9 00000000
40323ad0 md:md_admin_ioctl+24 (157ffff, 5605, ffbffa3c, 100003, 40323ba8, 0)
  %l0-7: 00005605 ffbffa3c 00100003 0157ffff 0aa64245 00000000 7efefeff 81010100
40323b48 md:mdioctl+e4 (157ffff, 5605, ffbffa3c, 100003, 7016db60, 40323c7c)
  %l0-7: 0157ffff 00005605 ffbffa3c 00100003 0003ffff 70995598 70995570 0147c800
40323bb0 genunix:ioctl+1dc (3, 5605, ffbffa3c, ffffff8, ffffffe0, ffbffa65)
  %l0-7: 0114c57c 70937428 ff3f26a0 00000000 00000001 ff3b10d4 0aa64245 00000000

panic:
stopped at      edd000d8:       ta      %icc,%g0 + 125
```

```
Type  ’go’ to resume

ok boot -s
Resetting ...

Sun Ultra 5/10 UPA/PCI (UltraSPARC-IIi 270MHz), No Keyboard
OpenBoot 3.11, 128 MB memory installed, Serial #9841776.
Ethernet address 8:0:20:96:2c:70, Host ID: 80962c70.



Rebooting with command: boot -s
Boot device: /pci@1f,0/pci@1,1/ide@3/disk@0,0:a  File and args: -s
SunOS Release 5.9 Version s81_39 64-bit

Copyright 1983-2001 Sun Microsystems, Inc.  All rights reserved.
configuring IPv4 interfaces: hme0.
Hostname: dodo

metainit: dodo: stale databases

Insufficient metadevice database replicas located.

Use metadb to delete databases which are broken.
Ignore any "Read-only file system" error messages.
Reboot the system when finished to reload the metadevice database.
After reboot, repair any broken database replicas which were deleted.

Type control-d to proceed with normal startup,
(or give root password for system maintenance): root-password
single-user privilege assigned to /dev/console.
Entering System Maintenance Mode

Jun  7 08:57:25 su: ’su root’ succeeded for root on /dev/console
Sun Microsystems Inc.   SunOS 5.9       s81_39  May 2002
# metadb -i
        flags           first blk       block count
     a m  p  lu         16              8192            /dev/dsk/c0t0d0s7
     a    p  l          8208            8192            /dev/dsk/c0t0d0s7
     a    p  l          16400           8192            /dev/dsk/c0t0d0s7
    M     p             16              unknown         /dev/dsk/c1t1d0s0
    M     p             8208            unknown         /dev/dsk/c1t1d0s0
    M     p             16400           unknown         /dev/dsk/c1t1d0s0
    M     p             24592           unknown         /dev/dsk/c1t1d0s0
    M     p             32784           unknown         /dev/dsk/c1t1d0s0
    M     p             40976           unknown         /dev/dsk/c1t1d0s0
    M     p             49168           unknown         /dev/dsk/c1t1d0s0
# metadb -d c1t1d0s0
```

```
# metadb
        flags           first blk       block count
     a m  p  lu         16              8192            /dev/dsk/c0t0d0s7
     a    p  l          8208            8192            /dev/dsk/c0t0d0s7
     a    p  l          16400           8192            /dev/dsk/c0t0d0s7
#
```

システムパニックが発生するのは、スライス /dev/dsk/c1t1d0s0 上で状態データベースの複製を検出できなくなったからです。このスライスは障害ディスクに含まれているか、または障害コントローラに接続しています。最初の metadb - i コマンドによって、このスライス上の複製のマスターブロックに問題があることがわかります。

無効状態の状態データベースの複製を削除する場合、ルート (/) ファイルシステムは読み取り専用になっています。mddb.cf エラーメッセージが表示されますが、無視してください。

この時点で、システムは再び動作状態になります。ただし、状態データベースの複製は、本来の数より少なくなっているはずです。障害記憶領域の一部を使用していたボリュームも、障害、エラー、またはホットスペア使用のいずれかになります。このような問題は、速やかに解決しなければなりません。

ソフトパーティション障害からの回復

ここでは、ソフトパーティションの構成情報を復元する方法について説明します。次の手順を使用するのは、状態データベースの複製がすべて失われていて、なおかつ次のいずれもない場合に限定してください。

- metastat -p の出力の最新コピーまたは正確なコピー
- md.cf ファイルの最新コピーまたは正確なコピー
- 最新の md.tab ファイル

▼ ソフトパーティションの構成データを復元するには

各ソフトパーティションの先頭部分には、ソフトパーティションエクステントの開始位置を示すセクターがあります。このような隠れたセクターを「エクステントヘッダー」といいます。このヘッダーはソフトパーティションのユーザーには見えません。Solaris ボリュームマネージャの構成データがすべて失われた場合には、ディスクをスキャンして構成データを生成することができます。

この手順は、失われたソフトパーティションの構成情報を復元する最後の手段です。したがって、metarecover コマンドは、metadb ファイルと md.cf ファイルの両方が失われており、また md.tab ファイルも失われているか、md.tab ファイルが古い場合にのみ使用します。

注 – この手順が有効なのは、ソフトパーティション情報を復元する場合だけです。この手順で、その他の失われた構成を復元したり、他の Solaris ボリュームマネージャボリュームの構成情報を復元したりすることはできません。

注 – ソフトパーティション上に他の Solaris ボリュームマネージャボリュームが作成されている場合には、ソフトパーティションを復元してから他のボリュームを復元する必要があります。

ソフトパーティションの構成情報は、デバイスと状態データベースに格納されています。どちらかのソースが破壊されている可能性があるため、どちらが信頼できるソースなのかを metarecover コマンドに知らせる必要があります。

最初に、metarecover コマンドを使って 2 つのソースが一致するかどうかを判定します。両者が一致する場合は、metarecover コマンドを使って変更を行うことはできません。ただし、metarecover コマンドから不一致が伝えられた場合は、出力を丹念に調べ、ディスクが壊れているのか、それとも状態データベースが壊れているのかを判断する必要があります。さらに、metarecover コマンドを使用して、適切なソースに基づいて構成を再作成します。

1 156 ページの「ソフトパーティション構成の指針」を確認します。

2 metarecover コマンドを実行し、出力されたソフトパーティション回復情報を検討します。

```
# metarecover component -p -d
```

component	raw コンポーネントの *cntndnsn* 名を指定します。
-p	ソフトパーティションを再作成することを指定します。
-d	物理スライスをスキャンしてソフトパーティションのエクステントヘッダーを検出することを指定します。

例 25–2 ディスク上のエクステントヘッダーからソフトパーティションを復元する

```
# metarecover c1t1d0s1 -p -d
The following soft partitions were found and will be added to
your metadevice configuration.
 Name            Size     No. of Extents
    d10           10240         1
    d11           10240         1
    d12           10240         1
# metarecover c1t1d0s1 -p -d
The following soft partitions were found and will be added to
your metadevice configuration.
```

```
 Name            Size     No. of Extents
    d10           10240         1
    d11           10240         1
    d12           10240         1
WARNING: You are about to add one or more soft partition
metadevices to your metadevice configuration.  If there
appears to be an error in the soft partition(s) displayed
above, do NOT proceed with this recovery operation.
Are you sure you want to do this (yes/no)?yes
c1t1d0s1: Soft Partitions recovered from device.
bash-2.05# metastat
d10: Soft Partition
    Device: c1t1d0s1
    State: Okay
    Size: 10240 blocks
        Device              Start Block  Dbase Reloc
        c1t1d0s1                   0     No    Yes

        Extent              Start Block              Block count
             0                     1                   10240

d11: Soft Partition
    Device: c1t1d0s1
    State: Okay
    Size: 10240 blocks
        Device              Start Block  Dbase Reloc
        c1t1d0s1                   0     No    Yes

        Extent              Start Block              Block count
             0                 10242                   10240

d12: Soft Partition
    Device: c1t1d0s1
    State: Okay
    Size: 10240 blocks
        Device              Start Block  Dbase Reloc
        c1t1d0s1                   0     No    Yes

        Extent              Start Block              Block count
             0                 20483                   10240
```

この例では、すべての状態データベースの複製が誤って削除されているときに、ディスクから 3 つのソフトパーティションを復元します。

別のシステムからの記憶領域の回復

Solaris ボリュームマネージャ構成を元のシステムから別のシステムに回復できます。

▼ ローカルディスクセットから記憶領域を回復するには

システムの障害時には、記憶領域を別のシステムに接続し、ローカルディスクセットから完全な構成を回復できます。たとえば、6 つのディスクからなる外部ディスクパックと Solaris ボリュームマネージャ構成のシステムがあるとします。これらのディスクには、少なくとも 1 つの状態データベースの複製があります。システムの障害時には、そのディスクパックを新しいシステムに物理的に移動して、新しいシステムがその構成を認識できるように設定します。ここでは、ディスクを別のシステムに移動して、ローカルディスクセットから構成を回復する方法を示します。

注 – この回復手順を使用できるのは、Solaris 9 以降の Solaris ボリュームマネージャボリュームに限られます。

---

1 **Solaris** ボリュームマネージャ構成が含まれている **1** つまたは複数のディスクを、**Solaris** ボリュームマネージャ構成が存在していないシステムに追加します。

2 リブートしてシステムを再構成し、新たに追加したディスクをシステムが認識できるようにします。

```
# reboot -- -r
```

3 状態データベースの複製が含まれている (新たに追加したディスク上の) スライスのメジャー/マイナー番号を特定します。

`ls -lL` 出力のグループ名と日付の間にある 2 つの数字がこのスライスのメジャー/マイナー番号です。

```
# ls -Ll /dev/dsk/c1t9d0s7
brw-r-----   1 root     sys       32, 71 Dec  5 10:05 /dev/dsk/c1t9d0s7
```

4 必要であれば、`/etc/name_to_major` 内のメジャー番号を調べ、そのメジャー番号に対応するメジャー名を特定します。

```
# grep " 32" /etc/name_to_major  sd 32
```

5 有効な状態データベースの複製が新しいディスク上のどこにあるのかを **Solaris** ボリュームマネージャに指示する情報を使用して、`/kernel/drv/md.conf` ファイルを更新します。

たとえば、`mddb_bootlist1` で始まる行にある `sd` を、手順 4 で特定したメジャー名で置き換えます。また、同じ行の *71* を、手順 3 で特定したマイナー番号で置き換えます。

```
#pragma ident   "@(#)md.conf    2.2     04/02/02 SMI"
#
# Copyright 2004 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# The parameters nmd and md_nsets are obsolete.  The values for these
# parameters no longer have any meaning.
name="md" parent="pseudo" nmd=128 md_nsets=4;

# Begin MDD database info (do not edit)
mddb_bootlist1="sd:71:16:id0";
# End MDD database info (do not edit)
```

6 システムをリブートして、**Solaris** ボリュームマネージャに構成を再ロードさせます。

次のようなメッセージがコンソールに表示されます。

```
volume management starting.
Dec  5 10:11:53 host1 metadevadm: Disk movement detected
Dec  5 10:11:53 host1 metadevadm: Updating device names in
Solaris Volume Manager
The system is ready.
```

7 構成を確認します。`metadb` コマンドを実行して、状態データベースの複製の状態を確認します。各ボリュームの状態を表示するには、`metastat` コマンドを使用します。

```
# metadb
        flags           first blk       block count
     a m  p  luo        16              8192            /dev/dsk/c1t9d0s7
     a       luo        16              8192            /dev/dsk/c1t10d0s7
     a       luo        16              8192            /dev/dsk/c1t11d0s7
     a       luo        16              8192            /dev/dsk/c1t12d0s7
     a       luo        16              8192            /dev/dsk/c1t13d0s7
# metastat
d12: RAID
    State: Okay
    Interlace: 32 blocks
    Size: 125685 blocks
Original device:
    Size: 128576 blocks
        Device              Start Block  Dbase State        Reloc  Hot Spare
        c1t11d0s3                330     No    Okay         Yes
        c1t12d0s3                330     No    Okay         Yes
        c1t13d0s3                330     No    Okay         Yes
```

```
d20: Soft Partition
    Device: d10
    State: Okay
    Size: 8192 blocks
        Extent              Start Block              Block count
             0                     3592                     8192

d21: Soft Partition
    Device: d10
    State: Okay
    Size: 8192 blocks
        Extent              Start Block              Block count
             0                    11785                     8192

d22: Soft Partition
    Device: d10
    State: Okay
    Size: 8192 blocks
        Extent              Start Block              Block count
             0                    19978                     8192

d10: Mirror
    Submirror 0: d0
      State: Okay
    Submirror 1: d1
      State: Okay
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Size: 82593 blocks

d0: Submirror of d10
    State: Okay
    Size: 118503 blocks
    Stripe 0: (interlace: 32 blocks)
        Device              Start Block  Dbase State        Reloc  Hot Spare
        c1t9d0s0                   0     No    Okay         Yes
        c1t10d0s0               3591     No    Okay         Yes


d1: Submirror of d10
    State: Okay
    Size: 82593 blocks
    Stripe 0: (interlace: 32 blocks)
        Device              Start Block  Dbase State        Reloc  Hot Spare
        c1t9d0s1                   0     No    Okay         Yes
        c1t10d0s1                  0     No    Okay         Yes
```

```
Device Relocation Information:
Device       Reloc    Device ID
c1t9d0       Yes      id1,sd@SSEAGATE_ST39103LCSUN9.0GLS3487980000U00907AZ
c1t10d0      Yes      id1,sd@SSEAGATE_ST39103LCSUN9.0GLS3397070000W0090A8Q
c1t11d0      Yes      id1,sd@SSEAGATE_ST39103LCSUN9.0GLS3449660000U00904NZ
c1t12d0      Yes      id1,sd@SSEAGATE_ST39103LCSUN9.0GLS32655400007010H04J
c1t13d0      Yes      id1,sd@SSEAGATE_ST39103LCSUN9.0GLS3461190000701001T0
#
# metadb
        flags           first blk       block count
     a m  p  luo        16              8192            /dev/dsk/c1t9d0s7
     a       luo        16              8192            /dev/dsk/c1t10d0s7
     a       luo        16              8192            /dev/dsk/c1t11d0s7
     a       luo        16              8192            /dev/dsk/c1t12d0s7
     a       luo        16              8192            /dev/dsk/c1t13d0s7
# metastat
d12: RAID
    State: Okay
    Interlace: 32 blocks
    Size: 125685 blocks
Original device:
    Size: 128576 blocks
        Device              Start Block  Dbase State        Reloc  Hot Spare
        c1t11d0s3                330     No    Okay         Yes
        c1t12d0s3                330     No    Okay         Yes
        c1t13d0s3                330     No    Okay         Yes

d20: Soft Partition
    Device: d10
    State: Okay
    Size: 8192 blocks
        Extent              Start Block              Block count
             0                     3592                    8192

d21: Soft Partition
    Device: d10
    State: Okay
    Size: 8192 blocks
        Extent              Start Block              Block count
             0                    11785                    8192

d22: Soft Partition
    Device: d10
    State: Okay
    Size: 8192 blocks
```

```
        Extent              Start Block              Block count
             0                    19978                    8192

d10: Mirror
    Submirror 0: d0
      State: Okay
    Submirror 1: d1
      State: Okay
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Size: 82593 blocks

d0: Submirror of d10
    State: Okay
    Size: 118503 blocks
    Stripe 0: (interlace: 32 blocks)
        Device              Start Block  Dbase State        Reloc  Hot Spare
        c1t9d0s0                   0     No    Okay         Yes
        c1t10d0s0               3591     No    Okay         Yes


d1: Submirror of d10
    State: Okay
    Size: 82593 blocks
    Stripe 0: (interlace: 32 blocks)
        Device              Start Block  Dbase State        Reloc  Hot Spare
        c1t9d0s1                   0     No    Okay         Yes
        c1t10d0s1                  0     No    Okay         Yes


Device Relocation Information:
Device         Reloc    Device ID
c1t9d0         Yes     id1,sd@SSEAGATE_ST39103LCSUN9.0GLS3487980000U00907AZ1
c1t10d0        Yes     id1,sd@SSEAGATE_ST39103LCSUN9.0GLS3397070000W0090A8Q
c1t11d0        Yes     id1,sd@SSEAGATE_ST39103LCSUN9.0GLS3449660000U00904NZ
c1t12d0        Yes     id1,sd@SSEAGATE_ST39103LCSUN9.0GLS32655400007010H04J
c1t13d0        Yes     id1,sd@SSEAGATE_ST39103LCSUN9.0GLS346119000070l001T0
# metastat -p
d12 -r c1t11d0s3 c1t12d0s3 c1t13d0s3 -k -i 32b
d20 -p d10 -o 3592 -b 8192
d21 -p d10 -o 11785 -b 8192
d22 -p d10 -o 19978 -b 8192
d10 -m d0 d1 1
d0 1 2 c1t9d0s0 c1t10d0s0 -i 32b
d1 1 2 c1t9d0s1 c1t10d0s1 -i 32b
#
```

既知のディスクセットから記憶領域を回復する

Solaris ボリュームマネージャにディスクセットのデバイス ID サポートが組み込まれたことによって、既知のディスクセットから記憶領域を回復したり、ディスクセットを別のシステムにインポートしたりできるようになりました。`metaimport` 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知のディスクセットをインポートできます。両方のシステムにおいて、既存の Solaris ボリュームマネージャ構成でデバイス ID がサポートされている必要があります。デバイス ID サポートの詳細については、213 ページの「ディスクセット内で非同期的に共有される記憶領域」を参照してください。
`metaimport` コマンドの詳細については、`metaimport(1M)` のマニュアルページを参照してください。

▼ インポートに利用できるディスクセットのレポートを出力するには

1. スーパーユーザーになります。
2. インポートに利用できるディスクセットについてのレポートを取得します。

```
# metaimport -r -v
```

`-r` システム上にあってインポートに利用できる未構成のディスクセットのレポートを提供します。

`-v` 状態データベースの複製の場所についての詳細な情報と、システム上にあってインポートに利用できる未構成のディスクセットのディスクの状態についての詳細な情報を提供します。

例 25–3 インポートに利用できるディスクセットを報告する

次の例では、インポートに利用できるディスクセットについてのレポートを出力する方法を示します。

```
# metaimport -r
 Drives in regular diskset including disk c1t2d0:
   c1t2d0
   c1t3d0
 More info:
   metaimport -r -v c1t2d0
Import:   metaimport -s <newsetname> c1t2d0
 Drives in replicated diskset including disk c1t4d0:
   c1t4d0
   c1t5d0
 More info:
   metaimport -r -v c1t4d0
Import:   metaimport -s <newsetname> c1t4d0
```

```
# metaimport -r -v c1t2d0
Import: metaimport -s <newsetname> c1t2d0
Last update: Mon Dec 29 14:13:35 2003
Device       offset       length replica flags
c1t2d0           16         8192      a        u
c1t3d0           16         8192      a        u
c1t8d0           16         8192      a        u
```

▼ あるシステムから別のシステムにディスクセットをインポートするには

1. スーパーユーザーになります。
2. ディスクセットがインポート可能であることを確認します。

   ```
   # metaimport -r -v
   ```

3. 利用できるディスクセットをインポートします。

   ```
   # metaimport -s diskset-name drive-name
   ```

   - s *diskset-name* 作成するディスクセットの名前です。

   drive-name インポートするディスクセットの状態データベースの複製を含むディスク (c#t#d#) を指定します。

4. ディスクセットがインポートされたことを確認します。

   ```
   # metaset -s diskset-name
   ```

例 25–4 ディスクセットのインポート

次の例では、ディスクセットをインポートする方法を示します。

```
# metaimport -s red c1t2d0
Drives in diskset including disk c1t2d0:
  c1t2d0
  c1t3d0
  c1t8d0
More info:   metaimport -r -v c1t2d0# metaset -s red


Set name = red, Set number = 1

Host                Owner
  host1             Yes
```

```
Drive    Dbase

c1t2d0   Yes

c1t3d0   Yes

c1t8d0   Yes
```

ディスクセットの問題からの回復

この節では、ディスクセットの特定の問題から回復する方法について説明します。

ディスクセットの所有権を取得できないときには

システム障害、ディスク障害、通信リンクの障害などが原因で、どのノードからもディスクセットの所有権を取得できなくなり、ディスクセットのレコードを削除できない場合には、現在のホスト上にある Solaris ボリュームマネージャ状態データベースの複製からディスクセットのレコードを削除する方法があります。

ディスクセットのレコードを削除しても、ディスクセットに含まれる状態データベースの情報には影響しないため、そのディスクセットはあとでインポートできます。このためには、`metaimport` コマンドを使用します (229 ページの「ディスクセットのインポート」を参照)。

Sun Cluster 構成からディスクセットを削除する必要がある場合は、次の手順を使用します。ただし、Sun Cluster 構成が存在しないときに使用する `-P` オプションではなく、`-C` オプションを使用します。

▼ ディスクセットを削除するには

1 `metaset` コマンドを使用して、ディスクセットの取得を試みます。

```
# metaset -s setname -t -f
```

このコマンドは、*setname* で指定したディスクセットを強制的に (`-f`) に取得 (`-t`) します。ディスクセットを取得できた場合、このコマンドは成功します。このコマンドを実行したときに別のホストがこのディスクセットを所有していた場合、このホストはパニック状態になり、データの破損や損失が回避されます。このコマンドが成功した場合、ディスクセットをきれいに削除できます。ディスクセットを削除する必要はありません。

ディスクセットを取得できなかった場合、所有権レコードを削除する必要があります。

2 metaset コマンドに -P オプションを付けて使用して、現在のホストからディスクセットを削除します。

```
# metaset -s setname -P
```

このコマンドは、コマンドを実行したホストから、*setname* で指定したディスクセットを削除 (-P) します。

3 metaset コマンドを使用して、ディスクセットが削除されたことを確認します。

```
# metaset
```

例 25–5 ディスクセットを削除する

```
host1# metaset -s red -t -f
metaset: host1: setname "red": no such set

host2# metaset

Set name = red, Set number = 1

Host                Owner
  host2

Drive    Dbase

c1t2d0   Yes

c1t3d0   Yes

c1t8d0   Yes

host2# metaset -s red -P
host2# metaset
```

参照
- 第 18 章(ディスクセットに関連する概念)
- 第 19 章(ディスクセットに関連する作業)

ufsdump コマンドによるマウント済みファイルシステムのバックアップ

次の手順で、RAID-1 ボリューム上のマウント済みファイルシステムを ufsdump コマンドを使用してバックアップする場合に、ufsdump コマンドの性能を向上させる方法について説明します。

▼ RAID-1 ボリューム上のマウント済みファイルのバックアップを実行するには

ufsdump コマンドを使用すると、RAID-1 ボリューム上のマウント済みファイルシステムのファイルをバックアップできます。バックアップユーティリティーが ufsdump の場合は、ボリュームの読み取りポリシーを「first」に設定します。バックアップの実行速度が上がります。

1 スーパーユーザーになります。

2 metastat コマンドを実行して、ミラーが「正常 (Okay)」状態であることを確認します。

```
# metastat d40
d40: Mirror
    Submirror 0: d41
      State: Okay
    Submirror 1: d42
      State: Okay
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Size: 20484288 blocks (9.8 GB)
```

ミラーが「保守 (Maintenance)」状態の場合は、まずそれを修復する必要があります。

3 ミラーの読み取りポリシーを「**first**」に設定します。

```
# metaparam -r first d40
# metastat d40
d40: Mirror
    Submirror 0: d41
      State: Okay
    Submirror 1: d42
      State: Okay
    Pass: 1
    Read option: first
    Write option: parallel (default)
    Size: 20484288 blocks (9.8 GB)
```

4 ファイルシステムのバックアップを実行します。

```
# ufsdump 0f /dev/backup /opt/test
```

5 ufsdump コマンドの実行後、ミラーの読み取りポリシーを「**roundrobin**」に設定します。

```
# metaparam -r roundrobin d40
# metastat d40
d40: Mirror
    Submirror 0: d41
```

```
  State: Okay
Submirror 1: d42
  State: Okay
Pass: 1
Read option: roundrobin
Write option: parallel (default)
Size: 20484288 blocks (9.8 GB)
```

システムの復元

システムを復元するとき、DVD または CD メディア上にある Solaris OS インストールイメージからブートすると便利な場合があります。たとえば、root のパスワードをリセットできるというのも、インストールイメージから起動するときの利点です。

Solaris ボリュームマネージャ構成を使用している場合は、実際のディスクではなく、Solaris ボリュームマネージャボリュームをマウントします。この手順は、ルート (/) ファイルシステムがミラー化されている場合には特に重要になります。Solaris ボリュームマネージャは Solaris OS の一部であるため、Solaris ボリュームマネージャボリュームをマウントすると、すべての変更がミラーの両側に反映されます。

Solaris OS DVD または CD-ROM のインストールイメージから Solaris ボリュームマネージャボリュームにアクセスできるようにするには、次の手順を使用します。

▼ Solaris ボリュームマネージャ構成を使用してシステムを復元するには

Solaris OS インストール DVD または CD メディアからシステムをブートします。この手順は、Solaris miniroot の root プロンプトから実行します。

1. **Solaris** ボリュームマネージャ構成が含まれる実際のディスクを読み取り専用としてマウントします。

   ```
   # mount -o ro /dev/dsk/c0t0d0s0 /a
   ```

2. md.conf ファイルを /kernel/drv ディレクトリにコピーします。

   ```
   # cp /a/kernel/drv/md.conf /kernel/drv/md.conf
   ```

3. ファイルシステムを **miniroot** からマウント解除します。

   ```
   # umount /a
   ```

4. この構成を読み込むように、**Solaris** ボリュームマネージャドライバを更新します。update_drv コマンドから出力される警告メッセージはすべて無視します。

   ```
   # update_drv -f md
   ```

5 システムボリュームを構成します。

```
# metainit -r
```

6 **Solaris** ボリュームマネージャ構成に **RAID-1** ボリュームがある場合は、これらのボリュームの再同期をとります。

```
# metasync mirror-name
```

7 mount コマンドを使用すれば、**Solaris** ボリュームマネージャボリュームにアクセスできます。

```
# mount /dev/md/dsk/volume-name /a
```

例 25–6 Solaris ボリュームマネージャ構成を使用してシステムを復元する

```
# mount -o ro /dev/dsk/c0t0d0s0 /a
# cp /a/kernel/drv/md.conf /kernel/drv/md.conf
# umount /a
# update_drv -f md
Cannot unload module: md
Will be unloaded upon reboot.
Forcing update of md.conf.
devfsadm: mkdir fialed for /dev 0xled: Read-only file system
devfsadm: inst_sync failed for /etc/path_to_inst.1359: Read-only file system
devfsadm: WARNING: failed to update /etc/path_to_inst
# metainit -r
# metasync d0
# mount /dev/md/dsk/d0 /a
```

◆ ◆ ◆ 付 録 A

Solaris ボリュームマネージャの重要なファイル

この付録では、Solaris ボリュームマネージャのファイルについて説明します。これらのファイルは、参照目的で使用してください。この付録の構成は、次のとおりです。

システムファイルと始動ファイル

この節では、Solaris ボリュームマネージャが正常に動作するのに必要なファイルについて説明します。いくつかの特別な構成変更を行う場合を除き、これらのファイルを使用したり、変更したりする必要はありません。

- `/etc/lvm/mddb.cf`

注意 – このファイルを編集しないでください。このファイルを変更すると、Solaris ボリュームマネージャ構成を破壊することがあります。

`/etc/lvm/mddb.cf` ファイルには、状態データベースの複製の格納場所が記録されています。状態データベースの複製の格納場所が変わると、Solaris ボリュームマネージャは、すべての状態データベースの格納場所が記録されている `mddb.cf` ファイルにエントリを作成します。詳細については、`mddb.cf(4)` のマニュアルページを参照してください。

- `/etc/lvm/md.cf`

  `/etc/lvm/md.cf` ファイルには、自動的に生成された、デフォルト (名前のない、またはローカルの) ディスクセットの構成情報が格納されています。Solaris ボリュームマネージャ構成が変更されると、Solaris ボリュームマネージャは `md.cf` ファイルを自動的に更新します (使用中のホットスペアの情報を除く)。詳細については、`md.cf(4)` のマニュアルページを参照してください。

注 – このファイルを編集しないでください。このファイルを変更すると、Solaris ボリュームマネージャ構成が壊れたり、回復できなくなったりすることがあります。

状態データベースに保持されている情報が失われた場合でも、その後でボリュームが変更されたり、作成されたりしていなければ、`md.cf` ファイルを使ってこの構成を復元できます。242 ページの「構成ファイルを使って Solaris ボリュームマネージャを初期化するには」を参照してください。

- `/kernel/drv/md.conf`

  `md.conf` 構成ファイルは、Solaris ボリュームマネージャによって起動時に読み取られます。`md.conf` ファイルには、状態データベースの複製の構成情報が格納されています。Solaris 10 から、`nmd` パラメータと `md_nsets` パラメータは手動で編集できなくなりました。Solaris ボリュームマネージャが拡張され、必要に応じてボリュームが動的に構成されるようになっています。

手動で設定するファイル

`md.tab` ファイルの概要

`/etc/lvm/md.tab` ファイルには、Solaris ボリュームマネージャの構成情報が格納されています。この情報を使えば、Solaris ボリュームマネージャ構成を再構築することができます。Solaris ボリュームマネージャは、このファイルをコマンド行ユーティリティー `metainit`、`metadb`、`metahs` への入力として使用することによって構成を再構築できます。このファイルには、通常、ボリュームやディスクセット、ホットスペア集合のエントリが含まれています。このファイルを作成する方法については (`metastat -p > /etc/lvm/md.tab` コマンドを使用)、241 ページの「構成ファイルを作成するには」を参照してください。

注 – `/etc/lvm/md.tab` ファイルの構成情報と、実際に使用中のボリュームやホットスペア、状態データベースの複製が異なる場合があります。このファイルは、意図する構成を手動で保存するためのものです。Solaris ボリュームマネージャ構成を変更したあと、このファイルを作成し直し、バックアップコピーを保管しておいてください。

このファイルを作成または更新したら、`metainit` や `metahs`、`metadb` コマンドを使って、このファイルに指定したボリュームや、ホットスペア集合、状態データベースの複製をアクティブにすることができます。

`/etc/lvm/md.tab` ファイルの各行には、1 つの ボリュームの完全な構成エンティティーの情報を、`metainit`、`metadb`、`metahs` コマンドの構文に基づいて指定します。

注 – metainit -an コマンドを使って、md.tab 内にあるすべてのボリュームの初期化をシミュレートすると、md.tab で定義されている別のボリュームに依存するボリュームについてのエラーメッセージが表示されることがあります。こうしたエラーメッセージが表示されるのは、metainit -an コマンドの実行時に作成されたはずのボリュームの状態を Solaris ボリュームマネージャが維持していないからです。構成が存在している場合は、既存押構成に基づいて各行が評価されます。したがって、metainit -an コマンドが失敗するように見える場合でも、-n オプションを指定しなければ成功する可能性があります。

その上で metainit コマンドに -a オプションを指定すれば、/etc/lvm/md.tab ファイルに指定されているすべてのボリュームをアクティブにできます。あるいは、このファイルの特定のエントリに対応するボリュームを指定すれば、そのボリュームをアクティブにできます。

注 – Solaris ボリュームマネージャが /etc/lvm/md.tab ファイルに構成情報を書き込んだり、格納したりすることはありません。Solaris ボリュームマネージャコンポーネントを再作成するためには、ユーザーがこのファイルを手動で編集し、metainit 、metahs、または metadb コマンドを実行する必要があります。

詳細は、md.tab(4) のマニュアルページを参照してください。

◆ ◆ ◆ 付 録 B

Solaris ボリュームマネージャのコマンド行リファレンス

この付録では、Solaris ボリュームマネージャのコマンド行インタフェースで使用できるコマンドの要約を示します。

コマンド行リファレンス

次の表に、Solaris ボリュームマネージャの管理に使用するコマンドを示します。詳細は、各コマンドのマニュアルページを参照してください。

表 B–1 Solaris ボリュームマネージャのコマンド

Solaris ボリュームマネージャのコマンド	説明	マニュアルページ
`growfs`	UFS ファイルシステムを安全に拡張します。	`growfs(1M)`
`metaclear`	アクティブなボリュームやホットスペア集合を削除します。	`metaclear(1M)`
`metadb`	状態データベースの複製を作成および削除します。	`metadb(1M)`
`metadetach`	RAID-0 または RAID-1 (ミラー) ボリュームからボリュームを切り離します。またはトランザクションボリュームからログデバイスを切り離します。 注 – トランザクションボリュームはサポートされなくなりました。	`metadetach(1M)`
`metadevadm`	デバイス ID 構成をチェックします。	`metadevadm(1M)`
`metahs`	ホットスペアとホットスペア集合を管理します。	`metahs(1M)`

表 B–1 Solaris ボリュームマネージャのコマンド　(続き)

Solaris ボリュームマネージャのコマンド	説明	マニュアルページ
metaimport	ディスクセットにデバイス ID サポートが設定されている既存の Solaris ボリュームマネージャ構成に、ディスクセット (複製されたディスクセットを含む) をインポートします。	metaimport(1M)
metainit	ボリュームを作成します。	metainit(1M)
metaoffline	サブミラーをオフラインにします。	metaoffline(1M)
metaonline	サブミラーをオンラインにします。	metaonline(1M)
metaparam	ボリュームパラメータを変更します。	metaparam(1M)
metarecover	ソフトパーティションの構成情報を復元します。	metarecover(1M)
metarename	ボリューム名を変更または交換します。	metarename(1M)
metareplace	サブミラーおよび RAID-5 ボリュームのコンポーネントを置き換えます。	metareplace(1M)
metaroot	ルート (/) ファイルシステムをミラー化するためのシステムファイルを用意します。	metaroot(1M)
metaset	ディスクセットを管理します。	metaset(1M)
metastat	ボリュームまたはホットスペア集合の状態を表示します。	metastat(1M)
metasync	リブート時にボリューム間の再同期をとります。	metasync(1M)
metattach	RAID-0 または RAID-1 ボリュームにコンポーネントを接続します。	metattach(1M)

◆ ◆ ◆ 付 録 C

Solaris ボリュームマネージャの CIM/WBEM API

Solaris ボリュームマネージャの管理

Solaris ボリュームマネージャの CIM/WBEM アプリケーションプログラミングインタフェース (API) は、標準に準拠したオープンなプログラムインタフェースを提供し、Solaris ボリュームマネージャの管理や構成を可能にします。この API は、Distributed Management Task Force (DMTF) の Common Information Model (CIM) に基づいています。DMTF については、`http://www.dmtf.org` をご覧ください。

CIM は、「スキーマ」と呼ばれるデータモデルを定義します。このスキーマには、次のものが規定されています。

- Solaris ボリュームマネージャデバイスの属性と Solaris ボリュームマネージャデバイスに対する操作
- 各種 Solaris ボリュームマネージャデバイス間の関係
- Solaris ボリュームマネージャデバイスと、オペレーティングシステムの機能 (ファイルシステムなど) との関係

このモデルは、Solaris Web Based Enterprise Management (WBEM) SDK を通じて使用されます。WBEM SDK は、CIM で規定されているシステム管理機能へのアクセスを可能にする Java™ テクノロジに基づく API セットです。

CIM/WBEM SDK の詳細については、『Solaris WBEM 開発ガイド』を参照してください。

索引

O

P

R

S

V

あ

い

え

か

そ

た

て

と

な

に

め

ら

れ

ろ