Sun N1 Grid Engine 6.1 管理ガイド

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

Part No: 820–2163–10
2007 年 5 月

原典: Sun N1 Grid Engine 6.1 Administration Guide

Part No: 820-0698-10

070524@17466

目次

図目次

表目次

例目次

はじめに

『Sun N1 Grid Engine 6.1 管理ガイド』は、N1™ Grid Engine 6.1 ソフトウェアが稼働する、ネットワークコンピュータホストのシステム設定と管理の方法に関する基本的な情報について説明します。

対象読者

このガイドの内容説明と解説は、経験豊富なシステム管理者を対象としています。

内容の紹介

『Sun N1 Grid Engine 6.1 管理ガイド』は、次の章で構成されています。

- 第 1 章では、ホストとクラスタの一般的な内容説明を行い、それらの構成方法を詳細に解説します。
- 第 2 章では、さまざまなカテゴリのコンテナとして機能する、キューについて説明します。この章では、クラスタキューとキューインスタンスの構成方法を完全に解説します。
- 第 3 章では、Grid Engine システムがコンプレックスを使用して、(ユーザーがジョブに関して要求可能なリソースを含む) すべての関連情報を定義する方法について説明します。管理者は、環境の要件に適合するよう、コンプレックスリソース属性を構成します。この章では、リソース属性の構成方法を詳細に解説します。
- 第 4 章 では、Grid Engine システムのさまざまな種類のユーザーに関する内容説明を行います。この章では、ユーザーのアクセス権とプロジェクトのアクセス権を設定および管理する方法を解説します。
- 第 5 章では、使用可能なユーザーポリシーの種類の詳細な内容説明を行います。この章では、これらのポリシーをコンピューティング環境に適合させる方法を解説します。また、第 5 章では、スケジューラを構成および変更する方法についても説明します。
- 第 6 章では、N1 Grid Engine ソフトウェアのリソース割り当て機能を使用して、ユーザー、プロジェクト、ホスト、クラスタキュー、または並列環境別にリソースを制限する方法を説明します。

- 第 7 章では、Grid Engine システムがどのように並列環境と適合するかについて説明し、並列環境の構成方法を詳細に解説します。またこの章ではチェックポイント設定環境の設定方法と使用方法についても説明します。
- 第 8 章では、レポートおよびアカウンティングの統計の収集方法、Grid Engine システム構成ファイルの自動バックアップ方法、ファイルやスクリプトを使用したキュー、ホスト、環境などのオブジェクトの追加または変更の方法について説明します。
- 第 9 章では、Grid Engine システムを チューニングするいくつかの方法を説明します。また、Grid Engine システムによりエラーメッセージがどのように取得されるか、およびデバッグモードでソフトウェアを実行する方法についても説明します。
- 第 10 章では、ARCo 機能の一部である DBWriter の変更方法について説明します。

注 – このガイドの内容の一部は、もともとは Sun Grid Engine プロジェクトの Web サイトの「How-To」セクションに掲載されていたものです。頻繁に更新されるため、この Web サイトは Grid Engine システムの管理者にとって特別な価値があるため、アクセスする価値があります。

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。

表 P–1 表記上の規則 (続き)

字体または記号	意味	例
「」	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'`

コマンド例のシェルプロンプト

次の表に、C シェル、Bourne シェル、および Korn シェルのデフォルトの UNIX® システムプロンプト、およびスーパーユーザーのプロンプトを紹介します。

表 P–2 シェルプロンプト

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

◆ ◆ ◆ 第 1 章

1 ホストとクラスタの構成

この章では、Grid Engine システムのさまざまな側面の構成に関する内容説明を行います。この章では、次の作業について説明します。

ホストとデーモンについて

システム上でどのデーモンが実行中であるか、およびホストが sge_qmaster でどのように登録されているかに応じて、Grid Engine システムホストは 4 つのグループに分類されます。

- マスターホスト。マスターホストは、クラスタの活動全般の中心です。マスターホストは、マスターデーモン sge_qmaster を実行します。sge_qmaster は、キューやジョブなどのすべての Grid Engine システムコンポーネントを制御します。また、ユーザーのアクセス権などに関するコンポーネントの状態に関するテーブルも維持します。マスターホストは、通常、スケジューラ sge_schedd を実行します。マスターホストは、インストール手順により実行される構成以外の構成を必要としません。

  マスターホストの最初の設定方法については、『Sun N1 Grid Engine 6.1 インストールガイド』の「マスターホストをインストールする方法」を参照してください。マスターホストに対する動的な変更の構成方法については、23 ページの「シャドウマスターホストの構成」を参照してください。
- 実行ホスト。実行ホストは、ジョブを実行するアクセス権を持つノードです。そのため、実行ホストはキューインスタンスを提供し、実行デーモン sge_execd を実行します。実行ホストは、『Sun N1 Grid Engine 6.1 インストールガイド』の「実行ホストをインストールする方法」で説明されているインストール手順によって最初に設定されます。
- 管理ホスト。任意の種類の管理作業を行うために、マスターホスト以外のホストにもアクセス権を付与できます。管理ホストを設定するには、次のコマンドを使用します。

  ```
  qconf -ah ホスト名
  ```

  詳細については、qconf(1) のマニュアルページを参照してください。
- 発行ホスト。発行ホストは、バッチジョブの発行と制御のみが可能です。特に、発行ホストにログインしているユーザーは、qsub を使用してジョブを発行したり、qstat を使用してジョブの状態を制御したり、グラフィカルユーザーインタフェース QMON を実行できます。発行ホストを設定するには、次のコマンドを使用します。

  ```
  qconf -as ホスト名
  ```

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

注 – ホストは、複数のクラスに属することが可能です。マスターホストは、デフォルトでは管理ホストと発行ホストになっています。

別のホストへの qmaster の移行

スプーリングデータベースは NFS でマウントされたファイルシステムに配置できないため、次の手続きでは、Berkeley DB RPC サーバーをスプーリングに使用する必要があります。

スプーリングをローカルファイルシステムに対して構成する場合は、スプーリングデータベースを、新しい sge_qmaster ホスト上のローカルファイルシステムに転送する必要があります。

▼ スクリプトを使用して別のホストに qmaster を移行する

1 新しいマスターホストに読み取り/書き込みアクセス権があることを確認します。

新しいマスターホストには、現在のマスター同様、qmaster スプールディレクトリと common ディレクトリに対する読み取り/書き込みアクセス権が必要です。管理ユーザーがユーザー root (グローバルクラスタ構成で admin_user の設定を確認) の場合は、ユーザー root がそのユーザー名でこれらのディレクトリにファイルを作成できることを確認してください。

2 新しいマスターホストで移行スクリプトを実行します。

新しいマスターホストで、ユーザー root として次のスクリプトを実行してください。

```
# /etc/init.d/sgemaster -migrate
```

このコマンドは、元のマスターホスト上で sge_qmaster および sge_schedd を停止し、新しいマスターホスト上でこれらのコマンドを起動します。
$SGE_ROOT/$SGE_CELL/common/act_qmaster ファイルにあるマスターホスト名は、自動的に新しいマスターホストの名前に変更されます。qmaster が動作していない場合は、警告メッセージが表示され、1 分ほどして、新しいホスト上で qmaster が起動されます。

3 必要に応じて shadow_masters ファイルを変更します。

$SGE_ROOT/$CELL/common/shadow_masters ファイルが存在するかどうかを確認します。ファイルが存在する場合は、必要に応じて新しい qmaster ホストをこのファイルに追加し、元のマスターホストを削除できます。それぞれのマシンで次のコマンドを発行して、sge_shadowd デーモンの実行を停止、再開します。

```
/etc/init.d/sgemaster -shadowd stop
/etc/init.d/sgemaster -shadowd start
```

注 – システム全体の sgemaster 起動スクリプトがある場所は、使用されているオペレーティングシステムによって異なります。 $SGE_ROOT/default/common/sgemaster は常に使用できます。

参考 **移行に関する注意**

移行手続きは、sgemaster -migrate コマンドが発行されたホストに移行します。primary_qmaster ファイルが存在する場合、そのマシンにある sgemaster への呼び出しがそのファイルに含まれていると、移行せずにそのマシンに戻ります。こうした状況を回避するには、$SGE_ROOT/$SGE_CELL/common/primary_qmaster ファイルを変更または削除してください。

注 – primary_qmaster ファイルの存在は、qmaster が実際に動作していることを意味しません。

移行手続き中、ジョブは実行を継続するかもしれませんが、グリッドはアクティブでないはずです。移行時に、qsub または qstat などの実行中の SGE コマンドはエラーを返します。

現在の qmaster が停止すると、qmaster との交信待ちでタイムアウトするまで、スケジューラの停止に遅れが生じます。

shadow_masters ファイルによって、移行手続きが直接影響を受けることはありません。このファイルは、シャドウマスターが構成されていた場合にのみ存在します。シャドウマスターの構成方法の詳細は、23 ページの「シャドウマスターホストの構成」を参照してください。

▼ 別のホストに手動で qmaster を移行する

1 現在のマスターホストで、次のコマンドを入力して、マスターデーモンとスケジューラデーモンを停止します。

```
qconf -ks -km
```

2 次の手順に従い *sge-root*/*cell*/common/act_qmaster ファイルを編集します。

a. 新しいマスターホストの名前を確認します。

新しいマスターホストの名前を取得するには、そのマスターホストで次のコマンドを入力します。

sge-root/utilbin/$ARCH/gethostname

b. act_qmaster ファイルにある現在のホスト名を、gethostname ユーティリティーから返された新しいマスターホストの名前に置き換えます。

3 新しいマスターホストで、sge_qmaster と sge_schedd を起動します。

sge-root/*cell*/common/sge5

シャドウマスターホストの構成

シャドウマスターホストはクラスタ内のマシンで、マスターデーモンの障害を検出して、マスターホストとしてその役割を引き継ぐことができます。シャドウマスターデーモンが、マスターデーモン sge_qmaster に異常な障害が発生したことを検出すると、シャドウマスターデーモンが動作しているホスト上で新しい sge_qmaster を起動します。

注 – マスターデーモンが正常に停止した場合は、シャドウマスターデーモンは起動しません。マスターデーモンを正常に停止したあとにシャドウマスターデーモンに引き継ぎを任せる場合は、sge_qmaster スプールディレクトリにあるロックファイルを削除します。このスプールディレクトリのデフォルトの位置は、*sge-root*/*cell*/spool/qmaster です。

シャドウマスターホストでの sge_qmaster の自動フェイルオーバーの起動には、約 1 分かかります。その間、Grid Engine システムコマンドが実行されるたびに、エラーメッセージが表示されます。

注 – ファイル *sge-root*/ *cell*/common/act_qmaster には、実際に sge_qmaster デーモンを実行しているホストの名前が含まれています。

シャドウマスターホストの要件

ホストをシャドウマスターとして準備するには、次の要件が満たされている必要があります。

- シャドウマスターホストは sge_shadowd を実行する必要があります。
- シャドウマスターホストは、ディスクに記録されている sge_qmaster の状態情報、ジョブ構成、およびキュー構成を共有する必要があります。特に、シャドウマスターホストは、マスターホストのスプールディレクトリと、ディレクトリ *sge-root*/*cell*/common への読み取り/書き込みルートアクセス権が必要です。
- sge_qmaster のスプーリングには、Berkeley DB RPC サーバーまたは典型的な Grid Engine システムスプーリングを使用する必要があります。 詳細は、『Sun N1 Grid Engine 6.1 インストールガイド』の「データベースサーバーとスプールホスト」を参照してください。

- shadow-master-hostname ファイルは、そのホストをシャドウマスターホストとして定義する行を含む必要があります。

以上の要件が満たされるとすぐに、シャドウマスターホストの機能はこのホストに対して起動されます。機能の起動には、Grid Engine システム デーモンの再起動は必要ありません。

シャドウマスターホストファイル

シャドウマスターホストファイル *sge-root*/*cell*/common/shadow_masters には、次の要素が含まれています。

- マスターデーモン sge_qmaster が最初に実行されるマシンである、プライマリマスターホストの名前
- シャドウマスターホストの名前

シャドウマスターホスト名ファイルの書式は次のようになります。

- ファイルの最初の行がプライマリマスターホストを定義する
- それ以降の行が、1 行につき 1 ホスト、シャドウマスターホストを定義する

シャドウマスターホストの順序は重要です。プライマリマスターホストがファイルの最初の行になります。プライマリマスターホストで動作の続行に障害が生じた場合、2 行目で定義されているシャドウマスターが引き継ぎます。このシャドウマスターにも障害が発生した場合は、3 行目で定義されているシャドウマスターが引き継ぐ、となります。

シャドウマスターホストの起動

シャドウ sge_qmaster を起動するには、システムは、古い sge_qmaster が停止しているか、新しく起動されるシャドウ sge_qmaster に干渉するアクションを実行することなく停止することを確認する必要があります。

非常にまれな状況では、古い sge_qmaster がすでに停止しているか、これから停止するかを判別することが不可能な場合があります。このような場合は、シャドウマスターホストの sge_shadowd のメッセージログファイルにエラーメッセージが記録されます。第 9 章を参照してください。また、sge_qmaster デーモンへの tcp 接続を開く試みは、永続的に失敗します。このような事態になった場合は、実行中のマスターデーモンが存在しないことを確認してから、すべてのシャドウマスターマシン上で sge_qmaster を手動で再起動します。42 ページの「コマンド行からのデーモンの再起動」を参照してください。

シャドウマスターホスト環境変数の構成

次の 3 つの環境変数は、シャドウマスターのテイクオーバー時間に影響します。

- SGE_DELAY_TIME - この変数は、テイクオーバーの実行が失敗した場合に sge_shadowd が一時停止する間隔を制御します。この値は、複数の sge_shadowd インスタンスが存在し、それらがマスターになろうと競合している場合にのみ使用されます。デフォルトは 600 秒です。
- SGE_CHECK_INTERVAL - この変数は、sge_shadowd がハートビートファイルを確認する間隔を制御します。デフォルトでは 60 秒です。
- SGE_GET_ACTIVE_INTERVAL - この変数は、ハートビートファイルが変更されていない場合に sge_shadowd インスタンスがテイクオーバーを試みる間隔を制御します。

これらの変数は、次のように相互に作用します。

1. マスターホストはハートビートファイルを 30 秒ごとに更新します。
2. sge_shadowd デーモンは、 SGE_CHECK_INTERVAL 変数で定義されている秒数ごとにハートビートファイルの変更を確認します。このため、この値は 30 秒より大きくします。
3. sge_shadowd デーモンは、ハートビートファイルが更新されていることを検出すると、そのデーモンは再び待ち状態になり、もう一度ハートビートファイルを確認するタイミングになるまで待ちます。
4. sge_shadowd デーモンは、ハートビートファイルが更新されていないことを検出すると、SGE_CHECK_INTERVAL 変数で定義されている秒数が経過するまで待機します。この手順によって、sge_shadowd デーモンのテイクオーバーが頻繁に行われすぎないようにし、マスターホストが余裕を持ってハートビートファイルを更新できるようにします。
5. SGE_GET_ACTIVE_INTERVAL の期限が切れると、ハートビートファイルがまだ更新されていない場合は、sge_shadowd デーモンのテイクオーバーが行われます。

推奨する設定は、SGE_CHECK_INTERVAL が 45 秒、SGE_GET_ACTIVE_INTERVAL が 90 秒です。この設定にすると、約 2 分後にテイクオーバーが発生します。これらの環境変数の設定後にシャドウホストの動作を確認する場合は、マスターホストのネットワークケーブルを引き抜いて、障害のシミュレーションを行う必要があります。

ホストの構成

N1 Grid Engine 6.1 ソフトウェアは、マスターホスト用を除き、すべてのタイプのホストのオブジェクトリストを維持しています。管理ホストオブジェクトと発行ホストオブジェクトのリストは、あるホストが管理アクセス権と発行アクセス権のどちらを持っているかを示しています。実行ホストオブジェクトにはそのほかのパラメータも含まれています。これらのパラメータの中には、ホスト上で実行中の sge_execd により報告される負荷情報や、管理者により定義された負荷パラメータのスケーリング係数があります。

ホストオブジェクトの構成は、QMON を使用するか、コマンド行から行うことができます。

「QMON Host Configuration」ダイアログボックスには次の 4 つのタブがあります。

- 「Administration Host」タブ。図 1–3 を参照してください。
- 「Submit Host」タブ。図 1–4 を参照してください。
- 「Host Groups」タブ。図 1–5 を参照してください。
- 「Execution Host」タブ。図 1–1 を参照してください。

qconf コマンドは、ホストオブジェクトを管理するためのコマンド行インタフェースを提供します。

QMON を使用した実行ホストの構成

実行ホストを構成する前に、『Sun N1 Grid Engine 6.1 インストールガイド』の「実行ホストをインストールする方法」の説明に従って、実行ホストにソフトウェアをインストールしてください。

実行ホストを構成するには、「QMON Main Control」ウィンドウで「Host Configuration」ボタンをクリックしてから、「Execution Host」タブをクリックします。「Execution Host」タブは次の図のようになっています。

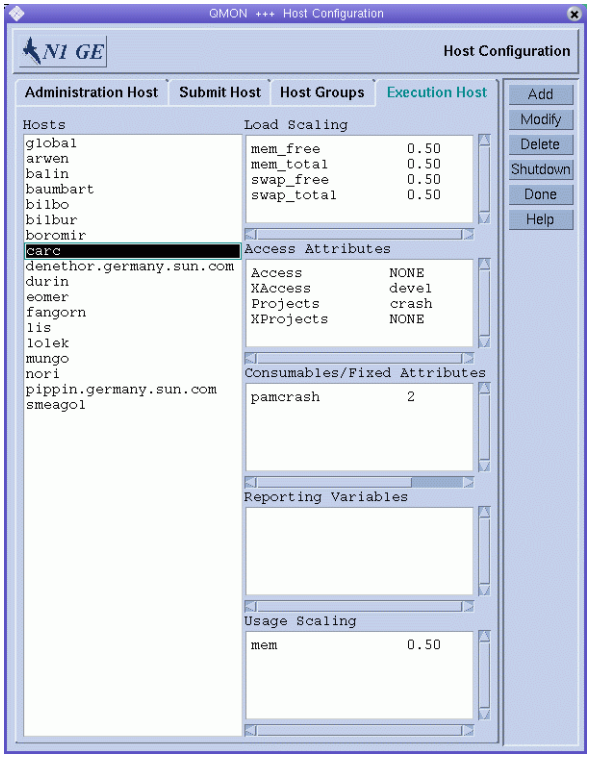


図 1–1 「Execution Host」タブ

注 – 実行ホストから管理または発行コマンドが使用できるのは、実行ホストも管理または発行ホストとして宣言されている場合のみです。33 ページの「QMON を使用した管理ホストの構成」および 34 ページの「QMON を使用した発行ホストの構成」を参照してください。

「Hosts」リストには、すでに定義されている実行ホストが表示されます。

「Load Scaling」リストには、選択した実行ホストに対して現在構成されている負荷スケーリング係数が表示されます。負荷パラメータの詳細については、90 ページの「負荷パラメータ」を参照してください。

「Access Attributes」リストにはアクセス権が表示されます。アクセス権の詳細については、第 4 章を参照してください。

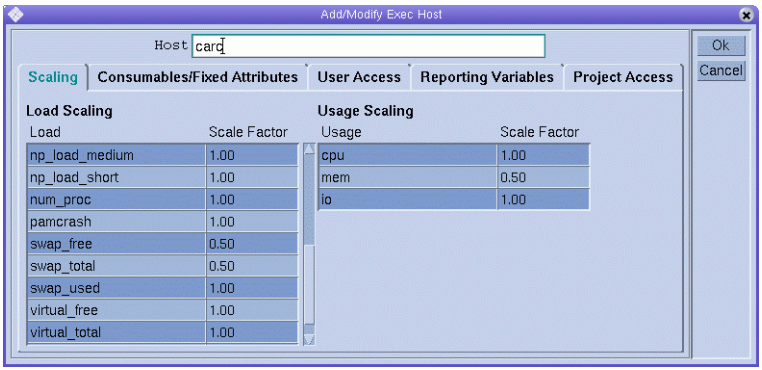
「Consumables/Fixed Attributes」リストには、ホストと関連付けられている消費可能リソース属性および固定リソース属性の、リソースの可用性が表示されます。リソース属性の詳細については、71 ページの「コンプレックスリソース属性」を参照してください。

「Reporting Variables」リストには、実行ホストから負荷レポートが受信された場合に、レポートファイルに書き込まれる変数が表示されます。レポート変数の詳細については、31 ページの「レポート変数の定義」を参照してください。

「Usage Scaling」リストには、さまざまなマシンの個別の使用率メトリック CPU、メモリー、および I/O の現在のスケーリング係数が表示されます。リソースの使用率は、現在実行中の各ジョブに関して、sge_execd によって定期的に報告されます。スケーリング係数は、ジョブを実行中のユーザーまたはプロジェクトに関して、特定のマシンでのリソース使用率の相対的なコストを示しています。たとえばこれらの係数は、400 MHz プロセッサでの 1 秒の CPU 時間のコストを、600 MHz CPU と比較することに使用できます。「Usage Scaling」ウィンドウに表示されないメトリックのスケーリング係数は 1 です。

実行ホストの追加または変更

実行ホストを追加または変更するには、「Add」または「Modify」をクリックします。「Add/Modify Exec Host」ダイアログボックスが表示されます。



「Add/Modify Exec Host」ダイアログボックスでは、実行ホストに関連付けられているすべての属性を変更できます。「Host」フィールドには、既存の実行ホストの名前が表示されます。

新しい実行ホストを追加する場合は、「Host」フィールドに名前を入力します。

スケーリング係数の定義

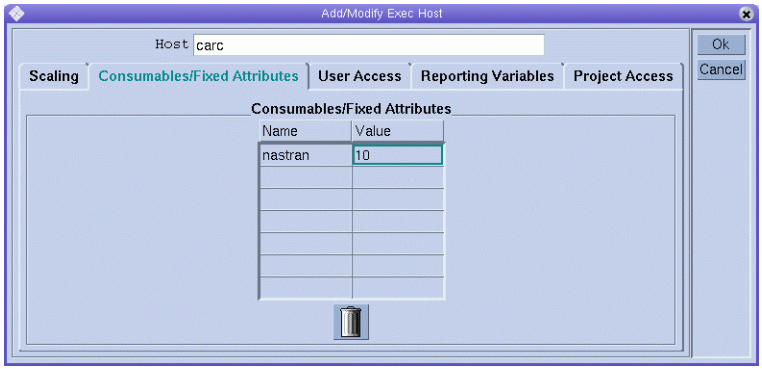
スケーリング係数を定義するには、「Scaling」タブをクリックします。

「Load Scaling」テーブルの「Load」カラムには使用可能なすべての負荷パラメータが表示され、「Scale Factor」カラムには、対応するスケーリングの定義が表示されます。「Scale Factor」カラムは編集できます。有効なスケーリング係数は、固定小数点または科学的記数法形式の正の浮動小数点数です。

「Usage Scaling」テーブルの「Usage」カラムには、使用率メトリック CPU、メモリー、および I/O の現在のスケーリング係数が表示されます。「Scale Factor」カラムには、対応するスケーリングの定義が表示されます。「Scale Factor」カラムは編集できます。有効なスケーリング係数は、固定小数点または科学的記数法形式の正の浮動小数点数です。

リソース属性の定義

リソース属性を定義してホストと関連付けるには、「Consumables/Fixed Attributes」タブをクリックします。



ホストと関連付けられているリソース属性は、「Consumables/Fixed Attributes」テーブルに表示されます。

現在のコンプレックス構成に関する詳細な情報が必要な場合、またはその情報を変更したい場合は、「Complex Configuration」ダイアログボックスを使用します。コンプレックスリソース属性の詳細については、71 ページの「コンプレックスリソース属性」を参照してください。

「Consumables/Fixed Attributes」テーブルには、値が現在定義されている対象であるすべてのリソース属性が表示されます。「Name」または「Value」カラム名のいずれかをクリックすることで、リストを拡張できます。「Attribute Selection」ダイアログボックスが表示されます。このダイアログボックスには、コンプレックスで定義されているすべてのリソース属性が含まれています。

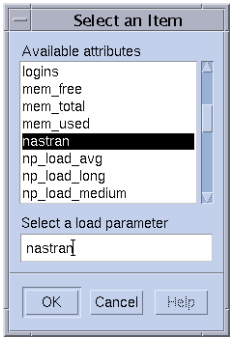


図 1–2 「Attribute Selection」ダイアログボックス

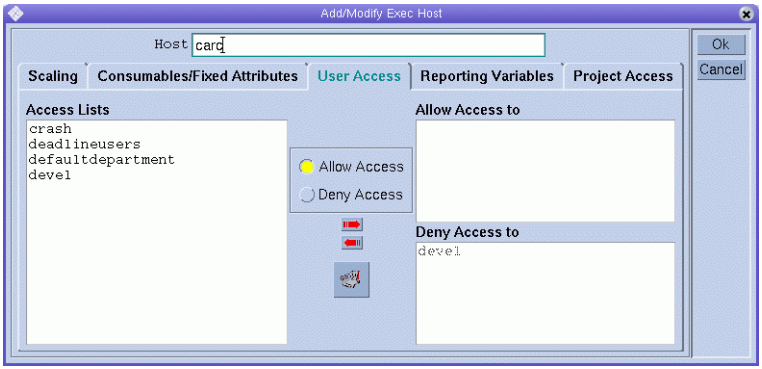
「Consumables/Fixed Attributes」テーブルに属性を追加するには、属性を選択してから「OK」をクリックします。

属性値を変更するには、「Value」フィールドをダブルクリックしてから値を入力します。

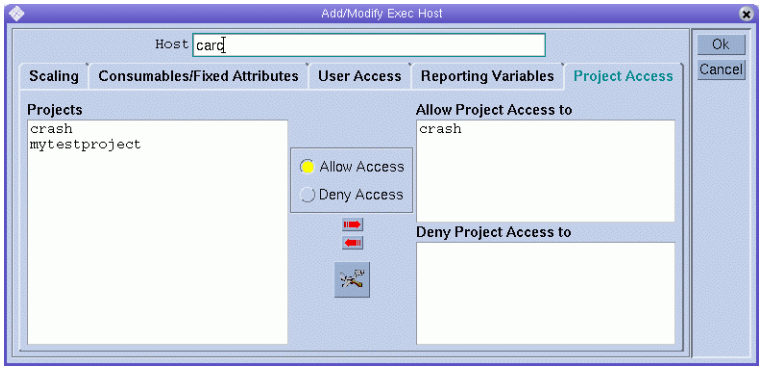
属性を削除するには、属性を選択してから Control + D キーを押すか、マウスボタン 3 をクリックします。「OK」をクリックして、その属性を削除することを確認します。

アクセス権の定義

以前に構成したユーザーアクセスリストに基づいて実行ホストに対するユーザーのアクセス権を定義するには、「User Access」タブをクリックします。

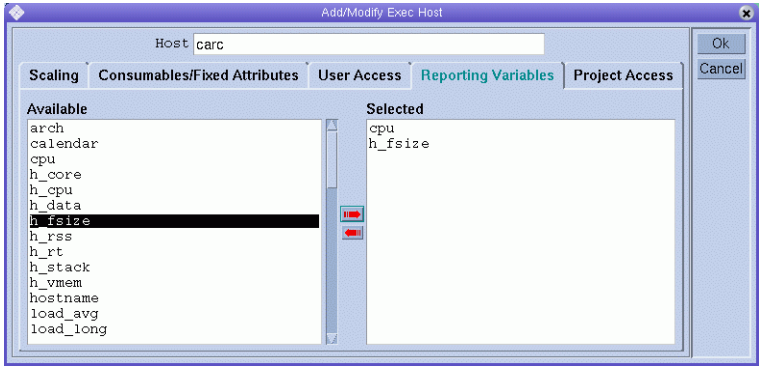


以前に構成したプロジェクトに基づいて実行ホストに対するプロジェクトのアクセス権を定義するには、「Project Access」タブをクリックします。



レポート変数の定義

レポート変数を定義するには、「Reporting Variables」タブをクリックします。



「Available」リストには、実行ホストから負荷レポートが受信された場合にレポートファイルに書き込み可能なすべての変数が表示されます。

「Available」リストからレポート変数を選択してから赤い右矢印をクリックし、選択した変数を「Selected」リストに追加します。

「Selected」リストからレポート変数を削除するには、変数を選択してから赤い左矢印をクリックします。

実行ホストの削除

実行ホストを削除するには、「QMON Main Control」ウィンドウで「Host Configuration」ボタンをクリックしてから、「Execution Host」タブをクリックします。

「Execution Host」ダイアログボックスで、削除するホストを選択してから「Delete」をクリックします。

実行ホストデーモンの停止

実行ホストデーモンを停止するには、「QMON Main Control」ウィンドウで「Host Configuration」ボタンをクリックしてから、「Execution Host」タブをクリックします。

「Execution Host」ダイアログボックスで、ホストを選択してから「Shutdown」をクリックします。

コマンド行からの実行ホストの構成

コマンド行から実行ホストを構成するには、qconf コマンドで次の引数を使用します。

- `qconf -ae [`*exec-host*`]`

  -ae オプション (実行ホストの追加) を使用すると、実行ホストの構成テンプレートが含まれるエディタが表示されます。このエディタは、デフォルトの vi エディタか、EDITOR 環境変数に対応するエディタのいずれかです。すでに構成されている実行ホストの名前である *exec-host* を指定すると、この実行ホストの構成がテンプレートとして使用されます。実行ホストの構成は、テンプレートを変更してディスクに保存することで行います。変更すべきテンプレートエントリの詳細については、host_conf(5) のマニュアルページを参照してください。

- `qconf -de` *hostname*

  -de オプション (実行ホストの削除) を使用すると、実行ホストのリストから、指定したホストが削除されます。実行ホスト構成のすべてのエントリは失われます。

- `qconf -me` *hostname*

  -me オプション (実行ホストの変更) を使用すると、指定した実行ホストの構成をテンプレートとして含むエディタが表示されます。このエディタは、デフォルトの vi エディタか、EDITOR 環境変数に対応するエディタのいずれかです。実行ホスト構成の変更は、テンプレートを変更してディスクに保存することで行います。変更すべきテンプレートエントリの詳細については、host_conf(5) のマニュアルページを参照してください。

- `qconf -Me` *filename*

  -Me オプション (実行ホストの変更) を使用すると、実行ホストの構成テンプレートとして *filename* の内容が使用されます。指定したファイルの構成では、既存の実行ホストを参照する必要があります。この実行ホストの構成は、ファイルの内容により置き換えられます。-Me オプションは手動による対話が不要であるため、qconf オプションは cron ジョブなどでのオフライン実行ホストの構成の変更に便利です。

- `qconf -se` *hostname*

`-se` オプション (実行ホストを表示) を使用すると、`host_conf` で定義されている指定の実行ホストの構成が表示されます。

- `qconf -sel`

  `-sel` オプション (実行ホストのリストの表示) を使用すると、実行ホストとして構成されているホストのリストが表示されます。

QMON を使用した管理ホストの構成

「QMON Main Control」ウィンドウで「Host Configuration」ボタンをクリックします。「Administration Host」タブとともに「Host Configuration」ダイアログボックスが表示されます。「Administration Host」タブは次の図のようになっています。

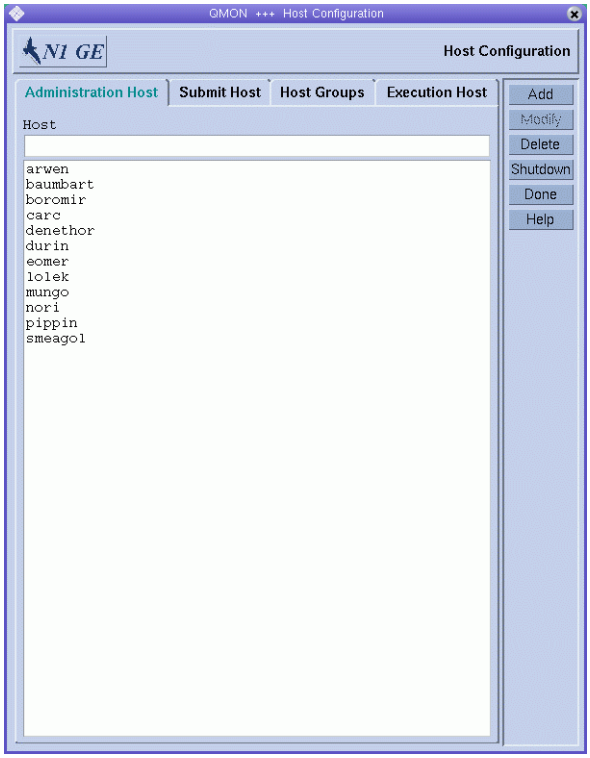


図 1–3 「Administration Host」タブ

注 – 「Host Configuration」ボタンをはじめてクリックすると、デフォルトでは「Administration Host」タブが表示されます。

「Administration Host」タブを使用して、管理コマンドが使用可能なホストを構成します。「Host」リストには、すでに管理アクセス権が設定されているホストが表示されます。

管理ホストの追加

新しい管理ホストを追加するには、「Host」フィールドに名前を入力してから、「Add」をクリックするか、Return キーを押します。

管理ホストの削除

リストから管理ホストを削除するには、ホストを選択してから「Delete」をクリックします。

コマンド行からの管理ホストの構成

コマンド行から管理ホストを構成するには、`qconf` コマンドで次の引数を使用します。

- `qconf -ah` *hostname*

  `-ah` オプション (管理ホストの追加) を使用すると、管理ホストのリストに、指定したホストが追加されます。

- `qconf -dh` *hostname*

  The `-dh` オプション (管理ホストの削除) を使用すると、管理ホストのリストから、指定したホストが削除されます。

- `qconf -sh`

  `-sh` オプション (管理ホストの表示) を使用すると、現在構成されているすべての管理ホストのリストが表示されます。

QMON を使用した発行ホストの構成

ホストが管理ホストとしても宣言されている場合を除き、発行ホストから管理コマンドを実行することはできません。詳細については、33 ページの「QMON を使用した管理ホストの構成」を参照してください。

発行ホストを構成するには、「QMON Main Control」ウィンドウで「Host Configuration」ボタンをクリックしてから、「Submit Host」タブをクリックします。次の図に「Submit Host」 タブを示します。

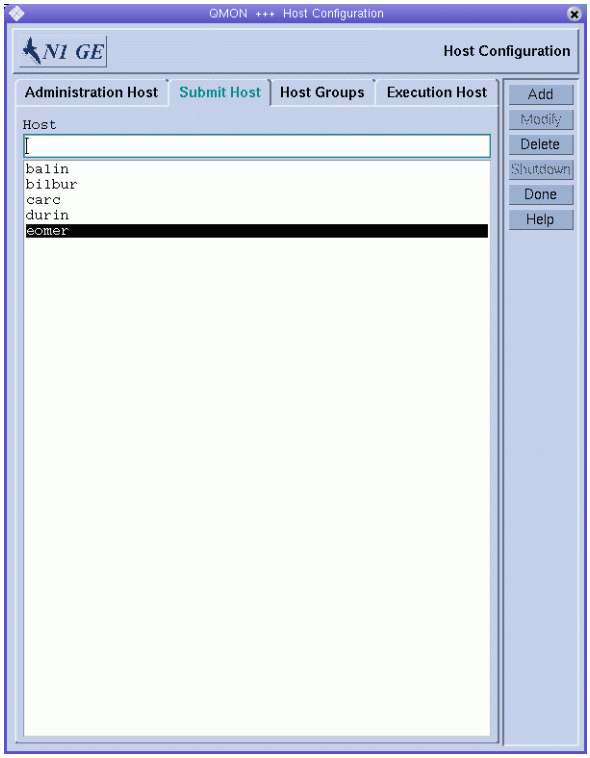


図 1–4 「Submit Host」タブ

「Submit Host」タブを使用して、ジョブの発行、監視、および制御を行うことができるホストを宣言します。「Host」リストには、すでに発行アクセス権が設定されているホストが表示されます。

発行ホストの追加

発行ホストを追加するには、「Host」フィールドに名前を入力してから、「Add」をクリックするか、Return キーを押します。

発行ホストの削除

発行ホストを削除するには、発行ホストを選択してから「Delete」をクリックします。

コマンド行からの発行ホストの構成

コマンド行から発行ホストを構成するには、qconf コマンドで次の引数を使用します。

- qconf -as *hostname*

  -as オプション (発行ホストの追加) を使用すると、発行ホストのリストに、指定したホストが追加されます。

- qconf -ds *hostname*

  -ds オプション (発行ホストの削除) を使用すると、発行ホストのリストから、指定したホストが削除されます。

- qconf -ss

  -ss オプション (発行ホストの表示) を使用すると、現在構成されているすべての発行ホストのリストが表示されます。

QMON を使用したホストグループの構成

ホストグループを使用すると、1 つの名前を使用して複数のホストを参照できます。同じようなホストは、ホストグループにグループ化してまとめることができます。ホストグループは、複数の個別ホストだけでなく、そのほかのホストグループを含むこともできます。別のホストグループのメンバーであるホストグループは、そのホストグループのサブグループになります。

たとえば、次のメンバーを含む @bigMachines というホストグループを定義できます。

```
@solaris64
@solaris32
fangorn
balrog
```

最初の @ 記号は、その名前がホストグループであることを示します。ホストグループ @bigMachines には、2 つのサブグループ @solaris64 および @solaris32 のメンバーであるすべてのホストが含まれます。また、@bigMachines には、2 つの個別ホスト fangorn および balrog も含まれます。

「QMON Main Control」ウィンドウで「Host Configuration」ボタンをクリックします。「Host Configuration」ダイアログボックスが表示されます。

「Host Groups」タブをクリックします。「Host Groups」タブは次の図のようになっています。

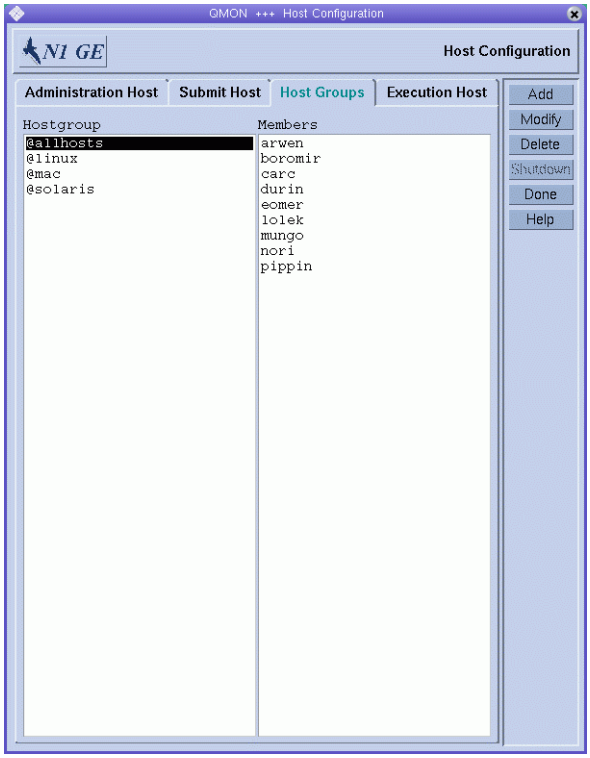
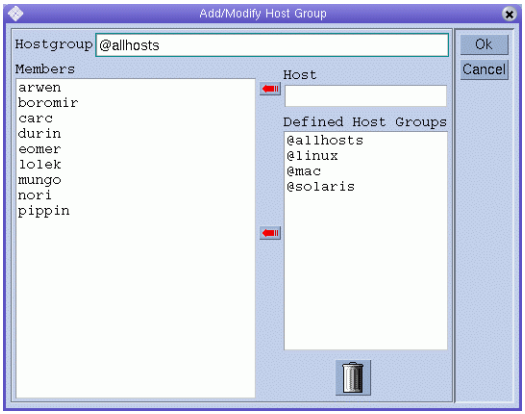


図 1–5 「Host Groups」タブ

ホストグループを構成するには、「Host Groups」タブを使用します。「Hostgroup」リストには、現在構成されているホストグループが表示されます。「Members」リストには、選択したホストグループのメンバーであるすべてのホストが表示されます。

ホストグループの追加または変更

ホストグループを追加するには「Add」をクリックし、ホストグループを変更するには「Modify」をクリックします。「Add/Modify Host Group」ダイアログボックスが表示されます。



新しいホストグループを追加する場合、「Hostgroup」フィールドにホストグループ名を入力します。ホストグループ名は「@」記号で始まる必要があります。

既存のホストグループを変更する場合は、「Hostgroup」フィールドにホストグループ名が入力されています。

現在構成中のホストグループにホストを追加するには、「Host」フィールドにホスト名を入力してから赤い矢印をクリックし、「Members」リストに名前を追加します。ホストグループをサブグループとして追加するには、「Defined Host Groups」リストからホストグループ名を選択してから赤い矢印をクリックし、その名前を「Members」リストに追加します。

「Members」リストからホストまたはホストグループを削除するには、ホストまたはホストグループを選択してから、ゴミ箱のアイコンをクリックします。

変更を保存するには「Ok」をクリックし、ダイアログボックスを閉じます。変更を保存せずにダイアログボックスを閉じるには、「Cancel」をクリックします。

ホストグループの削除

ホストグループを削除するには、「Hostgroup」リストからホストグループを選択してから「Delete」をクリックします。

コマンド行からのホストグループの構成

コマンド行からホストグループを構成するには、qconf コマンドで次の引数を使用します。

- `qconf -ahgrp [`*host-group-name*`]`

  -ahgrp オプション (ホストグループの追加) を使用すると、ホストグループのリストに新しいホストグループが追加されます。構成の書式の詳細な説明については、hostgroup(5) のマニュアルページを参照してください。

- `qconf -Ahgrp [`*filename*`]`

  -Ahgrp オプション (ファイルからのホストグループの追加) を使用すると、*filename* で定義されているホストグループ構成を含むエディタが表示されます。このエディタは、デフォルトの vi エディタか、EDITOR 環境変数に対応するエディタのいずれかです。ホストグループを構成するには、構成を変更してディスクに保存します。

- `qconf -dhgrp` *host-group-name*

  -dhgrp オプション (ホストグループの削除) を使用すると、ホストグループのリストから、指定したホストグループが削除されます。ホストグループ構成のすべてのエントリは失われます。

- `qconf -mhgrp` *host-group-name*

  -mhgrp オプション (ホストグループの変更) を使用すると、テンプレートとして、指定したホストグループの構成を含むエディタが表示されます。このエディタは、デフォルトの vi エディタか、EDITOR 環境変数に対応するエディタのいずれかです。ホストグループ構成の変更は、テンプレートを変更してディスクに保存することで行います。

- `qconf -Mhgrp` *filename*

  -Mhgrp オプション (ファイルからのホストグループの変更) を使用すると、*filename* の内容をホストグループの構成テンプレートとして使用します。指定したファイルの構成は、既存のホストグループを参照する必要があります。このホストグループの構成は、ファイルの内容により置き換えられます。

- `qconf -shgrp` *host-group-name*

  -shgrp オプション (ホストグループの表示) を使用すると、指定したホストグループの構成が表示されます。

- `qconf -shgrp_tree` *host-group-name*

  -shgrp_tree オプション (ツリーとしてホストグループを表示) を使用すると、指定したホストグループとそのサブホストグループの構成がツリーとして表示されます。

- `qconf -shgrp_resolved` *host-group-name*

-shgrp_resolved オプション (解釈処理されたホストリストとともにホストグループを表示) を使用すると、解釈処理されたホストリストとともに、指定したホストグループの構成が表示されます。

- qconf -shgrpl

  -shgrpl オプション (ホストグループリストの表示) を使用すると、すべてのホストグループのリストが表示されます。

qhost を使用した実行ホストの監視

実行ホストの状態の概要をすばやく取得するには、次のように qhost コマンドを使用します。

```
% qhost
```

このコマンドにより、次の例のような出力が作成されます。

例 1–1 qhost の出力例

```
HOSTNAME                ARCH         NCPU  LOAD  MEMTOT  MEMUSE  SWAPTO  SWAPUS
----------------------------------------------------------------------------------
global                  -               -     -       -       -       -       -
arwen                   aix43           1     -       -       -       -       -
baumbart                irix65          2  0.00    1.1G   91.5M  128.0M     0.0
boromir                 hp11            1     -  128.0M       -  256.0M       -
carc                    lx24-amd64      2  0.00    3.8G  989.8M    1.0G     0.0
denethor                aix51           1 4.54G       -       -       -       -
durin                   lx24-x86        1  0.37  123.1M   46.5M  213.6M   26.6M
eomer                   sol-sparc64     1  0.13  256.0M  248.0M  513.0M   93.0M
lolek                   tru64           1  0.02    1.0G  790.0M    1.0G    8.0K
mungo                   lx22-alpha      1  1.00  248.9M   78.8M  129.8M    2.5M
nori                    sol-x86         2  0.38 1023.0M  372.0M  512.0M   37.0M
pippin                  darwin          1  0.00  640.0M  264.0M     0.0     0.0
smeagol                 hp11            1  0.35  512.0M  425.0M    1.0G   95.0M
```

出力形式とそのほかのオプションの詳細については、qhost(1) のマニュアルページを参照してください。

無効なホスト名

無効、予約済みなどにより使用できないホスト名のリストを次に示します。

```
global
template
```

```
all
default
unknown
none
```

コマンド行からのデーモンの終了

コマンド行から Grid Engine システムデーモンを終了するには、次のいずれかのコマンドを使用します。

```
% qconf -ke[j] {hostname,... | all}
% qconf -ks
% qconf -km
```

これらのコマンドを使用するには、管理者特権またはオペレータ特権が必要です。管理者特権およびオペレータ特権の詳細については、第 4 章を参照してください。

- `qconf –ke` コマンドは、実行デーモンを停止します。ただし、アクティブなジョブは取り消しません。システム上で `sge_execd` が実行中でない間に終了するジョブは、`sge_execd` が再起動するまで、`sge_qmaster` に報告されません。ただし、ジョブのレポートは失われません。

  `qconf -kej` コマンドは、現在アクティブなすべてのジョブを終了し、すべての実行デーモンをダウンさせます。

  クラスタ内のすべての実行ホストを停止するには、停止させる実行ホストから成る、コンマで区切られたリストを使用するか、`all` を指定します。
- `qconf -ks` コマンドは、スケジューラ `sge_schedd` を停止します。
- `qconf -km` コマンドは、強制的に `sge_qmaster` プロセスを終了させます。

停止手続きを実行する前にすべてのアクティブなジョブが終了するのを待機したい場合は、すでに説明した `qconf` シーケンスを実行する前に、各クラスタキュー、キューインスタンス、またはキュードメインに対して `qmod -dq` コマンドを使用します。 クラスタキュー、キューインスタンス、およびキュードメインの詳細については、47 ページの「キューの構成」を参照してください。

```
% qmod -dq {cluster-queue | queue-instance | queue-domain}
```

`qmod -dq` コマンドは、新しいジョブが、使用不可のキューインスタンスに対してスケジュールされることを防ぎます。デーモンを終了する前には、キューインスタンス内で実行中のジョブがなくなるまで待機する必要があります。

コマンド行からのデーモンの再起動

Grid Engine システムデーモンを再起動するマシンに、ルートとしてログインします。

次のコマンドを入力して、起動スクリプトを実行します。

```
% sge-root/cell/common/sgemaster
% sge-root/cell/common/sgeexecd
```

これらのスクリプトは、このホストで通常は実行中であるデーモンを探し、対応するデーモンを起動します。

基本クラスタ構成

基本クラスタ構成は、サイトの依存関係を反映し、Grid Engine システムの動作に影響を与えるよう構成された情報のセットです。サイトの依存関係には、`mail` や `xterm` などのプログラム用の有効なパスが含まれます。グローバル構成は、マスターホストだけでなく、Grid Engine システムプール内のあらゆるホストに用意されています。また、各ホストに対してローカルな構成を使用して、グローバル構成の特定のエントリを無効にするよう、システムを構成できます。

クラスタ管理者は、インストール後ただちに、グローバル構成とローカルホスト構成をサイトのニーズに適合させる必要があります。そのあと、構成を最新の状態に保つ必要があります。

`sge_conf`(5) のマニュアルページには、構成のエントリの詳細な説明があります。

QMON を使用したクラスタ構成の表示

「`QMON` Main Control」ウィンドウで「Cluster Configuration」ボタンをクリックします。「Cluster Configuration」ダイアログボックスが表示されます。

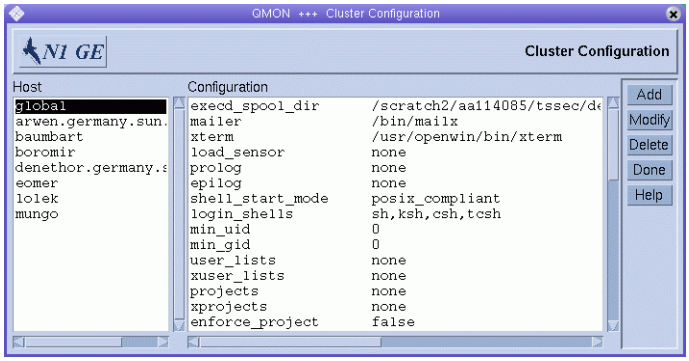


図 1–6 「Cluster Configuration」ダイアログボックス

「Host」リストで、ホストの名前を選択します。「Configuration」に、選択したホストの現在の構成が表示されます。

QMON を使用したグローバルクラスタ構成の表示

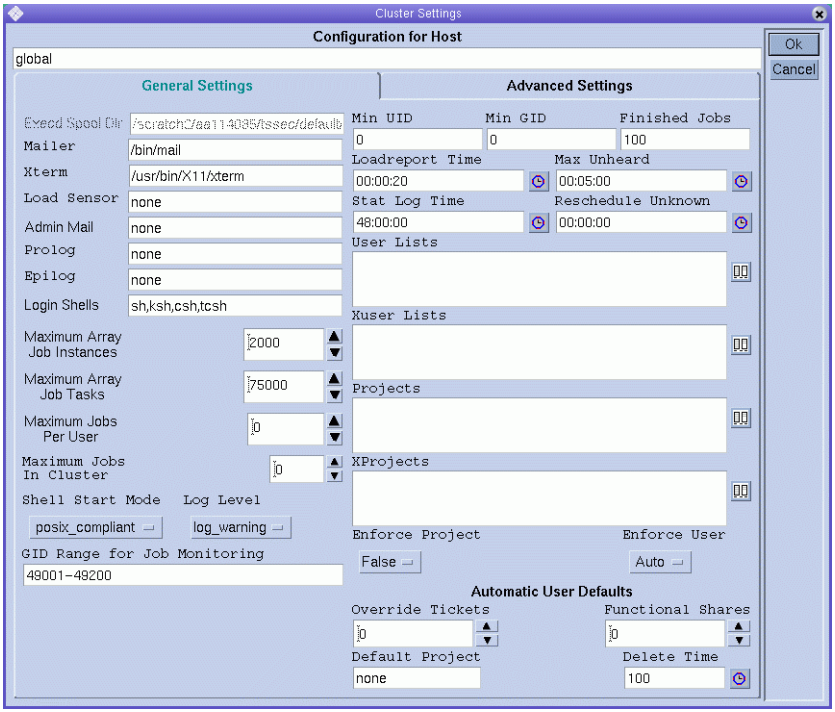
「QMON Main Control」ウィンドウで「Cluster Configuration」ボタンをクリックします。

「Host」リストで「global」を選択します。

sge_conf(5) のマニュアルページで説明されている書式で、構成が表示されます。

QMON を使用したグローバルおよびホスト構成の追加と変更

「Cluster Configuration」ダイアログボックス (図 1–6) で、ホスト名または名前「global」を選択してから「Add」または「Modify」をクリックします。「Cluster Settings」ダイアログボックスが表示されます。

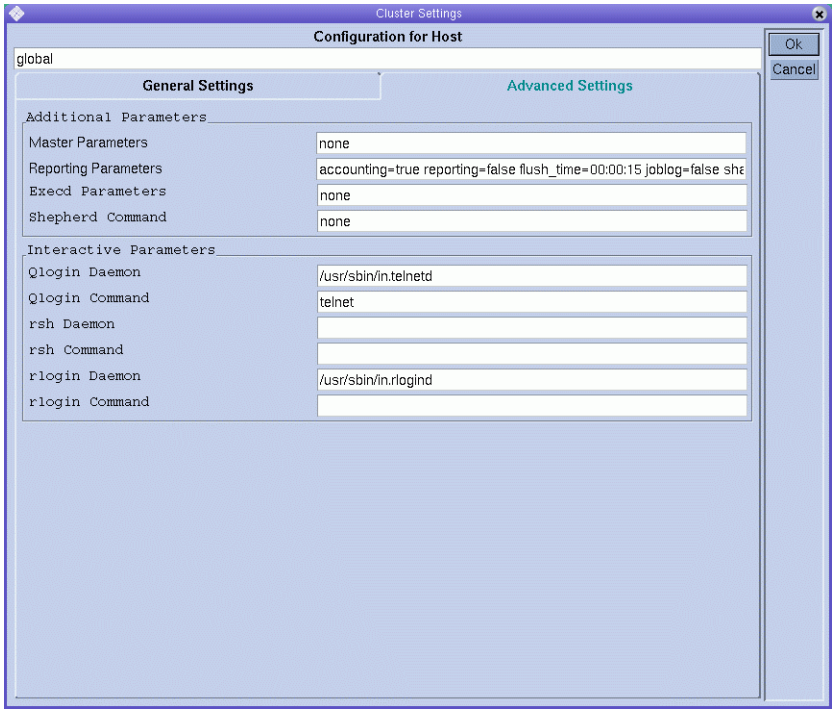


「Cluster Settings」ダイアログボックスでは、グローバル構成またはローカルホスト構成のすべてのパラメータを変更できます。

ダイアログボックスのすべてのフィールドにアクセスできるのは、グローバル構成を変更している場合のみです。ローカルホストを変更する場合、その構成はダイアログボックスに反映されます。ユーザーは、ローカルホストの変更に適用可能なパラメータのみ変更できます。

新しいローカルホスト構成を追加する場合、ダイアログボックスのフィールドは空になっています。

「Advanced Settings」タブでは、ユーザーが構成を変更しているか、新しい変更を追加しているかに応じて、対応する動作が表示されます。「Advanced Settings」タブからは、あまり使用しないクラスタ構成パラメータにアクセスできます。



変更を完了したら、「OK」をクリックして変更を保存し、ダイアログボックスを閉じます。変更を保存せずにダイアログボックスを閉じるには、「Cancel」をクリックします。

すべてのクラスタ構成パラメータの完全な説明については、`sge_conf(5)` のマニュアルページを参照してください。

QMON を使用したクラスタ構成の削除

「QMON Main Control」ウィンドウで「Cluster Configuration」ボタンをクリックします。

「Host」リストで、構成を削除するホストの名前を選択してから「Delete」をクリックします。

コマンド行からの基本クラスタ構成の表示

現在のクラスタ構成を表示するには `qconf -sconf` コマンドを使用します。詳細な説明は、`qconf(1)` のマニュアルページを参照してください。

次のいずれかのコマンドを入力します。

```
% qconf -sconf
% qconf -sconf global
% qconf -sconf host
```

- qconf –sconf コマンドと qconf –sconf global コマンドは機能が同等です。これらのコマンドはグローバル構成を表示します。
- qconf -sconf *host* コマンドは、指定したローカルホストの構成を表示します。

コマンド行からの基本クラスタ構成の変更

注 – qconf コマンドを使用してクラスタ構成を変更するには、管理者である必要があります。

次のいずれかのコマンドを入力します。

```
% qconf -mconf global
% qconf -mconf host
```

- qconf -mconf global コマンドは、グローバル構成を変更します。
- qconf -mconf *host* コマンドは、指定した実行ホストまたはマスターホストのローカル構成を変更します。

ここで説明されている qconf コマンドは、多くの使用可能な qconf コマンドの例にすぎません。そのほかの例については、qconf(1) マニュアルページを参照してください。

◆ ◆ ◆ 第 2 章

キューおよびキューカレンダの構成

この章では、キューおよびキューカレンダの構成に関する内容説明を行います。また、これらの構成方法に関する解説も行います。

この章で解説されている具体的なタスクのリストを次に示します。

キューの構成

キューとは、さまざまなカテゴリのジョブのコンテナです。キューは、同じカテゴリに属する複数のジョブの並行実行のための、対応するリソースを提供します。

N1 Grid Engine 6 では、1 つのキューは、1 つまたは複数のホストと関連付けることができます。キューは複数のホストにまたがることができるため、そのようなキューはクラスタキューと呼ばれます。クラスタキューを使用すると、1 つのクラスタキューの構成で実行ホストのクラスタを管理できます。

クラスタキューに関連付けられている各ホストは、そのホスト上に存在する、そのクラスタキューのインスタンスを受け取ります。本ガイドでは、このようなインスタンスをキューインスタンスと呼びます。クラスタキュー内では、各キューインスタンスを別々に構成できます。個別のキューインスタンスを構成することで、1 つのクラスタキュー構成で、実行ホストの異機種システム混在クラスタを管理できます。

あるクラスタキューを変更する場合、そのキューインスタンスのすべては同時に変更されます。1 つのクラスタキュー内で、複数のキューインスタンスの構成の違いを指定できます。その結果、一般的な設定には少数のクラスタキューのみが含まれ、これらのクラスタキューにより制御されるキューインスタンスは、大部分がバックグラウンドのままになります。

注 – クラスタキューとキューインスタンスの区別は重要です。たとえば、ジョブは常にキューインスタンスで実行され、クラスタキューでは実行されません。

クラスタキューを構成する際には、次のホストオブジェクトの任意の組み合わせを、クラスタキューに関連付けることができます。

- 1 つの実行ホスト
- 独立した実行ホストのリスト
- 1 つまたは複数のホストグループ

注 – 並列環境でキューが正しく動作できるようにするには、キューを並列環境に関連付けます。この関連付けによって、リソースのより多くの制御が可能になり、並列作業負荷の処理に特定のキューを割り当てることができます。

`queue_conf(5)` の `pe_list` 属性を使用して、適した並列環境を特定してください。次に、QMON ユーティリティーか次の形式の `qconf` コマンドを使用して、並列環境とキューを関連付けます。

```
# qconf -mq <queue_name>
```

ホストグループは、まとめて同じものとして扱うことができるホストのグループです。ホストグループを使用すると、1 つのホストグループ構成によって複数のホストを管理することができます。ホストグループの詳細については、36 ページの「QMON を使用したホストグループの構成」を参照してください。

個別ホストをクラスタキューに関連付ける場合、各ホスト上で結果として作成されるキューインスタンスの名前は、クラスタキュー名とホスト名を結び付けたものです。クラスタキュー名とホスト名は、@ 記号で区切ります。たとえば、ホスト `myexechost` とクラスタキュー `myqueue` を関連付ける場合、`myexechost` 上のキューインスタンスの名前は `myqueue@myexechost` になります。

ホストグループとクラスタキューを関連付ける場合は、キュードメインと呼ばれるものを作成します。キュードメインを使用すると、同じクラスタキューの一部であり、割り当てられたホストが同じホストグループの一部である、キューインスタンスのグループを管理できます。キュードメイン名は、@ を区切り記号として、クラスタキュー名とホストグループ名を結合したものです。たとえば、ホストグループ `myhostgroup` をクラスタキュー `myqueue` と関連付ける場合は、キュードメインの名前は `myqueue@@myhostgroup` になります。

注 – すべてのホストグループ名は @ 記号で始まるため、キュードメイン名には常に 2 つの @ 記号が含まれます。

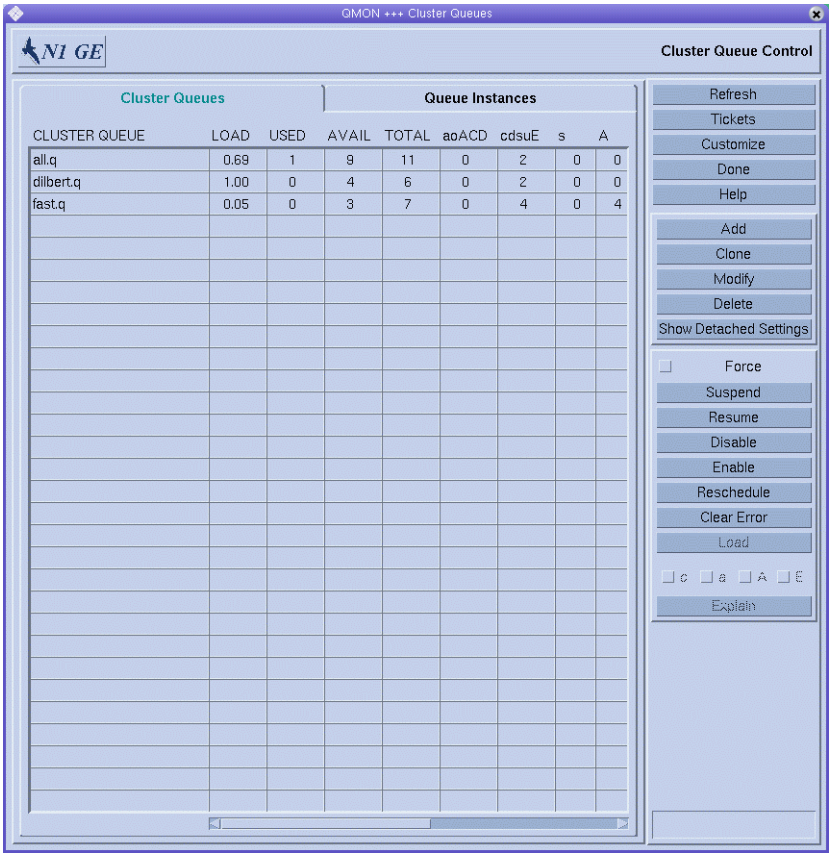
ジョブは、キューインスタンス内では待機しません。ジョブは、振り分けられるとすぐに実行を開始します。スケジューラの保留中のジョブのリストは、ジョブの唯一の待機領域です。

キューを構成すると、`sge_qmaster` にキュー属性が登録されます。キューは構成されるとすぐに、クラスタ全体と、Grid Engine システムに属する全ホスト上の全ユーザーに表示されるようになります。

詳細については、`queue_conf(5)` のマニュアルページを参照してください。

QMON を使用したキューの構成

「QMON Main Control」ウィンドウで「Queue Control」ボタンをクリックします。「Cluster Queues」ダイアログボックスが表示されます。



クラスタキューとキューインスタンスの状態を監視および操作するための、「Cluster Queues」ダイアログボックスとその機能は、『Sun N1 Grid Engine 6.1 ユーザーズガイド』の「QMON によるキューの監視と制御」で説明されています。

新しいクラスタキューを追加するには、「Add」をクリックします。

既存のクラスタキューを変更するには、「Cluster Queue」リストからクラスタキューを選択してから「Modify」をクリックします。

「Clone」ボタンを使用すると、既存のクラスタキューのすべてのパラメータをインポートできます。既存のキューのリストから、複製するキューを選択します。

「Add」をクリックすると「Queue Configuration – Add」ダイアログボックスが表示されます。「Modify」をクリックすると「Modify キュー名」ダイアログボックスが表示されます。「Queue Configuration」ダイアログボックスがはじめて表示される際には、「General Configuration」タブが表示されます。

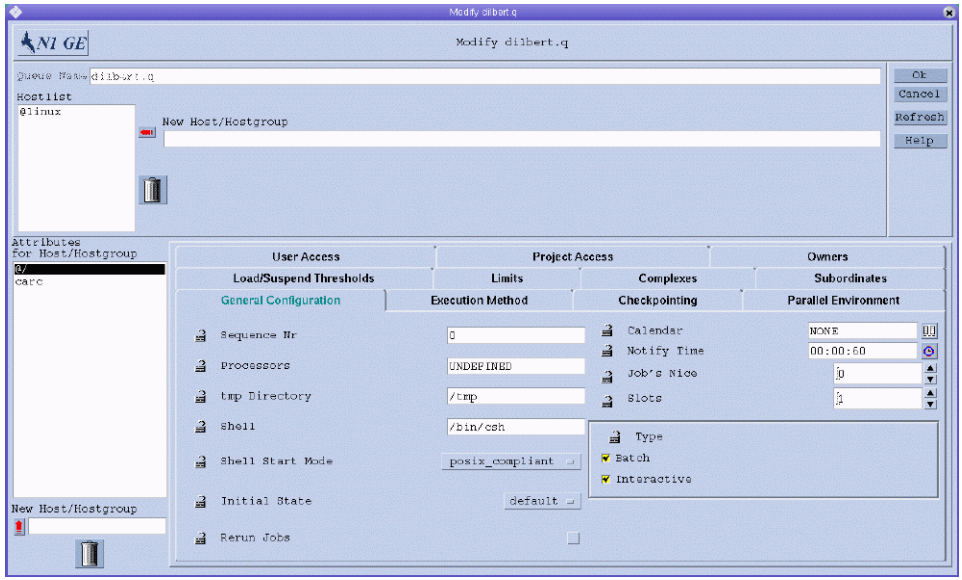


図 2–1 「Queue Configuration」–「General Configuration」タブ

既存のキューを変更する場合、「Queue Name」フィールドにキューの名前が表示されます。キューインスタンスが存在するホストが、「Hostlist」フィールドに表示されます。

新しいクラスタキューを追加する場合、キューの名前と、キューインスタンスが存在するホストの名前を指定する必要があります。

「Hostlist」フィールドでは、個別のホストの名前を指定できます。また、以前に定義したホストグループの名前も指定できます。このクラスタキューのキューインスタンスは、(すべてのホストサブグループのメンバーを含む) ユーザーが指定するホストグループのすべての個別ホストとすべてのメンバー上に存在します。ホストグループの詳細については、36 ページの「QMON を使用したホストグループの構成」を参照してください。

パラメータのセットを指定するための次の 11 のタブは、キューの定義に使用できます。

- 「General Configuration」 – 52 ページの「一般的なパラメータの構成」を参照
- 「Execution Method」 – 53 ページの「実行方法パラメータの構成」を参照
- 「Checkpointing」 – 54 ページの「チェックポイント設定パラメータの構成」を参照
- 「Parallel Environment」 – 55 ページの「並列環境の構成」を参照
- 「Load/Suspend Thresholds」 – 56 ページの「負荷および一時停止しきい値の構成」を参照
- 「Limits」 – 58 ページの「制限の構成」を参照

- 「Complex」 – 59 ページの「コンプレックスリソース属性の構成」を参照
- 「Subordinates」 – 60 ページの「従属キューの構成」を参照
- 「User Access」 – 61 ページの「ユーザーアクセスパラメータの構成」を参照
- 「Project Access」 – 62 ページの「プロジェクトアクセスパラメータの構成」を参照
- 「Owners」 – 63 ページの「所有者パラメータの構成」を参照

クラスタキューのデフォルトのパラメータを設定するには、「Attributes for Host/Hostgroup」リストで「@/」を選択してから、設定するパラメータが含まれるタブをクリックします。

デフォルトのパラメータは、「Hostlist」に表示されるすべてのホスト上のすべてのキューインスタンスに対して設定されます。ユーザーは、ユーザーが指定するホストまたはホストグループのデフォルトのパラメータの値を無効にすることができます。ホストまたはホストグループの override パラメータを設定するには、まず「Attributes for Host/Hostgroup」リストから名前を選択します。続いて設定するパラメータを含むタブをクリックします。ユーザーが設定したパラメータの値は、選択したホストまたはホストグループ上の、クラスタキューのデフォルトのパラメータを無効にします。

ホスト固有のパラメータを設定するには、まず構成に対してパラメータを使用可能にする必要があります。設定するパラメータの左側にある錠前のアイコンをクリックしてから、そのパラメータの値を変更します。

「Refresh」ボタンをクリックすると、「Queue Configuration」ダイアログボックスが開かれていた間に変更されたそのほかのオブジェクトの設定が読み込まれます。

すべてのキュー構成の変更を `sge_qmaster` に登録するには、「OK」をクリックしてダイアログボックスを閉じます。変更を保存せずにダイアログボックスを閉じるには、「Cancel」をクリックします。

一般的なパラメータの構成

一般的なパラメータを構成するには、「General Configuration」タブをクリックします。図 2–1 に「General Configuration」タブを示します。

次のパラメータを指定できます。

- Sequence Nr。キューのシーケンス番号です。
- Processors。そのキューで実行中のジョブにより使用するよう設定された、プロセッサの指示子です。一部のオペレーティングシステムアーキテクチャーに関しては、この指示子は、1 から 4、8、10 などの範囲を取ることも、プロセッサセットの整数の識別子のみを取ることもできます。詳細については、使用中の N1 Grid Engine 6.1 ソフトウェア配布の `doc` ディレクトリにある `arc_depend_*.asc` ファイルを参照してください。

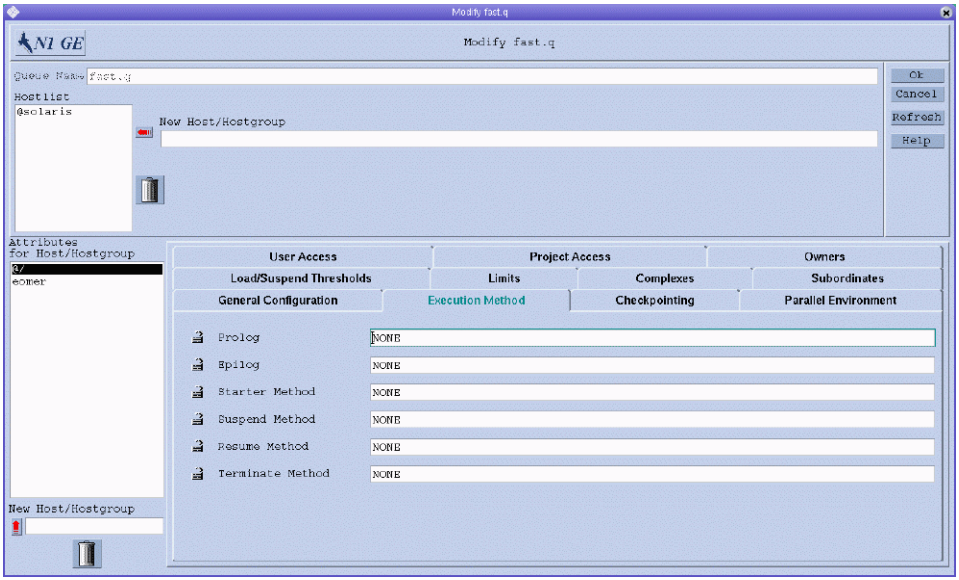
注意 – この値は、確実に必要な場合以外は変更しないでください。

- tmp Directory。一時ディレクトリのパスです。
- Shell。ジョブスクリプトを実行するために使用する、デフォルトのコマンドインタプリタです。
- Shell Start Mode。ジョブスクリプトを開始するモードです。
- Initial State。新しく追加されたキューの状態です。また、キューインスタンスホスト上で実行中の `sge_execd` が再起動した場合に、キューインスタンスが復旧する状態を示すこともあります。
- Rerun Jobs。システムクラッシュなどにより異常終了したジョブに対して強制される、キューのデフォルトのリターンポリシーです。ユーザーは `qsub -r` コマンドまたは「Submit Job」ダイアログボックスを使用することで、このポリシーを無効にできます。『Sun N1 Grid Engine 6.1 ユーザーズガイド』の「拡張ジョブの例」を参照してください。
- Calendar。キューに関連付けられたカレンダです。このカレンダは、キューの *on-duty* および *off-duty* 時間を定義します。
- Notify Time。`SIGUSR1/SIGUSR2` 通知シグナルを送信してから一時停止または 終了シグナルを送信するまでの待ち時間です。
- Job's Nice。このキューでジョブを開始するのに使用する *nice* 値です。`0` は、システムデフォルトを使用することを意味します。
- Slots。キューでの並行実行が可能なジョブの数です。スロットは、ジョブスロットとも呼ばれます。
- 型。キューおよびこのキューで実行可能なジョブの種類です。種類は Batch、Interactive のいずれか、または両方を取ることができます。

これらのパラメータの詳細については、`queue_conf(5)` のマニュアルページを参照してください。

実行方法パラメータの構成

実行方法パラメータを構成するには、「Execution Method」タブをクリックします。次の図に「Execution Method」タブを示します。



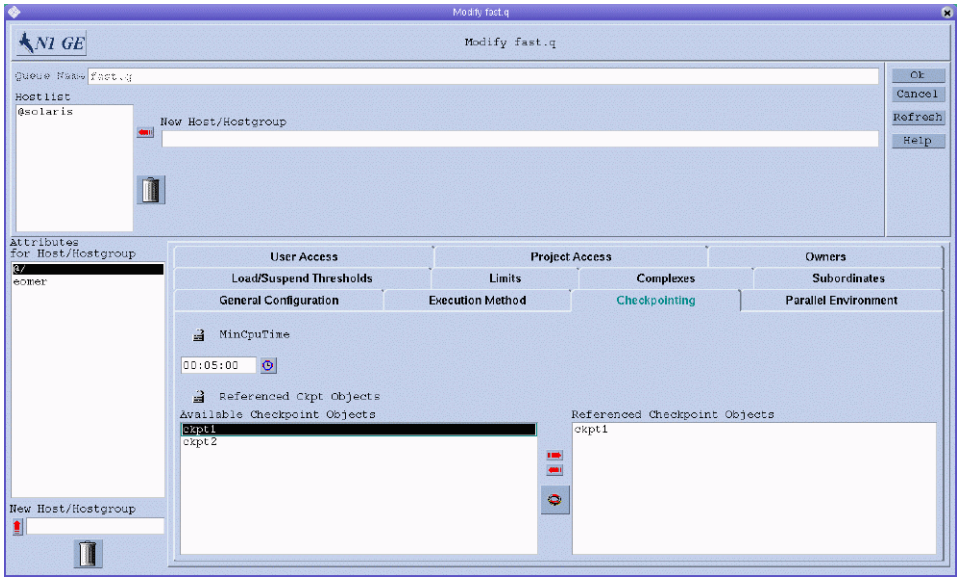
次のパラメータを指定できます。

- Prolog。キュー固有のプロローグスクリプトです。プロローグスクリプトは、ジョブスクリプトが起動される前のジョブと同じ環境で実行されます。
- Epilog。キュー固有のエピローグスクリプトです。エピローグスクリプトは、ジョブが完了したあとのジョブと同じ環境で実行されます。
- Starter Method、Suspend Method、Resume Method、Terminate Method。これらのフィールドを使用してデフォルトの方法を無効にし、これらのアクションをジョブに適用します。

これらのパラメータの詳細については、`queue_conf(5)` のマニュアルページを参照してください。

チェックポイント設定パラメータの構成

チェックポイント設定パラメータを構成するには、「Checkpointing」タブをクリックします。次の図に「Checkpointing」タブを示します。



次のパラメータを指定できます。

- MinCpuTime。定期的なチェックポイントの間隔です。
- Referenced Ckpt Objects。キューと関連付けられたチェックポイント設定環境のリストです。

キューからチェックポイント設定環境を参照するには、「Available」リストからチェックポイント設定環境の名前を選択し、右矢印をクリックしてそれを「Referenced」リストに追加します。

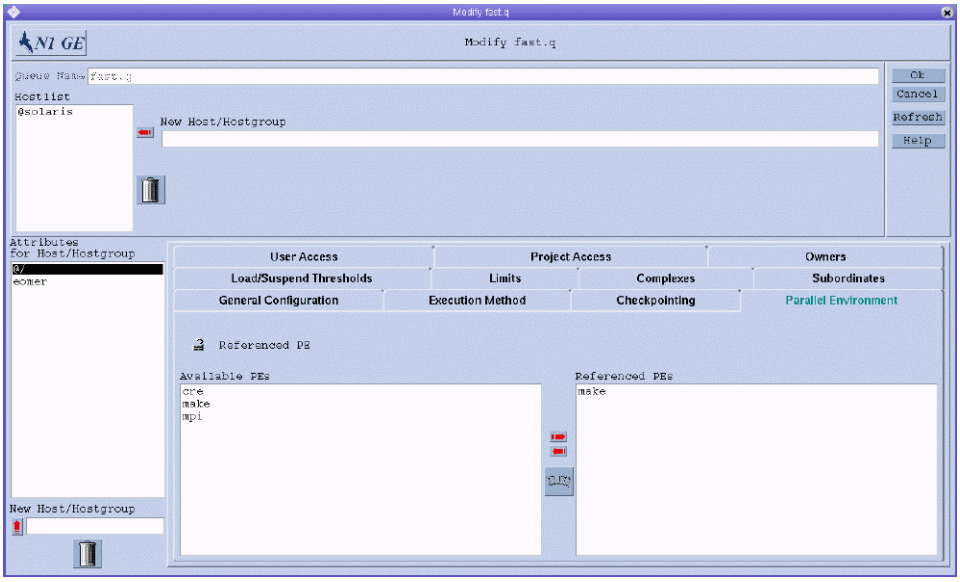
「Referenced」リストからチェックポイント設定環境を削除するには、それを選択してから左矢印をクリックします。

チェックポイント設定環境を追加または変更するには、赤い矢印の下にあるボタンをクリックして、「Checkpointing Configuration」ダイアログボックスを開きます。詳細については、187 ページの「QMON を使用したチェックポイント設定環境の構築」を参照してください。

これらのパラメータの詳細については、queue_conf(5) のマニュアルページを参照してください。

並列環境の構成

並列環境を構成するには、「Parallel Environment」タブをクリックします。次の図に「Parallel Environment」タブを示します。



次のパラメータを指定できます。

- Referenced PE。キューと関連付けられた並列環境のリストです。

キューから並列環境を参照するには、「Available PEs」リストから並列環境の名前を選択し、右矢印をクリックしてそれを「Referenced PEs」リストに追加します。

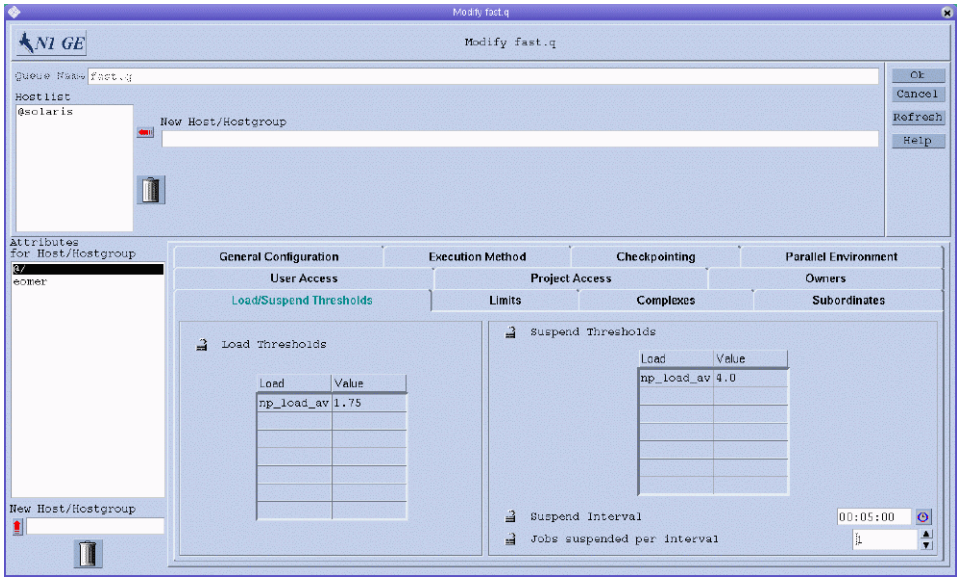
「Referenced PEs」リストからチェックポイント環境を削除するには、それを選択してから左矢印をクリックします。

並列環境を追加または変更するには、赤い矢印の下にあるボタンをクリックして、「Parallel Environment Configuration」ダイアログボックスを開きます。詳細については、176 ページの「QMON を使用した並列環境の構成」を参照してください。

このパラメータの詳細については、queue_conf(5) のマニュアルページを参照してください。

負荷および一時停止しきい値の構成

負荷および一時停止しきい値を構成するには、「Load/Suspend Thresholds」タブをクリックします。次の図に「Load/Suspend Thresholds」タブを示します。



次のパラメータを指定できます。

- 負荷パラメータの過負荷しきい値、および消費可能リソース属性を定義する、「Load Thresholds」および「Suspend Thresholds」テーブルです。71 ページの「コンプレックスリソース属性」を参照してください。

  負荷しきい値に達した場合は、過負荷により、キューがさらにジョブを受け取ることはありません。一時停止しきい値に達した場合は、負荷を軽減するため、過負荷によりキューのジョブが一時停止されます。

  テーブルには、現在構成されているしきい値が表示されます。

  既存のしきい値を変更するには、それを選択してから、対応する「Value」フィールドをダブルクリックします。

  新しいしきい値を追加するには、「Load」または「Value」をクリックします。キューに関連付けられているすべての有効な属性とともに、選択リストが表示されます。図 1–2 に「Attribute Selection」ダイアログボックスを示します。対応するしきい値テーブルの「Load」カラムに属性を追加するには、属性を選択してから「OK」をクリックします。

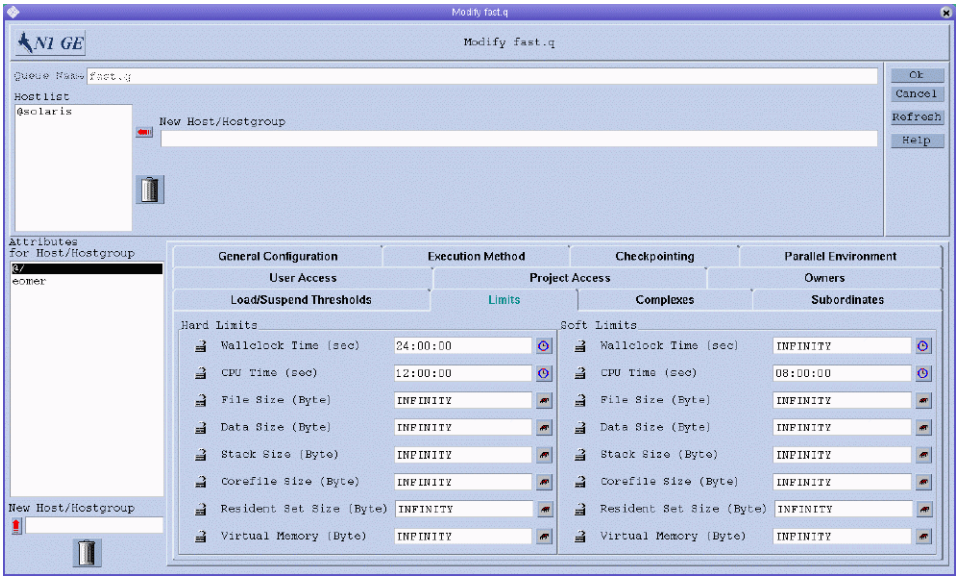
  既存のしきい値を削除するには、それを選択してから Control + D キーを押すか、マウスボタン 3 をクリックします。選択項目の削除を確認するプロンプトが表示されます。

- Suspend interval。依然として一時停止しきい値を上回っている場合の、そのほかのジョブの、一時停止の間の時間間隔。
- Jobs suspended per interval。構成したキューをホスティングしているシステム上の負荷を軽減するために、時間間隔ごとに一時停止するジョブの数です。

これらのパラメータの詳細については、queue_conf(5) のマニュアルページを参照してください。

制限の構成

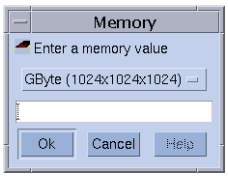
制限のパラメータを構成するには、「Limits」タブをクリックします。次の図に「Limits」 タブを示します。

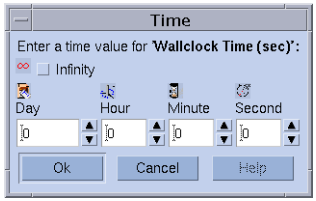


次のパラメータを指定できます。

- Hard Limit および Soft Limit。キューで実行中のジョブに対して課す強い制限値および弱い制限値です。

制限値を変更するには、値を変更するフィールドの右側にあるボタンをクリックします。Memory または Time 制限値のいずれかを入力できるダイアログボックスが表示されます。

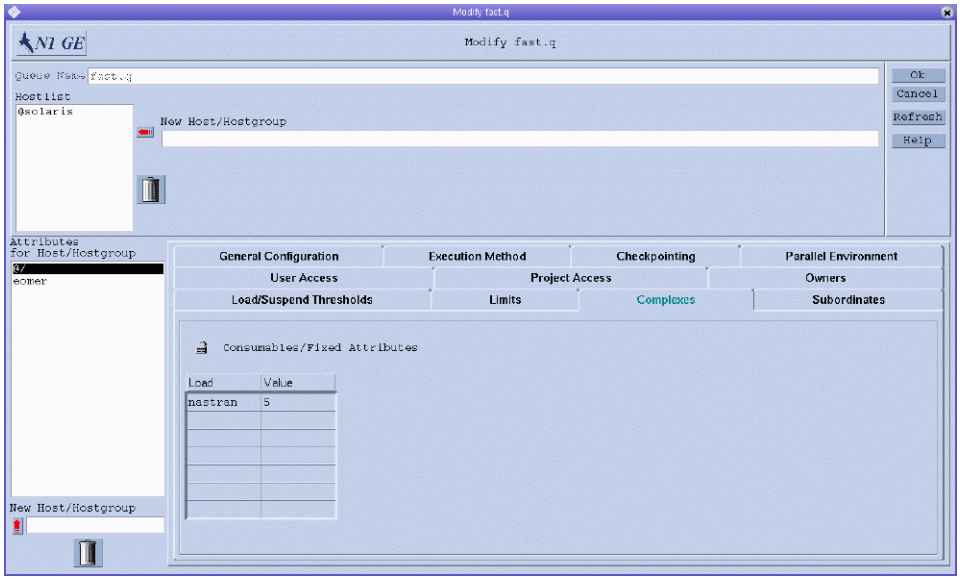




制限パラメータと、さまざまなオペレーティングシステムアーキテクチャーにおけるそれらの解釈の詳細については、queue_conf(5) および setrlimit(2) のマニュアルページを参照してください。

コンプレックスリソース属性の構成

リソース属性を構成するには、「Complex」タブをクリックします。次の図に「Complex」タブを示します。



次のパラメータを指定できます。

- Consumables/Fixed Attributes。このキューに使用可能なリソース属性のセットから選択された属性の値の定義。

  使用可能なリソース属性は、デフォルトでコンプレックスからアセンブルされます。

  リソース属性は、消費可能または固定のいずれかです。消費可能な値の定義は、キューにより管理される容量を定義します。固定値の定義は、キュー固有の値を定義します。詳細については、71 ページの「コンプレックスリソース属性」を参照してください。

  値が明示的に定義されている対象の属性は、「Consumable/Fixed Attributes」テーブルに表示されます。属性を変更するには、それを選択してから、対応する「Value」フィールドをダブルクリックします。

  新しい属性の定義を追加するには、「Load」または「Value」をクリックします。キューに関連付けられているすべての有効な属性のリストとともに、「Attribute Selection」ダイアログボックスが表示されます。図 1–2 に「Attribute Selection」ダイアログボックスを示します。

  属性テーブルの「Load」カラムに属性を追加するには、それを選択してから「OK」をクリックします。

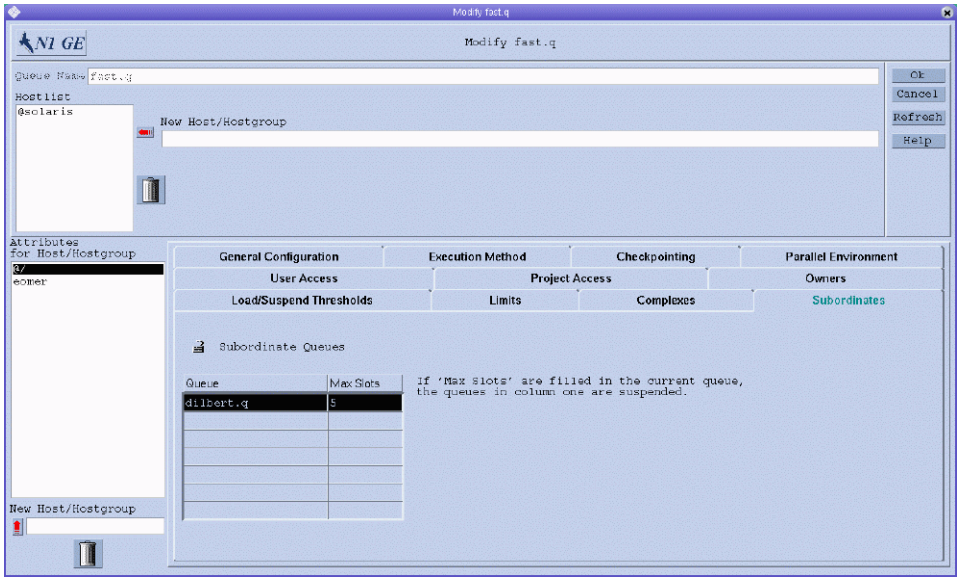
  属性を削除するには、属性を選択してから Control + D キーを押すか、マウスボタン 3 をクリックします。その属性を削除することを確認するプロンプトが表示されます。

これらの属性の詳細については、`queue_conf`(5) のマニュアルページを参照してください。

ユーザー定義リソース属性をキューに関連付ける前、またはキューからユーザー定義リソース属性を切り離す前には、「Complex Configuration」ダイアログボックスを使用して、現在のコンプレックス構成を確認または変更します。「Complex Configuration」ダイアログボックスにアクセスするには、「`QMON` Main Control」ウィンドウの「Complex Configuration」ボタンをクリックします。例は、図 3–1 を参照してください。

従属キューの構成

従属キューを構成するには、「Subordinates」タブをクリックします。次の図に「Subordinates」 タブを示します。



従属キューの機能を使用して、高優先順位および低優先順位のキューだけでなく、スタンドアロンキューを実装します。

次のパラメータを指定できます。

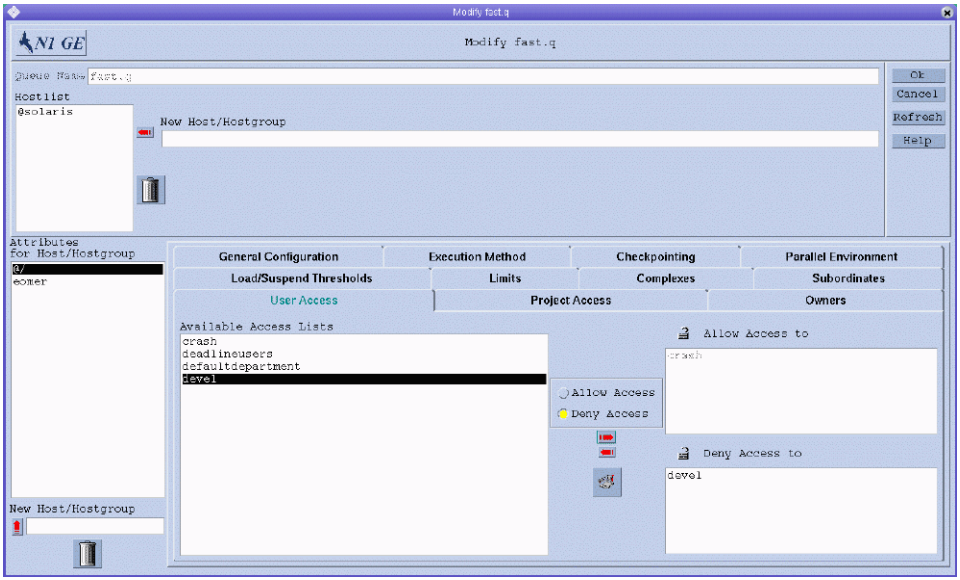
- Queue。構成したキューに対して従属しているキューのリストです。

  構成したキューがビシーになった場合、従属キューは一時停止されます。構成したキューがビシーでなくなれば、従属キューは再開されます。
- Max Slots。すべての従属キューに関して、一時停止をトリガーするために、構成したキューで満たされなければならないジョブスロットの数を構成できます。最大スロット値が指定されている場合、対応するキューの一時停止をトリガーするためには、すべてのジョブスロットが満たされていなければなりません。

これらのパラメータの詳細については、`queue_conf`(5) のマニュアルページを参照してください。

ユーザーアクセスパラメータの構成

ユーザーアクセスパラメータを構成するには、「User Access」タブをクリックします。次の図に「User Access」 タブを示します。



次のパラメータを指定できます。

- Available Access Lists。キューの「Allow Access」リストまたは「Deny Access」リストに追加できるユーザーアクセスリストです。

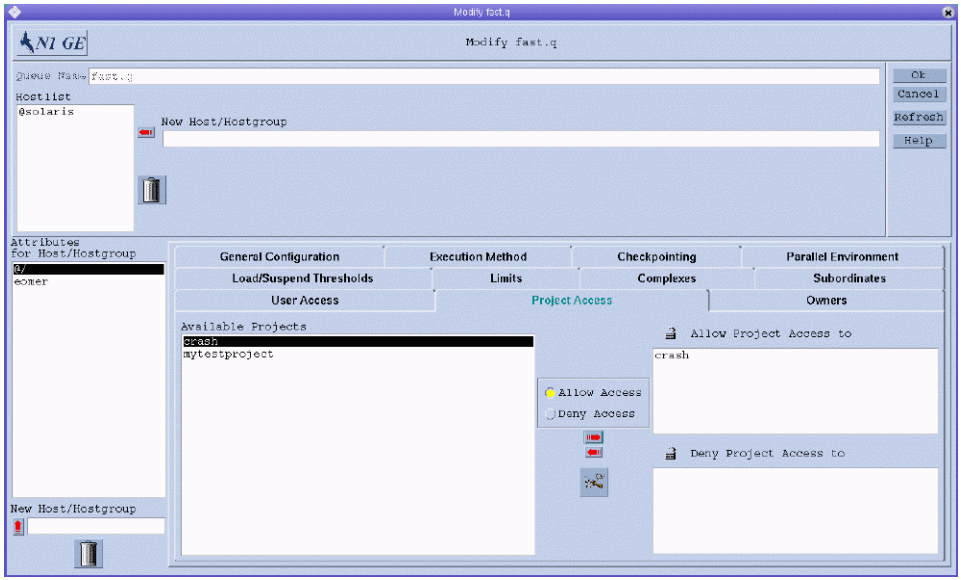
  「Allow Access」リストに含まれるアクセスリストに属するユーザーまたはユーザーグループは、キューへのアクセス権があります。「Deny Access」リストに含まれるユーザーは、キューにアクセスできません。「Allow Access」リストが空である場合、「Deny Access」リストで明示的に指定してある場合を除き、アクセスは無制限になります。

ユーザーアクセスリストを追加または変更するには、「Available Access Lists」と「Allow Access」および「Deny Access」リストの間にあるボタンをクリックして、「User Configuration」ダイアログボックスを開きます。詳細については、101 ページの「QMON を使用したユーザーアクセスリストの構成」を参照してください。

これらのパラメータの詳細については、queue_conf(5) のマニュアルページを参照してください。

プロジェクトアクセスパラメータの構成

プロジェクトアクセスパラメータを構成するには、「Project Access」タブをクリックします。次の図に「Project Access」 タブを示します。



次のパラメータを指定できます。

- Available Projects。キューへのアクセスが許可または拒否されているプロジェクトです。

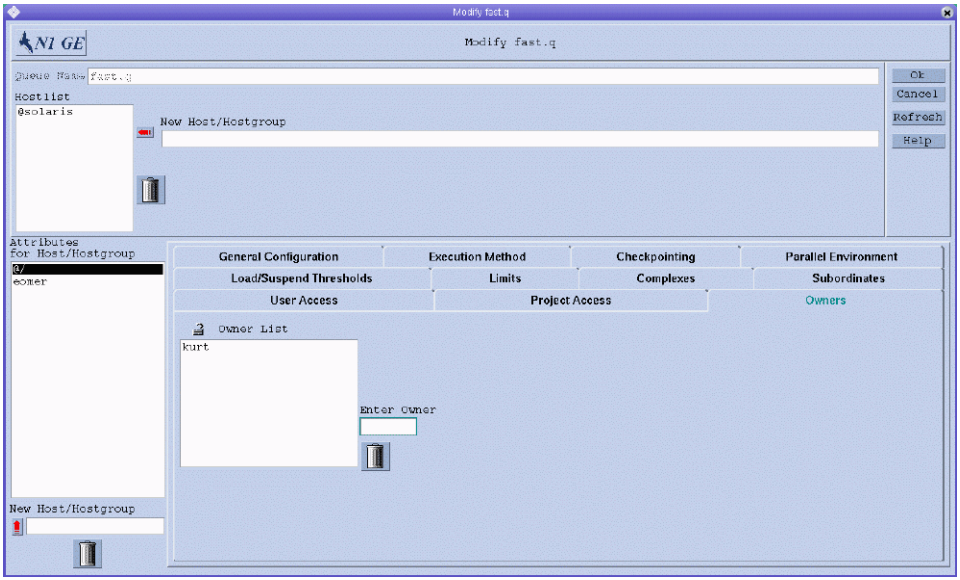
  許可されたプロジェクトのリストに属するプロジェクトに発行されたジョブは、キューへのアクセス権があります。拒否されたプロジェクトに発行されたジョブは、キューに振り分けられません。

プロジェクトアクセス権を追加または変更するには、「Available Projects」リストと「Allow Project Access」および「Deny Project Access」リストの間にあるボタンをクリックして、「Project Configuration」ダイアログボックスを開きます。詳細については、107 ページの「QMON を使用したプロジェクトの定義」を参照してください。

これらのパラメータの詳細については、queue_conf(5) のマニュアルページを参照してください。

所有者パラメータの構成

所有者パラメータを構成するには、「Owners」タブをクリックします。次の図に「Owners」 タブを示します。



次のパラメータを指定できます。

- Owner List。キューの所有者のリストです。

  一般的に、必要な場合にジョブの一時停止または使用不可をユーザーに許可するため、ユーザーはある種のキューインスタンスの所有者に設定されています。たとえば、重要な作業のためにある種のマシンがユーザーに必要になり、またこれらのマシンが、バックグラウンドで実行中のジョブにより大きな影響を受ける場合があります。 キューの所有者は次の操作を実行できます。

  - **Suspend**。キューで実行中のすべてのジョブの実行を停止し、キューを閉じます
  - **Resume**。キューの一時停止を解除して、それを開きます
  - **Disable**。キューを閉じますが、実行中のジョブには干渉しません
  - **Enable**。キューを開きます

  キューの一時停止中に明示的に一時停止されているジョブは、キューが再開された場合も再開されません。明示的に一時停止されたジョブは、明示的に再開する必要があります。

  すべての使用可能なユーザーアカウントは、所有者リストに追加できます。
  キュー所有者のリストからユーザーアカウントを削除するには、ユーザーアカウントを選択してからゴミ箱のアイコンをクリックします。

これらのパラメータの詳細については、`queue_conf`(5) のマニュアルページを参照してください。

コマンド行からのキューの構成

コマンド行からキューを構成するには、適切なオプションを使用して次のコマンドを入力します。

```
# qconf オプション
```

qconf コマンドには次のオプションがあります。

- qconf -aq [*cluster-queue*]

  -aq オプション (クラスタキューの追加) を使用すると、クラスタキュー構成用のテンプレートを含むエディタが表示されます。このエディタは、デフォルトの vi エディタか、EDITOR 環境変数により定義されたエディタのいずれかです。*cluster-queue* が指定されている場合、このクラスタキューの構成がテンプレートとして使用されます。クラスタキューを構成するには、テンプレートを変更してからそれを保存します。変更すべきテンプレートのエントリの詳細な説明については、queue_conf(5) のマニュアルページを参照してください。

- qconf -Aq *filename*

  -Aq オプション (ファイルからのクラスタキューの追加) を使用すると、ファイル *filename* を使用してクラスタキューを定義します。定義ファイルは、qconf -sq *queue* コマンドによって作成されたものである場合があります。

- qconf -cq *queue*[,...]

  -cq オプション (キューのクリーニング) を使用すると、指定したクラスタキュー、キュードメイン、またはキューインスタンスの状態がクリーニングされ、アイドル状態になり、実行中のジョブから解放されます。状態は、現在の状態に関係なくリセットされます。このオプションはエラー状態を解除するときに便利ですが、通常の運用モードでは使用しないでください。

- qconf -dq *cluster-queue*[,...]

  -dq オプション (クラスタキューの削除) を使用すると、使用可能なキューのリストから、引数リストで指定されているクラスタキューが削除されます。

- qconf -mq *cluster-queue*

  -mq オプション (クラスタキューの変更) を使用すると、指定したクラスタキューが変更されます。-mq オプションを使用すると、変更すべきクラスタキューの構成を含むエディタが表示されます。このエディタは、デフォルトの vi エディタか、EDITOR 環境変数により定義されたエディタのいずれかです。クラスタキューを変更するには、構成を変更してから変更を保存します。

- qconf -Mq *filename*

  -Mq オプション (ファイルからのクラスタキューの変更) を使用すると、ファイル *filename* を使用して、変更したクラスタキュー構成を定義します。定義ファイルは、qconf -sq *queue* コマンドとそのあとの変更によって作成されたものである場合があります。

- `qconf -sq [queue[,...]]`

  引数なしで `-sq` オプション (キューの表示) を使用すると、デフォルトのテンプレートクラスタキュー、キュードメイン、またはキューインスタンス構成が表示されます。引数付きで `-sq` オプションを使用すると、指定したキューの現在の構成が表示されます。

- `qconf -sql`

  `-sql` オプション (クラスタキューリストの表示) を使用すると、現在構成されているすべてのクラスタキューのリストが表示されます。

`qconf` コマンドには次のオプションのセットがあり、これらを使用すると特定のキュー属性を変更できます。

`-aattr` – 属性の追加
`-Aattr` – ファイルからの属性の追加
`-dattr` – 属性の削除
`-Dattr` – ファイルに表示されている属性の削除
`-mattr` – 属性の変更
`-Mattr` – ファイルからの属性の変更
`-rattr` – 属性の置換
`-Rattr` – ファイルからの属性の置換
`-sobjl` – 構成オブジェクトのリストの表示

これらのオプションの使用法の説明と、その使用例については、204 ページの「ファイルを使用した、キュー、ホスト、および環境の変更」を参照してください。これらのオプションの詳細については、`qconf`(1) のマニュアルページを参照してください。

キューカレンダの構成

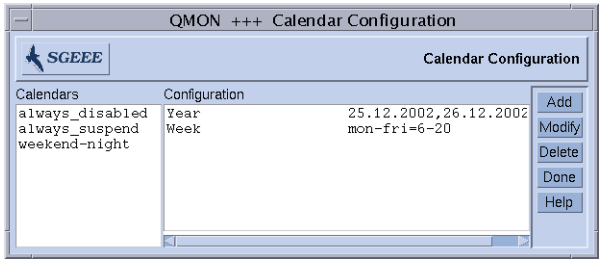
キューカレンダは、年の日付、週の曜日、または 1 日の時刻に従って、キューの可用性を定義します。特定の時間でキューの状態が変更されるよう、キューを構成できます。キューの状態は、使用不能、使用可能、一時停止、再開 (停止解除) に変更できます。

Grid Engine システムには、サイトに固有のカレンダのセットを定義する機能があります。それぞれのカレンダは、状態変更とその変更が発生する時間を指定します。これらのカレンダは、キューと関連付けることができます。各キューを 1 つのカレンダに関連付け、その関連付けたカレンダで定義されている可用性プロファイルを適用することができます。

カレンダ形式の構文は、`calendar_conf`(5) のマニュアルページで詳細に説明されています。以降の節では、対応する機能を説明するとともに、いくつか例を紹介します。

QMON を使用したキューカレンダの構成

「QMON Main Control」ウィンドウで「Calendar Configuration」ボタンをクリックします。「Calendar Configuration」ダイアログボックスが表示されます。



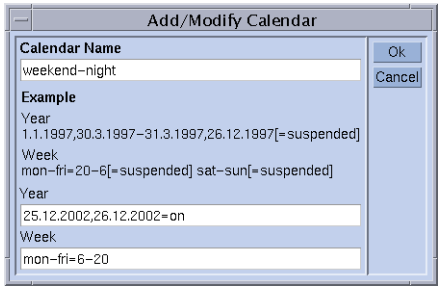
「Calendars」リストには使用可能なカレンダが表示されます。

「Calendars」リストで、変更または削除するカレンダ構成をクリックします。

次のいずれかの操作を行います。

- 選択したカレンダを削除するには「Delete」をクリックします。
- 選択したカレンダを変更するには「Modify」をクリックします。
- アクセスリストを追加するには「Add」をクリックします。

いずれの場合も「Add/Modify Calendar」ダイアログボックスが表示されます。



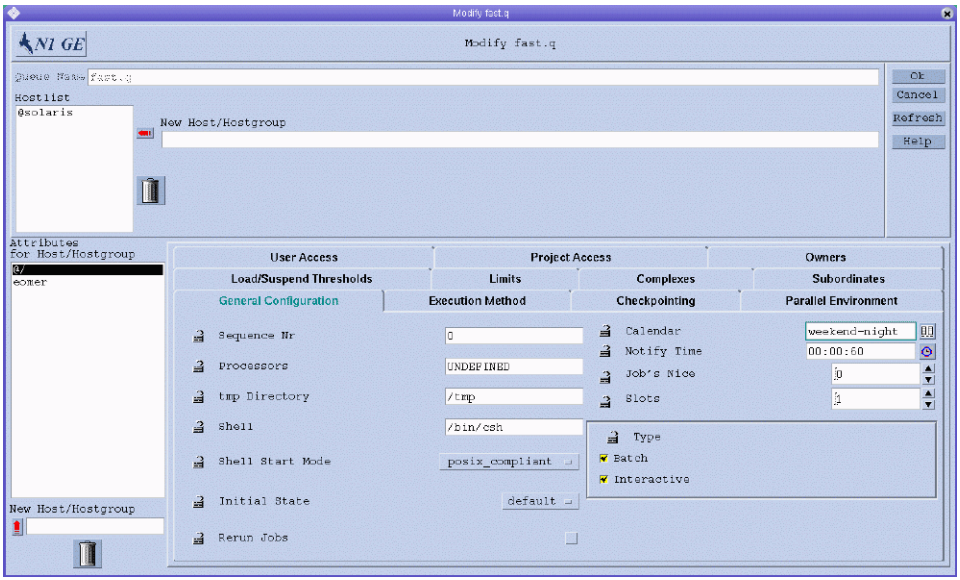
「Modify」または「Delete」をクリックした場合は、「Calendar Name」フィールドには選択したカレンダの名前が表示されます。「Add」をクリックした場合は、定義するカレンダの名前を入力します。

「Year」および「Week」フィールドでは、calendar_conf(5) のマニュアルページで説明されている構文を使用して、カレンダイベントを定義できます。

上記のカレンダ構成例は、営業時間外と週末に使用可能なキューに適しています。また、週末と同様に扱うようにクリスマス休暇を定義しています。

構文の詳細な説明とそのほかの例については、calendar_conf(5) のマニュアルページを参照してください。

キューにカレンダ構成を関連付けることによって、そのカレンダに定義されている可用性プロファイルがキューに設定されます。カレンダの関連付けは、「Modify キュー名」ダイアログボックスの「General Configuration」タブで行います。「Calendar」フィールドには、関連付けるカレンダの名前が含まれています。「Calendar」フィールドの隣りにあるボタンをクリックすると、現在構成されているカレンダが表示されます。キュー構成の詳細については、47 ページの「キューの構成」を参照してください。



コマンド行からのキューカレンダの構成

コマンド行からキューカレンダを構成するには、適切なオプションを使用して次のコマンドを入力します。

```
% qconf オプション
```

次のオプションを使用できます。

- `qconf -acal` *calendar-name*

  `-acal` オプション (カレンダの追加) を使用すると、*calendar-name* という名前の新しいカレンダ構成がクラスタに追加されます。テンプレート構成が付属するエディタが表示され、カレンダを定義できます。

- `qconf -Acal` *filename*

  `-Acal` オプション (ファイルからのカレンダの追加) を使用すると、新しいカレンダ構成がクラスタに追加されます。追加されたカレンダは、指定したファイルから読み取られます。

- `qconf -dcal` *calendar-name* `[ ,...]`

  `-dcal` オプション (カレンダの削除) を使用すると、指定したカレンダが削除されます。

- `qconf -mcal` *calendar-name*

  `-mcal` オプション (カレンダの変更) を使用すると、*calendar-name* という名前の既存のカレンダ構成が変更されます。エディタによって *calendar-name* が開かれ、定義に対して変更を行うことができます。

- `qconf -Mcal` *filename*

  `-Mcal` オプション (ファイルからのカレンダの変更) を使用すると、既存のカレンダ構成が変更されます。変更するカレンダは、指定したファイルから読み取られます。

- `qconf -scal` *calendar-name*

  `-scal` オプション (カレンダの表示) を使用すると、*calendar-name* の構成が表示されます。

- `qconf -scall`

  `-scall` オプション (カレンダリストの表示) を使用すると、構成されているすべてのカレンダのリストが表示されます。

◆ ◆ ◆ 第 3 章

3 コンプレックスリソース属性の構成

この章では、リソース属性の定義の構成方法を説明します。リソース属性の定義は、Grid Engine システムコンプレックスと呼ばれるエンティティーに格納されます。コンプレックスおよびそれに関連する概念に関する内容説明に加えて、この章では次のタスクを完了する方法を詳細に解説します。

- 72 ページの「QMON を使用したコンプレックスリソース属性の構成」
- 79 ページの「消費可能リソースの設定」
- 89 ページの「コマンド行からのコンプレックスリソース属性の構成」
- 91 ページの「独自の負荷センサーの記述」

コンプレックスリソース属性

コンプレックス構成は、`qsub -l` または `qalter -l` コマンドを使用してユーザーがジョブに対して要求できるリソース属性に関連するすべての情報を提供します。またコンプレックス構成は、Grid Engine システムがこれらのリソース属性をどのように解釈すべきであるかに関する情報も提供します。

またコンプレックスは、システムの消費可能リソース機能のフレームワークも構築します。コンプレックスで定義されるリソース属性は、グローバルクラスタ、ホスト、またはキューインスタンスに関連付けることができます。関連付けられた属性は、リソースと、関連する機能を結び付けます。スケジューリングプロセスでは、リソースとジョブ要件の可用性が考慮されます。また Grid Engine システムは、消費可能リソースの過剰な予約を防止するために必要なブックキーピングと容量計画を実行します。

一般的な消費可能リソース属性には次のものが含まれます。

- 使用可能な空きメモリー
- 使用されていないソフトウェアパッケージのライセンス
- 空きディスク容量
- ネットワーク接続の使用可能な帯域幅

Grid Engine のコンプレックスにおける属性の定義は、リソース属性をどのように解釈する必要があるかを定義します。

リソース属性の定義には次の要素が含まれます。

- 属性の名前
- 属性名を参照するためのショートカット
- STRING や TIME など、属性の値の型
- スケジューラにより使用される関係演算子
- ユーザーがジョブの属性を要求できるかどうかを決定する、要求可能なフラグ
- 属性を消費可能リソースと特定する、消費可能フラグ
- ジョブが明示的に属性に対して要求を指定しない場合に、消費可能属性に関して考慮される、デフォルトの要求値
- リソースごとにジョブの優先順位を決定する、緊急度の値

コンプレックスリソース属性を定義するには、図 3–1 に示されている「QMON Complex Configuration」ダイアログボックスを使用します。

QMON を使用したコンプレックスリソース属性の構成

「QMON Main Control」ウィンドウで「Complex Configuration」ボタンをクリックします。「Complex Configuration」ダイアログボックスが表示されます。

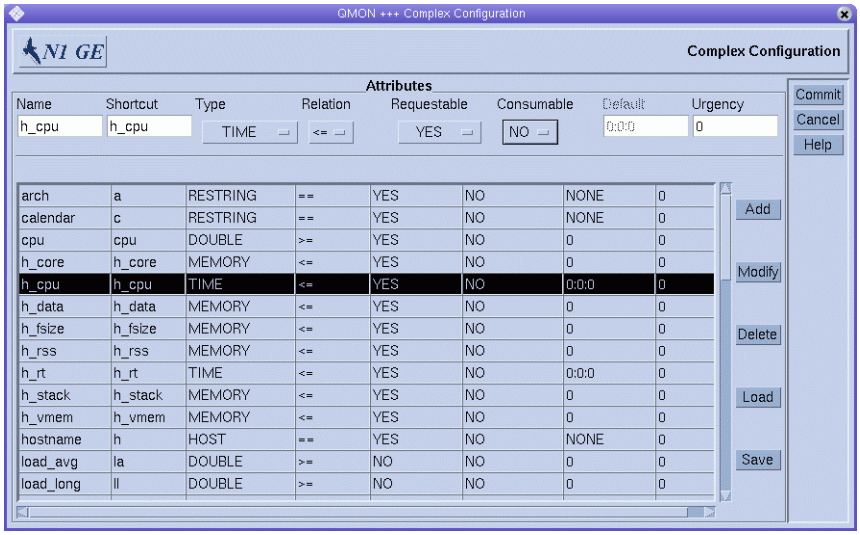


図 3–1 「Complex Configuration」ダイアログボックス

「Complex Configuration」ダイアログボックスでは、コンプレックスリソース属性の追加、変更、または削除を行うことができます。

新しい属性を追加するには、まず「Attributes」テーブルで行が選択されていないことを確認します。「Attributes」テーブル上のフィールドで、必要な値を入力または選択してから「Add」をクリックします。

注 – 新しい属性を追加する必要があり、既存の属性が選択されている場合は、選択をクリアする必要があります。選択されている属性を選択解除するには、Control キーを押したままマウスボタン 1 をクリックします。

既存の属性をコピーしてからそれを変更することによって、新しい属性を追加できます。属性名とそのショートカットが一意であることを確認します。

「Attributes」テーブルに表示されている属性を変更するには、それを選択します。選択した属性の値は、「Attributes」テーブル上に表示されます。属性値を変更してから「Modify」をクリックします。

構成の変更をファイルに保存するには、「Save」をクリックします。ファイルの値をコンプレックス構成に読み込むには、「Load」をクリックして、表示されるリストからファイルの名前を選択します。

「Attribute」テーブルの属性を削除するには、属性を選択してから「Delete」をクリックします。

テーブルの行とカラムの意味の詳細については、complex(5) のマニュアルページを参照してください。

新しいコンプレックス構成または変更したコンプレックス構成を sge_qmaster に登録するには、「Commit」をクリックします。

キュー、ホスト、およびグローバルクラスタへのリソース属性の割り当て

リソース属性は次のように使用することができます。

- キューリソース属性として
- ホストリソース属性として
- グローバルリソース属性として

各キューおよびホストには、デフォルトのリソース属性のセットがすでに関連付けられています。デフォルトのリソース属性はシステムに組み込まれ、削除したり、その型を変更することはできません。

ユーザー定義のリソース属性は、キューインスタンス、ホスト、またはグローバルクラスタに割り当てる前に、まずコンプレックスで定義する必要があります。リソース属性をこれらのターゲットの 1 つに割り当てる場合には、属性の値を指定します。

次の節では、各属性の型を詳細に説明しています。

キューリソース属性

デフォルトのキューリソース属性は、キュー構成で定義されているパラメータのセットです。これらのパラメータは、queue_conf (5) のマニュアルページで説明されています。

デフォルトの属性には新しいリソース属性を追加できます。新しい属性は、ユーザーが変更したキューインスタンスにのみ関連付けられます。特定のキューインスタンスの構成が、コンプレックスで定義されているでリソース属性を参照している場合、そのキュー構成は属性定義の値を提供します。キュー構成の詳細については、47 ページの「キューの構成」を参照してください。

たとえば、キュー構成値 h_vmem は、仮想記憶サイズの制限に使用されます。この値は、各ジョブが消費できる合計メモリーの量を制限します。キュー構成の complex_values リストにあるエントリが、ホスト上、またはキューに割り当てられた使用可能な仮想記憶の総量を定義します。 消費可能リソースの詳細については、78 ページの「消費可能リソース」を参照してください。

ホストリソース属性

ホストリソース属性は、ホストごとに管理されるパラメータです。

デフォルトのホスト関連の属性は、負荷値です。すでに74 ページの「キューリソース属性」で説明したように、デフォルト属性には新しいリソース属性を追加できます。

各 sge_execd は、定期的に sge_qmaster に対して負荷を報告します。90 ページの「負荷パラメータ」で説明されているように、報告される負荷値は、CPU 負荷の平均などの標準的な負荷値か、管理者によって定義された負荷値です。

標準的な負荷値の定義はデフォルトのホストリソース属性の一部ですが、管理者定義の負荷値は、ホストリソース属性の拡張を必要とします。

一般的にホスト関連の属性は、標準ではない負荷パラメータを含むよう拡張されます。ホスト関連の属性も、ホストに割り当てられているソフトウェアライセンスの数や、ホストのローカルファイルシステムで使用可能なディスク容量など、ホスト関連のリソースを管理するよう拡張されます。

ホスト関連の属性が、ホスト、またはそのホスト上のキューインスタンスに関連付けられている場合、特定のホストリソース属性の具体的な値は、次のいずれかの項目によって決定されます。

- キュー構成 (属性がキュー構成にも割り当てられている場合)
- 報告される負荷値
- 対応するホスト構成の complex_values エントリにある値の明示的な定義詳細については、25 ページの「ホストの構成」を参照してください。

場合によっては、これらの値が 1 つも使用できないことがあります。たとえば、値が負荷パラメータであると想定されていても、 sge_execd がそのパラメータの負荷値を報告しない場合などです。このような場合、属性は定義されておらず、また qstat –F コマンドではその属性が適用不可であることが示されます。

たとえば、空き仮想記憶の合計の属性 h_vmem は、キュー構成で制限として定義され、また標準的な負荷パラメータとしても報告されます。ホスト上の仮想記憶の使用可能な総量は、ホストの complex_values リストで定義されます。ホスト上のキューインスタンスに関連付けられた仮想記憶の使用可能な総量は、そのキューインスタンスの complex_values リストで定義できます。 h_vmem を消費可能リソースとして定義することで、(多くの場合スワッピングが原因でシステムパフォーマンスを低下させる) メモリーの過剰な予約のリスクを冒すことなく、マシンのメモリーを効率的に活用できます。消費可能リソースの詳細については、78 ページの「消費可能リソース」を参照してください。

注 – デフォルトのリソース属性に関しては、「Shortcut」、「Relation」、「Requestable」、「Consumable」、および「Default」の各カラムのみ変更可能です。デフォルトの属性は削除できません。

グローバルリソース属性

グローバルリソース属性は、ファイルサーバーの使用可能なネットワーク帯域幅や、ネットワーク全体の使用可能なファイルシステム上の空きディスク容量など、クラスタ全体のリソース属性です。

対応する負荷レポートに、90 ページの「負荷パラメータ」で説明されている GLOBAL 識別子が含まれる場合、グローバルリソース属性は、負荷レポートと関連付けることもできます。クラスタ内の任意のホストからグローバル負荷値を報告させることもできます。グローバル負荷値はデフォルトでは報告されないため、デフォルトのグローバルリソース属性は存在しません。

グローバルリソース属性の具体的な値は、次の項目によって決定されます。

- グローバル負荷レポート。
- global ホスト構成の complex_values パラメータの明示的な定義。25 ページの「ホストの構成」を参照してください。
- 特定のホストまたはキューとの関連付け、および対応する complex_values リストにおける明示的な定義

場合によっては、上記のケースがいずれも適用されません。たとえば、負荷値がまだ報告されていない場合などです。このような場合、その属性は存在しません。

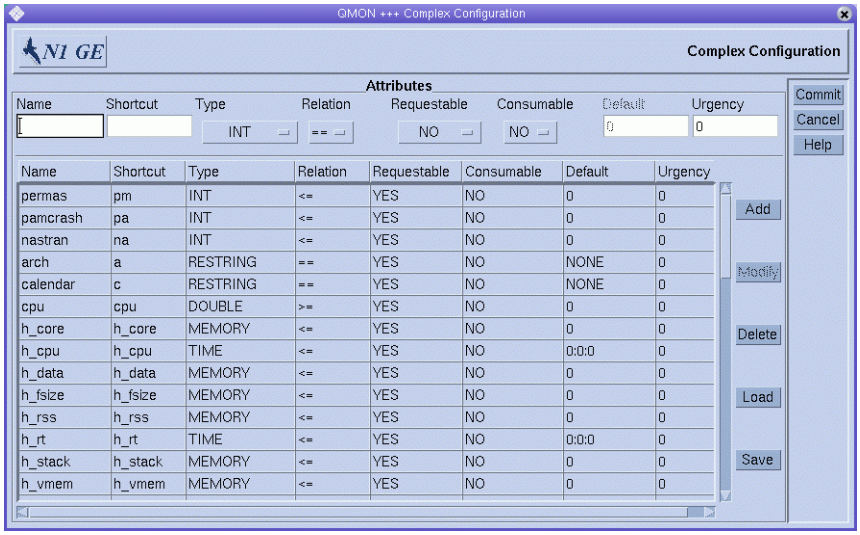
コンプレックスへのリソース属性の追加

コンプレックスにリソース属性を追加することにより、管理者は Grid Engine システムにより管理される属性のセットを拡張できます。また管理者は、ユーザー定義属性の影響を、特定のキューやホスト、またはその両方に制限することもできます。

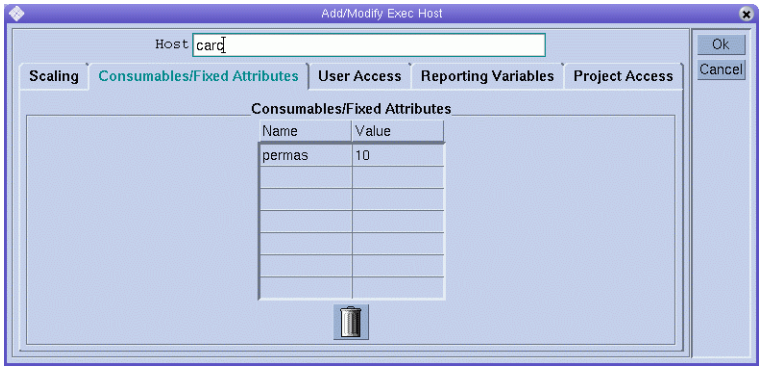
ユーザー定義属性は、Grid Engine ソフトウェアがどのように属性を処理するかに関する、対応する定義を持つ、属性の名前付きコレクションです。1 つまたは複数のユーザー定義属性を、1 つのキュー、1 つのホスト、またはグローバルにクラスタ内の全ホストに対して関連付けることができます。キュー構成およびホスト構成に対しては、complex_values パラメータを使用します。詳細については、47 ページの「キューの構成」および25 ページの「ホストの構成」を参照してください。定義済みの属性は、デフォルトのリソース属性に加えて、それぞれキューおよびホストに対して使用可能になります。

キュー構成とホスト構成の complex_values パラメータは、キューおよびホストに対して関連付けられているユーザー定義属性に対して、具体的な値を設定する必要があります。

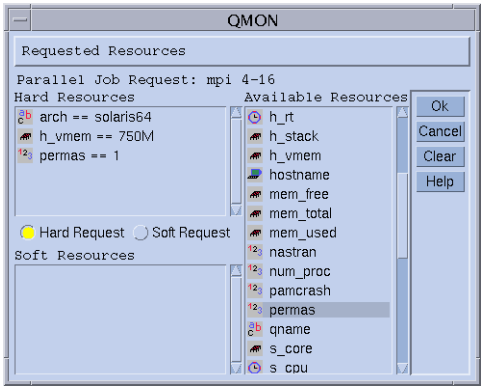
たとえば、次の図に示すユーザー定義リソース属性 permas、 pamcrash、および nastran が定義されます。



「Modify *queue-name*」ダイアログボックスの「Complex」タブに表示されているように、少なくとも 1 つ以上のキューに関して、関連付けられたユーザー定義属性のリストに、リソース属性を追加します。キューの構成方法の詳細については、47 ページの「キューの構成」とその関連の節を参照してください。



また、表示されているキューは、ソフトウェアパッケージ `permas` の最大 10 個のライセンスを管理するよう構成されています。さらに、「Requested Resources」ダイアログボックスの「Available Resources」リストに示すように、属性 `permas` はジョブに対して要求可能になります。



ジョブを発行する方法の詳細については、『Sun N1 Grid Engine 6.1 ユーザーズガイド』の第 3 章「ジョブの発行」を参照してください。

または、ユーザーは次のようにコマンド行からジョブを発行し、属性を要求することができます。

```
% qsub -l pm=1 permas.sh
```

注 – 完全な属性名 `permas` の代わりに、`pm` ショートカットを使用できます。

その結果、これらのジョブに対して適格なキューは、ユーザー定義リソース属性と関連付けられ、`permas` ライセンスが構成され使用可能になっているキューのみになります。

消費可能リソース

消費可能リソースは、使用可能なメモリー、ファイルシステム上の空き容量、ネットワーク帯域幅、浮動ソフトウェアライセンスなど、制限のあるリソースを管理するための効率的な手段になります。消費可能リソースはコンシューマブルとも呼ばれます。コンシューマブルの使用可能な総量は、管理者によって定義されます。対応するリソースの消費は、Grid Engine ソフトウェア内部のブックキーピングにより監視されます。Grid Engine システムは、すべての実行中のジョブに関してこのリソースの消費を把握します。ジョブが振り分けられるのは、内部のブックキーピングにより十分な消費可能リソースが使用可能であることが示された場合のみです。

コンシューマブルは、デフォルトの負荷パラメータや、ユーザー定義の負荷パラメータと結び付けることができます。消費可能属性に対しては、負荷値を報告できます。逆に、負荷属性に対しては Consumable フラグを設定できます。負荷は、リ

ソースの可用性の測定値になります。消費可能リソースの管理では、負荷と、内部のブックキーピングの両方が考慮され、両方とも指定の制限を超えません。負荷パラメータの詳細については、90 ページの「負荷パラメータ」を参照してください。

消費可能リソースの管理を有効にするには、リソースの総量を定義する必要があります。リソースの容量の定義は、指定したホスト、および指定したキューに対してだけでなく、クラスタに対してグローバルに行うことができます。これらのカテゴリは、列挙したのとは逆の順序で別のカテゴリに優先することができます。したがって、ホストがグローバルリソースの可用性を制限し、キューがホストリソースとグローバルリソースを制限することも可能です。

リソースの容量を定義するには、キューおよびホストの構成で `complex_values` 属性を使用します。`global` ホストの `complex_values` 定義が、グローバルクラスタのコンシューマブルの設定を定義します。詳細については、`host_conf`(5) および `queue_conf`(5) のマニュアルページだけでなく、47 ページの「キューの構成」および 25 ページの「ホストの構成」も参照してください。

`complex_values` リスト内の各消費可能属性に対して、そのリソースの使用可能な最大量を示す値が割り当てられます。内部のブックキーピングは、この合計から、(ジョブのリソース要求によって表現された) すべての実行中のジョブによる推定のリソース消費量を差し引きます。

並列ジョブは、ジョブスロットを消費するのと同じ数だけ消費可能リソースを消費します。たとえば、次のコマンドは合計 800M バイトのメモリーを消費します。

```
qsub -l mem=100M -pe make=8
```

メモリーの使用は、ジョブが実行されるキューとホストに分割されます。ホスト A で 4 つのタスクが実行され、ホスト B で 4 つのタスクが実行されている場合、そのジョブは各ホストで 400M バイトを消費します。

消費可能リソースの設定

コンシューマブルとしては数値属性のみが構成可能です。数値属性とは、その型が `INT`、`DOUBLE`、`MEMORY`、または `TIME` である属性です。

「`QMON` Main Control」ウィンドウで「Complex Configuration」ボタンをクリックします。図 3–1 に示すように、「Complex Configuration」ダイアログボックスが表示されます。

属性に関するコンシューマブルの管理を使用可能にするには、コンプレックス構成の属性に対して Consumable フラグを設定します。たとえば次の図は、`virtual_free` メモリーリソースに対して Consumable フラグが設定されていることを示します。

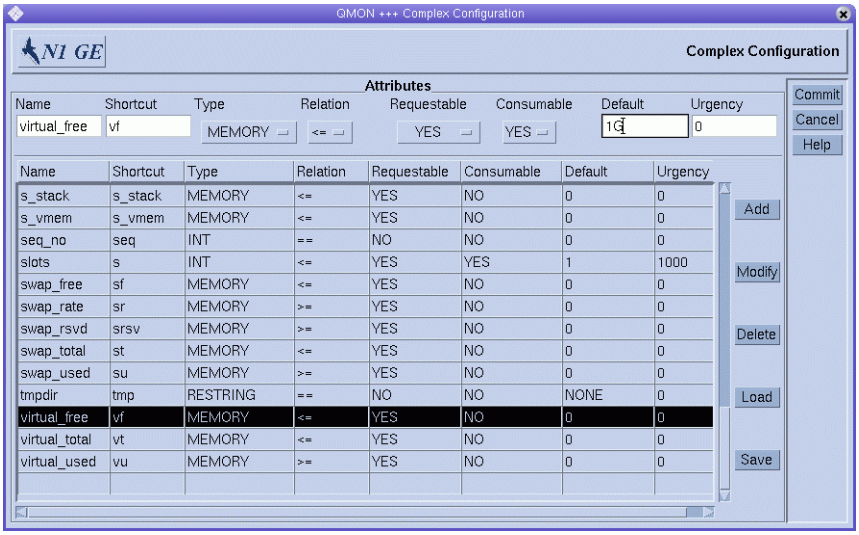


図 3–2 「Complex Configuration」ダイアログボックス: `virtual_free`

次の各節で詳細が説明されている例をガイドとして、そのほかの消費可能リソースを設定します。

- 81 ページの「例 1: 浮動ソフトウェアライセンスの管理」
- 85 ページの「例 2: 仮想記憶の容量の共有」
- 87 ページの「例 3: 使用可能なディスク容量の管理」

続いて、Grid Engine ソフトウェアに必要な容量計画を実行させる対象の各キューまたはホストに対して、`complex_values` リストで容量を定義する必要があります。次の図に示す例では、1G バイトの仮想記憶が現在のホストの容量値として定義されています。

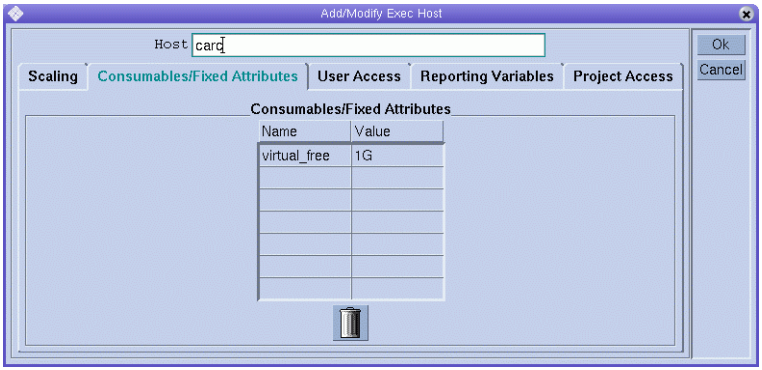


図 3–3 Add/Modify Exec Host: `virtual_free`

そのホスト上のすべてのキューで並行して実行中のすべてのジョブの仮想記憶の要件は、累積されます。続いて使用可能な仮想記憶を決定するため、1G バイトの容量から要件が差し引かれます。virtual_free に対するジョブ要求が使用可能な量を超える場合、ジョブはそのホスト上のキューに対して振り分けられません。

注 – Requestable パラメータの FORCED 値を介して、リソースを要求し、仮定の消費量を指定するよう、ジョブを強制することができます。

ジョブによって明示的には要求されていない消費可能属性に関しては、管理者はリソース消費量に対してデフォルト値を事前に定義できます。このような作業に意味があるのは、前の注で説明したように、属性の要求が強制されていない場合のみです。デフォルト値として 200M バイトが設定されています。

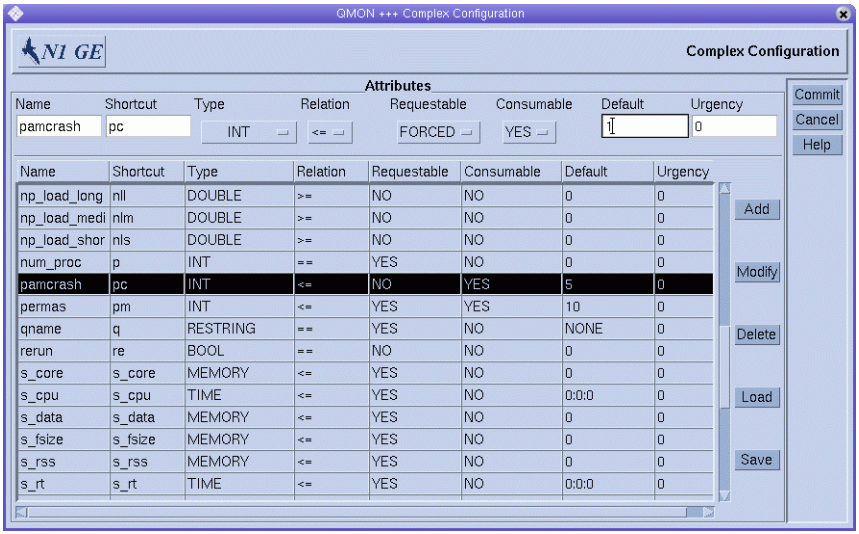
消費可能リソースの設定の例

サイトに対して消費可能リソースを設定する場合は、次の例をガイドとして使用してください。

例 1:浮動ソフトウェアライセンスの管理

クラスタでソフトウェアパッケージ pam-crash を使用し、10 個の浮動ライセンスへのアクセス権があると仮定します。ソフトウェアのアクティブな呼び出しが 10 を超えないかぎり、あらゆるシステムで pam-crash を使用できます。目標は、実行中のそのほかの pam-crash ジョブによって 10 個のライセンスがすべて占有される間は pam-crash ジョブのスケジューリングを避けるように Grid Engine システムを構成することです。

消費可能リソースを使用すると、この目標を簡単に達成できます。まず、グローバル消費可能リソースとして、使用可能な pam-crash ライセンスの数を、コンプレックス構成に追加する必要があります。



消費可能属性の名前は `pam-crash` に設定されています。その代わりに、`qalter -l`、`qselect -l`、`qsh -l`、`qstat -l`、または `qsub -l` コマンドで、ショートカットとして `pc` を使用できます。

属性の型は整数カウンタに定義されています。

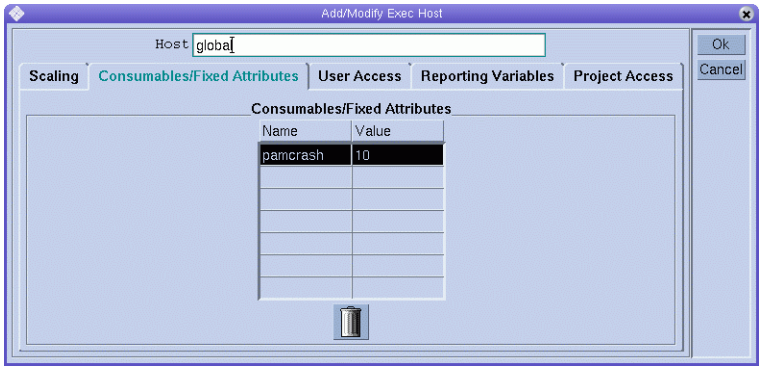
Requestable フラグは `FORCED` に設定されています。この設定では、ジョブが発行された時点で、ジョブがいくつの `pam-crash` ライセンスを占有するかをユーザーが要求しなければならないことが指定されています。

Consumable フラグは、その属性が消費可能リソースであることを指定します。

Requestable は (すべてのジョブでこの属性に対して要求値を受信する必要があることを意味する) `FORCED` に設定されているため、設定 Default は無関係になります。

コンシューマブルは、`complex_values` リストを介して、グローバル、ホスト、またはキュー構成からその値を受け取ります。`host_conf`(5) および `queue_conf`(5) のマニュアルページだけでなく、47 ページの「キューの構成」 および 25 ページの「ホストの構成」を参照してください。

この属性とクラスタに対してリソース計画をアクティブにするには、使用可能な `pam-crash` ライセンスの数をグローバルホスト構成で定義する必要があります。



属性 pam-crash の値は 10 に設定され、これは 10 個の浮動ライセンスに対応します。

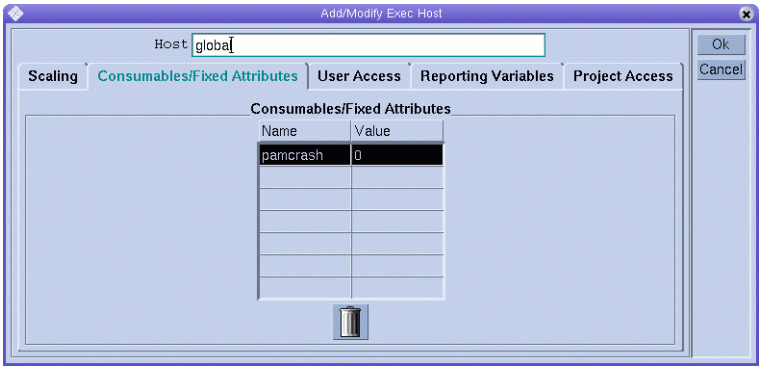
注 – テーブル Consumables/Fixed Attributes は、ホスト構成ファイル書式 host_conf(5) で説明されている complex_values エントリに対応します。

ユーザーが次のジョブを発行すると仮定します。

```
% qsub -l pc=1 pam-crash.sh
```

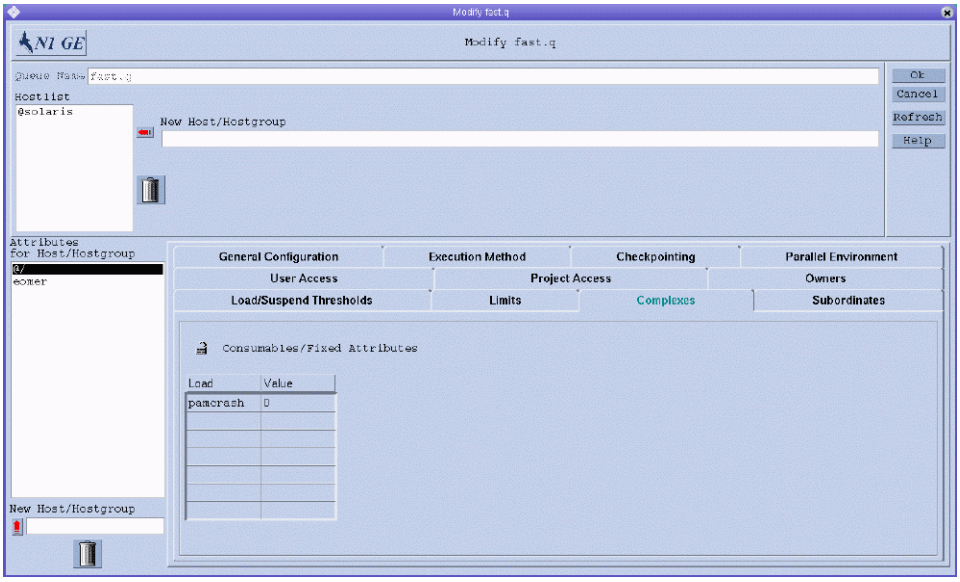
ジョブは、10 未満の pam-crash ライセンスが現在占有されている場合にのみ起動します。ジョブはクラスタ内の任意の場所で実行可能ですが、実行時間の全体において 1 つの pam-crash ライセンスを占有します。

クラスタ内のホストの 1 つを、浮動ライセンスに含めることができない場合があります。たとえば、そのホストに対しては pam-crash バイナリを使用できません。このような場合、pam-crash ライセンス管理からそのホストを除外できます。消費可能属性 pam-crash に関して、そのホストに関連する容量をゼロに設定することで、そのホストを排除できます。「Host Configuration」ダイアログボックスの「Execution Host」タブを使用します。



注 – コンプレックスのグローバル属性はすべての実行ホストにより継承されるため、pam-crash 属性は暗黙に実行ホストに対して使用可能になります。容量をゼロに設定することで、1 つのホストが管理できるライセンスの数を、2 などのゼロ以外の値に制限することもできます。この場合、そのホストには最大 2 つの pam-crash ジョブが共存できます。

同様に、あるキューが pam-crash ジョブを実行することを防止したい場合があります。たとえば、キューが、pam-crash には適切ではないメモリーと CPU 時間の制限を持つ、エクスプレスキューである場合です。この場合、次の図に示すように、キュー構成で対応する容量をゼロに設定します。



注 – コンプレックスのグローバル属性はすべてのキューにより継承されるため、pam-crash 属性は暗黙にキューに対して使用可能になります。

例 2: 仮想記憶の容量の共有

メモリーの過剰予約を原因とするパフォーマンスの低下、そしてその結果としてマシンでスワップが起きないようにシステムをチューニングすることは、システム管理者がよく行う仕事です。Grid Engine ソフトウェアは、消費可能リソースの機能を介して、ユーザーが行うこのタスクをサポートできます。

標準的な負荷パラメータ virtual_free は、使用可能な空き仮想記憶、つまり使用可能なスワップ容量と使用可能な物理メモリーの結合量を報告します。 スワッピングを回避するには、スワップ容量の使用量を最小限にする必要があります。理想的なケースでは、ホストで実行中のすべてのプロセスで必要なすべてのメモリーが、物理メモリーに収まっています。

Grid Engine ソフトウェアは、次の前提と構成の下では、Grid Engine システムを介して開始されるすべてのジョブに必要なメモリーが使用可能であることを保証できます。

- virtual_free が消費可能リソースとして構成され、各ホスト上のその容量が使用可能な物理メモリー以下に設定されている。
- ジョブは、想定されるメモリー使用量を要求し、実行中に要求値を超えない。

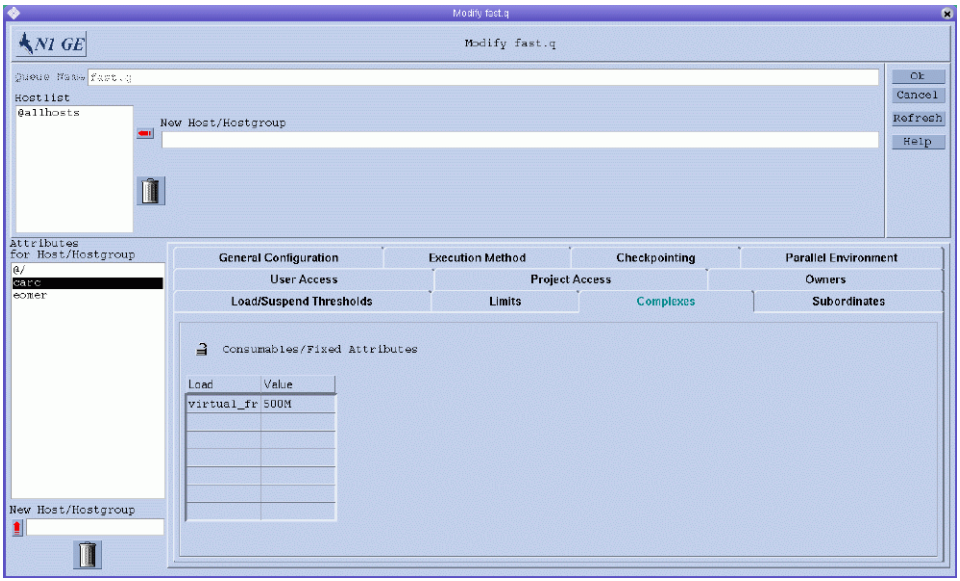
考えられる virtual_free リソース定義の例は、図 3–2 にあります。1G バイトのメインメモリーを搭載したホストの、対応する実行ホストの構成は、図 3–3 にあります。

virtual_free リソース定義の例では、グローバル構成の例と同じように、Requestable フラグは、FORCED ではなく YES に設定されています。これは、ユーザーはジョブのメモリー要件を指定する必要がないことを意味します。明示的なメモリーの要求がない場合に、「Default」フィールドの値が使用されます。この場合のデフォルト要求としての 1G バイトの値は、要求を持たないジョブがすべての使用可能な物理メモリーを占有すると想定されていることを意味します。

注 – virtual_free は、Grid Engine システムの標準的な負荷パラメータの 1 つです。仮想記憶の容量計画では、最近のメモリー統計の追加の可用性が、システムにより自動的に考慮されます。空き仮想記憶の負荷レポートが、Grid Engine ソフトウェアの内部のブックキーピングにより取得される値を下回る場合、メモリーの過剰予約を防ぐために負荷値が使用されます。Grid Engine システムを使用せずにジョブが起動された場合は、報告される負荷値と内部のブックキーピングの違いは容易に生じます。

1 つのマシンでさまざまなメモリー要件のさまざまなジョブクラスを実行する場合は、これらのジョブクラスが使用するメモリーを分割したい場合があります。この機能は、容量の共有と呼ばれます。各ジョブクラスに対してキューを構成することで、この機能を実現できます。続いて、そのホスト上の合計メモリーの一部を各キューに割り当てます。

例では、キュー構成が、ホスト carc に使用可能な合計メモリーの半分を、ホスト carc の fast.q キューに割り当てます。そのため、ホスト carc 上のキュー fast.q で実行中のすべてのジョブの累積メモリー消費量は、500M バイトを超えることができません。そのほかのキューのジョブは考慮されません。それにもかかわらず、ホスト carc 上で実行中のすべてのジョブの合計メモリー消費量は、1G バイトを超えることができません。



注 – 属性 virtual_free は、コンプレックスからの継承を介して、すべてのキューに使用可能です。

ユーザーは、次のいずれかの形式の例と同じように構成されているシステムに対して、ジョブを発行する場合があります。

```
% qsub -l vf=100M honest.sh
% qsub dont_care.sh
```

最初のコマンドにより発行されるジョブは、少なくとも 100M バイトのメモリーが使用可能になるとすぐに開始できます。この量は、virtual_free の消費可能なリソースの容量計画において考慮されます。2 番目のジョブは暗黙にすべての使用可能なメ

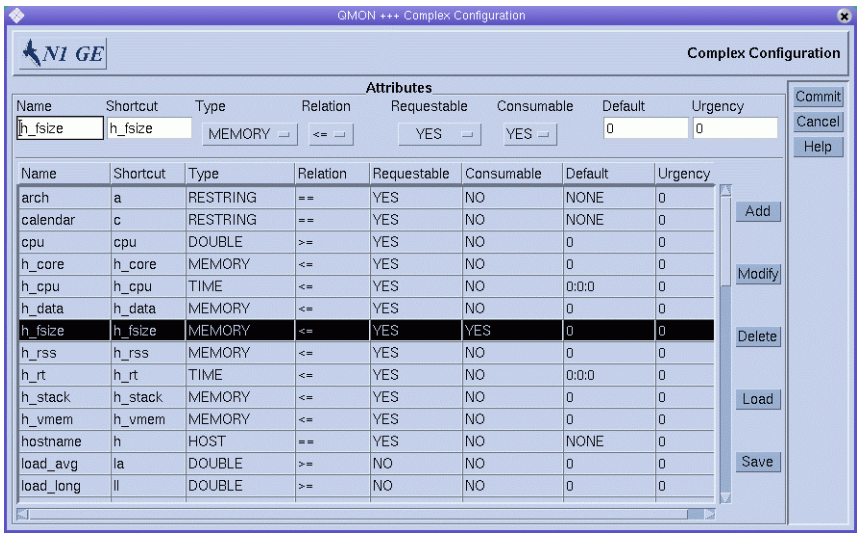
モリーを要求するため、2 番目のジョブが実行されるのは、システム上にそのほかのジョブが存在しない場合のみです。また、2 番目のジョブはキューのメモリー容量を超えるため、このジョブはキュー `fast.q` では実行できません。

例 3: 使用可能なディスク容量の管理

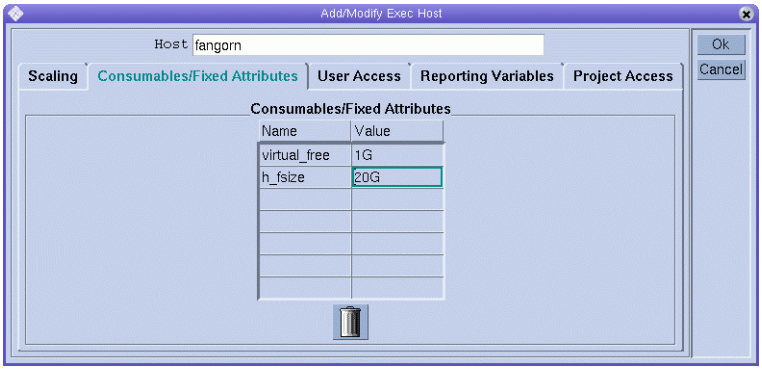
一部のアプリケーションでは、ファイルに格納されている大量のデータセットを操作する必要があります。このようなアプリケーションは、実行時間全体を通じて、十分なディスク容量を利用できる必要があります。この要件は、前の例で説明した使用可能なメモリーの容量の共有に似ています。主な違いは、Grid Engine システムでは、標準的な負荷パラメータの 1 つとして空きディスク容量が用意されていない点です。ディスクは通常、サイト固有の方法でファイルシステムに分割されているため、空きディスク容量は標準的な負荷パラメータではありません。サイト固有のパーティション分割により、対象のファイルシステムの識別を自動で行うことができません。

それにもかかわらず、使用可能なディスク容量は、消費可能リソースの機能を通じて、システムによって効率的に管理することができます。この目的のためには、ホストリソース属性 `h_fsize` を使用する必要があります。

まず、次の図に示すように、属性を消費可能リソースとして構成する必要があります。



ローカルホストファイルシステムの場合、次の図に示すように、ディスク容量のコンシューマブルに関するリーズナブルな容量の定義を、ホスト構成に配置することができます。



このように構成された Grid Engine システムに対するジョブの発行は、以前の例と同じように機能します。

```
% qsub -l hf=5G big-sort.sh
```

ここで `h_fsize` 属性が推奨されているのは、`h_fsize` は、キュー構成において固定ファイルサイズ制限値としても使用されているためです。ファイルサイズの制限値は、ジョブ発行時に指定されるよりも大きなサイズのファイルを作成するジョブの権限を制限します。この例の `qsub` コマンドは、5G バイトのファイルサイズの制限値を指定しています。ジョブが属性を要求しない場合、キュー構成またはホスト構成からの対応する値が使用されます。例で `h_fsize` の Requestable フラグが `FORCED` に設定されている場合、`qsub` コマンドには要求が含まれている必要があります。Requestable フラグが設定されていない場合は、`qsub` コマンドでは要求はオプションになります。

消費可能リソースとしてキュー制限値を使用することで、ジョブスクリプトによる実際のリソース消費量の代わりに、ユーザーが指定する要求を制御します。制限値の違反は制裁されるため、最終的にはジョブが異常終了します。キュー制限値により、Grid Engine システム内部の容量計画のベースとなる、リソース要求の信頼性が確保されます。詳細については、`queue_conf`(5) および `setrlimit`(2) のマニュアルページを参照してください。

注 – 一部のオペレーティングシステムでは、プロセスごとのファイルサイズの制限値のみ使用できます。この場合は、ジョブは制限値までのサイズの複数のファイルを作成する場合があります。ジョブごとのファイルサイズの制限値をサポートしているシステムでは、Grid Engine システムは `h_fsize` 属性とともにこの機能を使用します。詳細については、`queue_conf`(5) のマニュアルページを参照してください。

Grid Engine システムに発行されないアプリケーションに、同時にディスク容量を占有させたい場合があります。このような場合、ディスク容量の不足により、内部のブックキーピングではアプリケーション障害を防ぐのに十分ではない場合がありま

す。この問題を回避するため、ディスク容量の使用量に関する統計を定期的に受信することができます。そうした統計によって、Grid Engine システムの外部で発生しているものも含めて、全体のディスク領域の消費量が分かります。

負荷センサーインタフェースを使用すると、ファイルシステム上で使用可能なディスク容量など、サイト固有の情報を持つ、標準的な負荷パラメータのセットを強化することができます。詳細については、91 ページの「サイト固有の負荷パラメータの追加」を参照してください。

適切な負荷センサーを追加し、 `h_fsize` に関して空きディスク容量を報告することで、消費可能リソースの管理とリソースの可用性の統計を組み合わせることができます。Grid Engine システムは、ディスク容量のジョブ要件と、使用可能な容量、および報告された最新の負荷値を比較します。使用可能な容量は、内部のリソース計画から得られます。ジョブがホストに振り分けられるのは、両方の基準が満たされた場合のみです。

コマンド行からのコンプレックスリソース属性の構成

コマンド行からコンプレックスを構成するには、適切なオプションを使用して次のコマンドを入力します。

```
% qconf オプション
```

`qconf` コマンドの書式と有効な構文の詳細な定義については、`qconf(1)` のマニュアルページを参照してください。

次のオプションを使用すると、Grid Engine システムのコンプレックスを変更することができます。

- `-mc` – `-mc` オプションを使用すると、テンプレートコンプレックス構成、または変更用の既存のコンプレックス構成が入力されているエディタが開きます。
- `-Mc` – `qconf -Mc` オプションを使用すると、コンプレックス構成ファイルを引数として扱います。

次のコマンドを使用すると、`complex(5)` のマニュアルページで定義されているファイル形式で、現在のコンプレックス構成が標準出力ストリームに出力されます。

```
% qconf -sc
```

次の例に、サンプルの出力を示します。

例 3–1　qconf -sc の出力例

```
#name      shortcut  type  relop  requestable   consumable  default  urgency
#-------------------------------------------------------------------------
nastran    na        INT   <=     YES           NO          0        0
pam-crash  pc        INT   <=     YES           YES         1        0
permas     pm        INT   <=     FORCED        YES         1        0
#---- # start a comment but comments are not saved across edits -----------
```

負荷パラメータ

この節では、Grid Engine システムの負荷パラメータについて説明します。また、独自の負荷センサーの記述法も解説します。

デフォルトの負荷パラメータ

デフォルトでは、sge_execd は sge_qmaster に対して、定期的にいくつかの負荷パラメータとそれに対応する値を報告します。これらの値は、20 ページの「ホストとデーモンについて」で説明されている、sge_qmaster 内部ホストオブジェクトに格納されます。ただし、値が内部で使用されるのは、対応する名前を持つコンプレックスリソース属性が定義されている場合のみです。このようなコンプレックスリソース属性には、負荷値をどのように解釈するかに関する定義が含まれています。詳細については、74 ページの「キュー、ホスト、およびグローバルクラスタへのリソース属性の割り当て」を参照してください。

最初のインストール後、負荷パラメータの標準的なセットが報告されます。標準的な負荷パラメータに必要なすべての属性は、ホスト関連の属性として定義されます。N1 Grid Engine 6.1 ソフトウェアの今後のリリースでは、デフォルトの負荷パラメータの拡張セットが提供される可能性があるため、デフォルトで報告される負荷パラメータのセットは、ファイル *sge-root*/doc/load_parameters.asc に記載されています。

負荷属性の定義方法は、それらのアクセス可能性を決定します。負荷パラメータをグローバルリソース属性として定義すると、それらは、クラスタ全体とすべてのホストに対して使用可能になります。負荷パラメータをホスト関連の属性として定義すると、あらゆるホストにその属性が提供されますが、グローバルクラスタには提供されません。

注 – 負荷属性は、キュー属性としては定義しないでください。キュー属性は、ホストに対してもクラスタに対しても使用可能にはなりません。

サイト固有の負荷パラメータの追加

デフォルトの負荷パラメータのセットは、クラスタにおける負荷の状況を完全に記述するのには不適切である場合があります。特に、サイト固有のポリシー、アプリケーション、および構成に関してはこの可能性が高くなります。したがって、Grid Engine ソフトウェアには負荷パラメータのセットを拡張する手段が用意されています。この目的のため、sge_execd は、負荷パラメータと現在の負荷値を sge_execd に供給するインタフェースを提供します。そのあと、これらのパラメータはデフォルトの負荷パラメータと同じように扱われます。デフォルトの負荷パラメータに関しては、サイト固有の負荷パラメータを有効にするには、対応する属性をコンプレックスで定義する必要があります。詳細については、90 ページの「デフォルトの負荷パラメータ」を参照してください。

独自の負荷センサーの記述

sge_execd に追加の付加情報を供給するには、負荷センサーを用意する必要があります。負荷センサーには、スクリプトとバイナリ実行ファイルを使用できます。どちらの場合も、負荷センサーの標準入出力ストリームの処理および制御フローは次の規則に従っている必要があります。

- 負荷センサーは、あるポイントで STDIN からの入力を待機する無限ループとして記述する必要があります。
- STDIN から文字列 quit が読み取られた場合、負荷センサーは終了することが想定されています。
- STDIN から行末が読み取られるとすぐに、データを読み込むための取得サイクルが開始することが想定されています。

続いて負荷センサーは、目的の負荷値を計算するために必要なすべての処理を実行します。サイクルの終了時点で、負荷センサーは結果を STDOUT に書き込みます。

注 – 負荷の取得に長時間を要する場合、負荷レポートを送信した直後に負荷測定プロセスを開始できます。quit が受信されると、負荷値は送信できるようになります。

負荷センサーの規則の書式

負荷センサーの規則の書式は次のとおりです。

- 負荷値のレポートは、単語 begin のみが含まれる行で始まります。
- 各負荷値は復帰改行で区切られます。
- 各負荷値は、コロン (:) によって区切られた 3 つの部分から構成され、空白を含みません。
- 負荷値の最初の部分は、負荷の報告先のホストの名前か、特別な名前 global のいずれか一方です。

- 負荷センサーの 2 番目の部分は、コンプレックスで定義されている、負荷値の記号名です。詳細については、complex(5) のマニュアルページを参照してください。コンプレックスにエントリのない負荷値が報告された場合、報告された負荷値は使用されません。
- 負荷センサーの 3 番目の部分は、測定された負荷値です。負荷値レポートは、単語 end が含まれる行で終わります。

負荷センサースクリプトの例

次の例に負荷センサーを示します。負荷センサーは Bourne シェルスクリプトです。

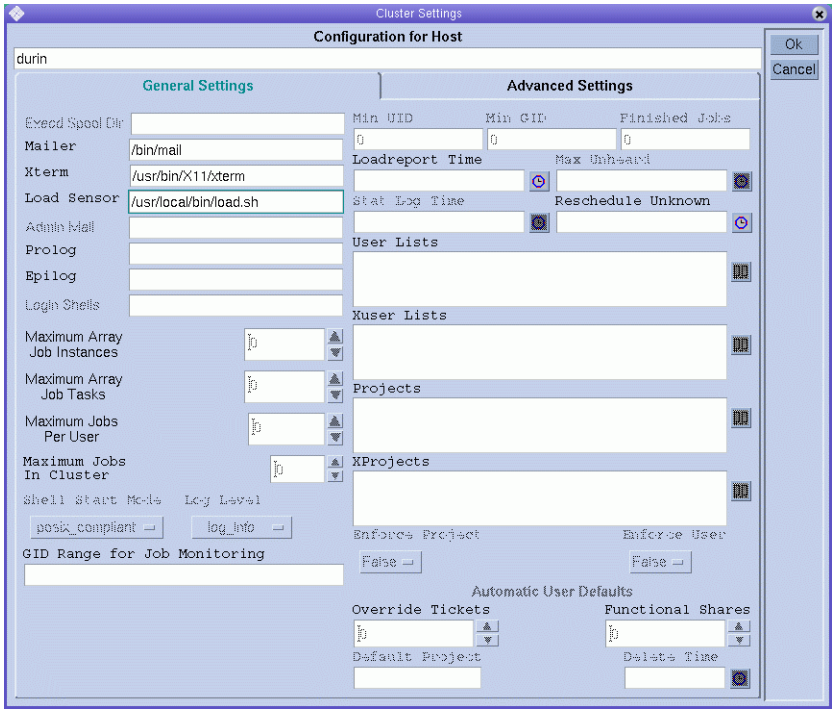
例 3–2　負荷センサー – Bourne シェルスクリプト

```
#!/bin/sh

myhost=‘uname -n‘

while [ 1 ]; do
     # wait for input
     read input
     result=$?
     if [ $result != 0 ]; then
          exit 1
     fi
     if [ $input = quit ]; then
          exit 0
     fi
     #send users logged in
     logins=‘who | cut -f1 -d" " | sort | uniq | wc -l‘ | sed "s/^ *//"
     echo begin
     echo "$myhost:logins:$logins"
     echo end
done

# we never get here

exit 0
```

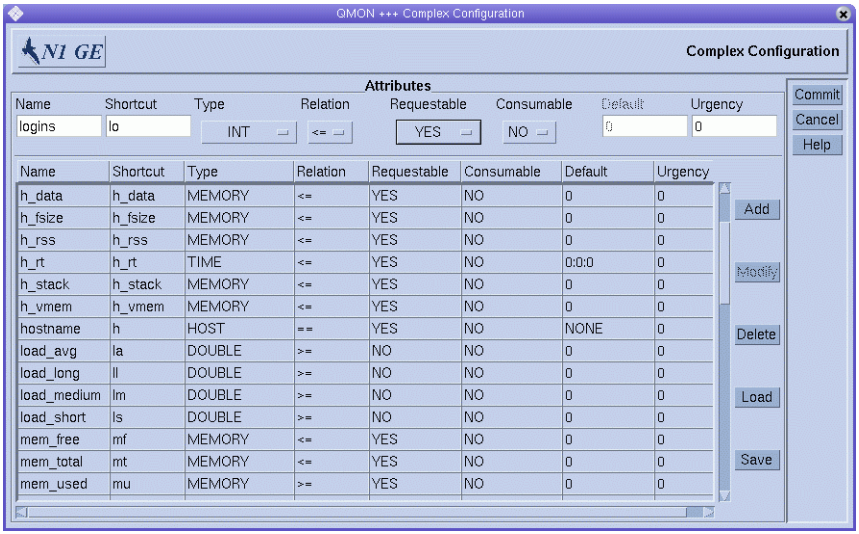
このスクリプトを、ファイル load.sh に保存します。chmod コマンドを使用して、このファイルに実行可能権限を割り当てます。コマンド行からスクリプトを対話形式でテストするには、load.sh と入力し、Return キーを繰り返し押します。

手続きが機能するようになるとすぐ、任意の実行ホストにインストールできます。手続きをインストールするには、クラスタ構成、グローバル構成、またはホスト固有の構成の load_sensor パラメータとして、負荷センサーのパスを構成します。詳細については、42 ページの「基本クラスタ構成」または sge_conf(5) のマニュアルページを参照してください。

対応する QMON ウィンドウは次の図のようになります。



報告される負荷パラメータ logins は、対応する属性がコンプレックスに追加されるとすぐに使用できるようになります。必要な定義は、次の図に示す最後のテーブルエントリのようになります。


QMON +++ Complex Configuration
N1 GE
Complex Configuration
Attributes
Name
Shortcut
Type
Relation
Requestable
Consumable
Default
Urgency
logins
lo
INT
<=
YES
NO
0
Commit
Cancel
Help
Add
Modify
Delete
Load
Save
Name Shortcut Type Relation Requestable Consumable Default Urgency
h_data h_data MEMORY <= YES NO 0 0
h_fsize h_fsize MEMORY <= YES NO 0 0
h_rss h_rss MEMORY <= YES NO 0 0
h_rt h_rt TIME <= YES NO 0:0:0 0
h_stack h_stack MEMORY <= YES NO 0 0
h_vmem h_vmem MEMORY <= YES NO 0 0
hostname h HOST == YES NO NONE 0
load_avg la DOUBLE >= NO NO 0 0
load_long ll DOUBLE >= NO NO 0 0
load_medium lm DOUBLE >= NO NO 0 0
load_short ls DOUBLE >= NO NO 0 0
mem_free mf MEMORY <= YES NO 0 0
mem_total mt MEMORY <= YES NO 0 0
mem_used mu MEMORY >= YES NO 0 0

◆ ◆ ◆ 第 4 章

ユーザーアクセスの管理

この章では、ユーザーアカウント、およびそのほかの関連するアカウントの管理について説明します。この章の内容は次のとおりです。

- ユーザーアクセス
- プロジェクトおよびプロジェクトアクセス
- パスの別名設定
- デフォルトの要求

内容説明に加えて、この章では次のタスクの実行方法を詳細に解説します。

ユーザーの設定

Grid Engine システムのユーザーを設定するには、次のタスクを実行する必要があります。

- 必要なログインの割り当て。

  ホスト A から、ホスト B で実行するようジョブを発行するには、ユーザーは両方のホストで同じアカウントを持っている必要があります。このアカウントには、同じユーザー名が必要です。`sge_qmaster` が動作しているマシンではログインは必要はありません。

- アクセス権の設定。

  Grid Engine ソフトウェアには、クラスタ全体やキュー、並列環境に対するユーザーアクセスを制限する機能が用意されています。詳細については、103 ページの「ユーザーの構成」を参照してください。

  また、特定のキューを一時停止または使用可能にする権限をユーザーに与えることができます。詳細については、63 ページの「所有者パラメータの構成」を参照してください。

- **Grid Engine** システムユーザーの宣言。

  共有ツリーにユーザーを追加したり、ユーザーに対して機能ポリシーや優先ポリシーを定義するには、これらのユーザーを Grid Engine システムに対して宣言する必要があります。詳細については、132 ページの「QMON を使用したポリシーに基づくリソース管理の構成」および104 ページの「QMON を使用したユーザーオブジェクトの構成」を参照してください。

- プロジェクトアクセスの設定。

  共有ベース、機能、または優先ポリシーの定義に対してプロジェクトを使用する場合、1 つ以上のプロジェクトに対してユーザーアクセスを与える必要があります。このように設定しないと、ユーザーのジョブは可能なもっとも低い優先順位クラスで終了し、ジョブがアクセスできるリソースは極めて少なくなります。詳細については、132 ページの「QMON を使用したポリシーに基づくリソース管理の構成」を参照してください。

- ファイルアクセス制限の設定。

  Grid Engine システムのユーザーは、*sge-root*/ *cell*/`common` ディレクトリに対する読み取りアクセス権を持っている必要があります。

  実行デーモンは、ジョブを開始する前にそのジョブ用に一時作業ディレクトリを作成し、そのディレクトリの所有権をジョブの所有者に移します。実行デーモンは root で動作します。この一時ディレクトリは、ジョブが完了するとただちに削除されます。この一時作業ディレクトリは、キュー構成パラメータ `tmpdir` で定義されたパスの下に作成されます。詳細については、`queue_conf`(5) の マニュアルページを参照してください。

tmpdir の場所の下に一時ディレクトリを作成できることを確認します。ディレクトリは、Grid Engine システムのユーザー所有権に設定する必要があります。ユーザーは、その一時ディレクトリに書き込みを行える必要があります。

- サイト依存の設定。

  定義上、バッチジョブには端末接続はありません。このため、コマンドインタプリタの起動リソースファイル (csh に対する .cshrc など) に stty などの UNIX コマンドが含まれていると、エラーになることがあります。起動ファイルに stty がないか調べ、『Sun N1 Grid Engine 6.1 インストールガイド』の第 6 章「インストールの検証」の説明に従って、そのようなコマンドを使用しないようにしてください。

  通常、バッチジョブはオフラインで実行されるため、エラーイベントなどをジョブの所有者に通知する方法は 2 つしかありません。1 つはファイルにエラーメッセージを記録する方法、もう 1 つは電子メールを送信する方法です。

  まれな状況ですが、たとえばエラーログファイルを開けない場合は、電子メールがユーザーに直接通知する唯一の手段になります。この場合でも、エラーメッセージは Grid Engine システムのログファイルに記録されますが、通常、ユーザーはシステムログファイルの内容を見ません。このため、Grid Engine ユーザーのために電子メールシステムを正しくインストールしておくことを推奨します。

- **Grid Engine** システム定義ファイルの設定。 Grid Engine ユーザー用に次の定義ファイルを設定できます。
  - qmon – Grid Engine システム GUI 用のリソースファイル。『Sun N1 Grid Engine 6.1 ユーザーズガイド』の「QMON のカスタマイズ」を参照してください。
  - sge_aliases – 現在の作業ディレクトリのパスの別名。109 ページの「パスの別名設定の使用」を参照してください。
  - sge_request – デフォルトの要求定義ファイル。112 ページの「デフォルト要求の構成」を参照してください。

ユーザーアクセスの構成

Grid Engine システムには、次の 4 つのユーザーのカテゴリがあります。

- 管理者。管理者は、Grid Engine システムの操作に関する全権を持ちます。デフォルトでは、マスターホストと、キューのホストとなるマシンのスーパーユーザーが、管理者特権を持ちます。
- オペレータ。 オペレータは、キューの追加、削除、変更ができないことを除けば、管理者が実行するのと同じコマンドの多くを実行できます。
- 所有者。キュー所有者は、所有するキューの一時停止/再開、あるいは使用不可/使用可能操作だけを行うことができます。qidle を正しく使用するには、これらの特権が必要です。一般にユーザーは、使用しているデスクトップワークステーション上のキューインスタンスの所有者として宣言されます。

- ユーザー。103 ページの「ユーザーの構成」で説明されているように、ユーザーにはある種のアクセス権がありますが、ユーザーはクラスタまたはキューを管理できません。

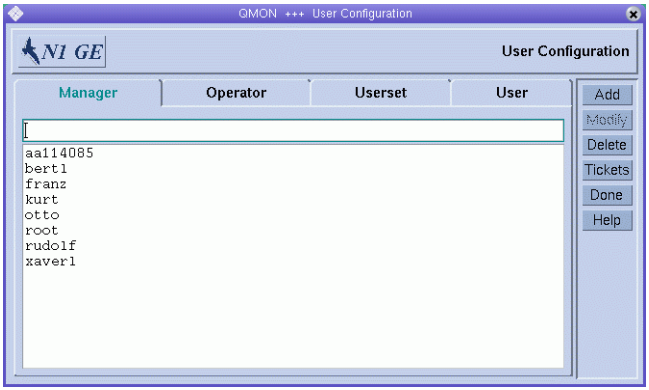
次では、各カテゴリをより詳細に説明します。

管理者アカウントの構成

QMON またはコマンド行から管理者アカウントを構成できます。

QMON を使用した管理者アカウントの構成

「QMON Main Control」ウィンドウで「User Configuration」ボタンをクリックします。「Manager」タブが表示されます。このタブで、どのアカウントが管理コマンドを実行できるかを宣言できます。



このタブには、管理権限を持っていることがすでに宣言されているすべてのアカウントが表示されます。

新しい管理者アカウントを追加するには、管理者アカウントのリストの上にあるフィールドに名前を入力してから、「Add」をクリックするか、Return キーを押します。

管理者アカウントを削除するには、管理者アカウントを選択してから「Delete」をクリックします。

コマンド行からの管理者アカウントの構成

コマンド行から管理者アカウントを構成するには、適切なオプションを使用して、次のコマンドを入力します。

```
# qconf オプション
```

次のオプションを使用できます。

- `qconf -am` *user-name* [,...]

  `-am` オプション (管理者の追加) を使用すると、Grid Engine システムの管理者のリストに 1 人以上のユーザーが追加されます。デフォルトでは、信頼されたすべてのホストの root アカウントは、Grid Engine システムの管理者になります。詳細については、20 ページの「ホストとデーモンについて」を参照してください。

- `qconf -dm` *user-name* [,...]

  `-dm` オプション (管理者の削除) を使用すると、Grid Engine システムの管理者のリストから、指定したユーザーが削除されます。

- `qconf -sm`

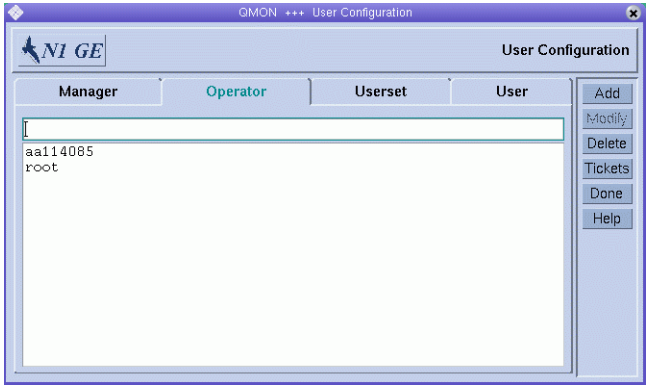
  `-sm` オプション (管理者の表示) を使用すると、Grid Engine システムのすべての管理者のリストが表示されます。

オペレータアカウントの構成

QMON またはコマンド行からオペレータアカウントを構成できます。

QMON を使用したオペレータアカウントの構成

「QMON Main Control」ウィンドウで「User Configuration」ボタンをクリックしてから「Operator」タブをクリックします。



「Operator」タブでは、アカウントが管理者アカウントとしても宣言されている場合を除き、どのアカウントに制限された管理権限を付与するかを宣言できます。98 ページの「QMON を使用した管理者アカウントの構成」を参照してください。

このタブには、オペレータ権限を持っていることがすでに宣言されているすべてのアカウントが表示されます。

新しいオペレータアカウントを追加するには、オペレータアカウントのリストの上にあるフィールドに名前を入力してから、「Add」をクリックするか、Return キーを押します。

オペレータアカウントを削除するには、オペレータアカウントを選択してから「Delete」をクリックします。

コマンド行からのオペレータアカウントの構成

コマンド行からオペレータアカウントを構成するには、適切なオプションを使用して、次のコマンドを入力します。

```
# qconf オプション
```

次のオプションを使用できます。

- `qconf -ao` *user-name*`[ ,...]`

  `-ao` オプション (オペレータの追加) を使用すると、Grid Engine システムのオペレータのリストに 1 人以上のユーザーが追加されます。

- `qconf -do` *user-name*`[,...]`

  `-do` オプション (オペレータの削除) を使用すると、Grid Engine システムのオペレータのリストから、指定したユーザーが削除されます。

- `qconf -so`

  `-so` オプション (オペレータの表示) を使用すると、Grid Engine システムのすべてのオペレーターのリストが表示されます。

ユーザーアクセスリストの構成

少なくとも 1 つの発行ホストおよび実行ホストで有効なログイン ID を持つユーザーは、Grid Engine システムを使用できます。ただし、Grid Engine システムの管理者は、一部のキューまたはすべてのキューに対する一部のユーザーのアクセスを禁止できます。また、管理者は特定の並列環境などの機能の使用を制限できます。詳細については、175 ページの「並列環境の構成」を参照してください。

アクセス権を定義するには、ユーザーアクセスリストを定義する必要があります。ユーザーアクセスリストは、名前付きのユーザーのセットで構成されます。ユーザーアクセスリストの定義には、ユーザー名と UNIX グループ名を使用します。ユーザーアクセスリストは、次の任意の構成で、特定のリソースへのアクセスの拒否や許可の指定に使用されます。

- クラスタ構成 – 42 ページの「基本クラスタ構成」を参照
- キュー構成 – 60 ページの「従属キューの構成」を参照
- 並列環境インタフェースの構成 – 176 ページの「QMON を使用した並列環境の構成」を参照。

QMON を使用したユーザーアクセスリストの構成

「QMON Main Control」ウィンドウで「User Configuration」ボタンをクリックしてから「Userset」タブをクリックします。「Userset」タブが表示されます。

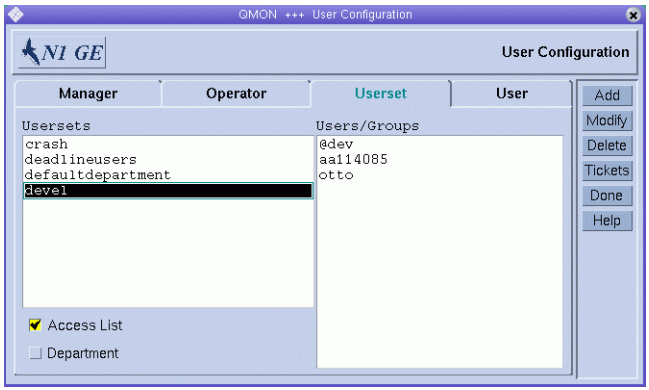


図 4–1 「Userset」タブ

Grid Engine システムでは、ユーザーセットにはアクセスリストと部署の一方またはその両方を使用できます。「Usersets」リストの下の 2 つのチェックボックスには、選択したユーザーセットの種類が表示されます。このセクションにアクセスリストの詳細が表示されます。部署については、103 ページの「プロジェクトおよび部署としてのユーザーセットの定義」で説明されています。

「Usersets」リストには、使用可能なすべてのアクセスリストが表示されます。アクセスリストの内容を表示するには、アクセスリストを選択します。内容は「Users/Groups」リストに表示されます。

注 – グループ名の先頭には @ 記号が付いています。

新しいユーザーセットを追加するには「Add」をクリックします。

既存のユーザーセットを変更するには、ユーザーセットを選択してから「Modify」をクリックします。

ユーザーセットを削除するには、ユーザーセットを選択してから「Delete」をクリックします。

「Add」または「Modify」をクリックすると、「Access List Definition」ダイアログボックスが表示されます。

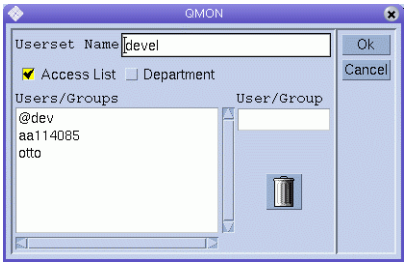


図 4–2 「Access List Definition」ダイアログボックス

新しいアクセスリストの定義を追加するには、「Userset Name」フィールドにアクセスリストの名前を入力します。既存のアクセスリストを変更している場合は、その名前が「Userset Name」フィールドに表示されます。

アクセスリストに新しいユーザーまたはグループを追加するには、「User/Group」フィールドにユーザー名またはグループ名を入力します。必ず、グループ名の先頭には @ 記号を付けてください。

「Users/Groups」リストには、現在定義されているすべてのユーザーとグループが表示されます。

「Users/Groups」リストからユーザーまたはグループを削除するには、ユーザーまたはグループを選択してから、ゴミ箱のアイコンをクリックします。

変更を保存して、ダイアログボックスを閉じるには、「OK」をクリックします。変更を保存せずにダイアログボックスを閉じるには、「Cancel」をクリックします。

コマンド行からのユーザーアクセスリストの構成

コマンド行からユーザーアクセスリストを構成するには、適切なオプションを使用して、次のコマンドを入力します。

```
# qconf オプション
```

次のオプションを使用できます。

- `qconf -au` *user-name*`[ ,...]`*access-list-name*`[,...]`

  `-au` オプション (ユーザーの追加) を使用すると、指定したアクセスリストに 1 人以上のユーザーが追加されます。

- `qconf -Au` *filename*

  `-Au` オプション (ファイルからのユーザーアクセスリストの追加) を使用すると、構成ファイル *filename* を使用してアクセスリストを追加します。

- `qconf -du` *user-name*`[ ,...]` *access-list-name* `[,... ]`

  `-du` オプション (ユーザーの削除) を使用すると、指定したアクセスリストから、1 人以上のユーザーが削除されます。

- `qconf -dul` *access-list-name* `[,...]`

  `-dul` オプション (ユーザーリストの削除) を使用すると、ユーザーセットのリストが完全に削除されます。

- `qconf -mu` *access-list-name*

  `-mu` オプション (ユーザーアクセスリストの変更) を使用すると、指定したアクセスリストが変更されます。

- `qconf -Mu` *filename*

  `-Mu` オプション (ファイルからのユーザーアクセスリストの変更) を使用すると、構成ファイル *filename* を使用して、指定したアクセスリストを変更します。

- `qconf -su` *access-list-name* `[,...]`

  `-su` オプション (ユーザーアクセスリストの表示) を使用すると、指定したアクセスリストが表示されます。

- `qconf -sul`

  `-sul` オプション (ユーザーアクセスリストの表示) を使用すると、現在定義されているすべてのアクセスリストが表示されます。

プロジェクトおよび部署としてのユーザーセットの定義

ユーザーセットは、Grid Engine システムのプロジェクトと部署の定義にも使用できます。詳細については、106 ページの「プロジェクトの定義」を参照してください。

部署は、機能ポリシーと優先ポリシーの構成に使用します。ユーザーは 1 つの部署のみのメンバーになることができるのに対し、1 人のユーザーを複数のアクセスリストに含めることができるという点において、部署はアクセスリストと異なります。詳細については、154 ページの「機能ポリシーの構成」および 159 ページの「優先ポリシーの構成」を参照してください。

ユーザーセットが部署であることは、図 4–1 と図 4–2 に示す「Department」フラグで示されます。ユーザーセットは、同時に部署とアクセスリストの両方として定義できます。ただし、複数の部署にユーザーを登録できないという制限が適用されます。

ユーザーの構成

ユーザーに共有ベース、機能、または優先ポリシーを定義する前には、ユーザー名を宣言する必要があります。132 ページの「QMON を使用したポリシーに基づくリソース管理の構成」を参照してください。

ポリシーを定義する前に明示的にユーザー名を宣言したくない場合、Grid Engine システムでは、定義済みのデフォルト値に基づいて、自動的にユーザーを作成できます。自動的にユーザーを作成することで、多くのユーザーを抱えるサイトの管理上の負担を大幅に軽減できます。

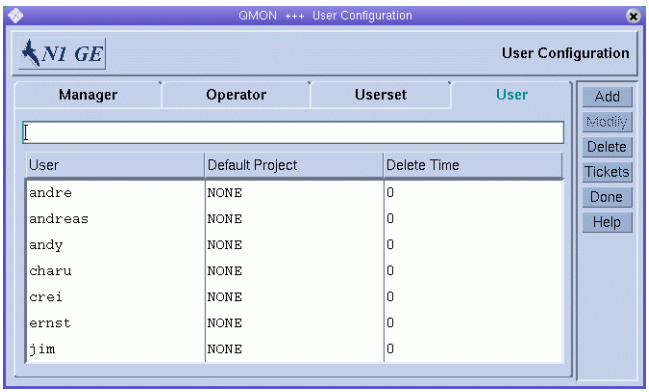
システムにユーザーを自動的に作成させるには、「Cluster Settings」ダイアログボックスの「Enforce User」パラメータを「Auto」に設定します。自動的に作成されるユーザーのデフォルト値を設定するには、「Cluster Settings」ダイアログボックスで、次の「Automatic User Defaults」の値を指定します。

- 「Override Tickets」
- 「Functional Shares」
- 「Default Project」
- 「Delete Time」

クラスタ構成の詳細については、42 ページの「基本クラスタ構成」を参照してください。

QMON を使用したユーザーオブジェクトの構成

「QMON Main Control」ウィンドウで「User Configuration」ボタンをクリックしてから「User」タブをクリックします。「User」タブは次の図のようになっています。



新しいユーザーを追加するには、「User」リスト上のフィールドにユーザー名を入力してから、「Add」をクリックするか Return キーを押します。

ユーザーを削除するには、「User」リストでユーザー名を選択してから「Delete」をクリックします。

「Delete Time」カラムは読み取り専用です。このカラムには、自動的に作成されたユーザーが Grid Engine システムから削除される時間が表示されます。ゼロは、そのユーザーが削除されないことを示します。

各ユーザーにはデフォルトのプロジェクトを割り当てることができます。ユーザーが、アクセスできる別のプロジェクトを要求する場合を除き、ユーザーが発行する各ジョブにはデフォルトのプロジェクトが関連付けられます。詳細については、106 ページの「プロジェクトの定義」を参照してください。

デフォルトのプロジェクトを割り当てるには、ユーザーを選択してから、「Default Project」カラムヘッダーをクリックします。「Project Selection」ダイアログボックスが表示されます。

選択状態のユーザーエントリのプロジェクトを選択します。

デフォルトのプロジェクトを割り当て、ダイアログボックスを閉じるには、「OK」をクリックします。デフォルトのプロジェクトを割り当てることなくダイアログボックスを閉じるには、「Cancel」をクリックします。

コマンド行からのユーザーオブジェクトの構成

コマンド行からユーザーオブジェクトを構成するには、適切なオプションを使用して、次のコマンドを入力します。

```
# qconf オプション
```

次のオプションを使用できます。

- `qconf -auser`

  `-auser` オプション (ユーザーの追加) を使用すると、エディタでテンプレートユーザー構成が開かれます。`user(5)` のマニュアルページを参照してください。このエディタは、デフォルトの `vi` エディタか、`EDITOR` 環境変数により指定されたエディタのいずれかです。 変更を保存してエディタを終了すると、変更は `sge_qmaster` に登録されます。

- `qconf -Auser` *filename*

  `-Auser` オプション (ファイルからのユーザーの追加) を使用すると、指定したファイルが解析され、ユーザー構成が追加されます。

  このファイルは、ユーザー構成テンプレートの形式を有する必要があります。

- `qconf -duser` *user-name*[,...]

  `-duser` オプション (ユーザーの削除) を使用すると、1 つ以上のユーザーオブジェクトが削除されます。

- `qconf -muser` *user-name*

-muser オプション (ユーザーの変更) を使用すると、既存のユーザーエントリを変更できます。このオプションにより、ユーザー構成がエディタに読み込まれます。このエディタは、デフォルトの vi エディタか、EDITOR 環境変数により指定されたエディタのいずれかです。 変更を保存してエディタを終了すると、変更は sge_qmaster に登録されます。

- qconf -Muser *filename*

  -Muser オプション (ファイルからのユーザーの変更) を使用すると、指定したファイルが解析され、ユーザー構成が変更されます。

  このファイルは、ユーザー構成テンプレートの形式を有する必要があります。

- qconf -suser *user-name*

  -suser オプション (ユーザーの表示) を使用すると、指定したユーザーの構成が表示されます。

- qconf -suserl

  -suserl オプション (ユーザーリストの表示) を使用すると、現在定義されているすべてのユーザーのリストが表示されます。

プロジェクトの定義

プロジェクトは、複数のユーザーの共同計算タスクをまとめる手段となります。またプロジェクトは、そのようなプロジェクトに属するすべてのジョブに対してリソース使用ポリシーを定義します。

プロジェクトは、次の 3 つのスケジューリングポリシーの領域で使用されます。

- 共有ベース – プロジェクトに配分が割り当てられます (141 ページの「共有ベースポリシーの構成」 を参照)
- 機能 – プロジェクトは一定の割合の機能チケットを受け取ります (154 ページの「機能ポリシーの構成」 を参照)
- 優先 – 管理者によってプロジェクトに優先チケットが付与されます (159 ページの「優先ポリシーの構成」 を参照)

これらの 3 つのポリシーでプロジェクトを使用するには、前もってプロジェクトを宣言する必要があります。

Grid Engine システムの管理者は、プロジェクトに名前を付け、いくつかの属性を設定することによってプロジェクトを定義します。Grid Engine のユーザーは、ジョブの発行時に、ジョブをプロジェクトに関連付けることができます。共有ベース、機能、優先チケットのプロジェクトの配分に応じて、プロジェクトへのジョブの関連付けは、ジョブの振り分けに影響します。

QMON を使用したプロジェクトの定義

Grid Engine システムの管理者は、「Project Configuration」ダイアログボックスを使用して、プロジェクトを定義し、定義を更新することができます。

プロジェクトを定義するには、「QMON Main Control」ウィンドウで「Project Configuration」ボタンをクリックします。「Project Configuration」ダイアログボックスが表示されます。

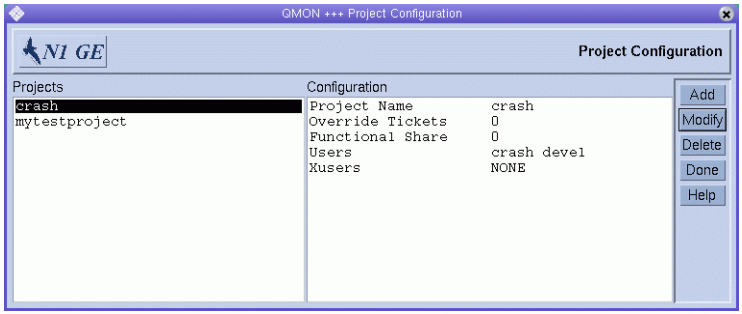


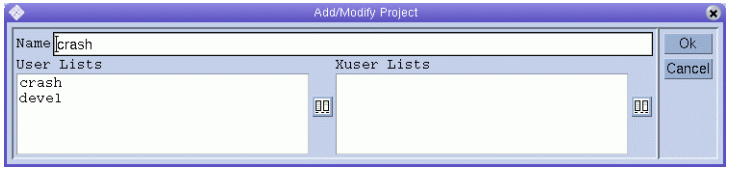
図 4–3 「Project Configuration」ダイアログボックス

現在定義されているプロジェクトは、「Projects」リストに表示されます。

選択したプロジェクトのプロジェクト定義が「Configuration」の下に表示されます。

プロジェクトをただちに削除するには、プロジェクトを選択してから「Delete」をクリックします。

新しいプロジェクトを追加するには「Add」をクリックします。プロジェクトを変更するには、プロジェクトを選択してから「Modify」をクリックします。「Add」または「Modify」をクリックすると、「Add/Modify Project」ダイアログボックスが開きます。



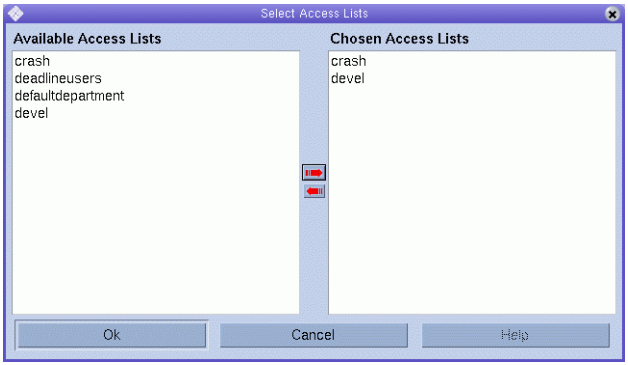
選択したプロジェクトの名前が「Name」フィールドに表示されます。プロジェクトにより、プロジェクトへのアクセスが許可または拒否されているユーザーのアクセスリストが定義されます。

「User Lists」の下のアクセスリストに含まれているユーザーは、プロジェクトにアクセスするアクセス権を持っています。「Xuser Lists」の下のアクセスリストに含まれているユーザーは、プロジェクトへのアクセスを拒否されます。詳細については 103 ページの「ユーザーの構成」を参照してください。

両方のリストが空である場合、すべてのユーザーがプロジェクトにアクセスできます。「User Lists」と「Xuser Lists」の両方に関連付けられている、異なるアクセスリストに含まれるユーザーは、プロジェクトへのアクセスを拒否されます。

「User Lists」または「Xuser Lists」にアクセスリストを追加したり、いずれかのリストからアクセスリストを削除することができます。このためには、「User Lists」または「Xuser Lists」の右側のボタンをクリックします。

「Select Access Lists」ダイアログボックスが表示されます。



「Select Access Lists」ダイアログボックスには、「Available Access Lists」の下で現在定義されているすべてのアクセスリストが表示されます。ダイアログボックスの「Chosen Access Lists」の下には、関連付けられているリストが表示されます。一方のリストでアクセスリストを選択できます。赤い矢印を使用すると、一方のリストからもう一方のリストにアクセスリストを起動できます。

変更を保存するには「OK」をクリックし、ダイアログ ボックスを閉じます。変更を保存せずにダイアログボックスを閉じるには、「Cancel」をクリックします。

コマンド行からのプロジェクトの定義

コマンド行からプロジェクトを構成するには、適切なオプションを使用して、次のコマンドを入力します。

```
# qconf オプション
```

次のオプションを使用できます。

- `qconf -aprj`

  `-aprj` オプション (プロジェクトの追加) を使用すると、エディタでテンプレートプロジェクト構成が開かれます。`project(5)` のマニュアルページを参照してください。このエディタは、デフォルトの `vi` エディタか、`EDITOR` 環境変数により指定されたエディタのいずれかです。 変更を保存してエディタを終了すると、変更は `sge_qmaster` に登録されます。

- `qconf -Aprj` *filename*

  `-Aprj` オプション (ファイルからのプロジェクトの追加) を使用すると、指定したファイルが解析され、新しいプロジェクト構成が追加されます。このファイルは、プロジェクト構成テンプレートの形式を有する必要があります。

- `qconf -dprj` *project-name*[,...]

  `-dprj` オプション (プロジェクトの削除) を使用すると、1 つ以上のプロジェクトが削除されます。

- `qconf -mprj` *project-name*

  `-mprj` オプション (プロジェクトの変更) を使用すると、既存のユーザーエントリを変更できます。このオプションにより、プロジェクト構成がエディタに読み込まれます。このエディタは、デフォルトの `vi` エディタか、`EDITOR` 環境変数により指定されたエディタのいずれかです。 変更を保存してエディタを終了すると、変更は `sge_qmaster` に登録されます。

- `qconf -Mprj` *filename*

  `-Mprj` オプション (ファイルからのプロジェクトの変更) を使用すると、指定したファイルが解析され、既存のプロジェクト構成が変更されます。このファイルは、プロジェクト構成テンプレートの形式を有する必要があります。

- `qconf -sprj` *project-name*

  `-sprj` オプション (プロジェクトの表示) を使用すると、特定のプロジェクトの構成が表示されます。

- `qconf -sprjl`

  `-sprjl` オプション (プロジェクトリストの表示) を使用すると、現在定義されているすべてのプロジェクトのリストが表示されます。

パスの別名設定の使用

Solaris などのネットワーク UNIX 環境では、ユーザーがいくつかのマシンに同じホームディレクトリ (あるいはその一部) を持つことがよくあります。たとえば、ディレクトリが NFS でアクセス可能になっている場合などです。しかし、すべてのマシンでホームディレクトリのパスが完全には同じでないことがあります。

たとえばNFSとオートマウンタで使用可能なユーザーのホームディレクトリを考えてみます。ユーザーがNFSサーバー上に/home/fooというホームディレクトリを持っている場合、オートマウンタを実行する適切にインストールされたすべてのNFSクライアント上では、このパスでこのホームディレクトリはアクセスできます。しかし、クライアントの/home/fooは、/tmp_mnt/home/fooへのシンボリックリンクにすぎません。オートマウンタがディレクトリを物理的にマウントしているNFSサーバー上の実際の場所は、/tmp_mnt/home/fooです。

クライアントホスト上のユーザーがqsub -cwdコマンドを使用してホームディレクトリツリー内のある場所からジョブを発行した場合、-cwdフラグは、そのジョブが現在の作業ディレクトリで実行されることを要求します。ただし、実行ホストがNFSサーバーである場合、Grid Engineシステムは、そのホストで現在の作業ディレクトリを見つけられない可能性があります。これは、発行ホスト上の現在の作業ディレクトリが、(発行ホスト上の物理的な位置である)/tmp_mnt/home/fooであるためです。このパスは実行ホストに渡されます。ただし、実行ホストがNFSサーバーである場合、物理的なホームディレクトリパスは/home/fooで、/tmp_mnt/home/fooではないため、パスが解決できません。

そのほか、同じような問題を引き起こすケースとしては、次のような場合があります。

- マシンによってマウントポイントのパスが異なる固定NFSマウント。例としては、あるホストでは/usr/peopleにホームディレクトリをマウントし、別のホストで/usr/usersの下にホームディレクトリをマウントする場合があります。
- ネットワークから利用可能なファイルシステムへの外部からのシンボリックリンク

このような問題を防ぐため、Grid Engineソフトウェアでは、管理者とユーザーがどちらもパス別名設定ファイルを構成できます。このような2つのファイルは、次の場所にあります。

- *sge-root*/*cell*/common/sge_aliases — クラスタ用のグローバルクラスタパス別名設定ファイル
- $HOME/.sge_aliases — ユーザー別のパス別名設定ファイル

注– グローバルファイルの変更は、管理者だけが行なってください。

パス別名設定ファイルの形式

パス別名設定ファイルは、両方とも同じ形式です。

- 空白行と、先頭に # 記号がある行は無視されます。
- 空白行と # で始まる行以外の各行には、任意の数の空白文字またはタブで区切った 4 つの文字列が含まれる必要があります。

  最初の文字列がソースパス、2 つ目が発行ホスト、3 つ目が実行ホスト、4 つ目がソースパス置換を指定します。
- 発行ホストおよび実行ホストの文字列には、任意のホストに一致する * (アスタリスク) 記号を使用できます。

パス別名設定ファイルの解釈法

ファイルは、次のように解釈されます。

1. qsub が物理的な現在の作業ディレクトリのパスを検索したあとで、グローバルパス別名設定ファイルが存在する場合は、そのファイルが読み取られます。ユーザーパス別名設定ファイルは、グローバルファイルの最後に付加されているかのように、あとで読み取られます。
2. 無視されないファイルは、ファイルの先頭から1 行ずつ読み取られます。必要に応じて、それらの行により指定された置換内容が格納されます。 置換内容が保存されるのは、次の条件がいずれも真である場合のみです。
   - 発行ホストの文字列が、qsub コマンドが実行されるホストに一致する。
   - ソースパスが、すでに格納されている現在の作業ディレクトリまたはソースパス置換の先頭部分の構成要素になっている。
3. 両方のファイルが読み取られると、格納されているパス別名設定情報が、発行されたジョブとともに実行ホストに渡されます。
4. 実行ホストで、別名設定情報が評価されます。実行ホストの文字列が実行ホストに一致する場合は、現在の作業ディレクトリの先頭部分が、ソースパス置換により置き換えられます。この場合、現在の作業ディレクトリ文字列が変更されます。以降のパス別名は、適用されるためには、置換された作業ディレクトリパスに一致する必要があります。

例 4–1 は、上記の NFS オートマウンタの問題が、別名設定ファイルエントリを使用してどのように解決できるかを示す例です。

例 4–1 パス別名設定ファイルの例

```
# cluster global path aliases file
# src-path  subm-host   exec-host   dest-path
/tmp_mnt/   *           *           /
```

デフォルト要求の構成

通常、バッチジョブは、要求プロファイルを基にキューに割り当てられます。ユーザーは、特定のジョブに対して要求プロファイルを定義します。ユーザーは、ジョブを正しく実行するために満たされなければならない要求のセットを組み立てます。スケジューラは、このジョブに関する要求のセットを満たすキューだけを考慮します。

ユーザーがジョブに関する要求を指定しない場合、スケジューラは、ユーザーがアクセス可能なあらゆるキューを考慮します。この場合、キューがアクセス可能であること以外の制約はありません。ただし、Grid Engine ソフトウェアでは、ユーザーがリソース要求を明示的に指定しなくてもジョブのリソース要求を定義するデフォルト要求を構成することができます。

デフォルト要求は、クラスタのユーザー全員にグローバルに構成することも、任意のユーザーに対して個人用として構成することもできます。デフォルト要求の構成は、デフォルト要求ファイルに格納されます。グローバル要求ファイルは *sge-root/cell* `/common/sge_request` にあります。ユーザー別の要求ファイルは、ユーザーのホームディレクトリまたは現在の作業ディレクトリのいずれかに置くことができます。作業ディレクトリは、`qsub` コマンドが実行される場所です。ユーザー別の要求ファイルは、`.sge_request` と呼ばれます。

これらのファイルが存在する場合は、あらゆるジョブでその評価が行われます。評価の順序は次のとおりです。

1. グローバルデフォルト要求ファイル
2. ユーザーのホームディレクトリにあるユーザー別デフォルト要求ファイル
3. 現在の作業ディレクトリにあるユーザー別デフォルト要求ファイル

注 – ジョブスクリプトまたは `qsub` コマンドで指定された要求は、デフォルト要求ファイルの要求より優先されます。ジョブに対する明示的なリソース要求方法についての詳細は、『Sun N1 Grid Engine 6.1 ユーザーズガイド』の第 3 章「ジョブの発行」を参照してください。

(以前の要求の指定を破棄する) `qsub -clear` コマンドを使用することで、Grid Engine システムがデフォルト要求ファイルを使用しないようにすることができます。

デフォルト要求ファイルの形式

ローカルおよびグローバル両方のデフォルト要求ファイルの形式は次のようになります。

- デフォルト要求ファイルには、任意の数の行を含むことができます。空白行と、先頭に # 記号がある行は無視されます。
- qsub(1) のマニュアルページで説明されているように、無視する行以外の各行には、任意の qsub オプションを含めることができます。1 行に複数のオプションを指定することができます。バッチスクリプトファイルと、バッチスクリプトに対する引数オプションは、qsub オプションとみなされません。このため、これらはデフォルト要求ファイルでは使用できません。
- qsub -clear コマンドは、現在評価されている要求ファイルまたは以前に処理された要求ファイルの、以前のあらゆる要求指定を廃棄します。

あるユーザーのローカルのデフォルト要求ファイルが例 4–2 のスクリプト、test.sh と同じ構成であると仮定します。

例 4–2 デフォルト要求ファイルの例

```
# Local Default Request File
# exec job on a sun4 queue offering 5h cpu
-l arch=solaris64,s_cpu=5:0:0
# exec job in current working dir
-cwd
```

このスクリプトを実行するには、ユーザーは次のコマンドを入力します。

```
% qsub test.sh
```

次のように、ユーザーが直接コマンド行ですべての qsub オプションを指定した場合、test.sh スクリプトを実行した結果と同じになります。

```
% qsub -l arch=solaris64,s_cpu=5:0:0 -cwd test.sh
```

注 – qsub を使用して発行したバッチジョブ同様、qsh を使用して発行した対話形式のジョブでもデフォルト要求ファイルは考慮されます。QMON を使用して発行した対話形式またはバッチジョブでも、これらの要求ファイルが考慮されます。

◆ ◆ ◆ 第 5 章

ポリシーとスケジューラの管理

この章では、Grid Engine システムのポリシーを説明します。この章の内容は次のとおりです。

- スケジューリング
- ポリシー

内容説明に加えて、この章では次のタスクの実行方法を詳細に解説します。

スケジューラの管理

この節では、Grid Engine システムが実行用にジョブをどのようにスケジューリングするかを説明します。また、さまざまな種類のスケジューリング戦略と、スケジューラを構成する方法についても説明します。

スケジューリングについて

Grid Engine システムでは、次のジョブスケジューリング処理が行われます。

- 振り分け前の決定。キューがいっぱいであるか過負荷状態であるためキュー削除する、現在実行には適切でないジョブをスプールする、優先順位の高いジョブにリソースを要約するなどの処理
- 振り分け。そのほかの保留中のジョブおよび実行中のジョブを考慮したジョブの重要性の決定、クラスタ内の全てのマシン上の負荷の検知、および構成済みの選択基準に基づいて選択されたマシン上のキューへのジョブの送信
- 振り分け後の監視。 ジョブがリソースを取得した時点、および独自の相対的な重要性を持つそのほかのジョブがシステムに出入りした時点での、ジョブの相対的な重要性の調整

Grid Engine ソフトウェアは、次の基準に基づいて、コンピュータの異機種システム混在クラスタでジョブをスケジューリングします。

- クラスタの現在の負荷
- ジョブの相対的な重要性
- ホストの相対的なパフォーマンス
- CPU、メモリー、および I/O 帯域幅など、ジョブのリソース要求

スケジューリングに関する決定は、サイトの戦略と、クラスタ内の各コンピュータの瞬間的な負荷の特性に基づいて行われます。サイトのスケジューリング戦略は、Grid Engine システムの構成パラメータを介して表されます。負荷特性は、動作中のシステムのパフォーマンスデータを収集することによって確認されます。

スケジューリング戦略

管理者は、次のスケジューリング業務について戦略を立てることができます。

- 動的リソース管理。Grid Engine システムは、実行中のジョブに割り当てられるリソースのエンタイトルメントを動的に制御および調整します。つまり、システムが CPU の配分を変更します。
- キューのソート。ソフトウェアは、キューを埋める順番に従ってクラスタ内のキューをランク付けします。
- ジョブのソート。ジョブのソートは、Grid Engine システムがジョブをスケジューリングする順番を決定します。
- リソース予約およびバックフィリング。リソース予約は、ジョブのリソースを予約し、 低優先順位のジョブによる使用をブロックします。バックフィリングにより、リソースを使用しても予約に干渉しない場合は、低優先順位のジョブが、ブロックされたリソースを使用できます。

動的リソース管理

Grid Engine ソフトウェアは、次の 3 つのチケットに基づくポリシーの重み付けされた組み合わせを使用して、ジョブスケジューリング戦略の自動化を実現します。

- 共有ベース
- 機能 (プライオリティと呼ばれる場合もあり)
- 優先

Grid Engine システムを設定して、日常的に共有ベースポリシーと機能ポリシーのいずれか、またはその両方を使用するよう構成できます。これらのポリシーは、任意に組み合わせることができます。たとえば、1 つのポリシーにゼロの重みを与え、2 つ目のポリシーのみを使用することができます。また、両方のポリシーに同じ重みを与えることもできます。

日常的なポリシーとともに、管理者は共有ベースおよび機能スケジューリングを一時的に無効にしたり、エクスプレスキューなどの目的のために永続的に無効にすることができます。優先は、1 つのジョブに対して適用することも、ユーザー、部署、プロジェクト、ジョブクラス (すなわち、キュー) に関連付けられているすべてのジョブに適用することもできます。

Grid Engine システムには、すべてのジョブ間の調停をするためのこれら 3 つのポリシーのほかに、ユーザーが自分のジョブに優先順位を設定する機能もあります。たとえば、あるユーザーには、ジョブ 1 と 2 の重要性は同じであり、ジョブ 3 はジョブ 1 と 2 よりも重要である場合があります。ユーザーは、ポリシーの組み合わせに共有ベースポリシー、機能ポリシー、またはその両方が含まれる場合、独自のジョブの優先順位を設定することができます。また、ジョブには機能チケットを付与する必要があります。

チケット

共有ベース、機能、および優先スケジューリングポリシーは、チケットを使用することで実現されます。各ポリシーにはチケットのプールがあります。ジョブが複数のマシンから成る Grid Engine システムに入力された時点で、ポリシーはチケットをジョブに割り当てます。有効である日常的な各ポリシーは、一部のチケットを新しい各ジョブに割り当てます。ポリシーは、各スケジューリング間隔で、実行中のジョブにチケットを再割り当てする場合もあります。

チケットは、3 つのポリシーに重みを与えます。たとえば、機能ポリシーにチケットが割り当てられていない場合、そのポリシーは使用されません。機能チケットプールと共有ベースチケットプールに同じ数のチケットがある場合、ジョブの重要性を決定する際、両方のポリシーは等しい重みを持ちます。

チケットは、Grid Engine システム管理者によるシステム構成で、定期ポリシーに割り当てられます。管理者とオペレータは任意の時点でチケットの割り当てを変更でき、この変更はただちに効力を持ちます。優先を指示するには、システムに一時的に追加チケットを注入します。ポリシーはチケットの割り当てによって組み合わせ

られます。複数のポリシーにチケットが割り当てられている場合、ジョブは各ポリシーのチケットの一部を受け取ります。このことは、有効な各ポリシーにおけるジョブの重要性を示します。

Grid Engine システムは、システムに入るジョブにチケットを付与することによって、有効な各ポリシーにおけるその重要性を指示します。各スケジューリング間隔で、実行中の各ジョブはチケットを取得したり、チケットを失ったり、また同じ数のチケットを維持することができます。たとえば、ジョブは優先からチケットを取得する場合があります。ジョブは、正当なリソースの配分以上のチケットを取得しているため、チケットを失う場合があります。ジョブが保持するチケット数は、 Grid Engine システムが各スケジューリング間隔時にそのジョブに付与しようとするリソース配分を表します。

インストール時には、サイトの動的リソース管理戦略を構成します。まず、管理者は共有ベースポリシーと機能ポリシーにチケットを割り当てます。続いて、共有ツリーと機能配分を定義します。共有ベースチケットの割り当てと機能チケットの割り当ては、任意の時点で自動的に変更できます。管理者は、手動でチケットを割り当てたり、削除したりします。

キューのソート

Grid Engine システムがキューを埋める順番を決定するために、次の手段が用意されています。

- 負荷レポート。 管理者は、ホストの負荷状態とそのキューインスタンスを比較するためにどの負荷パラメータを使用するかを選択できます。使用可能な幅広い標準的な負荷パラメータ、およびサイト固有の負荷センサーを使用してこのセットを拡張するためのインタフェースは、90 ページの「負荷パラメータ」で説明されています。
- 負荷スケーリング。さまざまなホストからの負荷レポートを正規化して、負荷状況を比較できるようにします。26 ページの「QMON を使用した実行ホストの構成」を参照してください。
- 負荷調整。ホストにジョブを振り分けたときに、前回報告された負荷を Grid Engine ソフトウェアが自動的に修正するよう構成することができます。修正された負荷は、最近のジョブの実行開始で発生すると予想される負荷状況の増加を表します。この人為的に増加された負荷は、それらのジョブの負荷に対する影響が明らかになると、自動的に減少させることができます。
- 連続番号。厳密な順序でキューをソートすることができます。

ジョブのソート

Grid Engine システムがジョブの振り分けを開始する前、ジョブは優先順位の高い順に、優先順位に組み入れられます。続いてシステムは、優先順位の順番でジョブに適したリソースを見つけようとします。

管理者の操作がなければ、順序は先入れ先出し (FIFO) です。管理者には、ジョブの順序を制御する次の手段があります。

- チケットに基づくジョブ優先順位。ジョブは常に、ジョブが有するチケットの数により定義される相対的な重要性にしたがって扱われます。保留中のジョブはチケット順にソートされます。管理者がチケットポリシーに適用するすべての変更は、ソート順も変更します。
- 緊急度に基づくジョブ優先順位。ジョブには、その相対的な重要性を決定する緊急度の値を割り当てることができます。保留中のジョブは、緊急度の値に従ってソートされます。緊急度ポリシーに適用されるすべての変更により、ソート順序も変更されます。
- **POSIX** 優先順位。 `qsub` コマンドに `–p` オプションを使用すると、サイト固有の優先順位ポリシーを実現できます。`–p` オプションにより、–1023 から 1024 の範囲の優先順位が指定されます。数字が大きければ、優先順位も高くなります。 ジョブのデフォルトの優先順位はゼロです。
- ユーザーまたはユーザーグループジョブの最大数。ユーザーまたは UNIX ユーザーグループが並行して実行可能なジョブの最大数を制限できます。制限を超えていないユーザーのジョブは優先されるため、この制限は保留中のジョブリストのソート順序に影響します。

各優先順位の種類に対して、重み付け係数を指定できます。この重み付け係数は、各種類の優先順位がジョブ全般の優先順位に影響を与える程度を決定します。各優先順位の種類の値の範囲を制御しやすくするため、生のチケットの値、緊急度の値、および POSIX 優先順位の値の代わりに、正規化された値が使用されます。

ジョブの優先順位値がどのように決定されるかは、次の式で表されます。

```
job_priority = weight_urgency * normalized_urgency_value +
weight_ticket * normalized_ticket_value +
weight_priority * normalized_POSIX_priority_value
```

ジョブの優先順位の監視には `qstat` コマンドを使用できます。

- POSIX 優先順位を含む、ジョブ優先順位全般を監視するには、`qstat –pri` を使用します。
- チケットポリシーに基づいてジョブ優先順位を監視するには、`qstat –ext` を使用します。
- 緊急度ポリシーに基づいてジョブ優先順位を監視するには、`qstat –urg` を使用します。
- 緊急度ポリシー、チケットに基づくポリシー、および *-p <priority>* を同時に使用した場合のジョブの優先順位に関する問題を診断するには、`qstat –pri` を使用します。
- エラー状態に基づいてさまざまなキューインスタンスを診断するには、`qstat –explain` を使用します。

緊急度ポリシーについて

緊急度ポリシーは各ジョブの緊急度の値を定義します。緊急度の値は、次の 3 つの情報の合計から得られます。

- リソース要求の情報
- 待機時間の情報
- 締め切りの情報

リソース要求の情報は、すべてのハードリソースの要求の合計から得られ、各要求に関する 1 つの加数になります。

リソース要求が型 *numeric* である場合、リソース要求の加数は、次の 3 つの要素の積になります。

- コンプレックスで定義されているリソースの緊急度の値。詳細については、72 ページの「QMON を使用したコンプレックスリソース属性の構成」を参照してください。
- ジョブの、想定されているスロット割り当て。
- `qsub –l` コマンドにより指定されるスロットごとの要求。

リソース要求が型 *string* である場合、リソース要求の加数は、コンプレックスで定義されているリソースの緊急度の値になります。

待機時間の情報は、秒単位でのジョブの待機時間と、「Policy Configuration」ダイアログボックスで指定されている waiting-weight 値の積になります。

締め切りの情報は、締め切りのないジョブではゼロになります。締め切りのあるジョブの場合、締め切りの情報は、(締め切り開始時間までの) 秒単位の空き時間で除算された、「Policy Configuration」ダイアログボックスで定義されている waiting-weight 値です。

緊急度ポリシーの構成の詳細については、135 ページの「緊急度ポリシーの構成」を参照してください。

リソース予約およびバックフィリング

リソース予約を使用すると、指定した保留中のジョブに、システムリソースを予約できます。ジョブに対してリソースを予約する場合、このようなリソースは優先順位の低いジョブには使用されません。

リソース要求、ジョブ優先順位、待機時間、リソース配分エンタイトルメントなどの基準に応じて、ジョブはリソースを予約できます。最も優先順位の高いジョブが一番早く使用可能なリソース割り当てを取得するよう、スケジューラは予約を強制します。これにより、「ジョブ枯渇」というよく知られた問題を回避できます。

リソース予約を使用すると、ジョブ優先順位の順序でリソースがジョブに振り分けられるようにすることができます。

次の例を考えます。ジョブ A はサイズの大きな保留中のジョブで、並列ジョブの可能性もあり、特定のリソースを大量に必要とします。比較的小さなジョブ B(i) のストリームは、同じリソースを必要としますが、比較的少量です。リソース予約がなければ、B(i) ジョブのストリームが停止しないという前提では、ジョブ A に対するリソース割り当てを保証できません。ジョブ A が B(i) ジョブよりも高い優先順位を持っている場合であっても、リソースを保証できません。

リソース予約により、ジョブ A は、低優先順位ジョブ B(i) をブロックする予約を取得します。可能なかぎり早い段階で、リソースはジョブ A に使用できるよう保証されます。

バックフィリングを使用すると、低優先順位ジョブは、リソース予約によりブロックされたリソースを使用できるようになります。バックフィリングが機能するのは、見込みの実行時間が十分短いため、元の予約に干渉することなくブロックされたリソースを使用できる、実行可能なジョブが存在する場合のみです。

上記の例では、期間が非常に短いジョブ C は、バックフィリングを使用してジョブ A の前に実行可能です。

リソース予約によりスケジューラはルックアヘッドを行うため、リソース予約を使用するとシステムのパフォーマンスに影響を与えます。小規模なクラスタでは、保留中のジョブがごく少数である場合、パフォーマンスへの影響は無視できる程度です。ただし、大規模なクラスタでは、多くの保留中のジョブがあるクラスタであれば、パフォーマンスへの影響は大きくなる場合があります。

発生しうるこのパフォーマンスの低下を相殺するために、スケジューリング間隔中に作成可能なリソース予約の総数を制限することができます。リソース予約は次の 2 つの方法で制限できます。

- スケジューリング間隔時に作成可能な予約の絶対数を制限するには、「Scheduler Configuration」ダイアログボックスで Maximum Reservation パラメータを設定します。たとえば、Maximum Reservation を 20 に設定した場合、間隔内では 20 までの予約しか作成できません。
- 重要なジョブに対してのみ予約スケジューリングを制限するには、`qsub` コマンドの `–R y` オプションを使用します。上記の例では、ジョブ A のリソース予約を保証するためには B(i) ジョブをスケジュールする必要はありません。`–R y` オプションを使用して発行しなければならないジョブは、ジョブ A だけです。

スケジューラを構成して、スケジューラがリソース予約によりどのような影響を受けるかを監視することができます。スケジューラを監視する場合は、各スケジューリング実行に関する情報はファイル *sge-root*/ *cell*/`common/schedule` に記録されます。

次の例に、スケジュール監視の実行内容を示します。グローバル `license` の消費可能リソースが 5 ライセンスに制限されているクラスタに、次のジョブのシーケンスが発行されると仮定します。

```
qsub -N L4_RR -R y -l h_rt=30,license=4 -p 100 $SGE_ROOT/examples/jobs/sleeper.sh 20
qsub -N L5_RR -R y -l h_rt-30,license=5        $SGE_ROOT/examples/jobs/sleeper.sh 20
qsub -N L1_RR -R y -l h_rt=31,license=1        $SGE_ROOT/examples/jobs/sleeper.sh 20
```

スケジューラ構成では、デフォルトの優先順位設定が使用されると仮定します。

```
weight_priority          1.000000
weight_urgency           0.100000
weight_ticket            0.010000
```

ジョブ L4_RR の –p 100 の優先順位は、ライセンスに基づく緊急度に取って代わるため、次の優先順位付けが行われます。

```
job-ID  prior    name
--------------------
   3127 1.08000 L4_RR
   3128 0.10500 L5_RR
   3129 0.00500 L1_RR
```

この場合、6 つのスケジュール間隔で、これらのジョブのトレースが schedule ファイルに見つかります。

```
::::::::
      3127:1:STARTING:1077903416:30:G:global:license:4.000000
      3127:1:STARTING:1077903416:30:Q:all.q@carc:slots:1.000000
      3128:1:RESERVING:1077903446:30:G:global:license:5.000000
      3128:1:RESERVING:1077903446:30:Q:all.q@bilbur:slots:1.000000
      3129:1:RESERVING:1077903476:31:G:global:license:1.000000
      3129:1:RESERVING:1077903476:31:Q:all.q@es-ergb01-01:slots:1.000000
      ::::::::
      3127:1:RUNNING:1077903416:30:G:global:license:4.000000
      3127:1:RUNNING:1077903416:30:Q:all.q@carc:slots:1.000000
      3128:1:RESERVING:1077903446:30:G:global:license:5.000000
      3128:1:RESERVING:1077903446:30:Q:all.q@es-ergb01-01:slots:1.000000
      3129:1:RESERVING:1077903476:31:G:global:license:1.000000
      3129:1:RESERVING:1077903476:31:Q:all.q@es-ergb01-01:slots:1.000000
      ::::::::
      3128:1:STARTING:1077903448:30:G:global:license:5.000000
      3128:1:STARTING:1077903448:30:Q:all.q@carc:slots:1.000000
      3129:1:RESERVING:1077903478:31:G:global:license:1.000000
      3129:1:RESERVING:1077903478:31:Q:all.q@bilbur:slots:1.000000
      ::::::::
      3128:1:RUNNING:1077903448:30:G:global:license:5.000000
      3128:1:RUNNING:1077903448:30:Q:all.q@carc:slots:1.000000
      3129:1:RESERVING:1077903478:31:G:global:license:1.000000
      3129:1:RESERVING:1077903478:31:Q:all.q@es-ergb01-01:slots:1.000000
      ::::::::
      3129:1:STARTING:1077903480:31:G:global:license:1.000000
```

```
3129:1:STARTING:1077903480:31:Q:all.q@carc:slots:1.000000
::::::::
3129:1:RUNNING:1077903480:31:G:global:license:1.000000
3129:1:RUNNING:1077903480:31:Q:all.q@carc:slots:1.000000
```

スケジュール間隔に関して、各セクションでは、考慮されたすべてのリソース利用量が示されます。RUNNING エントリは、間隔の開始時点ですでに実行中であったジョブの利用量を示しています。STARTING エントリは、間隔内で決定された即時の利用量を示しています。RESERVING エントリは、将来計画されている使用、つまり予約を示しています。

スケジュールファイルの書式は次のとおりです。

jobID	ジョブ ID
taskID	配列タスク ID (通常のジョブの場合は 1)
状態	RUNNING、SUSPENDED、 MIGRATING、STARTING、RESERVING を取ります
開始時刻	1.1.1070 以降の秒単位での開始時刻
継続時間	想定されている秒単位でのジョブの継続時間
レベル文字	P (並列環境の場合)、G (グローバルの場合)、H (ホストの場合)、または Q (キューの場合) を取ります
オブジェクト名	並列環境、ホスト、またはキューの名前
リソース名	消費可能リソースの名前
使用率	そのジョブにより発生するリソース使用率

行 :::::::: は、新しいスケジュール間隔の開始のマークとなります。

注 – schedule ファイルは、切り捨てられません。ファイルを切り捨てるように設定されている自動手続きがない場合は、必ず監視をオフにしてください。

スケジューラ間隔で行われる動作

スケジューラは、間隔での作業をスケジューリングします。スケジューリングアクションの間、Grid Engine システムは次の要素などの、重要なイベントに関する情報を保持します。

- ジョブの発行
- ジョブの完了
- ジョブの取り消し
- クラスタ構成の更新
- クラスタでの新しいマシンの登録

スケジューリングが行われる際には、スケジューラはまず次の処理を実行します。

- すべての重要なイベントの考慮
- 管理者の指定に従ったジョブとキューのソート
- すべてのジョブのリソース要求の考慮
- 将来に備えたスケジュールでのジョブに関するリソースの予約

必要に応じて、Grid Engine システムは次のタスクを実行します。

- 新しいジョブの振り分け
- 実行中のジョブの一時停止
- 実行中のジョブに割り当てられているリソースの増減
- 現状の維持

共有ベーススケジューリングが使用されている場合、そのユーザーまたはプロジェクトに対してすでに発生している利用量が、計算では考慮されます。

少なくとも一部でもスケジューリングが共有ベーススケジューリングでない場合は、計算では実行中および実行待ちのすべてのジョブがランク付けされます。クラスタ内のリソース (CPU、メモリー、および I/O 帯域幅) が可能なかぎり使用されるようになるまで、計算では最も重要なジョブが処理されます。

スケジューラの監視

ジョブが開始しない理由が不明である場合、そのジョブに対して `qalter -w v` コマンドを発行します。Grid Engine ソフトウェアでは、クラスタが空であることが仮定されていて、またジョブに適したキューが使用可能であるかどうかチェックされます。

詳細な情報は、`qstat -j` *job-id* コマンドを実行することで取得できます。このコマンドは、ジョブの要求プロファイルの要約を出力します。要約には、前回のスケジューリング実行でジョブがスケジューリングされなかった理由も含まれます。ジョブ ID を使用せずに `qstat -j` コマンドを実行すると、前回のスケジューリング間隔でスケジューリングされなかったすべてのジョブに関する理由が要約されます。

注 – スケジューラ構成 `sched_conf`(5) では、ジョブスケジューリング情報の収集をオンにする必要があります。`sched_conf`(5) のマニュアルページの `schedd_job_info` パラメータの説明、または 128 ページの「QMON を使用したスケジューラ構成の変更」を参照してください。

スケジューラ `sge_schedd` の決定に関する詳細な情報を取得するには、`qconf` コマンドの `-tsm` オプションを使用します。このコマンドを使用すると、`sge_schedd` は強制的にファイルにトレース出力を書き込みます。

スケジューラの構成

Grid Engine システムのリソース共有ポリシーのスケジューリング管理の詳細については、132 ページの「QMON を使用したポリシーに基づくリソース管理の構成」を参照してください。次の節では、スケジューラ構成 sched_conf の管理とそれに関連する問題に焦点を当てます。

デフォルトのスケジューリング

デフォルトのスケジューリングは、*FIFO* 方式です。すなわち、最初に発行されたジョブが、最初にスケジューラによって調べられ、キューに振り分けられます。保留中のジョブのリストの最初のジョブが、適切で使用可能なジョブを見つけると、そのジョブがまず開始されます。最初のジョブよりも下位のランクのジョブが開始できるのは、最初のジョブが適切な空きリソースを見つけられなかった場合のみです。

デフォルト戦略では、ジョブのリソース要求に対してキューが適切なサービスを提供するかぎり、最も負荷の低いホスト上のキューインスタンスを選択します。複数の適切なキューが同じ負荷を共有する場合、どのキューが選択されるか予測できません。

そのほかのスケジューリング方法

次のようなさまざまな方法で、ジョブのスケジューリングとキュー選択の戦略を変更できます。

- スケジューリングアルゴリズムの変更
- システム負荷のスケーリング
- 連続番号によるキューの選択
- 配分によるキューの選択
- ユーザー 1 人またはグループ 1 つあたりのジョブ数の制限

次では、これらの代替方法を詳しく説明します。

スケジューリングアルゴリズムの変更

スケジューラ構成パラメータ algorithm は、使用するスケジューリングアルゴリズムを選択できるようにします。詳細については、sched_conf(5) のマニュアルページを参照してください。ただし、現在指定できる設定は default だけです。

システム負荷のスケーリング

ジョブを実行するキューを選択するため、Grid Engine システムは、キューインスタンスをホスティングするマシンのシステム負荷情報を使用します。このキュー選択方式によって負荷分散状態が確立され、クラスタ内の使用可能なリソースの有効利用が保証されます。

ただし、システム負荷が常に真実を伝えるとはかぎりません。たとえば複数 CPU のマシンとシングル CPU のシステムを比較すると、通常、多くの場合マルチプロセッサシステムの方が実行しているプロセス数が多いため、マルチプロセッサシステムが報告する負荷値の方が大きくなります。システム負荷は、CPU へのアクセス権を取得しようとするプロセス数に大きく左右される測定値です。しかし、複数 CPU システムは、シングル CPU マシンに比べてはるかに大きな負荷に対処することができます。この問題は、`sge_execd` からデフォルトで報告される負荷値をプロセッサ数で調整することによって対処します。すなわち、生の負荷値ではなく、負荷パラメータを使用することによって、上記の問題に対処することができます。詳細については、90 ページの「負荷パラメータ」 および *sge-root*`/doc/load_parameters.asc` ファイルを参照してください。

負荷値が正しく判断されない可能性があるもう 1 つの例として、システムで潜在的なパフォーマンスあるいは価格パフォーマンス比に著しい差がある場合があります。どちらの場合も、負荷値が同じであるからといって、ジョブの実行用にどちらのホストを選択してもよいわけではありません。こうした場合、管理者は、実行ホストと負荷パラメータに対する負荷スケーリング係数を定義する必要があります。26 ページの「QMON を使用した実行ホストの構成」および関連の節を参照してください。

注 – スケーリングした負荷パラメータは、負荷しきい値リストの *load-thresholds* および *migr-load-thresholds* とも照合されます。詳細については、`queue_conf`(5) のマニュアルページを参照してください。

負荷パラメータに関連するもう 1 つの問題は、値とその相対的な重要性を、アプリケーションおよびサイトに依存して解釈する必要があることです。特定のサイトで一般的なある種のアプリケーションでは CPU 負荷が圧倒的であるのに対し、別のサイトや、サイトの計算クラスタが専門に扱うアプリケーションプロファイルでは、メモリー負荷がずっと重要であることがあります。この問題に対処するため、Grid Engine システムでは、管理者はスケジューラ構成ファイル `sched_conf` で負荷の式を指定できます。詳細については、`sched_conf`(5) のマニュアルページを参照してください。負荷の式でサイト定義の負荷パラメータと消費可能リソースを使用することによって、リソース使用率と容量計画に関するサイト固有の情報を考慮することができます。91 ページの「サイト固有の負荷パラメータの追加」および78 ページの「消費可能リソース」の各節を参照してください。

最後に、負荷パラメータの時間依存も考慮する必要があります。システムで実行中のジョブによって課される負荷は時間とともに変化します。しばしば、CPU 負荷などの負荷は、オペレーティングシステムが適切な報告するのにかなりの時間が必要になることがあります。ジョブの開始直後の場合、報告される負荷は、ジョブによってホストに課されている負荷を正確に表していないことがあります。報告される負荷は時間とともに実際の負荷に近づいていきます。しかし、報告される負荷が低すぎる間は、そのホストが過剰な予約を受ける可能性があります。Grid Engine システムでは、管理者は、スケジューラでこの問題の補正に使用される負荷調整係数を指定することができます。このような負荷調整係数の設定方法の詳細については、sched_conf(5) のマニュアルページを参照してください。

負荷調整は、ジョブが振り分けられたあとに、測定される負荷を仮想的に増加させるために使用します。過剰な予約が発生したマシンの場合、これは負荷しきい値との調整に役立ちます。負荷調整が必要でない場合は、負荷調整をオフにする必要があります。負荷調整は、ホストのソートと負荷しきい値の検証を伴って、スケジューラに追加の作業を課すことになります。

負荷調整を使用不可にするには、「Scheduler Configuration」ダイアログボックスの「Load Adjustment」でタブで「Decay Time」をゼロに設定し、テーブル内のすべての負荷調整の値を削除します。128 ページの「QMON を使用したスケジューラ構成の変更」を参照してください。

連続番号によるキューの選択

デフォルトのキュー選択方法を変更するもう 1 つの方法は、グローバルクラスタ構成パラメータ queue_sort_method をデフォルトの load から seq_no に変更する方法です。この設定にすると、キュー選択の第一の手段としてシステム負荷が使用されなくなります。その代わりに、キュー構成パラメータ seq_no によりキューに割り当てられた連続番号が、キュー選択の固定順序を定義します。キューは、検討対象のジョブに適切で、使用可能である必要があります。詳細については、queue_conf(5) および sched_conf (5) のマニュアルページを参照してください。

このキュー選択方法は、サイトでバッチサービスを提供するマシンが、ジョブ 1 つあたりの単純な価格の順序でランク付けされている場合に役立ちます。たとえば、マシン A であるジョブを実行すると 1 単位のコストがかかるとします。同じジョブは、マシン B では 10 単位、マシン C では 100 単位のコストがかかります。この場合の望ましいスケジューリング方法は、最初にホスト A を一杯にして、次にホスト B、そして代わりが残っていない場合だけホスト C を使用するという方法です。

注 – キュー選択方法を seq_no に変更して、検討対象のすべてのキューの連続番号が同じ場合、キューはデフォルトの load により選択されます。

配分量によるキューの選択

この方法の目標は、すべてのジョブに対して目標とするグロバールシステムリソース配分が達成されるようにジョブを配置することにあります。この方法では、すべてのシステムリソースとの関係で各ホストが表すリソース容量を考慮します。またこの方法では、各ホストのチケットの割合 (すなわち、ホストで実行中のすべてのジョブの総チケット数) と、特定のホストがシステムに対して表すリソース能力の割合のバランスを取ろうとします。ホストの容量の定義方法については、26 ページの「QMON を使用した実行ホストの構成」を参照してください。

重要性は二次的なものとなりますが、ソートではホストの負荷も考慮されます。共有ツリーポリシーを使用しているサイトでは、このソート方法を選択してください。

ユーザー 1 人またはグループ 1 つあたりのジョブ数の制限

管理者は、ユーザーまたは UNIX グループが任意の時点で実行できるジョブ数に上限を割り当てることができます。この機能を使用するには、次のいずれかを実行します。

- `sched_conf`(5) のマニュアルページで説明されているように、`maxujobs` か `maxgjobs`、またはその両方を設定します。
- 「Scheduler Configuration」ダイアログボックスの「General Parameters」タブで、「Max Jobs/User」フィールドを使用して、ユーザーまたはユーザーグループが並行して実行できるジョブの最大数を設定します。

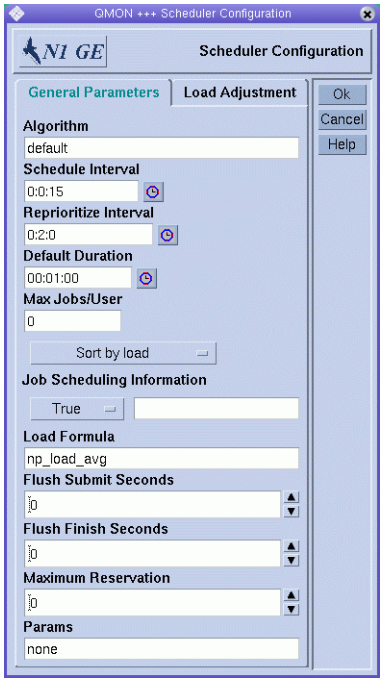
QMON を使用したスケジューラ構成の変更

「QMON Main Control」ウィンドウで、「Scheduler Configuration」ボタンをクリックします。

「Scheduler Configuration」ダイアログボックスが表示されます。このダイアログには、2 つのタブがあります。

- 「General Parameters」タブ
- 「Load Adjustment」タブ

一般的なスケジューリングパラメータを変更するには、「General Parameters」タブをクリックします。「General Parameters」タブは次の図のようになっています。



「General Parameters」タブを使用して、次のパラメータを設定します。

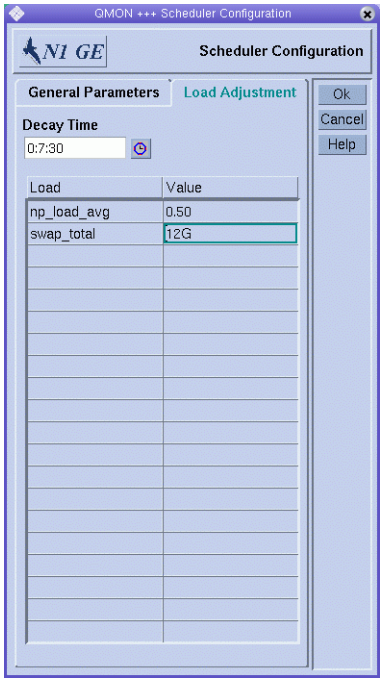
- **Algorithm**。スケジューリングアルゴリズムです。125 ページの「スケジューリングアルゴリズムの変更」を参照してください。
- **Schedule Interval**。スケジューラ実行の定期的な時間間隔です。
- **Reprioritize Interval**。ジョブを実行するための現在のチケット量に基づき、実行ホストに対してジョブの優先順位を再決定する定期的な時間間隔です。優先順位を再決定しない場合は、このパラメータをゼロに設定します。
- **Max Jobs/User**。1 人のユーザーおよび 1 つの UNIX グループにつき並行して実行可能なジョブの最大数です。128 ページの「ユーザー 1 人またはグループ 1 つあたりのジョブ数の制限」を参照してください。
- **Sort by**。負荷によるソート、または連続番号によるソートの、キューソート方法。127 ページの「連続番号によるキューの選択」を参照してください。
- **Job Scheduling Information**。`qstat -j` によりジョブスケジューリング情報へアクセス可能であるかどうか、またはジョブ ID 範囲に関してのみこの情報が収集されるかどうかを指定します。ジョブスケジューリング情報の一般的な収集は、保留中のジョブ数がきわめて多い場合にのみ一時的に使用してください。

  スケジューラ監視は、一部のジョブが振り分けられない理由を調べるのに役立ちます。ただし、すべてのジョブに関してこの情報を常に提供すると、リソースを消費することがあります。通常、このような情報は必要ありません。

- **Load Formula**。ホストとキューをソートするために使用する負荷の式です。
- **Flush Submit Seconds**。ジョブが発行されてから、スケジューラがトリガーされるまで、スケジューラが待機する秒数です。ジョブが発行されたあとのフラッシュを使用不可にするには、このパラメータをゼロに設定します。
- **Flush Finish Seconds**。ジョブが完了してから、スケジューラがトリガーされるまで、スケジューラが待機する秒数です。ジョブが完了したあとのフラッシュを使用不可にするには、このパラメータをゼロに設定します。
- **Maximum Reservation**。スケジューリング間隔内でスケジューリング可能なリソース予約の最大数です。120 ページの「リソース予約およびバックフィリング」を参照してください。
- **Params**。この設定を使用して、スケジューラに渡す追加のパラメータを指定します。Params は、`PROFILE` または `MONITOR` になります。`PROFILE` を指定した場合、スケジューラは各スケジューリング実行を要約したプロファイリング情報を記録します。`MONITOR` を指定した場合、スケジューラは各実行スケジューリングに関する情報を、ファイル *sge-root*/ *cell*/`common/schedule` に記録します。

デフォルトでは、Grid Engine システムは、固定スケジュール間隔でジョブ実行をスケジューリングします。「Flush Submit Seconds」および「Flush Finish Seconds」パラメータを使用すると、ただちにスケジュールを行うよう構成できます。詳細については、213 ページの「直接スケジューリング」を参照してください。

負荷調整パラメータを変更するには、「Load Adjustment」タブをクリックします。「Load Adjustment」タブは次の図のようになっています。



「Load Adjustment」タブを使用して、次のパラメータを設定します。

- **Decay Time**。負荷調整の減少時間です。
- 現在調整値が定義されているすべての負荷および消費可能属性を一覧表示する、負荷調整値のテーブルです。

  リストに負荷値を追加するには、「Load」または「Value」カラムヘッダーをクリックします。ホストに関連付けられているすべてのリソース属性とともに、選択リストが表示されます。

  図 1–2 に「Attribute Selection」ダイアログボックスを示します。「Consumable/Fixed Attributes」テーブルの「Load」カラムにリソース属性を追加するには、いずれかの属性を選択してから「OK」をクリックします。

  既存の値を変更するには、「Value」フィールドをダブルクリックします。

  リソース属性を削除するには、属性を選択してから Control + D キーを押すか、マウスボタン 3 をクリックします。削除を確認するダイアログボックスが表示されます。

内容説明については、126 ページの「システム負荷のスケーリング」を参照してください。スケジューラ構成の詳細については、`sched_conf`(5) のマニュアルページを参照してください。

ポリシーの管理

この節では、ポリシーを構成してクラスタリソースを管理する方法について説明します。

Grid Engine ソフトウェアは、管理者が管理する社内リソースポリシーに基づいて計算パワーの供給を調整します。システムでは、これらのポリシーを使用してグリッド内で使用可能なコンピュータリソースを調べます。システムは、グリッド全体でリソース使用率が最適化されるように、これらのリソースを収集し、リソースを自動的に割り当てて供給します。

グリッド内での協調を実現するため、プロジェクトの所有者は次の作業を実行する必要があります。

- ポリシーの折衝
- 固有のプロジェクト要件に対する、手動による優先のためのポリシーの柔軟性の確保
- 自動的なポリシーの監視および実施

管理者は、サイト用にカスタマイズされたハイレベルな使用ポリシーを定義できます。次のような 4 つのポリシーが使用可能です。

- 緊急度ポリシー – 135 ページの「緊急度ポリシーの構成」を参照
- 共有ベースポリシー – 141 ページの「共有ベースポリシーの構成」を参照
- 機能ポリシー – 154 ページの「機能ポリシーの構成」を参照
- 優先ポリシー – 159 ページの「優先ポリシーの構成」を参照

目標を達成するため、ポリシー管理はクラスタ内の共有リソースの使用を自動的に制御します。優先順位の高いジョブは優先的に振り分けられます。このようなジョブは、そのほかの低優先順位のジョブと競合した場合、より多くの CPU エンタイトルメントを受け取ります。Grid Engine ソフトウェアはすべてのジョブの進行状況を監視します。その状況に応じて、またポリシーに定義されている目標に従ってその相対的な優先順位を調整します。

このポリシーに基づくリソース割り当ては、各ユーザー、チーム、部署およびすべてのプロジェクトに、システムリソースの割り当て済みの配分を付与します。このリソースの割り当ては、1 週間、1 か月、または 1 四半期などの指定した期間で行われます。

QMON を使用したポリシーに基づくリソース管理の構成

「QMON Main Control」ウィンドウで、「Policy Configuration」ボタンをクリックします。「Policy Configuration」ダイアログボックスが表示されます。

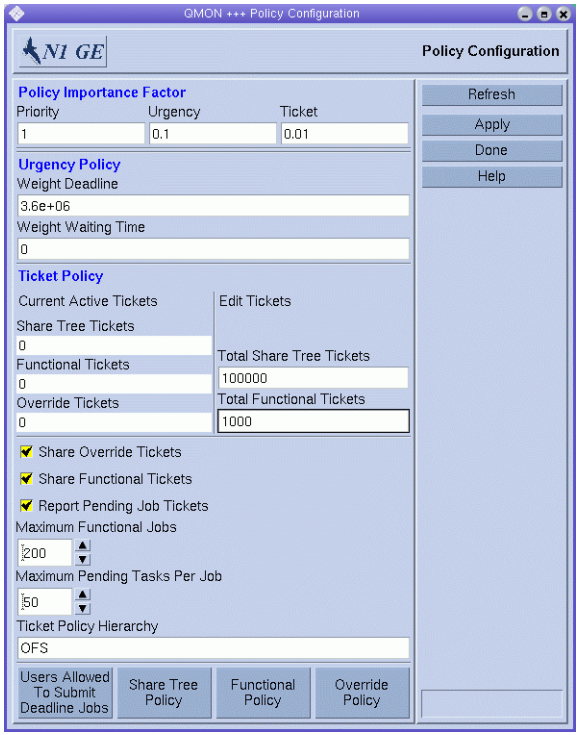


図 5–1 「Policy Configuration」ダイアログボックス

「Policy Configuration」ダイアログボックスには次の情報が表示されます。

- Policy Importance Factor
- Urgency Policy
- Ticket Policy。ポリシー関連のチケットを再調整できます。

このダイアログボックスから、3 つのチケットに基づくポリシー用の特定の構成ダイアログボックスにアクセスできます。

ポリシーの優先順位の指定

Grid Engine システムがジョブを振り分ける前に、ジョブは優先順位の高い順に、優先順位に組み込まれます。管理者の介入がなければ、この順番は FIFO (先入れ先出し) 方式です。

「Policy Configuration」ダイアログボックスでは、「Policy Importance Factor」で、ジョブのソート順を制御する、3 つの優先順位の種類の相対的な重要性を指定できます。

- **Priority**。POSIX 優先順位とも呼びます。`qsub` コマンドの `–p` オプションで、サイト固有の優先順位ポリシーを指定します。
- **Urgency Policy**。ジョブには、その相対的な重要性を決定する緊急度の値を割り当てることができます。保留中のジョブは、緊急度の値に従ってソートされます。
- **Ticket Policy**。ジョブは常に、ジョブが有するチケットの数により定義される相対的な重要性にしたがって扱われます。保留中のジョブはチケット順にソートされます。

ジョブの優先順位の詳細については、118 ページの「ジョブのソート」を参照してください。

各優先順位の種類に対しては、重み付け係数を指定できます。この重み付け係数は、各種類の優先順位がジョブ全般の優先順位に影響を与える程度を決定します。各優先順位の種類の値の範囲を制御しやすくするため、生のチケットの値、緊急度の値、および POSIX 優先順位の値の代わりに、正規化された値が使用されます。

ジョブの優先順位値がどのように決定されるかは、次の式で表されます。

```
ジョブ優先順位 = 緊急度 * 正規化された緊急度の値 +
チケット * 正規化されたチケットの値 +
優先順位 * 正規化された優先順位の値
```

「Priority」、「Ticket 」、および「Priority」は、「Policy Importance Factor」でユーザーが指定する 3 つの重み付け係数です。たとえば、「Priority」に 1、「Urgency」に 0.1、および「Ticket 」に 0.01 を指定した場合、`qsub –p` コマンドにより指定されるジョブの優先順位には最大の重みが与えられ、「Urgency Policy」により指定されるジョブの優先順位は次に考慮され、「Ticket Policy」により指定されるジョブの優先順位はもっとも低い重みが与えられます。

緊急度ポリシーの構成

「Urgency Policy」では各ジョブの緊急度の値を定義します。緊急度の値は、次の 3 つの関係要素の合計によって決定されます。

- リソース要求。コンプレックスで定義されている各リソース属性は、緊急度の値を取ることができます。リソース属性の緊急度の値の設定方法の詳細については、72 ページの「QMON を使用したコンプレックスリソース属性の構成」を参照してください。リソース属性に対する各ジョブ要求により、合計に対して、その属性の緊急度の値が追加されます。
- 締め切り優先。 締め切り優先ジョブの緊急度の値は、「Policy Configuration」ダイアログボックスで指定された「Weight Deadline」を、`qsub –dl` コマンドにより指定されたそのジョブの締め切り開始時間までの秒単位の空き時間で除算することで決定されます。
- 待機時間。 ジョブの待機時間に関する緊急度の値は、ジョブの待機時間と、「Policy Configuration」ダイアログボックスで指定されている「Weight Waiting Time」の積を計算することで決定されます。ジョブの待機時間は秒単位で測定されます。

Grid Engine システムが緊急度の値の合計をどのように算出するかの詳細については、120 ページの「緊急度ポリシーについて」を参照してください。

チケットに基づくポリシーの構成

個別のポリシーに現在割り当てられているチケットは、「Current Active Tickets」の下に表示されます。その数字は、ポリシーの相対的な重要性を反映し、特定のポリシーが現時点でクラスタを優先使用しているかどうか、あるいはポリシー間のバランスが取られているかどうかを示します。

チケットは量的な目安を提供します。たとえば、共有ベースポリシーへのチケット割り当て量が、機能ポリシーへの割り当て量の 2 倍の場合、共有ベースポリシーには機能ポリシーの 2 倍のリソースエンタイトルメントがあることを意味します。この意味では、チケットは企業の株式に非常によく似ています。

総チケット数に特別な意味はありません。重要なのはポリシー間の関係だけです。つまり、ポリシーの相対的な重要性を細かく調整できるよう、総チケット数は通常かなり大きな数字になります。

「Edit Tickets」では、共有ツリーポリシーと機能ポリシーに割り当てられたチケットの数を変更できます。詳細については、136 ページの「チケットの編集」を参照してください。

優先ポリシーにより分配される合計チケット量を制御するには、「Share Override Tickets」チェックボックスを選択します。そのほかのポリシーと優先カテゴリに使用可能なチケットプールを基準として、個別ジョブの重要性を制御するには、チェックボックスをクリアします。詳細については、137 ページの「優先チケットの共有」を参照してください。

そのすべてのジョブの合計に関して一定のエンタイトルメントレベルをカテゴリメンバーに与えるには、「Share Functional Tickets」チェックボックスを選択します。カテゴリメンバーのエンタイトルメントに基づいて、各ジョブに同じエンタイトルメントレベルを付与するには、チェックボックスをクリアします。詳細については、137 ページの「機能チケットの分配の共有」を参照してください。

機能ポリシーでスケジューリング可能なジョブの最大数を設定できます。デフォルト値は 200 です。

各配列ジョブに認められている保留中のサブタスクの最大数を設定できます。デフォルト値は 50 です。スケジューリングのオーバーヘッドを軽減するにはこの設定を使用します。

ポリシーが衝突するある種のケースを解決するには、「Ticket Policy Hierarchy」を指定できます。ポリシー間の衝突の解決は、特に保留中のジョブに関係します。詳細については、139 ページの「チケットポリシー階層の設定」を参照してください。

表示されている情報を更新するには、「Refresh」をクリックします。

「Policy Configuration」に対して行なった変更を保存するには、「Apply」をクリックします。変更を保存することなくダイアログボックスを閉じるには、「Done」をクリックします。

チケットの編集

共有ツリーチケットと機能チケットの合計数を編集できます。優先チケットは、優先ポリシーの構成を介して直接割り当てられます。そのほかのチケットプールは、ポリシーに関連付けられているジョブ間で実際のポリシー構成に基づいて自動的に分配されます。

注 – すべての共有ベースチケットと機能チケットは、これらのポリシーに関連付けられているジョブに分配されます。優先チケットは、現在アクティブなジョブには適用できない場合があります。その結果、優先ポリシーに定義済みのチケットがある場合であっても、アクティブな優先チケットはゼロになる場合があります。

優先チケットの共有

管理者は、優先カテゴリのさまざまなメンバー、つまり個別ユーザー、プロジェクト、部署、またはジョブにチケットを割り当てます。その結果、カテゴリメンバーに割り当てられたチケットの数が、そのカテゴリメンバーの下でジョブに割り当てられるチケットの数を決定します。たとえば、ユーザー A に割り当てられたチケットの数は、ユーザー A のすべてのジョブに割り当てられるチケットの数を決定します。

注 – ジョブカテゴリに割り当てられたチケットの数は、そのカテゴリのジョブに割り当てられるチケットの数を決定しません。

`sched_conf`(5) の `share_override_tickets` パラメータを設定するには、「Share Override Tickets」チェックボックスを使用します。このパラメータは、カテゴリメンバーのチケット値からどのようにジョブチケット値が得られるかを制御します。「Share Override Tickets」チェックボックスを選択すると、カテゴリメンバーのチケットは、このメンバーの下でジョブに均等に分配されます。「Share Override Tickets」チェックボックスをクリアすると、各ジョブは、カテゴリメンバーに対して定義されているチケット量を継承します。すなわち、カテゴリメンバーのチケットが、そのジョブのすべてのジョブに繰り返されます。

優先ポリシーにより分配される合計チケット量を制御するには、「Share Override Tickets」チェックボックスを選択します。この設定を使用すると、多くのジョブが 1 つのカテゴリメンバーの支配下にある場合、ジョブに割り当てられるチケットの量は無視できるほど少なくなります。たとえば、多くのジョブがユーザーカテゴリの 1 メンバーに属する場合、チケット量は減少する場合があります。

そのほかのポリシーと優先カテゴリに使用可能なチケットプールを基準とした各ジョブの重要性を制御するには、「Share Override Tickets」チェックボックスをクリアします。この設定を使用すると、カテゴリメンバーの支配下にあるジョブの数は重要ではなくなります。そのジョブは常に同じ数のチケットを取得します。ただし、優先チケットを受け取る権利を持つジョブの数が大きくなるにつれ、システム内の優先チケットの合計数は大きくなります。このような場合、そのほかのポリシーは 重要性を喪失することがあります。

機能チケットの分配の共有

機能ポリシーは、機能カテゴリのエンタイトルメントの分配を定義します。さらにこのポリシーは、これらの各カテゴリのすべてのメンバーに関する分配を定義します。そのため、機能ポリシーは、2 レベルの共有ツリーに類似しています。異なる点は、ジョブは複数のカテゴリに同時に関連付けることができる、という点です。たとえば、ジョブは特定のユーザーに属するだけでなく、ジョブはプロジェクト、部署、およびジョブクラスに属することも可能です。

ただし、共有ツリーと同様に、機能カテゴリからジョブが受けるエンタイトルメントの配分は、次の要素によって決まります。

- 対応するカテゴリメンバー (たとえばそのプロジェクト) に対して定義されている配分
- カテゴリ (ユーザー、部署などではなく、プロジェクト) に与えられている配分

`sched_conf`(5) の `share_functional_shares` パラメータを設定するには、「Share Functional Tickets」チェックボックスを使用します。このパラメータは、ジョブの配分の決定にカテゴリメンバーの配分をどのように使用するかを定義します。特定のユーザーやプロジェクトなどのカテゴリメンバーに割り当てられた配分をすべてのジョブに繰り返すことも、カテゴリメンバーのジョブの間で配分を振り分けることもできます。

- 「Share Functional Tickets」チェックボックスを選択すると、機能配分がジョブの間で繰り返されることを意味します。
- 「Share Functional Tickets」チェックボックスをクリアすると、機能配分がジョブの間で分配されることを意味します。

こうした配分は株式に例えることができます。このような配分は、同じカテゴリメンバーに属するジョブには何の意味もありません。どちらの場合も、同じカテゴリメンバーのすべてのジョブは同じ数の配分を受けます。しかし、同じカテゴリ内の配分量の比較では、配分数は意味を持ちます。「Share Functional Tickets」チェックボックスを選択すると、同じカテゴリメンバーに属する多くの兄弟を持つジョブが受け取る配分は比較的少なくなります。一方、「Share Functional Tickets」チェックボックスをクリアすると、すべての兄弟ジョブは、そのカテゴリーメンバーと同じ分配量を受け取ります。

そのすべてのジョブの合計に関して一定のエンタイトルメントレベルをカテゴリメンバーに与えるには、「Share Functional Tickets」チェックボックスを選択します。ただし、ジョブに多くの兄弟が存在する場合、各ジョブのエンタイトルメントは、無視できるほど小さくなることがあります。

カテゴリメンバーのエンタイトルメントに基づいて、各ジョブに同じエンタイトルメントレベルを与えるには、「Share Functional Tickets」チェックボックスをクリアします。システム内のジョブの兄弟の数は問題にはなりません。

注 – 多数のジョブを抱えるカテゴリメンバーは、機能ポリシーを優先使用する可能性があります。

---

share functional shares の設定は、分配される機能チケットの合計数を決定しないことに注意してください。合計数は常に、機能ポリシーのチケットプールに対して管理者により定義されたものです。share functional shares パラメータは、単に機能ポリシー内での機能チケットの分配方法に影響するだけです。

例 5–1 機能ポリシーの例

この例は、share override 機能ポリシーのチケット機能を理解することなく、SGE-5.3 スケジューラのオプション -user_sort true を N1GE 6.1 構成に変換しようとする場合の一般的なシナリオを表しています。

ユーザーに基づく単純な均等割当では、次のパラメータを使用してグローバル構成 sge_conf(5) を設定します。

```
-enforce_user auto
-auto_user_fshare 100
```

次に、スケジューラ構成 sched_conf(5) で -weight_tickets_functional 10000 を使用します。この操作によって、ユーザーごとに 100 の配分がスケジューリングされた、ユーザーに基づく均等割当に対して、機能ポリシーが使用されます。

スケジューリング実行時間のチューニング

保留中のジョブは、118 ページの「ジョブのソート」で説明されているように、各ジョブが持つチケットの数に従ってソートされます。スケジューラは、保留中の各ジョブが持つチケットの数を、マスターデーモン sge_qmaster に報告します。ただし、非常に多くの数のジョブを抱えるシステムでは、チケットの報告をオフにしたい場合があります。チケットの報告をオフにすると、チケットに基づくジョブ優先順位が使用不可になります。ジョブのソート順序は、各ジョブが発行される時間にのみ基づいて行われます。

保留中のジョブチケットによる sge_qmaster への報告をオフにするには、「Policy Configuration」ダイアログボックスの「Report Pending Job Tickets」チェックボックスをクリアします。これにより、 sched_conf(5) の report_pjob_tickets パラメータが false に設定されます。

チケットポリシー階層の設定

チケットポリシー階層は、衝突するチケットポリシーの一部のケースを解決する手段を提供します。チケットポリシー間の衝突の解決は、特に保留中のジョブに関係します。

そのようなケースは、共有ベースポリシーおよび機能ポリシーと結び付いて発生する場合があります。どちらのポリシーでも、同じリーフレベルのエンティティーに属するジョブへの優先順位への割り当ては、先着順で行われます。リーフレベルのエンティティーには、次のものが該当します。

- 共有ツリーのユーザーリーフ
- 共有ツリーのプロジェクトリーフ

- 機能ポリシーの、ユーザー、プロジェクト、部署、またはキューなどのカテゴリのすべてのメンバー

ジョブカテゴリーのメンバーは、リーフレベルのエンティティーには含まれません。そのため、同じユーザーの最初のジョブが最大量を受け取り、2 番目のジョブが 2 番目に多い量を受け取り、3 番目がそれに続く、となります。

別のポリシーが異なる順序を指定した場合に、衝突が発生する可能性があります。そのため、たとえば、優先ポリシーが 3 番目のジョブをもっとも重要なものとして定義し、発行される最初のジョブが最後に来る場合があります。

ポリシー階層は、共有ツリーポリシーや機能ポリシーよりも、優先ポリシーにより高い優先順位を与える場合があります。このようなポリシー階層により、優先ポリシーの高優先順位ジョブが、ほかの 2 つのポリシーのジョブよりも多くのエンタイトルメントを獲得します。このようなジョブは、共有ツリーで同じリーフレベルのエンティティー (ユーザーまたはプロジェクト) に属する必要があります。

チケットポリシー階層は、3 文字までの組み合わせから構成できます。これらの文字は、次の 3 つのチケットポリシーの名前の最初の文字です。

- S – Share-based (共有ベース)
- F – Functional (機能)
- O – Override (優先)

これらの文字を使用してチケットポリシーの階層を作成します。最初の文字で、最上位のポリシーを定義します。最後の文字で、階層の最下位を定義します。ポリシー階層に存在しないポリシーは、階層に影響を与えません。ただし、階層に存在しないポリシーも、ジョブのチケットのソースとなる場合があります。ただし、これらのチケットはそのほかのポリシーにおけるチケット計算には影響しません。各ジョブが全エンタイトルメントを定義できるように、すべてのポリシーのすべてのチケットは合計されます。

次の例では、2 つの設定と、これらの設定が保留中のジョブの順序にどのように影響するかを説明します。

```
policy_hierarchy=OS
```

1. 優先ポリシーは、保留中の各ジョブに適切な数のチケットを割り当てます。
2. 2 つのジョブが同じユーザーまたは同じリーフレベルのプロジェクトに属する場合、チケットの数は共有ツリーでのエンタイトルメントの割り当てを決定します。続いて共有ツリーのチケットは、保留中のジョブに対して計算されます。
3. 優先ポリシー、および共有ツリーポリシーからのチケットは、階層内にないそのほかのアクティブなポリシーとともに合算されます。結果のチケット数が最高であるジョブは、最高のエンタイトルメントを獲得します。

```
policy_hierarchy=OF
```

1. 優先ポリシーは、保留中の各ジョブに適切な数のチケットを割り当てます。続いて優先ポリシーからのチケットが合算されます。
2. 2 つのジョブが同じ機能カテゴリメンバーに属している場合、結果のチケット数は機能ポリシーのエンタイトルメントの割り当てに影響します。このエンタイトルメントの割り当てに基づいて、保留中のジョブに対する機能チケットが計算されます。
3. 結果の値は、優先ポリシーのチケット量に加算されます。結果のチケット数が最高であるジョブは、最高のエンタイトルメントを獲得します。

3 つの文字は理論上どのように組み合わせることもできますが、意味がある、あるいは現実的に妥当なのは一部の組み合わせだけです。最後の文字は常に S または F にする必要があります。これは、上記の例で説明した特徴があるため、影響を受ける可能性があるポリシーは、この 2 つしかないためです。

`policy_hierarchy` の設定には次の形式を推奨します。

```
[O][S|F]
```

優先ポリシーが存在する場合、影響を与える可能性があるポリシーは優先ポリシーのみであるため、O は最初の文字としてのみ使用する必要があります。影響を受ける可能性があるポリシーは、共有ベースポリシーと機能ポリシーのみです。そのため、S または F を最後の文字として使用する必要があります。

共有ベースポリシーの構成

共有ベーススケジューリングは、週、月、四半期などの累積期間中に、各ユーザーおよびプロジェクトに、システムリソースの割り当て済み配分を付与します。共有ベーススケジューリングは、共有ツリースケジューリングとも呼ばれます。このスケジューリングでは常に、次のスケジューリング間隔までの短い時間の間に、各ユーザーおよびプロジェクトに予定されているリソース配分が調整されます。共有ベーススケジューリングは、ユーザーやプロジェクト、またはその両方に対して定義します。

共有ベーススケジューリングにより、時間の経過とともに、共有ツリーで構成されているインスタンスに対し、定義済みの配分が保証されるようになります。システムがジョブを振り分ける場合には、過去のリソース消費量が予想より少なかった共有ツリーのブランチに関連付けられているジョブが優先されます。同時に、そのほかの共有ツリーのブランチに関連付けられている保留中のジョブに、未使用の配分比率が依然として使用可能であるため、完全なリソース使用率が保証されます。

各ユーザーまたはプロジェクトにできるかぎり目標に近い配分を付与することによって、ユーザーまたはプロジェクトのグループも目標の配分を得られるようにな

ります。部署や部門は、そのようなグループの例です。累積期間中にリソースエンタイトルメントを持つすべてのエンティティーがリソースを得ようとした場合にのみ、すべてのエンティティーに対する公平な配分を達成できます。ユーザー、プロジェクト、またはグループが特定の期間中にジョブを発行しなかった場合、リソースはジョブの発行者の間で分配されます。

共有ベーススケジューリングはフィードバック方式です。任意のユーザー/ユーザーグループ、またはプロジェクト/プロジェクトグループにエンタイトルメントが与えられている対象のシステムの配分は、構成パラメータです。ジョブにエンタイトルメントが与えられている対象のシステムの配分は、次の要素に基づいて決定されます。

- ジョブのユーザーまたはプロジェクトに割り当てられている配分
- 各ユーザー/ユーザーグループ、および各プロジェクト/プロジェクトグループの累積された過去の使用率。この使用率は減少係数で調整されます。「以前」の使用ほど影響は小さくなります。

Grid Engine ソフトウェアは、ユーザーおよびプロジェクトがすでに受け取った使用率を追跡します。スケジューリング間隔のたびに、スケジューラはすべてのジョブのリソース配分を調整します。このようにして、すべてのユーザー、ユーザーグループ、プロジェクト、プロジェクトグループが、累積期間の間にできるかぎりシステムの公平な配分を受けられるようにします。言い替えれば、リソース使用率を許可したり、拒否したりすることによって、誰もがほぼ目標に近い配分を受けられるようにします。

半減期係数

半減期とは、システムがユーザーのリソース消費を「忘れる」速さです。システム管理者は、それが 6 か月前であれ 6 日前であれ、ユーザーの大きなリソース消費にペナルティーを科すかどうかを決定できます。また管理者は、どのようにしてペナルティーを科すかも決定できます。Grid Engine ソフトウェアは、共有ツリーのすべてのノードについてユーザーのリソース消費記録を維持します。

共有ベースポリシーを設定する際、システム管理者はこの記録に基づいて、ユーザーの過少利用または過大利用を判断する際、どのくらい過去にさかのぼるかを決定できます。この意味でのリソース利用量は、「スライドする時間枠」で消費されたすべてのコンピュータリソースの数学的な合計です。

この時間枠の長さは「半減期」係数で決まり、Grid Engine システムではそれは 内部減少関数です。この減少関数は、経時的に発生するリソース消費の影響を小さくします。半減期が短いほど、リソースの過大消費の影響は短期間に小さくなり、半減期が長いほど、リソースの過大消費の影響は徐々に小さくなります。

この半減期減少関数は指定された時間単位に基づきます。たとえば、1,000 単位のリソース消費に 7 日の半減期を適用した場合を考えます。この半減期減少係数を使用した場合、経時で次の使用「ペナルティー」による調整が行われます。

- 7 日後 500
- 14 日後 250
- 21 日後 125
- 28 日後 62.5

半減期に基づく減少は、ペナルティーの効果が無視できるまで、経時のユーザーのリソース消費の影響を小さくします。

注 – 優先チケットは別のポリシーシステムに属するため、ユーザーが受け取る優先チケットは、過去の使用ペナルティーの影響を受けません。減少関数は、共有ツリーポリシーのみの特性です。

補正係数

比較では、場合によって、実際の利用量が目標使用率を大きく下回っていることが明らかになることがあります。このような場合は、ユーザーのリソースの配分またはプロジェクトのリソースの配分を調整することで、ユーザーがシステムを優先使用できます。このような調整は、目標の配分に到達するという目的に基づいています。ただし、この優先使用は望ましくない場合があります。

補正係数を使用すると、管理者は、ユーザーまたはプロジェクトが短期にリソースを優先使用できる度合いを制限できます。

たとえば補正係数 2 は、ユーザーまたはプロジェクトの現在の配分を目標配分の 2 倍に制限します。ユーザーまたはプロジェクトが、累積期間中にシステムリソースの 20% を受けられるようになっていると仮定すると、ユーザーまたはプロジェクトが現在受けている量がそれより非常に低い場合、短期に受け取れる最大量は、40% だけになります。

共有ベースポリシーは、共有ツリーに従ってユーザーまたはプロジェクトの長期のリソースエンタイトルメントを定義します。共有ベースポリシーと補正係数を組み合わせることによって、エンタイトルメントの自動的な調整が行われます。

ユーザーまたはプロジェクトが、定義された目標のエンタイトルメントを下回っているか上回っている場合、Grid Engine システムは補正を行います。システムは、ユーザーまたはプロジェクトの短期のエンタイトルメントを長期目標よりも上げるか、下げることによって補正をします。この補正は、共有ツリーアルゴリズムにより計算されます。

補正係数は、Grid Engine システムが割り当てる補正量を制御するもう 1 つの仕組みを提供します。この追加の補正係数 (CF) 計算は、次の条件が真である場合にのみ行われます。

- 短期エンタイトルメントが、長期エンタイトルメントと CF の積よりも大きい
- CF が 0 よりも大きい

上記の条件の一方または両方が真ではない場合は、共有ツリーアルゴリズムによって定義および実装されている補正が適用されます。

CF 値が小さいほど、その効果は大きくなります。値が 1 より大きい場合、Grid Engine システムの補正は限られた効果しかありません。補正の上限は、長期エンタイトルメントと CF の積で計算されます。上記で定義されているように、補正係数に基づく操作が行われるには、短期エンタイトルメントがこの上限を超えている必要があります。

CF が 1 の場合、Grid Engine システムはそのままの共有ツリーアルゴリズムと同じ方法で補正をします。このため、値 1 は値 0 と似た効果になります。唯一の違いは、実装の詳細です。CF = 1 の場合は、CF 計算が行われますが、影響はありません。CF = 0 の場合は、CF 計算が抑止されます。

値が 1 より小さい場合、Grid Engine システムは過剰補正を行います。ジョブは、共有ツリーアルゴリズムに基づいて得られるよりも大幅に多くの補正量を受けます。また、補正の実施条件が低い短期エンタイトルメント値で満たされるため、ジョブは早期にこの過剰補正を受けることになります。実施条件は「短期エンタイトルメント > 長期エンタイトルメント * CF」です。

階層形式の共有ツリー

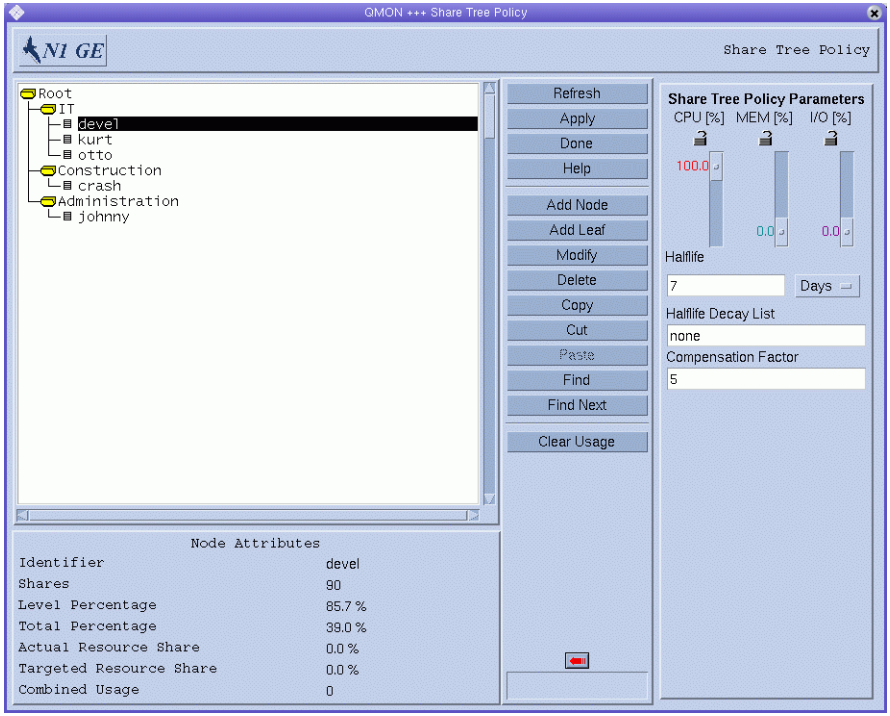
共有ベースポリシーは、共有ツリーを使用して実現されます。共有ツリーは、移動累積期間中にすべてのユーザーおよびプロジェクトの間でどのようにシステムリソースを配分するかを規定します。累積期間の長さは、構成可能な減少定数で決まります。Grid Engine システムは、共有ツリー内の各親ノードのその累積上限への到達度に基づいて、ジョブの共有エンタイトルメントを決定します。ジョブの共有エンタイトルメントは、そのリーフノードの配分割り当てに基づいて決まり、リーフノードの配分割り当てはその親ノードの割り当てに依存します。リーフノードに関連付けられているすべてのジョブ間で、関連付けられている配分が分配されます。

ジョブの最終的なエンタイトルメントは、共有ツリーから得られるエンタイトルメントと、機能ポリシーなどから得られるほかのエンタイトルメントを組み合わせることによって決定されます。共有ツリーには、共有ベーススケジューリングに対するチケットの合計数が割り当てられます。この数によって、4 通りあるスケジューリングポリシーにおける共有ベーススケジューリングの重みが決まります。

共有ツリーは、インストール中に定義し、いつでも変更することができます。共有ツリーを編集すると、次のスケジューリング間隔で新しい共有割り当てが有効になります。

QMON を使用した共有ツリーポリシーの構成

「QMON Policy Configuration」ダイアログボックス (図 5–1) で「Share Tree Policy」をクリックします。「Share Tree Policy」ダイアログボックスが表示されます。



「Node Attributes」

「Node Attributes」の下には、選択したノードの属性が表示されます。

- **Identifier**。ユーザー、プロジェクト、またはその集団の名前です。
- **Shares**。このユーザーまたはプロジェクトに割り当てられている配分数です。

注 – 配分は相対的な重要性を定義します。百分率値ではありません。また、数量的な意味もありません。大きな数字を使用すると重要性の関係を細かく調整できるため、一般には、百または千台の数字で配分を指定することを推奨します。

- **Level Percentage**。 このツリーの同じ親ノードレベルでの配分合計に対して、このノードが占める割合です。その兄弟の配分の合計で除算した、このノードの配分の数です。
- **Total Percentage**。共有ツリー全体での合計配分に対するこのノードの割合です。ノードの長期的な目標リソース配分です。
- **Actual Resource Usage**。 累積期間中にこのノードがこれまでに消費した、システムの全リソースに対する割合です。百分率は、共有ツリーのすべてのノードに対して表されたものです。
- **Targeted Resource Usage**。「Actual Resource Usage」と同じですが、共有ツリーで現在アクティブなノードだけが考慮されます。アクティブなノードとはシステムにジョブがあるノードです。短期には、Grid Engine システムはアクティブなノードの間でエンタイトルメントのバランスをとろうとします。
- **Combined Usage**。このノードの使用率の合計です。「Combined Usage」はこのノードで累積された使用率の合計です。リーフノードは、その下で実行されるすべてのジョブの使用率を累積します。内部ノードは、すべての子孫ノードの使用率を累積します。「Combined Usage」には「Share Tree Policy Parameters」で指定された割合に従い、CPU 、メモリー、および I/O 使用率が含まれます。「Combined Usage」はパラメータによって指定された半減期減少率で減少します。

ユーザーノードまたはプロジェクトノードを削除して、追加して戻しても、そのユーザーまたはプロジェクトの使用率は残ります。ノードは、共有ツリー内の同じ場所または別の場所に追加して戻すことができます。ノードを共有ツリーに追加して戻す前に、その使用率をゼロにすることができます。このためには、まず Grid Engine システムで構成したユーザーまたはプロジェクトからノードを削除します。続いて、その場所のユーザーまたはプロジェクトにノードを追加して戻します。

共有ツリーに存在しなかったにもかかわらず、ジョブを実行したユーザーまたはプロジェクトを、共有ツリーに追加すると、その使用率はゼロ以外の値になります。そのようなユーザーまたはプロジェクトをツリーに追加する際に、使用率をゼロにするには、Grid Engine システムで構成したユーザーまたはプロジェクトからそのユーザーまたはプロジェクトを削除します。続いて、そのユーザーまたはプロジェクトをツリーに追加します。

選択したノードの下に内部ノードを追加するには、「Add Node」をクリックします。空の「Node Info」ウィンドウが表示され、このウィンドウにノードの名前と配分の数を入力できます。任意のノード名または配分の数を入力できます。

選択したノードの下にリーフノードを追加するには、「Add Leaf」をクリックします。空の「Node Info」ウィンドウが表示され、このウィンドウにノードの名前と配分の数を入力できます。ノードの名前は、既存の Grid Engine ユーザー (104 ページの「QMON を使用したユーザーオブジェクトの構成」) またはプロジェクト (106 ページの「プロジェクトの定義」) でなければなりません。

リーフノードを追加する際には、次の規則が適用されます。

- 共有ツリー内のすべてのノードが固有のパスを持つこと。
- 共有ツリーで同じプロジェクトを複数回参照しないこと。
- プロジェクトサブツリーに同じユーザーが 1 回しか現れないこと。
- プロジェクトサブツリーの外部に同じユーザーが 1 回しか現れないこと。
- ユーザーがリーフ以外のノードに現れないこと。
- プロジェクトサブツリー内のあらゆるリーフノードが既知のユーザーか予約名の `default` を参照すること。この特殊なユーザーについての詳細は、148 ページの「特殊ユーザー `default` について」を参照してください。
- プロジェクトサブツリー内にサブプロジェクトを含まないこと。
- プロジェクトサブツリー内に存在しないあらゆるリーフノードが既知のユーザーかプロジェクトを参照すること。
- プロジェクトサブツリー内のあらゆるユーザーリーフノードにプロジェクトへのアクセス権があること。

選択したノードを編集するには、「Modify」をクリックします。「Node Info」ウィンドウが表示されます。ウィンドウにはモードの名前と、配分の数が表示されます。

選択したノードのカットまたはバッファーへのコピーを行うには、「Cut」または「Copy」をクリックします。直前にカットまたはコピーしたノードの内容を、選択したノードの下にペーストするには、「Paste」をクリックします。

選択したノードとそのすべての子孫を削除するには、「Delete」をクリックします。

共有ツリー階層全体をクリアするには、「Clear Usage」をクリックします。共有ベースポリシーを予算に合わせて、各予算編成時期に最初からやり直す必要がある場合に、階層をクリアします。「Clear Usage」機能は、N1 Grid Engine 6.1 ソフトウェアのテスト環境を設定したり、変更したりする際にも便利です。

`QMON` は、「Share Tree Policy」ダイアログボックスに表示される情報を定期的に更新します。ただちに表示を更新させるには、「Refresh」をクリックします。

行なったすべてのノードの変更を保存するには、「Apply」をクリックします。変更を保存することなくダイアログボックスを閉じるには、「Done」をクリックします。

共有ツリーでノード名を検索するには「Find」をクリックしてから検索文字列を入力します。検索文字列は大文字と小文字が区別され、その文字列から始まるノード名が表示されます。次に一致する検索文字列を検索するには、「Find Next」をクリックします。

オンラインヘルプシステムを開くには、「Help」をクリックします。

共有ツリーポリシーのパラメータ

「Share Tree Policy Parameters」を表示するには、「Node Attributes」の右側にある矢印をクリックします。

- **CPU [%]** スライダ — このスライダの設定は、CPU が使用合計に占める割合を示します。このスライダを動かすと、CPU の割合の変化を補正するため、MEM および I/O スライダも変化します。
- **MEM [%]** スライダ — このスライダの設定は、メモリーが使用合計に占める割合を示します。このスライダを動かすと、MEM の割合の変化を補正するため、CPU および I/O スライダも変化します。
- **I/O [%] slider** — このスライダの設定は、I/O が使用合計に占める割合を示します。このスライダを動かすと、I/O の割合の変化を補正するため、CPU および MEM スライダも変化します。

注 – CPU [%]、MEM [%]、および I/O [%] の合計は常に 100% になります。

- 錠のシンボル — 錠が開いていると、錠が保護するスライダが自由に動くことができます。直接操作されることによって動くことも、別のスライダが動かされたため、それを補正するために動くこともあります。

  錠が閉じていると、錠が保護するスライダは動きません。2 つの錠が閉じていて 1 つが開いている場合は、どのスライダも動かせません。
- 「**Half-life**」 — このフィールドを使用して、リソース使用に対する半減期を指定します。使用率は、スケジューリング間隔のたびに減少し、累積利用量に関わるリソース使用があると、半減期の経過後にその値が半分になります。
- 「**Days/Hours**」選択メニュー — 半減期の測定単位を日数または時間数のどちらにするか選択します。
- 「**Compensation Factor**」 — このフィールドは、正の整数値の補正係数を受け付けます。適切な値は 2 から 10 の範囲です。

  ユーザーまたはプロジェクトの実際の使用率が、目標使用率を大きく下回っていることがあります。補正係数は、このようなユーザーまたはプロジェクトが、はじめてリソースを取得するときにリソースを優先使用するのを防ぎます。詳細については、143 ページの「補正係数」を参照してください。

特殊ユーザー `default` について

多数のユーザーがいるサイトでは、特殊なユーザー `default` を使用して、共有ツリーを保守する作業量を減らすことができます。共有ツリーポリシーにおいては、

ジョブの優先順位は、ジョブが共有ツリー内で対応するノードに基づいて決定されます。共有ツリー内で明示的に名前が付けられていないユーザーは、(存在する場合) `default` ノードに割り当てられます。

1 つの `default` ノードを指定することで、シンプルな共有ツリーの作成が可能になります。このような共有ツリーにより、ユーザーに基づく公正な分配が可能になります。

また、大部分のユーザーに同じ共有エンタイトルメントが割り当てられている場合でも、`default` ユーザーを使用できます。同じ共有エンタイトルメントは、均等配分スケジューリングとも呼ばれます。

`default` ユーザーは、`default` ノードの下ですべてのユーザーエントリを構成し、各ユーザーに同じ配分量を与えます。ジョブを発行する各ユーザーは、`default` ユーザーに対して構成されているのと同じ共有エンタイトルメントを受け取ります。特定のユーザーに対してこの機能を有効にするには、Grid Engine ユーザーのリストにこのユーザーを追加する必要があります。

共有ツリーでは、`default` ノードに割り当てられているすべてのユーザーに関して、「仮想」ノードが表示されます。仮想ノードを表示することで、`default` ノードに割り当てられているユーザーに関して、使用率と、公正な配分スケジューリングのパラメータを調べることができます。

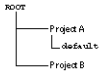
また、「ハイブリッド」の共有ツリーに対して `default` ユーザーを使用することもできます。このツリーでは、ユーザーは共有ツリー内のプロジェクトの従属下に置かれます。`default` ユーザーは、プロジェクトノードのリーフノードになることができます。

ユーザーの短期エンタイトルメントは、ユーザーが消費するリソース量の違いにより異なります。ただし、ユーザーの長期エンタイトルメントは同じままになります。

一部のユーザーにだけ低い、または高いエンタイトルメントを割り当てて、ほかのすべてのユーザーには同じ長期エンタイトルメントを維持したい場合があります。このためには、特別なエンタイトルメントを持つのユーザー用の `default` ユーザーの隣に、個別のユーザーエントリを含む共有ツリーを構成します。

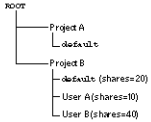
例 A では、プロジェクト A に発行するすべてのユーザーが同じ長期エンタイトルメントを得ます。プロジェクト B に発行するユーザーは、プロジェクト B の累積リソース消費量に関係します。プロジェクト B のユーザーのエンタイトルメントは管理されません。

例 5–2 例 A



例 A と例 B を比較してください。

例 5–3　例 B



例 B では、プロジェクト A に対する扱いは、例 A のときと同じです。しかし、ユーザー A および B を除き、プロジェクト B にジョブを発行するすべてのデフォルトユーザーは、同等の長期のリソースエンタイトルメントを得ます。デフォルトのユーザーは 20 の配分を持ちます。ユーザー A の配分は 10 で、受けるエンタイトルメントはデフォルトユーザーの半分です。ユーザー B の配分は 40 で、受けるエンタイトルメントはデフォルトユーザーの 2 倍です。

コマンド行からの共有ベースポリシーの構成

注 – 階層ツリーはグラフィカルな表示と編集に適しているため、QMON を使用して共有ツリーポリシーを構成します。ただし、共有ツリーの変更をシェルスクリプトで統合する必要がある場合は、qconf コマンドとそのオプションなどを使用できます。

コマンド行から共有ポリシーを構成するには、適切なオプションを使用して qconf コマンドを使用します。

- qconf のオプションである -astree、-mstree、-dstree、および -sstree を使用すると、次の操作を実行できます。
  - 新しい共有ツリーの追加
  - 既存の共有ツリーの変更
  - 共有ツリーの削除
  - 共有ツリーの構成の表示

  これらのオプションの詳細については、qconf(1) のマニュアルページを参照してください。share_tree(5) マニュアルページでは、共有ツリーの構成の書式が説明されています。
- -astnode、-mstnode、-dstnode、および -sstnode の各オプションは、共有ツリー全体ではなく、1 つのノードに対応します。ノードは、ディレクトリパスと同じように、すべての親ノードを通過して共有ツリーを下る、パスとして参照されま

す。上記のオプションを使用すると、ノードの追加、変更、削除、および表示を行うことができます。ノードに含まれる情報には、その名前と、関連付けられた配分があります。

- CPU、メモリー、およびI/Oの使用率パラメータの重みは、`usage_weight`としてスケジューラ構成に含まれています。半減期の重みは、`halftime`としてスケジューラ構成に含まれています。補正係数は、`compensation_factor`としてスケジューラ構成に含まれています。`qconf`の`-msconf`および`-ssconf`オプションを使用することで、コマンド行からスケジューラ構成にアクセスできます。書式の詳細については、`sched_conf`(5)のマニュアルページを参照してください。

▼ プロジェクトに基づく共有ツリースケジューリングの作成方法

この設定の目的は、経時でのすべてのクラスタリソースの配分割り当てを、異なるプロジェクトに保証することです。

1. スケジューラ構成で、共有ツリーチケットの数(**1000000**など)を指定します。

   132ページの「QMONを使用したポリシーに基づくリソース管理の構成」、および`sched_conf`(5)のマニュアルページを参照してください。

2. (オプション)スケジューリング関連のユーザーごとに、**1**人のユーザーを追加します。

   104ページの「QMONを使用したユーザーオブジェクトの構成」、および`user`(5)のマニュアルページを参照してください。

3. スケジューリング関連のプロジェクトごとに、**1**つのプロジェクトを追加します。

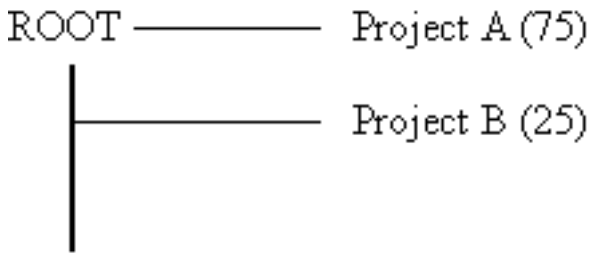
   107ページの「QMONを使用したプロジェクトの定義」、および`project` (5)のマニュアルページを参照してください。

4. QMONを使用して、すべてのスケジューリング関連のプロジェクトの構造をノードとして反映する、共有ツリーを設定します。

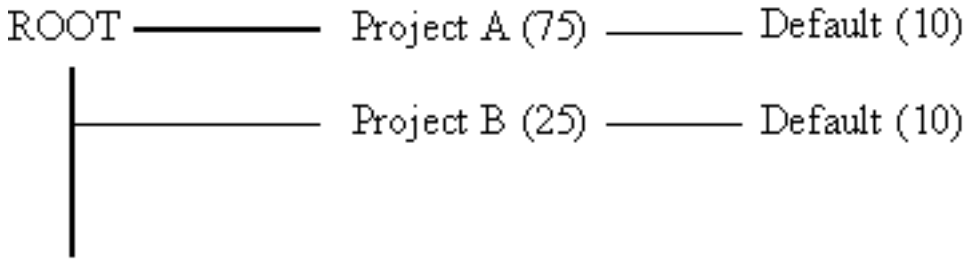
   145ページの「QMONを使用した共有ツリーポリシーの構成」を参照してください。

5. プロジェクトに共有ツリーの配分を割り当てます。

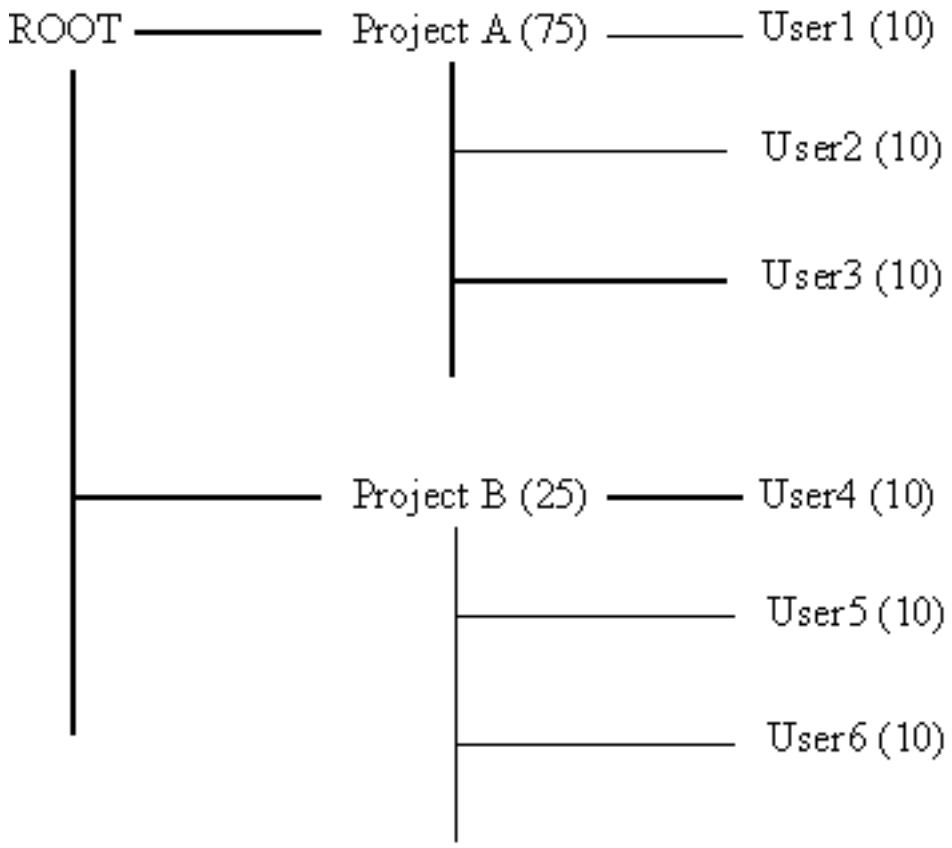
   たとえば、同じプロジェクトのジョブ間で優先順のスケジューリングを行う、プロジェクトに基づく共有ツリースケジューリングを作成する場合、次のようなシンプルな構造になります。



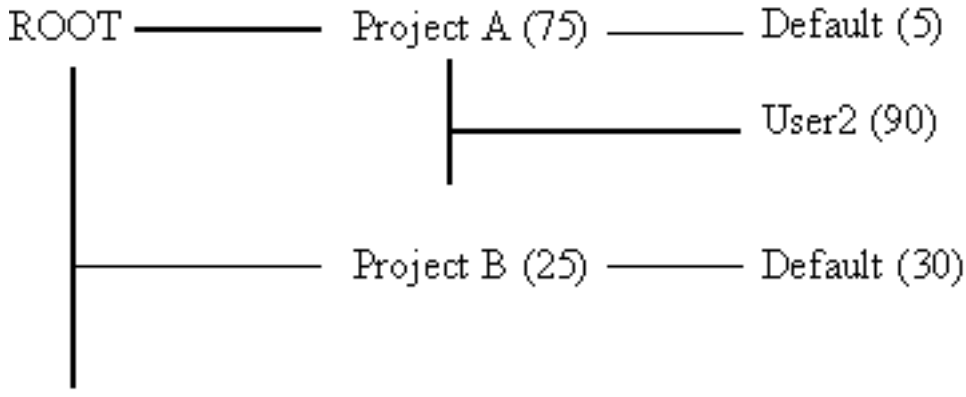
各ユーザーに等しい配分を設定する、プロジェクトに基づく共有ツリースケジューリングを作成する場合、次のようなシンプルな構造になります。



各プロジェクト内に個別のユーザー配分がある、プロジェクトに基づく共有ツリースケジューリングを作成する場合、プロジェクトに対するリーフとしてユーザーを追加します。続いて個別の配分を割り当てます。次のようなシンプルな構造になります。



個別の配分をごく少数のユーザーにのみ割り当てる場合、プロジェクトノードの下に、個別ユーザーと組み合わせて、ユーザー `default` を指定します。たとえば、上図のツリーを次のように要約することができます。



機能ポリシーの構成

機能スケジューリングは、ジョブの重要性を決定するための、非フィードバック方法です。機能スケジューリングは、ジョブと、発行を行うユーザー、プロジェクト、部署、およびジョブクラスを関連付けます。機能スケジューリングは、プライオリティスケジューリングと呼ばれることもあります。機能ポリシーの設定により、定義済みの配分が常に各ユーザー、プロジェクト、または部署に保証されます。予定よりも少ないリソースを使用したユーザー、プロジェクト、または部署のジョブが優先されるのは、システムがアイドル状態のリソースに対してジョブを振り分けた場合のみです。

同時に、未使用の配分の比率は、リソースを必要とするユーザー、プロジェクト、および部署で分配されるため、完全なリソースの使用が保証されます。過去のリソース消費は考慮されません。

ジョブの実際のエンタイトルメントを決定する際に、システムリソースに対する機能ポリシーのエンタイトルメントは、そのほかのエンタイトルメントと結び付けられます。たとえば、機能ポリシーのエンタイトルメントは、共有ベースポリシーのエンタイトルメントと結び付けられる場合があります。

機能ポリシーに割り当てられたチケットの合計数は、3 つのスケジューリングポリシーの間での機能スケジューリングの重みを決定します。インストール時に、管理者は機能チケットの合計数を、ユーザー、部署、プロジェクト、ジョブ、およびジョブクラスの機能カテゴリに分割します。

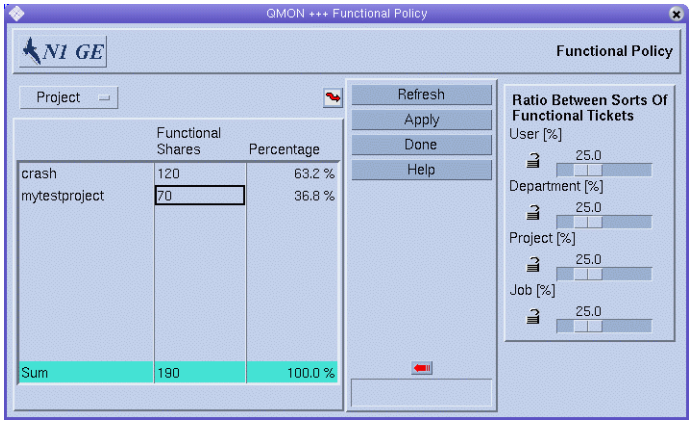
機能共有

機能共有は、各機能カテゴリ (ユーザー、部署、プロジェクト、ジョブ、ジョブクラス) のあらゆるメンバーに割り当てられます。それらの配分は、カテゴリのメンバーに関連付けられている各ジョブが受ける資格を持つ、そのカテゴリ用のチケット全体に占める割合を示します。たとえば、ユーザー `davidson` が 200、ユーザー `donlee` が 100 の配分の場合、存在するチケット数に関係なく、`davidson` が発行するジョブは `donlee` のジョブよりも 2 倍多くの user-functional-tickets を得ることができます。

各カテゴリに割り当てられた機能チケットは、特定のカテゴリに関連付けられているすべてのジョブ間で分配されます。

QMON を使用した機能共有ポリシーの構成

「QMON Policy Configuration」ダイアログボックスの下部にある「Functional Policy」をクリックします。「Functional Policy」ダイアログボックスが表示されます。



Function Category List

機能共有を定義する対象の機能カテゴリ (ユーザー、プロジェクト、部署、またはジョブ) を選択します。

「Functional Shares」テーブル

「Functional Shares」の下にあるテーブルはスクロール可能です。テーブルには次の情報が表示されます。

- 「Function Category」リストから現在選択されているカテゴリのメンバーのリスト。
- カテゴリの各メンバーの機能共有の数。配分は、機能カテゴリの各メンバーの相対的な重要性を表す便利な指標として使用できます。このフィールドは編集可能です。
- この機能共有の数が表す、機能チケットのこのカテゴリに関する、機能共有の割り当てのパーセンテージ。このフィールドはフィードバックデバイスで、編集はできません。

QMON は、「Functional Policy」ダイアログボックスに表示される情報を定期的に更新します。ただちに表示を更新させるには、「Refresh」をクリックします。

行なったすべてのノードの変更を保存するには、「Apply」をクリックします。変更を保存することなくダイアログボックスを閉じるには、「Done」をクリックします。

機能構成の変更

「Functional Shares」テーブルの上にある曲がった矢印をクリックして、構成ダイアログボックスを開きます。

- 「User」の機能共有の場合、「User Configuration」ダイアログボックスが表示されます。Grid Engine ユーザーの構成を変更するために、「User」タブを使用して適切なモードに切り替えます。104 ページの「QMON を使用したユーザーオブジェクトの構成」を参照してください。
- 「Department」の機能共有の場合、「User Configuration」ダイアログボックスが表示されます。ユーザーセットとして表示されているデパートメントの構成を変更するために、「Userset」タブを使用して適切なモードに切り替えます。103 ページの「プロジェクトおよび部署としてのユーザーセットの定義」を参照してください。
- 「Project」の機能共有の場合、「Project Configuration」ダイアログボックスが表示されます。107 ページの「QMON を使用したプロジェクトの定義」を参照してください。
- 「Job」の機能共有の場合、「Job Control」ダイアログボックスが表示されます。『Sun N1 Grid Engine 6.1 ユーザーズガイド』の「QMON を使用したジョブの監視と制御」を参照してください。

「Ratio Between Sorts Of Functional Tickets」

「Ratio Between Sorts Of Functional Tickets」を表示するには、「Functional Shares」テーブルの右側にある矢印をクリックします。

「User [%]」、「Department [%]」、「Project [%]」、「Job [%]」および「Job Class [%]」の合計は常に 100% になります。

いずれかのスライダを動かすと、変化を補うために、ロックされていないそのほかすべてのスライダが変化します。

ロックが開いている場合は、ロックが保護するスライダは自由に動きます。直接操作されることによって動くことも、別のスライダが動かされたために、動くこともあります。錠が閉じていると、錠が保護するスライダは動きません。4 つの錠が閉じていて 1 つが開いている場合は、どのスライダも動かせません。

- 「User」スライダ – 機能チケット全体からユーザーカテゴリに割り当てる割合を示します。
- 「Departments」スライダ – 機能チケット全体からデパートメントカテゴリに割り当てる割合を示します。
- 「Project」スライダ – 機能チケット全体からプロジェクトカテゴリに割り当てる割合を示します。

- 「Job」スライダ – 機能チケット全体からジョブカテゴリに割り当てる割合を示します。

コマンド行からの機能共有ポリシーの構成

注 – QMON を使用する場合にのみ機能共有をジョブに割り当てることができます。この機能に関してはコマンド行インタフェースは使用できません。

コマンド行から機能共有ポリシーを構成するには、適切なオプションを指定した qconf コマンドを使用します。

- ユーザーカテゴリを構成するには、qconf -muser コマンドを使用します。-muser オプションは、ユーザーエントリファイルの fshare パラメータを変更します。ユーザーエントリファイルの詳細については、user(5) のマニュアルページを参照してください。
- 部署カテゴリを構成するには、qconf -mu コマンドを使用します。-mu オプションは、アクセスリストファイルの fshare パラメータを変更します。部署を表すために使用されるアクセスリストファイルの詳細については、access_list(5) のマニュアルページを参照してください。
- プロジェクトカテゴリを構成するには、qconf -mprj コマンドを使用します。-mprj オプションは、プロジェクトエントリファイルの fshare パラメータを変更します。プロジェクトエントリファイルの詳細については、project(5) のマニュアルページを参照してください。
- ジョブクラスカテゴリを構成するには、qconf -mq コマンドを使用します。-mq オプションは、キュー構成ファイルの fshare パラメータを変更します。ジョブクラスを表すために使用されるキュー構成ファイルの詳細については、queue_conf (5) のマニュアルページを参照してください。
- さまざまなカテゴリ間の重みは、スケジューラ構成 sched_conf で定義し、qconf -msconf を使用して変更することができます。変更するパラメータは、weight_user、weight_department、weight_project、weight_job、および weight_jobclass です。これらのパラメータの値の範囲は 0 から 1 で、パラメータの合計は 1 になる必要があります。

▼ ユーザー、プロジェクト、および部署に基づく機能スケジューリングの作成方法

この設定を使用して、クラスタ内のすべてのリソースのある種の配分割り当てを、さまざまなユーザー、プロジェクト、または部署に対して作成します。同じユーザー、プロジェクト、または部署のジョブでは、先着順のスケジューリングが使用されます。

1 「**Scheduler Configuration**」ダイアログボックスで、「**Share Functional Tickets**」チェックボックスを選択します。

137 ページの「機能チケットの分配の共有」、および `sched_conf`(5) のマニュアルページを参照してください。

2 スケジューラ構成で、機能チケットの数 (**1000000** など) を指定します。

132 ページの「QMON を使用したポリシーに基づくリソース管理の構成」、および `sched_conf`(5) のマニュアルページを参照してください。

3 スケジューリング関連の項目を追加します。

- スケジューリング関連のユーザーごとに、**1** 人のユーザーを追加します。

  104 ページの「QMON を使用したユーザーオブジェクトの構成」、および `user`(5) のマニュアルページを参照してください。

- スケジューリング関連のプロジェクトごとに、**1** つのプロジェクトを追加します。

  107 ページの「QMON を使用したプロジェクトの定義」、および `project` (5) のマニュアルページを参照してください。

- 各スケジューリング関連の部署を追加します。

4 各ユーザー、プロジェクト、または部署に機能共有を割り当てます。

101 ページの「QMON を使用したユーザーアクセスリストの構成」、および `access_list`(5) のマニュアルページを参照してください。

全体に対するパーセンテージとして配分を割り当てます。次に例を示します。

ユーザーの場合

- UserA (10)
- UserB (20)
- UserC (20)
- UserD (20)

プロジェクトの場合

- ProjectA (55)
- ProjectB (45)

部署の場合

- DepartmentA (90)
- DepartmentB (5)
- DepartmentC (5)

優先ポリシーの構成

優先スケジューリングでは、Grid Engine システムの管理者またはオペレータは、ユーザー、部署、プロジェクト、ジョブクラスに関連付けられている、1 つまたはすべてのジョブの相対的な重要性を動的に調整することができます。この調整では、指定されたジョブ、ユーザー、部署、プロジェクト、またはジョブクラスにチケットを追加します。優先チケットを追加すると、優先スケジューリングでは、ユーザー、部署、プロジェクト、またはジョブが受け取るチケットの合計数が増加します。その結果、リソースの全体配分が増加します。

また、優先チケットを追加すると、システムのチケットの合計数も増加します。こうした追加のチケットによって、あらゆるジョブのチケットの価値が低下します。

優先チケットは、次の 2 つの用途に使用できます。

- 共有ベースポリシーや機能ポリシーの構成を変更しないで、それらのポリシーよりも一時的に優先させる。
- 関連付けられた一定量のチケットを有するリソースエンタイトルメントレベルを確立する。エンタイトルメントレベルの確立は、高、中、低のジョブクラス、または高、中、低の優先順位クラスなどのシナリオに適しています。

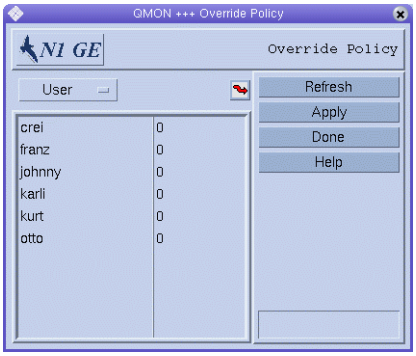
ジョブに直接割り当てられた優先チケットは、そのジョブが完了すると消滅します。ほかのすべてのチケットは元の価値に戻ります。ユーザー、部署、プロジェクト、ジョブクラスに割り当てられた優先チケットは、管理者が明示的にチケットを削除するまで、システムに留まります。

「Policy Configuration」ダイアログボックスには、システムでアクティブな優先チケットの現在の数が表示されます。

注 – 優先エントリは「Override」ダイアログボックスに残ります。不必要になった時点で管理者がこれらのエントリを明示的に削除しなかった場合、以降の運用に影響が出ることがあります。

QMON を使用した優先ポリシーの構成

「QMON Policy Configuration」ダイアログボックスの下部にある「Override Policy」をクリックします。「Override Policy」ダイアログボックスが表示されます。



「Override Category」リスト

優先チケットを定義する対象である業務優先カテゴリ (ユーザー、プロジェクト、部署、またはジョブ) を選択します。

優先テーブル

優先テーブルはスクロール可能です。次の情報が表示されます。

- チケットを定義する対象であるカテゴリのメンバーのリスト。カテゴリには、ユーザー、プロジェクト、部署、ジョブ、およびジョブクラスがあります。
- カテゴリの各メンバーの優先チケットの数。このフィールドは編集可能です。

`QMON` は、「Override Policy」ダイアログボックスに表示される情報を定期的に更新します。ただちに表示を更新させるには、「Refresh」をクリックします。

行なった優先に関するすべての変更を保存するには、「Apply」をクリックします。変更を保存することなくダイアログボックスを閉じるには、「Done」をクリックします。

優先構成の変更

優先テーブルの上にある曲がった矢印をクリックして、構成ダイアログボックスを開きます。

- 「User」の優先チケットの場合、「User Configuration」ダイアログボックスが表示されます。Grid Engine ユーザーの構成を変更するために、「User」タブを使用して適切なモードに切り替えます。104 ページの「QMON を使用したユーザーオブジェクトの構成」を参照してください。
- 「Department」の優先チケットの場合、「User Configuration」ダイアログボックスが表示されます。ユーザーセットとして表示されている部署の構成を変更するために、「Userset」タブを使用して適切なモードに切り替えます。103 ページの「プロジェクトおよび部署としてのユーザーセットの定義」を参照してください。
- 「Project」の優先チケットの場合、「Project Configuration」ダイアログボックスが表示されます。107 ページの「QMON を使用したプロジェクトの定義」を参照してください。
- 「Job」の優先チケットの場合、「Job Control」ダイアログボックスが表示されます。『Sun N1 Grid Engine 6.1 ユーザーズガイド』の「QMON を使用したジョブの監視と制御」を参照してください。

コマンド行からの優先ポリシーの構成

注 – QMON を使用する場合にのみ優先チケットをジョブに割り当てることができます。この機能に関してはコマンド行インタフェースは使用できません。

コマンド行から優先ポリシーを構成するには、適切なオプションを指定した qconf コマンドを使用します。

- ユーザーカテゴリを構成するには、qconf -muser コマンドを使用します。-muser オプションは、ユーザーエントリファイルの oticket パラメータを変更します。ユーザーエントリファイルの詳細については、user(5) のマニュアルページを参照してください。
- 部署カテゴリを構成するには、qconf -mu コマンドを使用します。-mu オプションは、アクセスリストファイルの oticket パラメータを変更します。部署を表すために使用されるアクセスリストファイルの詳細については、access_list (5) のマニュアルページを参照してください。
- プロジェクトカテゴリを構成するには、qconf -mprj コマンドを使用します。-mprj オプションは、プロジェクトエントリファイルの oticket パラメータを変更します。プロジェクトエントリファイルの詳細については、project(5) のマニュアルページを参照してください。

- ジョブクラスカテゴリを構成するには、`qconf -mq` コマンドを使用します。`-mq` オプションは、キュー構成ファイルの `oticket` パラメータを変更します。ジョブクラスを表すために使用されるキュー構成ファイルの詳細については、`queue_conf` (5) のマニュアルページを参照してください。

◆ ◆ ◆ 第 6 章

リソース割り当ての管理

この章では、N1 Grid Engine 6.1 ソフトウェアのリソース割り当て機能を使用して、ユーザー、プロジェクト、ホスト、クラスタキュー、並列環境別にリソースを制限する方法を説明します。こうした制限は、ユーザーアクセスリスト、部署、ホストグループを使用して表すことができます。

この章では、次の内容について説明します。

リソース割り当ての概要

ユーザーが使用可能なすべてのリソースを消費しないようにするため、N1 Grid Engine 6.1 ソフトウェアは、グローバル、キュー、またはホスト層で設定可能なコンプレックス属性をサポートしています。この階層型のリソース管理手法は強力ではありますが、ギャップが生じます。多数のさまざまなカスタムリソース、ユーザーグループ、およびプロジェクトで構成される大規模な導入では特に重要な意味を持ちます。リソース割り当て機能は、ボトルネックリソースが不足したときに、あるプロジェクトまたは部署がそのリソースを譲るように制御して、こうした企業環境を管理できるようにすることで、このギャップを埋めます。

リソース割り当て機能によって、数種類のリソース、数種類のリソース消費者、クラスタ内のすべてのジョブ、また消費者の組み合わせに対して制限を適用することができます。この意味では、リソースは N1 Grid Engine 構成で認識されている、定義済みコンプレックス属性のすべてです。コンプレックス属性の詳細は、`complex(5)` のマニュアルページを参照してください。リソースには、`slots`、 `arch`、`mem_total`、`num_proc`、`swap_total`、組み込みリソース、あるいは `compiler_license`

のようなカスタム定義のリソースを指定できます。リソース消費者は、ユーザー、キュー、ホスト、プロジェクト、および並列環境単位になります。

リソース割り当て機能は、消費者が任意の時点で使用可能なリソースを制限する手段を提供します。この制限は、ユーザー、部署、およびプロジェクトに優先順位を付ける間接的な手段になります。ユーザーがリソースを取得する優先順位を直接定義するには、135 ページの「緊急度ポリシーの構成」および 141 ページの「共有ベースポリシーの構成」で説明しているリソースの緊急度および共有ベースポリシーを利用します。

N1 Grid Engine 6.1 ソフトウェアを使用してリソースを制限するには、`qquota` および `qconf` コマンド、または QMON グラフィカルインタフェースを使用します。詳細は、`qquota(1)` および `qconf(1)` のマニュアルページを参照してください。

リソース割り当てセットについて

リソース割り当てセットを使用して、任意のジョブ要求に対して最大リソース消費量を指定することができます。リソース割り当てセットを定義すると、スケジューラはそのセットを利用し、割り当て量の超過がないことに注意しながら、次に実行可能なジョブを選択することができます。リソース割り当て量を設定することの最大の意味は、リソース割り当て量を超えないジョブだけがスケジューリングされて、実行されることです。

リソース割り当てセットでは、特定のジョブ要求に対する最大リソース割り当て量を定義します。構成された規則セットのすべてが、常に適用されます。複数のリソース割り当てセットが定義されている場合は、もっとも制限が厳しいセットが適用されます。リソース割り当てセットはどれも、1 つ以上のリソース割り当て規則で構成されます。これらの規則が順に評価され、特定の要求に合致する最初の規則が使用されます。つまり、リソース割り当てセットの中の 1 つのリソース割り当て規則が特定の要求に対する実効規則になります。

リソース割り当てセットは、次の情報で構成されます。

- 名前 (name) – リソース割り当てセットの名前。
- 有効 (enabled) – スケジューリングの判定時にリソース割り当てセットを考慮するかどうかを示すブール型値。enabled が true の場合は、リソース割り当てセットは有効になり、スケジューリングの判定時に考慮されます。デフォルト値は false です。
- 説明 (description) – セットを説明する任意の文字列 (省略可能)。デフォルト値は NONE です。
- 制限規則 (limit rule) – どのリソース割り当てセットにも、少なくとも 1 つの制限規則が必要です。この規則を limit フィールドに含めます。たとえば、次の制限規則はユーザー全体で最大スロット割り当て数を 10 個に制限します。`limit users * to slots=10`. 制限規則には、次の情報が含まれます。

- 名前 (name) – 規則の名前 (省略可能)。使用する場合は、リソース割り当てセット内で一意でなければなりません。
- フィルタスコープ (filter scope) – この割り当てを適用するリソース消費者をリスト指定します。各リソース消費者は、1 つのキーワードのあとに消費者のコンマ区切りのリストで設定します。使用できるキーワードは次のとおりです。`users`、 `projects`、`queues` (クラスタキュー)、`hosts`、および `pes` (並列環境)。たとえば、`users {user1, user2}` はリソース消費者の例です。`users {user1, user2} hosts *` はフィルタスコープの例です。このフィルタスコープは、`user1` および `user2` に対する最大割り当て量をホストとは別に設定された制限の最大数に制限します。

  リソース割り当て定義に拡張可能なリストを含めるには、リソース消費者リストを中括弧 {} で囲みます。

  リストから特定のリソースタイプを除外するには、感嘆符 ! を使用します (「否定」記号とも呼ばれることがある)。
- 制限 (limit) – リソースに対する実際の制限を定義した属性と値のペア。例: `virtual_free=2G`。属性と値のペアはコンマで区切って複数指定することができます。例: `virtual_free=2G,swap_free=1.5G`。

例 6–1 リソース割り当てセット例

次のリソース割り当てセット例は、`user1` と `user2` に対して、ホストグループ `lx_hosts` の各ホストの未使用仮想空間の最大割り当て量を 2 ギガバイトに制限します。

```
{
   name         max_virtual_free_on_lx_hosts
   description  "resource quota for virtual_free restriction"
   enabled      true
   limit        users {user1,user2} hosts {@lx_host} to virtual_free=2g
}
```

静的リソース割り当てと動的リソース割り当て

リソース割り当て規則では、必ず、使用可能なリソースの最大値を定義します。だいたいの場合、これらは静的であり、一致するすべてのフィルタスコープに対して同じです。いくつかの異なる規則を、異なるスコープに適用することもできますが、それらの規則はほぼ同じです。この場合は、規則を複製する代わりに、動的制限を定義できます。

動的制限では、代数式を使用して規則制限値を導出します。代数式でコンプレックス属性を参照し、その値を使用して制限値を計算することができます。

例 6–2 動的制限例

次は、動的制限の使用例です。ユーザーは、すべての Linux ホストで 1 CPU 当たり 5 つのスロットの使用が許可されます。

```
limit hosts {@linux_hosts} to slots=$num_proc*5
```

num_proc の値は、ホスト上のプロセッサ数です。制限は式 $num_proc*5 で求められ、ホストによって異なることがあります。上記の例を展開すると、たとえば次のような制限が考えられます。

- CPU 2 つのホストの場合、ユーザーはジョブの実行に 10 個のスロットを使用できます。
- CPU 1 つのホストの場合、ユーザーはジョブの実行に 5 個のスロットを使用できます。

num_proc の代わりに、負荷値あるいは消費可能リソースのいずれかとしてホストが認識するほかのコンプレックス属性をすべて使用することができます。

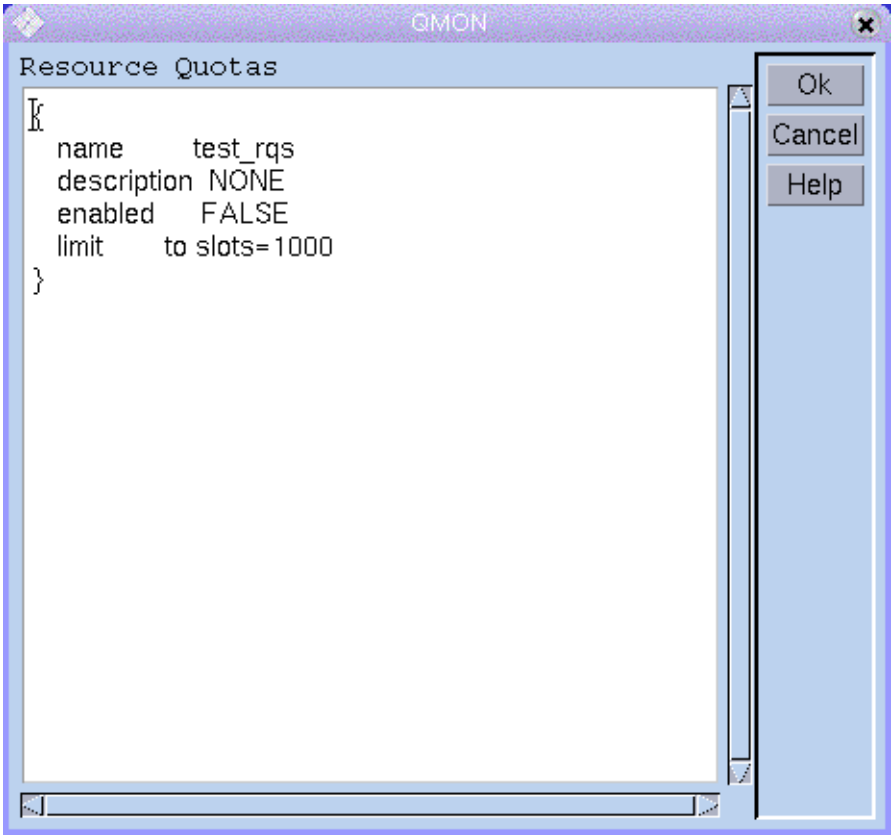
QMON によるリソース割り当ての管理

次のタスクは、QMON グラフィカルインタフェースを使用してリソース割り当て量を設定する例です。

▼ QMON を使用してリソース割り当て量を設定する

1 「**QMON Main Control**」ウィンドウで「**Resource Quota Configuration**」ボタンをクリックします。

2 テキストフィールドにリソース割り当て情報を入力します。
次の画面例で示すように、qconf コマンドに対するのと同じ構文を使用してください。



コマンド行からのリソース割り当て利用状況の監視

qquota コマンドを使用して、現在の N1 Grid Engine リソース割り当てに関する情報を表示できます。qquota コマンドは、少なくとも 1 回使用されているか、静的制限を定義した各リソース割り当てを一覧表示します。qquota は、該当するリソース割り当てごとに次の情報を表示します。

- リソース割り当て規則 – 規則セットの名前と規則の名前または個数
- 制限 – リソース名、およびそのリソースの使用可能数と使用数
- フィルタ – 実効リソース割り当てセットフィルタ。164 ページの「リソース割り当てセットについて」 で説明したフィルタスコープを適用することで決定されます。

qquota コマンドには、特定のホスト、クラスタキュー、プロジェクト、並列環境、リソースまたはユーザーに情報を絞り込むためのオプションがいくつかあります。

オプションが使用されなかった場合、qquota は、コマンドの呼び出し元ユーザー名に該当するリソースセットの情報を表示します。詳細は、qquota(1) のマニュアルページを参照してください。

例 6–3 qquota コマンド例

次の例は、ユーザー user1 に該当するリソース割り当てセットの情報を示しています。

```
$ qquota -u user1
resource quota    limit               filter
--------------------------------------------------------------------------------
maxujobs/1         slots=5/20          -
max_linux/1        slots=5/5           hosts @linux
max_per_host/1     slots=1/2           users user1 hosts host2
```

コマンド行からのリソース割り当ての設定

qconf コマンドを使用して、リソース割り当てセットおよび規則を追加、変更、削除できます。

- テキストエディタを呼び出して、リソース割り当てセットを追加する。

  ```
  $ qconf -arqs [name]
  ```

  ファイル中にすでに定義されているセットを追加するには、qconf -Arqs *filename* を使用します。
- エディタを呼び出して、リソース割り当てセットの情報を変更する。

  ```
  $ qconf -mrqs [name]
  ```

  ファイルに含まれる情報を使用してセットを変更するには、 qconf -Mrqs *filename* [*name*] を使用します。

  注 – -mrqs または -Mrqs オプションで名前を指定しなかった場合は、現在設定されているすべての規則セットが新しい規則セットで置き換えられます。

- リソース割り当てセットを削除する。

  ```
  $ qconf -drqs [name_list]
  ```
- 定義されているリソース割り当てセットを一覧表示する。

  ```
  $ qconf -srqsl
  ```

- 定義されている特定のリソース割り当てセットの詳細情報を表示する。

```
$ qconf -srqs [name_list]
```

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

例

次の例は、リソース割り当てにさまざまなコマンドを使用する方法を示しています。例 6–4 に示す規則セットが定義している制限は、次のとおりです。

- ユーザー全体でスロットの割り当て量が 20 個を超えてはならない。
- 各 Linux ホストで、ユーザー全員のスロット割り当ては最大 5 個。
- ユーザー 1 人当たりのスロット割り当て量は Linux ホスト 1 台当たり 1 スロット。ただし、MyUser ユーザーの割り当て量は 2 つ。ほかのホストのすべてのスロットはゼロに制限。
- ホストグループ @linux には、host1 および host2 がある。

この規則セットを設定するには、次の形式の qconf コマンドを使用します。

- qconf -arqs <*rule-set-name*> (規則セットごと実行する場合)
- qconf -arqs (すべての規則セットを一度に実行する場合)

複数のユーザーでジョブを発行すると、qstat コマンドは、例 6–5 に示すような情報を出力します。

例 6–4 ルールセット

```
{
 name maxujobs
 limit users * to slots=20
}

{
 name max_linux
 limit users * hosts @linux to slots=5
}

{
 name max_per_host
 limit users MyUser hosts {@linux} to slots=2
 limit users {*} hosts {@linux} to slots=1
 limit users * hosts * to slots=0
}
```

例 6–5 qstat の出力

```
$ qstat
job-ID  prior   name       user       state submit/start at     queue       slots ja-task-ID
-----------------------------------------------------------------------------------------
     27 0.55500 Sleeper    MyUser     r     02/21/2006 15:53:10 all.q@host1   1
     29 0.55500 Sleeper    MyUser     r     02/21/2006 15:53:10 all.q@host1   1
     30 0.55500 Sleeper    MyUser     r     02/21/2006 15:53:10 all.q@host2   1
     26 0.55500 Sleeper    MyUser     r     02/21/2006 15:53:10 all.q@host2   1
     28 0.55500 Sleeper    user1      r     02/21/2006 15:53:10 all.q@host2   1
```

例 6–6 qquota の出力

```
$ qquota # as user MyUser
resource quota rule    limit                filter
--------------------------------------------------------------------------------
maxujobs/1         slots=5/20           -
max_linux/1        slots=5/5            hosts @linux
max_per_host/1     slots=2/2            users MyUser hosts host2
max_per_host/1     slots=2/2            users MyUser hosts host1

$ qquota -h host2 # as user MyUser
resource quota    limit                filter
--------------------------------------------------------------------------------
maxujobs/1         slots=5/20           -
max_linux/1        slots=5/5            hosts @linux
max_per_host/1     slots=2/2            users MyUser hosts host2

$ qquota -u user1
resource quota    limit                filter
--------------------------------------------------------------------------------
maxujobs/1         slots=5/20           -
max_linux/1        slots=5/5            hosts @linux
max_per_host/1     slots=1/2            users user1 hosts host2

$ qquota -u *
resource quota    limit                filter
--------------------------------------------------------------------------------
maxujobs/1         slots=5/20           -
max_linux/1        slots=5/5            hosts @linux
max_per_host/1     slots=2/2            users MyUser hosts host1
max_per_host/1     slots=2/2            users MyUser hosts host2
max_per_host/1     slots=1/2            users user1 hosts host2
```

パフォーマンス上の検討事項

効率的な規則セット

キュー内のジョブおよびリソースの処理効率を最大限に高めるには、もっとも厳しい規則を規則セットの先頭に置きます。この仕様に従うと、N1 Grid Engineスケジューラが特に効率的な方法で適切なキューインスタンスの量を絞り込むのに役立ちます。なぜなら、先頭の規則は、同じ規則セット内の後続の規則に影響されることはなく、常にその規則の代表になるためです。

このことを具体的に理解するために、次のような環境を考えてみます。

- 4 つのキュー Q001 ～ Q004
- 4 つの管理対象リソース F001 ～ F004
- F001 などの特定の管理対象リソースを要求するジョブは、Q001 などの関連付けられているキューで実行するよう制限されている
- ジョブは 5 つあるプロジェクト P001 ～ P005 の 1 つに発行する

こうした環境で、次のような規則セットを 1 つ定義したと仮定します。

```
{
      name         30_for_each_project
      description  "not more than 30 per project"
      enabled      TRUE
      limit projects {*} queues Q001 to F001=30
      limit projects {*} queues Q002 to F002=30
      limit projects {*} queues Q003 to F003=30
      limit projects {*} queues Q004 to F004=30
      limit to F001=0,F002=0,F003=0,F004=0
   }
```

この 1 つの規則セットは、プロジェクト 1 つ当たりの各管理対象リソースの使用量を 30 に制限し、同時に適格のキュー内のジョブを制限します。この規則は問題なく機能しますが、多数のホストからなる大規模なクラスタでは、ジョブの割り当てに時間がかかる原因になります。

N1 Grid Engineスケジューラが仲介中にできるだけ多くのキューインスタンスを除外できるようにするには、この単一規則セットを 4 つの規則セットに分ける方が優れています。

```
{
      name         30_for_each_project_in_Q001
      description  "not more than 30 per project of F001 in Q001"
      enabled      TRUE
```

```
   limit queues !Q001 to F001=0
   limit projects {*} queues Q001 to F001=30
}

{
   name         30_for_each_project_in_Q002
   description  "not more than 30 per project of F002 in Q002"
   enabled      TRUE
   limit queues !Q002 to F002=0
   limit projects {*} queues Q002 to F002=30
}

{
   name         30_for_each_project_in_Q003
   description  "not more than 30 per project of F003 in Q003"
   enabled      TRUE
   limit queues !Q003 to F003=0
   limit projects {*} queues Q003 to F003=30
}

{
   name         30_for_each_project_in_Q004
   description  "not more than 30 per project of F004 in Q004"
   enabled      TRUE
   limit queues !Q004 to F004=0
   limit projects {*} queues Q004 to F004=30
}
```

この4つの規則セットは、単一の規則セットと同じプロジェクト単位のリソース割り当て量制限を課します。しかし、4つの規則セットは、不適切なキューインスタンスが最初にシールドされているため、ずっと効率的に処理できます。この場合、これらのシールドを1つのリソース割り当てセットにまとめることはできません。

注 – 前述の例の目的は、リソース 1 つ当たりクラスタキュー 1 つを推奨することではありません。実際は、その逆です。次に示すように、キューを少なくすることで、常にシールドをより少なく、より強力にできます。

```
{
    name         30_for_each_project_in_Q001
    description  "not more than 30 per project of F001/F002 in Q001"
    enabled      TRUE
    limit queues !Q001 to F001=0,F002=0
    limit projects {*} queues Q001 to F001=30,F002=30
}

{
    name         30_for_each_project_in_Q002
    description  "not more than 30 per project of F003/F004 in Q002"
    enabled      TRUE
    limit queues !Q002 to F003=0,F004=0
    limit projects {*} queues Q002 to F003=30,F004=30
}
```

この例では、キューが Q001 ～ Q004 から Q001 ～ Q002 に整理されています。しかし、実際にはこうすることで、クラスタ全体の稼働率とスループットが向上します。

◆ ◆ ◆ 第 7 章

7 特別な環境の管理

この章では、次の特別な環境を管理する方法について説明します。

- 並列環境
- チェックポイント環境

これらの環境に関する内容説明に加えて、この章では次のタスクの実行方法を詳細に解説します。

- 176 ページの「QMON を使用した並列環境の構成」
- 181 ページの「コマンド行からの並列環境の構成」
- 187 ページの「QMON を使用したチェックポイント設定環境の構築」
- 188 ページの「コマンド行からのチェックポイント設定環境の構成」

並列環境の構成

並列環境 (PE) は、ネットワーク環境での並列プラットフォームにおける並列コンピューティングを実現するソフトウェアパッケージです。

この何年もの間にさまざまなシステムが発展を遂げ、さまざまなハードウェアプラットフォームで分散、並列処理技術が実用的なものになってきました。もっとも一般的なメッセージ引き渡し環境には、次の 2 つの例があります。

- PVM – Parallel Virtual Machine (Oak Ridge National Laboratories)
- MPI – Message Passing Interface (the Message Passing Interface Forum)

両方のツールとも、ハードウェアベンダー提供の実装だけでなく、パブリックドメインのものもあります。

これらのシステムはどれも異なる特徴を持ち、要求される使用条件がそれぞれに異なります。そうしたシステム上で実行中の並列ジョブに対応できるよう、Grid Engine システムには、さまざまなニーズを満たす柔軟で強力なインタフェースが用意されています。

Grid Engine システムには、次のプログラムにより並列ジョブを実行する手段が用意されています。

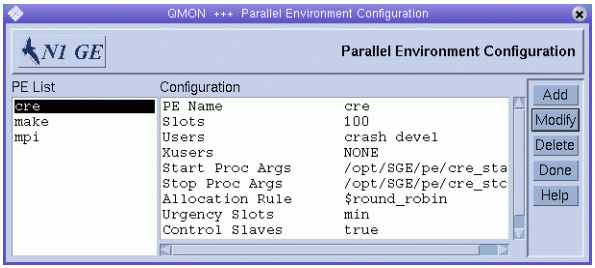
- PVM や MPI などの任意のメッセージ引き渡し環境。詳細については、『PVM User's Guide』および『 MPI User's Guide』を参照してください。
- 分散メモリー並列ジョブのための、単一キュー内に収まる、または複数のキューおよびマシンに分散された、複数スロット上の共有メモリー並列プログラム。

任意の数のさまざまな並列環境インタフェースを同時並行的に構成することができます。

起動および停止手続きが用意されていれば、並列環境と Grid Engine システムとの間でインタフェースを実装できます。起動手続きと停止手続きは、それぞれ182 ページの「並列環境の起動手続き」および183 ページの「並列環境の終了」で説明されています。

QMON を使用した並列環境の構成

「QMON Main Control」ウィンドウで「Parallel Environment Configuration」ボタンをクリックします。「Parallel Environment Configuration」ダイアログボックスが表示されます。



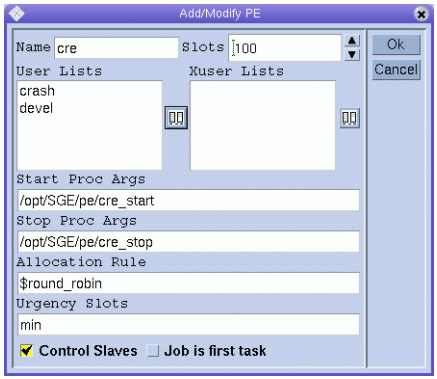
現在構成されている並列環境は、「PE List」の下に表示されます。

並列環境の内容を表示するには、並列環境を選択します。選択した並列環境の構成は、「Configuration」の下に表示されます。

並列環境を削除するには、並列環境を選択してから「Delete」をクリックします。

新しい並列環境を追加するには「Add」をクリックし、並列環境を変更するには並列環境を選択してから「Modify」をクリックします。

「Add」または「Modify」をクリックすると、「Add/Modify PE」ダイアログボックスが表示されます。

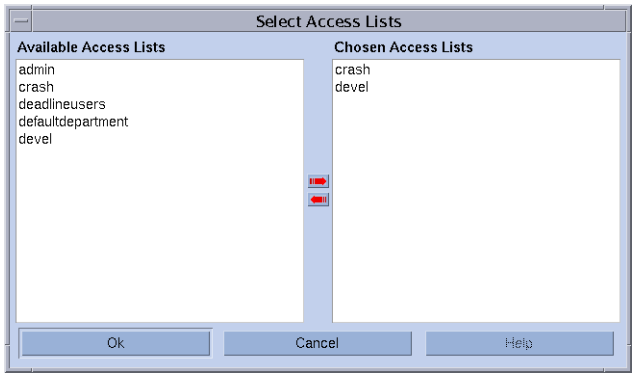


新しい並列環境を追加する場合、「Name」フィールドにその名前を入力します。並列環境を変更する場合、「Name」フィールドにはその名前が表示されます。

「Slots」ボックスでは、並行して実行中のすべての並列環境ジョブにより占有可能なジョブスロットの合計数を入力します。

「User Lists」には、並列環境にアクセスできるユーザーアクセスリストが表示されます。「Xuser Lists」には、並列環境にアクセスできないユーザーアクセスリストが表示されます。ユーザーアクセスリストの詳細については、100 ページの「ユーザーアクセスリストの構成」を参照してください。

リストの内容を変更するには、各リストの右側にあるアイコンをクリックします。「Select Access Lists」ダイアログボックスが表示されます。



「Start Proc Args」フィールドおよび「Stop Proc Args」フィールドはオプションです。これらのフィールドを使用して、並列環境の起動および停止手続きの正確な起動シーケンスを入力します。それぞれ、182 ページの「並列環境の起動手続き」および 183 ページの「並列環境の終了」を参照してください。ある種の並列環境にこのような手続きが必要でない場合は、これらのフィールは空のままにできます。

通常、先頭の引数は、起動または停止手続きそのものの名前になります。残りのパラメータは、その手続きに対するコマンド行引数です。

内部実行時情報を手続きに渡すための各種の特殊な識別子 ($ 接頭辞から始まる) が用意されています。使用可能なすべてのパラメータのリストは、 sge_pe(5) のマニュアルページに記載されています。

「Allocation Rule」フィールドでは、並列環境で使用する各マシンに割り当てる並列プロセス数を定義します。正の整数で、適切な各ホストのプロセス数を決定します。特別な分母 $pe_slots を使用すると、ジョブのプロセスの完全な範囲を、シングルホスト (SMP) 上に割り当てることになります。分母 $fill_up および $round_robin を使用すると、各ホストでのプロセスの分配がアンバランスになります。これらの割り当て規則の詳細については、 sge_pe(5) のマニュアルページを参照してください。

「Urgency Slots」フィールドでは、スロット範囲を持つ保留中のジョブが取得するスロット数のアクセスに Grid Engine システムが使用する方法を指定します。想定されたスロット割り当てが意味を持つのは、数値リソースに関するリソース要求ベースの優先順位関係を決定する場合です。スロットの数には、整数値を指定できます。min を指定すると、スロット範囲の最小値を使用します。max を指定すると、スロット範囲の最大値を使用します。avg を指定すると、ジョブの並列環境の範囲要求内で発生するすべての数値の平均を使用します。

「Control Slaves」チェックボックスでは、Grid Engine システムが並列タスクを生成するかどうか、または対応する並列環境が独自のプロセスを作成するかどうかを指定します。Grid Engine システムでは、sge_execd および sge_shepherd を使用して並列タスクを生成します。システムにより正確なアカウンティングとリソース制御が実現されるため、Grid Engine システムによりスレーブタスクを完全に制御することを推奨します。ただし、この機能が使用可能であるのは、Grid Engine システムシステム用に特別にカスタマイズされた並列環境インタフェースのみです。詳細については、184 ページの「並列環境と Grid Engine ソフトウェアの密統合」を参照してください。

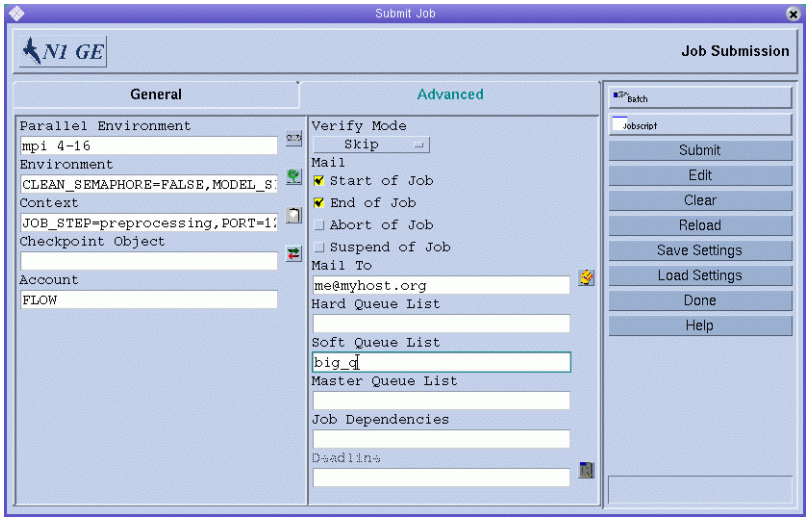
「Job Is First Task」チェックボックスが意味を持つのは、「Control Slaves」が選択されている場合のみです。「Job Is First Task」を選択した場合は、ジョブスクリプト、またはその子プロセスの 1 つが、並列アプリケーションの並列タスクの 1 つとして動作します。たとえば PVM では、通常、ジョブスクリプトを並列アプリケーションの一部にする必要があります。「Job Is First Task」チェックボックスをクリアすると、ジョブスクリプトは並列アプリケーションを開始しますが、参加はしません。たとえば MPI では、mpirun を使用する場合、通常はジョブスクリプトを並列アプリケーションの一部にする必要はありません。

変更を保存するには「OK」をクリックし、ダイアログ ボックスを閉じます。変更を保存せずにダイアログボックスを閉じるには、「Cancel」をクリックします。

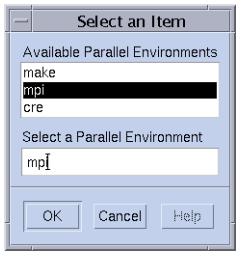
QMON を使用した構成済み並列環境インタフェースの表示

「QMON Main Control」ウィンドウで「Parallel Environment Configuration」ボタンをクリックします。「Parallel Environment Configuration」ダイアログボックスが表示されます。詳細については、176 ページの「QMON を使用した並列環境の構成」を参照してください。

次の例では、発行される並列ジョブを定義しています。ジョブは、4 から 16 個のプロセスとともに、並列環境インタフェース mpi (メッセージ引き渡しインタフェース) を使用するよう要求しています。推奨値は 16 です。



使用可能な並列環境のリストから並列環境を選択するには、「Parallel Environment」フィールドの右側にあるボタンをクリックします。選択ダイアログボックスが表示されます。



「Parallel Environment」フィールドの並列環境名のあとに、ジョブにより開始される並列タスクの数の範囲を追加できます。

以前に説明した並列ジョブの指定に対応する qsub コマンドは、次のようになります。

```
% qsub -N Flow -p -111 -P devel -a 200012240000.00 -cwd \
 -S /bin/tcsh -o flow.out -j y -pe mpi 4-16 \
 -v SHARED_MEM=TRUE,MODEL_SIZE=LARGE \
 -ac JOB_STEP=preprocessing,PORT=1234 \
 -A FLOW -w w -r y -m s,e -q big_q\
 -M me@myhost.com,me@other.address \
 flow.sh big.data
```

この例は、qsub -pe コマンドを使用して、同等の要求を定式化する方法を示しています。qsub(1) のマニュアルページには、-pe オプションの詳細が記載されています。

次の考慮事項に留意し、並列ジョブに適した並列環境インタフェースを選択します。

- 並列環境インタフェースはさまざまなメッセージ引き渡しシステムを使用でき、またメッセージシステムを使用しない場合もあります。
- 並列環境インタフェースは、単一のホストにプロセスを割り当てることもあれば、複数のホストに割り当てることもあります。
- 一部のユーザーに対しては、並列環境へのアクセスを拒否されることもあります。
- 特定のキューしか利用しない並列環境インタフェースもあります。
- いつでも特定の個数のキュースロットしか占有しない並列環境インタフェースもあります。

ご使用の種類の並列ジョブに最適な、使用可能な並列環境インタフェースについては、Grid Engine システムの管理者にお問い合わせください。

並列環境要求とともにリソース要件を指定することができます。リソース要求を指定すると、並列環境インタフェースに適したキューのセットが、要求に合うキューにさらに絞られます。『Sun N1 Grid Engine 6.1 ユーザーズガイド』の「リソース要件の定義」を参照してください。

たとえば、次のコマンドを実行する場合を考えます。

```
% qsub -pe mpi 1,2,4,8 -l nastran,arch=osf nastran.par
```

このジョブに適したキューは、並列環境の構成で並列環境インタフェース mpi に関連付けられたキューになります。適したキューは、qsub -l コマンドで指定されたリソース要求の指定も満たします。

注 – 並列環境インタフェースは、構成の自由度が大きい機能です。特に、管理者は、サイトに固有のニーズに合わせて並列環境の起動および停止手続きを構成することができます。詳細については、sge_pe(5) のマニュアルページを参照してください。起動および停止手続きにジョブを発行するユーザーから、情報を渡すには、qsub -v および qsub -V コマンドを使用します。これらの 2 つのオプションは環境変数をエクスポートします。特定の環境変数をエクスポートする必要があるかどうかについて不明な点がある場合は、管理者にお尋ねください。

コマンド行からの並列環境の構成

適切なオプションを使用して、qconf コマンドを入力します。

qconf オプション

次のオプションを使用できます。

- qconf -ap *pe-name*

  -ap オプション (並列環境の追加) を使用すると、並列環境の構成テンプレートが含まれるエディタが表示されます。このエディタは、デフォルトの vi エディタか、EDITOR 環境変数により定義されたエディタのいずれかです。*pe-name* で並列環境の名前を指定します。この名前は、テンプレートの対応するフィールドにすでに入力されています。並列環境の構成は、テンプレートを変更してディスクに保存することで行います。変更すべきテンプレートエントリの詳細については、sge_pe(5) のマニュアルページを参照してください。

- qconf -Ap *filename*

  -Ap オプション (ファイルからの並列環境の追加) を使用すると、指定したファイル *filename* が解析され、新しい並列環境の構成が追加されます。

  ファイルは、並列環境の構成テンプレートの形式である必要があります。

- qconf -dp *pe-name*

  -dp オプション (並列環境の削除) を使用すると、指定した並列環境が削除されます。

- qconf -mp *pe-name*

  -mp オプション (並列環境の変更) を使用すると、構成テンプレートとして指定した並列環境を含むエディタが表示されます。このエディタは、デフォルトの vi エディタか、EDITOR 環境変数により定義されたエディタのいずれかです。並列環境の変更は、テンプレートを変更してディスクに保存することで行います。変更すべきテンプレートエントリの詳細については、sge_pe(5) のマニュアルページを参照してください。

- qconf -Mp *filename*

`-Mp` オプション (ファイルからの並列環境の変更) を使用すると、指定したファイル *filename* が解析され、既存の並列環境の構成が変更されます。

ファイルは、並列環境の構成テンプレートの形式である必要があります。

- `qconf -sp` *pe-name*

  `-sp` オプション (並列環境の表示) を使用すると、指定した並列環境の構成が標準出力に出力されます。

- `qconf -spl`

  `-spl` オプション (並列環境リストの表示) を使用すると、現在構成されているすべての並列環境の名前が表示されます。

注 – 並列ジョブを実行するには、キューを並列環境に関連付ける必要があります。`queue_conf(5)` の `pe_list` 属性を使用して、適した並列環境を特定してください。次に、QMON ユーティリティーか次の形式の `qconf` コマンドを使用して、並列環境とキューを関連付けます。

 # qconf -mq <*queue_name***>**

並列環境の起動手続き

Grid Engine システムは、`exec` システムコールを使用して起動手続きを呼び出すことによって並列環境を起動します。起動用の実行可能ファイル名とそのファイルに渡すパラメータは、Grid Engine システムの中から設定することができます。

Grid Engine システムの配布ツリーには、PVE 環境用のこのような起動手続きのサンプルが含まれています。起動手続きは、シェルスクリプト 1 つとそのスクリプトによって呼び出される C プログラム 1 つで構成されています。シェルスクリプトは C プログラムを使用して、PVM をクリーンに起動します。そのほかの必要な処理はすべて、シェルスクリプトが行います。

このシェルスクリプトのパスは *sge-root*`/pvm/startpvm.sh` です。C プログラムファイルのパスは *sge-root* `/pvm/src/start_pvm.c` です。

注 – 起動手続きが、C プログラム 1 つだけであってもかまいません。シェルスクリプトを使用することによって、サンプルの起動手続きのカスタマイズが容易になります。

サンプルスクリプトの `startpvm.sh` には、次の 3 つの引数が必要です。

- Grid Engine ソフトウェアによって生成されたホストファイルのパス (PVM が起動されるホスト名を含む)
- `startpvm.sh` 手続きの起動元のホスト

- 通常は、PVM_ROOT 環境変数に含まれる、PVM ルートディレクトリのパス

これらのパラメータは、176 ページの「QMON を使用した並列環境の構成」で説明されている方法で、起動スクリプトに渡すことができます。実行中、Grid Engine システムによって並列環境の起動および停止スクリプトに提供されるパラメータは、このほかにもあります。たとえば、必要なホストファイルは Grid Engine システムによって生成されます。そのファイル名は、並列環境の構成で特殊パラメータ名 $pe_hostfile を使用して起動手続きに渡すことができます。使用可能なすべてのパラメータの説明は、 sge_pe(5) のマニュアルページに記載されています。

ホストファイルの形式は次のとおりです。

- ファイルの各行は、並列プロセスを実行するキューを表します。
- 各行の最初のエントリはキュー名を示します。
- 2 つ目のエントリは、このキューで実行する並列プロセス数を示します。
- 3 つ目のエントリは、キューを示します。
- 4 つ目のエントリは、マルチプロセッサマシンの場合に使用するプロセッサ範囲を示します。

このファイル形式は Grid Engine システムにより生成されます。このファイル形式は固定されています。異なるファイル形式を必要とする並列環境は、起動手続き内で形式を変換する必要があります。startpvm.sh ファイルを参照してください。異なるファイル形式を必要とする並列環境の例は PVM です。

Grid Engine システムによって並列環境の起動手続きが実行されると、起動手続きにより並列環境が起動されます。起動手続きは、ゼロの終了ステータスで終了するようにします。起動手続きの終了ステータスがゼロ以外の場合、Grid Engine ソフトウェアはエラーを報告し、並列ジョブを実行しません。

注 – 最初は Grid Engine システムを使用せずに、コマンド行から起動手続きをテストする必要があります。こうすることで、手続きを Grid Engine システムの枠組みに組み込んだ場合の、追跡が困難なすべてのエラーを回避できます。

並列環境の終了

並列ジョブが正常終了するか、qdel などによって異常終了すると、その並列環境を停止する手続きが呼び出されます。この手続きの定義と構文は、起動プログラムに関して説明したものと非常によく似ています。停止手続きは、並列環境の構成で定義することもできます。たとえば、176 ページの「QMON を使用した並列環境の構成」を参照してください。

停止手続きの目的は、並列環境を停止して、関係するすべてのプロセスを刈り取ることにあります。

注 – 停止手続きが並列環境プロセスの後処理に失敗すると、Grid Engine システムが、並列環境の制御下で動作しているプロセスの情報を得られないことがあります。このため、停止手続きはこれらのプロセスの後処理を行えません。当然、Grid Engine ソフトウェアは、システムが起動したジョブスクリプトに直接関係するすべてのプロセスの後処理をします。

Grid Engine システムの配布ツリーには、PVM 並列環境用のサンプル停止手続きも含まれています。このサンプルは *sge-root*/pvm/stoppvm.sh の下にあり、次の 2 つの引数を取ります。

- Grid Engine システムによって生成されたホストファイルのパス
- 停止手続きの実行元ホスト名

起動手続きと同様に、停止手続きは、実行成功時にゼロの終了ステータス、失敗時にゼロ以外の終了ステータスを返すとみなされます。

注 – 最初は Grid Engine システムを使用せずに、コマンド行から停止手続きをテストする必要があります。こうすることで、手続きを Grid Engine システムの枠組みに組み込んだ場合の、追跡が困難なすべてのエラーを回避できます。

並列環境と Grid Engine ソフトウェアの密統合

176 ページの「QMON を使用した並列環境の構成」 では、sge_execd および sge_shepherd を使用して並列タスクを作成する方が、独自の並列タスクを作成する並列環境よりメリットが多いと記載されています。これは、UNIX オペレーティングシステムでは、プロセス階層の作成者だけが信頼性の高いリソース制御を行えるためです。並列アプリケーションのための適切なアカウンティング、リソースの利用制限、プロセス制御などの機能は、すべての並列タスクの作成者だけが適用できます。

大部分の並列環境には、こうした機能は実装されていません。このため、Grid Engine システムなどのリソース管理システムと統合するための十分なインタフェースが用意されていません。この問題を克服するために、Grid Engine システムの側で、並列環境と密に統合するための高度な並列環境インタフェースを提供しています。この並列環境インタフェースは、タスクの生成の仕事を並列環境から Grid Engine ソフトウェアに移します。

Grid Engine システムの配布には、PVM のパブリックドメイン版用の例と、Argonne National Laboratories の MPICH MPI 実装用の例の、このような密統合の 2 つの例が用意されています。例は、それぞれディレクトリ *sge-root*/pvm および *sge-root*/mpi に含まれています。これらのディレクトリには、使用方法と現在の制限事項を記した README ファイルも含まれています。詳細については、これらの README ファイルを参照してください。

比較のため、*sge-root*/mpi/sunhpc/loose-integration ディレクトリには、Sun HPC ClusterTools™ ソフトウェアとの疎統合のサンプルが含まれ、*sge-root*/mpi ディレクトリには、比較用に疎統合版のインタフェースが含まれています。

注 – 並列環境との密統合は高度な作業であり、その並列環境と Grid Engine システムの並列環境インタフェースに関する専門的な知識が必要になることがあります。サポートについては、Sun のサポート担当者にお問い合わせください。

チェックポイント設定環境の構成

チェックポイント設定は、次のタスクを実行する機能です。

1. 実行中のジョブまたはアプリケーションの状態を凍結する
2. この状態 (チェックポイント) をディスクに保存する
3. システム停止などの原因で、そのジョブまたはアプリケーションが終了しなかった場合に、そのチェックポイントからジョブまたはアプリケーションを再開する

チェックポイントをあるホストから別のホストに移動すると、チェックポイント設定機能により、リソースをあまり失うことなく、クラスタ内のジョブまたはアプリケーションを移行できます。つまり、チェックポイント設定機能の助けにより、動的な負荷分散を実現できます。

Grid Engine システムは、次の 2 つのレベルのチェックポイント設定をサポートしています。

- ユーザーレベルのチェックポイント設定。

  このレベルでのチェックポイント生成メカニズムは、完全にユーザーまたはアプリケーションの責任で実現します。ユーザーレベルのチェックポイント設定の例には、次のものがあります。

  - アプリケーション再起動時のファイルの適切な処理とともに行われる、重要なアルゴリズムステップでの、アプリケーションで符号化された再起動ファイルの定期的な書き込み。
  - アプリケーションにリンクする必要があり、チェックポイント設定メカニズムをインストールするチェックポイントライブラリの利用。

注 – さまざまなサードパーティ製アプリケーションが、再起動ファイルの書き込みに基づく、統合されたチェックポイント機能を提供します。チェックポイントライブラリは、ハードウェアベンダーまたはパブリックドメインから入手できます。たとえば、University of Wisconsin の Condor プロジェクトにアクセスしてください。

- カーネルレベルの透過的チェックポイント設定。

  このレベルのチェックポイント設定は、任意のジョブに適用できるオペレーティングシステム (またはその拡張機能) によって実現する必要があります。カーネルレベルのチェックポイント設定を使用するために、ソースコードを変更したり、アプリケーションを再リンクしたりする必要はありません。

カーネルレベルのチェックポイント設定は、ジョブ全体、すなわち、ジョブによって作成されたプロセス階層に適用できます。対照的に、ユーザーレベルのチェックポイント設定は、通常単一プログラムに制限されます。つまり、そうしたプログラムが埋め込まれているジョブは、ジョブ全体を再開した場合に、この問題に正しく対処する必要があります。

チェックポイント設定ライブラリに基づくチェックポイント設定ばかりでなく、カーネルレベルのチェックポイント設定は、非常に多くのリソースを消費する可能性があります。チェックポイント生成時にジョブまたはアプリケーションが使用している仮想アドレス空間全体をディスクにダンプしなければならないためです。これに比べて、再起動ファイルに基づくユーザーレベルのチェックポイント設定では、チェックポイントに書き込むデータを重要情報に関するものだけに制限することができます。

チェックポイント設定環境について

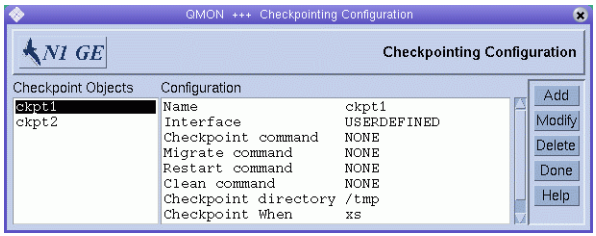
Grid Engine システムでは、使用する各チェックポイント設定の方法に関する、構成可能な属性記述が用意されています。さまざまな属性記述は、さまざまなチェックポイント設定方法だけでなく、さまざまなオペレーティングシステムアーキテクチャーでそれらの方法から派生する可能性がある多様な方法も反映します。

この属性記述をチェックポイント設定環境と呼びます。Grid Engine システムの配布には、デフォルトのチェックポイント設定環境が用意されており、サイトのニーズに応じて変更することができます。

原理的には、新しいチェックポイント設定方法を組み込むこともできます。ただし、新しい方法の組み込みは難しい作業になる可能性があります。このような組み込み作業は、経験の豊富なスタッフまたは Grid Engine システムサポートチームだけが行うようにしてください。

QMON を使用したチェックポイント設定環境の構築

「QMON Main Control」ウィンドウで、「Checkpoint Configuration」ボタンをクリックします。「Checkpoint Configuration」ダイアログボックスが表示されます。

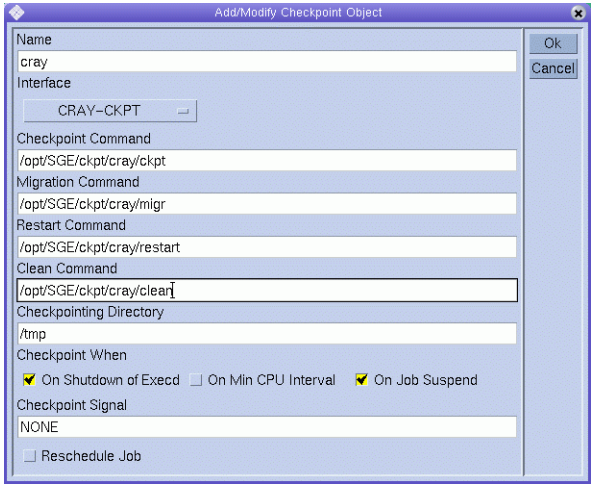


構成済みのチェックポイント設定環境の表示

以前に構成したチェックポイント設定環境を表示するには、「Checkpoint Objects」の下に表示されているチェックポイント設定環境名のいずれかを選択します。「Configuration」の下に、対応する構成が表示されます。

チェックポイント設定環境の追加

「Checkpointing Configuration」ダイアログボックスで「Add」をクリックします。ユーザーが編集可能なテンプレート構成とともに「Add/Modify Checkpoint Object」ダイアログボックスが表示されます。



テンプレートに必要な情報を入力します。

変更を sge_qmaster に登録するには「OK」をクリックします。変更を保存せずにダイアログボックスを閉じるには、「Cancel」をクリックします。

チェックポイント設定環境の変更

「Checkpoint Objects」リストで、変更する構成済みのチェックポイント設定環境の名前を選択してから「Modify」をクリックします。選択したチェックポイント設定環境の現在の構成とともに「Add/Modify Checkpoint Object」ダイアログボックスが表示されます。

「Add/Modify Checkpoint Object」ダイアログボックスでは次の情報を変更できます。

- 名前
- 「Checkpoint Command」、「Migration Command」、「Restart Command」および「Clean Command」の文字列
- チェックポイント設定ファイルを格納するディレクトリ
- チェックポイントを開始しなければならない時点
- チェックポイントの開始時点でジョブまたはアプリケーションに送信するシグナル。

これらのパラメータの詳細については、`checkpoint`(5) のマニュアルページを参照してください。

また、使用するインタフェースを定義する必要があります。インタフェースはチェックポイント設定方法とも呼ばれます。「Name」の下にある「Interface」リストからインタフェースを選択します。さまざまなインタフェースの意味の詳細については、`checkpoint` (5) のマニュアルページを参照してください。

注 – Grid Engine システムの配布に付属するチェックポイント設定環境の場合は、「Name」のパラメータと「Checkpointing Directory」のパラメータのみ変更します。

変更を `sge_qmaster` に登録するには「OK」をクリックします。変更を保存せずにダイアログボックスを閉じるには、「Cancel」をクリックします。

チェックポイント設定環境の削除

構成済みのチェックポイント設定環境を削除するには、チェックポイント設定環境を選択してから「Delete」をクリックします。

コマンド行からのチェックポイント設定環境の構成

コマンド行からチェックポイント設定環境を構成するには、適切なオプションを使用して `qconf` コマンドを入力します。

次のオプションを使用できます。

- `qconf -ackpt` *ckpt-name*

  `-ackpt` オプション (チェックポイント設定環境の追加) を使用すると、チェックポイント設定環境の構成テンプレートが含まれるエディタが表示されます。このエディタは、デフォルトの `vi` エディタか、`EDITOR` 環境変数に対応するエディタのいずれかです。パラメータ *ckpt-name* で、チェックポイント設定環境の名前を指定します。このパラメータは、テンプレートの対応するフィールドにすでに入力されています。チェックポイント設定環境の構成は、テンプレートを変更してディスクに保存することで行います。変更すべきテンプレートエントリの詳細については、 `checkpoint`(5) のマニュアルページを参照してください。

- `qconf -Ackpt` *filename*

  `-Ackpt` オプション (ファイルからのチェックポイント設定環境の追加) を使用すると、指定したファイルが解析され、新しいチェックポイント設定環境の構成が追加されます。

  このファイルは、チェックポイント設定環境テンプレートの形式を有する必要あります。

- `qconf -dckpt` *ckpt-name*

  `-dckpt` オプション (チェックポイント設定環境の削除) を使用すると、指定したチェックポイント設定環境が削除されます。

- `qconf -mckpt` *ckpt-name*

  `-mckpt` オプション (チェックポイント設定環境の変更) を使用すると、構成テンプレートとして、指定したチェックポイント設定環境が含まれるエディタが表示されます。このエディタは、デフォルトの `vi` エディタか、`EDITOR` 環境変数に対応するエディタのいずれかです。チェックポイント設定環境の変更は、テンプレートを変更してディスクに保存することで行います。変更すべきテンプレートエントリの詳細については、`checkpoint`(5) のマニュアルページを参照してください。

- `qconf -Mckpt` *filename*

  `-Mckpt` オプション (ファイルからのチェックポイント設定環境の変更) を使用すると、指定したファイルが解析され、既存のチェックポイント設定構成が変更されます。

  このファイルは、チェックポイント設定環境テンプレートの形式を有する必要あります。

- `qconf -sckpt` *ckpt-name*

  `-sckpt` オプション (チェックポイント設定環境の表示) を使用すると、指定したチェックポイント設定環境の構成が標準出力に出力されます。

- `qconf -sckptl`

  `-sckptl` オプション (チェックポイント設定環境リストの表示) を使用すると、現在構成されているすべてのチェックポイント設定環境の名前のリストが表示されます。

◆ ◆ ◆ 第 8 章

そのほかの管理作業

この章では、ファイルやスクリプトを使用して、キュー、ホスト、および環境などの Grid Engine システムのオブジェクトを追加または変更する方法について説明します。

この章で説明する内容は、次のとおりです。

アカウンティングおよびレポート統計の収集

Grid Engine システムには、2 種類のレポートとアカウンティングの機能が用意されています。

- アカウンティングおよびレポートコンソール (ARCo)
- `qacct` コマンドおよび `accounting` ファイル

レポート統計 (ARCo)

オプションのアカウンティングおよびレポートコンソール (ARCo) を使用すると、Grid Engine システムから実際のアカウンティングとレポートのデータを収集し、標準的な SQL データベースであるレポートデータベースにデータを格納できます。ARCo は、次の SQL データベースシステムをサポートしています。

- PostgreSQL
- Oracle
- MySQL

また ARCo には、レポートデータベースに関する情報クエリを収集し、表形式またはグラフィック形式で結果を取得するための、Web ベースのツールが用意されてい

ます。ARCo を使用すると、あとに使用できるようクエリを格納し、定義済みのクエリを実行し、バッチモードでクエリを実行することができます。ARCo の使用法の詳細については、『Sun N1 Grid Engine 6.1 ユーザーズガイド』の第 5 章「アカウンティングとレポート」を参照してください。ARCo のインストール方法の詳細については、『Sun N1 Grid Engine 6.1 インストールガイド』の第 8 章「アカウンティングおよびレポートコンソールのインストール」を参照してください。

生のレポートデータは sge_qmaster により生成されます。この raw データはレポートファイルに格納されます。dbwriter プログラムはレポートファイルの raw データを読み取り、SQL レポートデータベースにそのデータを書き込みます。SQL レポートデータベースは ARCo によりアクセス可能です。

dbwriter プログラムについて

dbwriter プログラムは、次のタスクを実行します。

- レポートファイルから raw データを読み取り、この raw データをレポートデータベースに書き込む。
- 派生値を計算する。ユーザーは、計算を制御する規則だけでなく、どの値を計算するかを構成できます。
- 期限切れのデータを削除する。ユーザーは、データを保持する期間を構成できます。

dbwriter は、起動時に、派生値を計算します。また、dbwriter は起動時に、期限切れレコードの削除も行います。dbwriter が継続モードで実行中である場合、dbwriter は継続して、1 時間ごとの間隔またはユーザーが指定する間隔で、派生値を計算し、期限切れレコードを削除します。

XML ファイルで、計算する値と削除するレコードを指定できます。この XML ファイルのパスを指定するには、dbwriter コマンドの -calculation オプションを使用します。

派生値の計算の詳細については、193 ページの「dbwriter を使用した派生値の計算」を参照してください。

期限切れレコードの削除の詳細については、196 ページの「dbwriter を使用した期限切れレコードの削除」を参照してください。

レポートファイルの有効化

レポートファイルには次の種類のデータが含まれています。

- ホスト負荷値および消費可能リソース
- キュー消費可能リソース
- ジョブのログ記録
- ジョブのアカウンティング

- 共有ツリーの使用率

Grid Engine システムを最初にインストールした時点では、レポートファイルは使用不可になっています。ARCo を使用するためには、クラスタに対してレポートファイルを使用可能にする必要があります。使用可能にすると、レポートファイルは `sge_qmaster` により生成されます。デフォルトでは、レポートファイルの位置は *sge-root*/ *cell*/`common` です。`dbwriter` コマンドの `-reporting` オプションを使用すると、デフォルトを変更できます。

レポートファイルの生成の構成については、`sge_conf`(5) のマニュアルページの `reporting_params` パラメータ、および `host_conf`(5) のマニュアルページの `report_variables` パラメータを参照してください。

`QMON` でレポートファイルを使用可能にするには、「Main Control」ウィンドウで「Cluster Configuration」ボタンをクリックし、「`global`」ホストを選択してから「Modify」をクリックします。

「Cluster Settings」ダイアログボックスで「Advanced Settings」タブをクリックします。

「Reporting Parameters」フィールドでは次のパラメータを設定します。

- `accounting` を `true` に設定します。`true` がデフォルト値です。
- `reporting` を `true` に設定します。
- `flush_time` を `00:00:15` に設定します。`00:00:15` がデフォルト値です。
- `joblog` を `true` に設定します。
- `sharelog` を `00:00:00` に設定します。`00:00:00` がデフォルト値です。

コマンド行からレポートファイルを使用可能にするには、上記の説明のように、`qconf –mconf` コマンドを使用して `reporting_params` 属性を設定します。

レポートファイルが使用可能になれば、`dbwriter` はレポートファイルから raw データを読み取り、それをレポートデータベースに書き込むことができるようになります。

レポートファイルの構成の詳細については、 `reporting`(5) のマニュアルページを参照してください。ARCo のインストールと設定の詳細については、『Sun N1 Grid Engine 6.1 インストールガイド』の第 8 章「アカウンティングおよびレポートコンソールのインストール」を参照してください。

dbwriter を使用した派生値の計算

派生値を計算するための規則は、`DbWriterConfig` タグのサブタグである派生タグで指定します。次の表に、派生タグの属性を示します。

属性	説明
`object`	データを集計する対象のオブジェクト。オブジェクトは次のいずれかです。

- ホスト
- キュー
- プロジェクト
- 部署
- ユーザー
- グループ

`interval` 派生値を計算する頻度を指定する時間範囲です。時間範囲は次のいずれかです。

- `hour`
- `day`
- `month`
- `year`

`variable` 計算済みのデータを保持する変数の名前です。

次の表に、派生タグのサブ要素を示します。

`sql` 派生値を計算する SQL 文です。この文は次のカラムを作成する必要があります。

- `time_start` – `time_end` とともに、計算済みの値の期間を指定します。
- `time_end`
- `value` – 計算済みの派生値

SQL 文は次のプレースホルダを含むことができます。`dbwriter` は、規則に基づいて各クエリのプレースホルダを次のように置換します。

- `__time_start__` – クエリーの開始時間。`dbwriter` は、この規則から、直前に計算された最後の派生値を検索し、このタイムスタンプを次のクエリーの開始時間として使用します。
- `__time_end__` – クエリーの終了時間。このタイムスタンプは、最後に経過した時間範囲の終了を指定します。たとえば、時間範囲が `day` であり、派生値が `00:30` で計算された場合、`00:00` が `time_end` として取得されます。
- `__key_0__`, `__key_1__`, . . , `__key_n__` – 指定したオブジェクト型の主キーのコンポーネント。たとえば、`sge_hosts` テーブルには主キー `h_hostname` があります。ホストオブジェクト型に対して規則が処理された場合、`sge_hosts` テーブル内のエントリごとに 1 つのクエリーが実行され、SQL 文内の `__key_0__` プレースホルダはホスト名により置き換えられます。

  `sge_queue` テーブルは、`q_qname` および `q_hostname` から構成された複合的な主キーを持ちます。

`auto` `dbwriter` は、派生値の計算用の SQL 文を生成します。

自動生成された SQL 文は、次のテンプレートのようになります。

```
SELECT time_start, time_end, <function>(<value_field>) as value
FROM ( SELECT TRUNC( <timestart_field>, <interval> ) as time_start
              TRUNC( <timestart_field>, <interval> ) +
                    INTERVAL '1' <interval> as time_end,
              <value_field>
       FROM <object value table>
       WHERE <primary key field 0> = __key_0__
         AND <primary key field 1> = __key_1__
         AND . . .
         AND <parent key field> =
             (SELECT <parent key field> FROM <parent table>
              WHERE <parent filter> )
         AND <timestart_field> <= {ts __time_start__ }
         AND <timeend_field>   >  {ts __time_end__ }
GROUP BY time_start, time_end
```

SQL テンプレートパラメータには次のものがあります。

パラメータ	説明
<function>	派生値を計算するための集計関数です。XML ファイルの auto タグの関数属性に由来します。
<value_field>	派生値のオブジェクトに依存します。
<timestart_field>	派生値のオブジェクトに依存します。
<timeend_field>	派生値のオブジェクトに依存します。
<interval>	派生タグの内部属性に由来します。
<object value table>	値が格納されるデータベーステーブルの名前です。オブジェクトに依存します (host => host_values, user => user_values, .)。
<primary key field n>	値テーブルを親テーブルに結合するのに必要な主キーです。オブジェクトに依存します。
<parent key field>	親の ID を保持するフィールドの名前です。オブジェクトに依存します。
<parent table>	親データベーステーブルの名前です。オブジェクトに依存します (host => host, user => user)。
<parent filter>	親テーブル用のフィルタです。親テーブルの各エントリの派生値が計算されます (例: u_user = 'user1')。

自動生成された SQL 文の例を次に示します。

```
<derive object="host" interval="day" variable="d_load">
   <auto function="AVG" variable="h_load" />
```

```
</derive>

SELECT time_start, time_end, AGE(hv_dvalue)
FROM ( SELECT TRUNC( hv_time_start, 'day' ) as time_start,
              TRUNC( hv_time_start, 'day' ) +
                            INTERVAL '1' day as time_end,
              hv_dvalue
       FROM sge_host_values
       WHERE hv_variable = 'h_load' AND
             hv_parent =
               (SELECT h_id FROM sge_host
                WHERE h_hostname = 'foo.bar') AND
             hv_time_start <= {ts '2004-05-21 00:00:00.0'} AND
             hv_time_end    > {ts '2004-05-17 00:00:00.0'} )
GROUP BY time_start, time_end
```

dbwriter を使用した期限切れレコードの削除

レポートデータベースで期限切れレコードを削除するには、削除タグで削除規則を指定する必要があります。次の表に、削除タグの属性を示します。

属性	説明
scope	削除するデータの型です。有効なエントリには次のものがあります。 ■ job ■ job_log ■ share_log ■ host_values ■ queue_values ■ project_values ■ department_values ■ user_values ■ group_values
time_range	time_amount の単位です。
time_amount	レコードを保持する単位 (time_range) の数です。

次の表に、削除タブのサブ要素を示します。

sub_scope	一部のスコープに関しては、サブスコープを構成できます。サブスコープは、削除の追加条件を指定します。サブスコープは、すべての *_values スコープと、share_log スコープに対して構成できます。 サブスコープが *_values 規則に対して構成されている場合、そのサブスコープには、空白で区切られた、削除する変数のリストが含まれています。

サブスコープが share_log に対して指定されている場合、そのサブスコープには、空白で区切られた、削除する共有ツリーノードのリストが含まれています。

サブスコープを使用する場合、サブスコープにより明示的に名前が付けられていないすべてのオブジェクトを削除する、サブスコープのないフォールバック規則を常に用意する必要があります。

削除タグの例を次に示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<DbWriterConfig>
  <!-- keep host values for 2 years -->
  <delete scope="host_values" time_range="year" time_amount="2"/>

  <!-- keep queue values one month -->
  <delete scope="queue_values" time_range="month" time_amount="1">
    <sub_scope>slots</sub_scope>
    <sub_scope>state</sub_scope>
  </delete>
</DbWriterConfig>
```

アカウンティングおよび使用率の統計 (qacct)

qacct コマンドを使用すると、英数字のアカウンティング統計を生成できます。オプションを指定しないと、qacct により、完了したすべてのジョブによって生成され、クラスタアカウンティングファイル *sge-root*/*cell*/common/accounting に含まれている、クラスタのすべてのマシンに関する使用率集計情報が表示されます。この場合、qacct は秒単位で次の 3 つの時間を報告します。

- REAL 時間 – ジョブの開始と終了までの時計時間
- USER 時間 – ユーザープロセスで費やされた CPU 時間
- SYSTEM 時間 – システムコールで費やされた CPU 時間

キューあるいはユーザーなどに関するアカウンティング情報の報告に、いくつかのオプションを使用可能です。特に、qacct -l コマンドを使用すると、すでに完了していて、リソース要求指定に一致するすべてのジョブに関する情報を要求できます。

qacct -j [*job-id* | *job-name*] コマンドを使用すると、Grid Engine システムによって格納された、リソース使用率情報の全体に直接アクセスできます。 この情報には、getrusage システムコールにより提供された情報が含まれています。

-j オプションは、*job-id* または *job-name* を持つジョブのリソース使用率エントリを報告します。引数なしで実行された場合は、参照されたアカウンティングファイルに含まれるすべてのジョブが表示されます。ジョブ ID が指定され、複数のエントリが表示された場合は、次のいずれかが真になります。

- ジョブ ID 番号が折り返される。ジョブ ID の範囲は 1 から 999999 です。
- 移動したチェックポイント設定ジョブが表示される。

詳細については、qacct(1) のマニュアルページを参照してください。

Grid Engine システム構成のバックアップ

Grid Engine システムの構成ファイルを自動的にバックアップすることができます。自動バックアッププロセスでは、backup_template.conf という名前の構成ファイルを使用します。デフォルトでは、バックアップ構成ファイルの位置は *sge-root*/util/install_modules/backup_template.conf です。

バックアップ構成ファイルでは、次の要素を定義する必要があります。

- Grid Engine システムのルートディレクトリ。
- Grid Engine システムのセルディレクトリ。
- Grid Engine システムのバックアップディレクトリ。
- バックアップの種類。Grid Engine システムの構成ファイルを単にバックアップするだけでなく、バックアップを、構成ファイルが含まれる圧縮 tar ファイルにすることもできます。
- バックアップファイルのファイル名。

バックアップテンプレートファイルは、次の例のようになります。

```
##############################################
# Autobackup Configuration File Template
##############################################

# Please, enter your SGE_ROOT here (mandatory)
SGE_ROOT=""

# Please, enter your SGE_CELL here (mandatory)
SGE_CELL=""

# Please, enter your Backup Directory here
# After backup you will find your backup files here (mandatory)
# The autobackup will add a time /date combination to this dirname
# to prevent an overwriting!
BACKUP_DIR=""
```

```
# Please, enter true to get a tar/gz package
# and false to copy the files only (mandatory)
TAR="true"

# Please, enter the backup file name here. (mandatory)
BACKUP_FILE="backup.tar"
```

自動バックアッププロセスを開始するには、sge_qmaster ホストで次のコマンドを入力します。

```
inst_sge -bup -auto backup-conf
```

backup-conf は、バックアップ構成ファイルのフルパスです。

注 – 構成ファイルをバックアップする前に Grid Engine システムデーモンを停止する必要はありません。

バックアップは、BACKUP_FILE により指定されたディレクトリに作成されます。このディレクトリには、install.*pid* という名前のバックアップログファイルも作成されます。*pid* はプロセス ID 番号です。

▼ 手動バックアップを実行する

1 次のコマンドを入力して、手動バックアップを開始します。

```
inst_sge -bup
```

2 <*sge-root*> ディレクトリを入力するか、デフォルトをそのまま使用します。

```
SGE Configuration Backup
------------------------

This feature does a backup of all configuration you made
within your cluster.
Please enter your SGE_ROOT directory.
Default: [/home/user/ts/u10]
```

3 <*sge-cell*> 名を入力するか、デフォルトをそのまま使用します。

```
Please enter your SGE_CELL name. Default: [default]
```

4 バックアップの作成先のディレクトリを入力するか、デフォルトをそのまま使用します。

```
Where do you want to save the backupfiles?
Default: [/home/user/ts/u10/backup]
```

5 圧縮形式の **tar** バックアップファイルを作成するかどうかを選択します。

使用している tar のバージョン (gnu tar/ solaris tar) が異なると、このオプションで問題が起きることがあります。場合によっては、tar パッケージが壊れていることがあります。圧縮および圧縮解除を問題なく行うには、同じ tar バイナリを使用してください。

```
Shall the backup function create a compressed tarpackage with your files? (y/n) [y] >>
```

6 バックアップファイルの名前を入力します。デフォルトが示されます。

```
... starting with backup


Please enter a filename for your backupfile. Default: [backup.tar] >>
```

7 バックアップが実行され、その情報が表示されます。

```
2007-01-11_22_43_22.dump
bootstrap
qtask
settings.sh
act_qmaster
sgemaster
settings.csh
sgeexecd
jobseqnum

... backup completed
All information is saved in
[/home/user/ts/u10/backup/backup.tar.gz[Z]]
```

▼ バックアップから復元する

始める前に 復元作業を開始する前に `qmaster` デーモンを停止してください。復元中にスプーリングデータベースが変更され、復元機能と qmaster が同じデータにアクセスしようとしていると、データが失われることがあります。

1 次のコマンドを入力して、復元作業を開始します。

```
inst_sge -rst
```

2 画面のメッセージを読んで、対応します。

```
SGE Configuration Restore
-------------------------

This feature restores the configuration from a backup you made
previously.

Hit, <ENTER> to continue!
```

3 <*sge-root*> ディレクトリを入力するか、デフォルトをそのまま使用します。

```
Please enter your SGE_ROOT directory.
Default: [/home/user/ts/u10]
```

4 <*sge-cell*> 名を入力するか、デフォルトをそのまま使用します。

```
Please enter your SGE_CELL name. Default: [default]
```

5 バックアップファイルの形式を確認します。
バックアップファイルが、圧縮形式の tar ファイルでないことがあります。

```
Is your backupfile in tar.gz[Z] format? (y/n) [y]
```

6 バックアップファイルへのフルパスを入力します。

```
Please enter the full path and name of your backup file.
Default: [/home/user/ts/u10/backup/backup.tar.gz]
```

7 スプーリングデータベースの情報を確認します。
復元機能はバックアップファイルを圧縮解除し、システム情報を読み取ります。データが失われないようにするには、適切なスプーリングデータベースが検出されていることを確認します。

```
Copying backupfile to /tmp/bup_tmp_22_51_40

/home/user/ts/u10/backup/backup.tar.gz
2007-01-11_22_43_22.dump
bootstrap
qtask
settings.sh
act_qmaster
sgemaster
settings.csh
sgeexecd
jobseqnum

Spooling Method: berkeleydb detected!
```

```
The path to your spooling db is [/tmp/dom/spooldb]
If this is correct hit <ENTER> to continue, else enter the path. >>
```

8 qmaster を再起動します。

管理作業用のファイルとスクリプトの使用法

この節では、ファイルとスクリプトを使用して、キュー、ホスト、環境などの Grid Engine システムのオブジェクトを追加または変更する方法を説明します。

Grid Engine システムでのすべての管理作業の実行には、QMON グラフィカルユーザーインタフェースを使用できます。また、シェルプロンプトで入力し、シェルスクリプト内から呼び出すコマンドを介して、Grid Engine システムを管理することもできます。多くの経験豊富な管理者にとっては、ファイルとスクリプトを使用する方が、設定を変更するより柔軟、迅速、かつ強力な手段です。

ファイルを使用したオブジェクトの追加または変更

ファイル内に作成した指定に従ってオブジェクトを追加するには、次のオプションを付けて qconf コマンドを使用します。

```
qconf -Ae
qconf -Aq
qconf -Au
qconf -Ackpt
qconf -Ap
```

ファイル内に作成した指定に従ってオブジェクトを変更するには、次のオプションを付けて qconf コマンドを使用します。

```
qconf -Me
qconf -Mq
qconf -Mu
qconf -Mckpt
qconf -Mp
```

–Ae および –Me オプションで、実行ホストの追加または変更を行います。

–Aq および –Mq オプションで、キューの追加または変更を行います。

–Au および –Mu オプションで、ユーザーセットの追加または変更を行います。

–Ackpt および –Mckpt オプションで、チェックポイント設定環境の追加または変更を行います。

–Ap および –Mp オプションで、並列環境の追加または変更を行います。

qconf –s コマンドとこれらのオプションを組み合わせて使用し、既存のオブジェクトを取得し、それを変更します。既存のオブジェクトを更新したり、新しいオブジェクトを作成することができます。

例 8–1 チェックポイント環境の移動コマンドの変更

```
#!/bin/sh
# ckptmod.sh: modify the migration command
# of a checkpointing environment
# Usage: ckptmod.sh <checkpoint-env-name> <full-path-to-command>
TMPFILE=tmp/ckptmod.$$

CKPT=$1
MIGMETHOD=$2

qconf -sckpt $CKPT | grep -v '^migr_command' > $TMPFILE
echo "migr_command $MIGMETHOD" >> $TMPFILE
qconf -Mckpt $TMPFILE
rm $TMPFILE
```

ファイルを使用した、キュー、ホスト、および環境の変更

コマンド行から、個別のキュー、ホスト、並列環境、およびチェックポイント設定環境を変更することができます。そのほかのコマンドと組み合わせて qconf コマンドを使用します。

- すでにファイルを用意している場合は、次のように適切なオプションを付けて qconf コマンドを入力します。

```
qconf -Me
qconf -Mq
qconf -Mckpt
qconf -Mp
```

- ファイルを用意していない場合は、次のように適切なオプションを付けて qconf コマンドを入力します。

```
qconf -me
qconf -mq
qconf -mckpt
qconf -mp
```

–Me および –me オプションで、実行ホストを変更します。

–Mq および –mq オプションで、キューを変更します。

–Mckpt および –mckpt オプションで、チェックポイント設定環境を変更します。

–Mp および –mp オプションで、並列環境を変更します。

大文字の –M オプションと小文字の –m オプションの違いは、qconf コマンドの結果に影響します。–M と –m は両方とも「modify (変更)」を意味しますが、大文字の –M は既存のファイルからの変更を表すのに対し、小文字の –m は既存のファイルからの変更ではありません。その代わりに、小文字の –m は、エディタで一時ファイルを開きます。このファイルに対して行なった変更を保存し、エディタを終了すると、システムではこれらの変更がただちに反映されます。

ただし、多くのオブジェクトを一度に変更する場合や、対話形式を使用せずにオブジェクト構成を変更する場合は、オブジェクト属性を変更するオプション (–Aattr、–Mattr など) を付けて qconf コマンドを使用します。

次のコマンドは、ファイルでの指定に従って変更を行います。

```
qconf -Aattr {queue | exechost | pe | ckpt} filename
qconf -Mattr {queue | exechost | pe | ckpt} filename
qconf -Rattr {queue | exechost | pe | ckpt} filename
```

```
qconf -Dattr {queue | exechost | pe | ckpt} filename
```

次のコマンドは、コマンド行での指定に従って変更を行います。

```
qconf -aattr {queue | exechost | pe | ckpt} attribute value {queue-list | host-list}
qconf -mattr {queue | exechost | pe | ckpt} attribute value {queue-list | host-list}
qconf -rattr {queue | exechost | pe | ckpt} attribute value {queue-list | host-list}
qconf -dattr {queue | exechost | pe | ckpt} attribute value {queue-list | host-list}
```

–Aattr および –aattr オプションで、属性を追加します。

–Mattr および –mattr オプションで、属性を変更します。

–Rattr および –rattr オプションで、属性を置換します。

–Dattr および –dattr オプションで、属性を削除します。

filename は、属性値のペアが含まれるファイルの名前です。

attribute は、変更するキューまたはホストの属性です。

value は、変更する属性の値です。

–aattr、–mattr、および –dattr オプションを使用すると、値のリストにある個別の値を操作できます。–rattr オプションにより、値のリスト全体が、コマンド行またはファイルで指定した新しいリストに置き換えられます。

例 8–2 キューの型の変更

次のコマンドにより、tcf27–e019.q のキューの型が、バッチのみに変更されます。

```
% qconf -rattr queue qtype batch tcf27-e019.q
```

例 8–3 キューの型とシェル起動動作の変更

次のコマンドでは、ファイル new.cfg を使用して、tcf27–e019.q のキューの型とシェル起動動作を変更します。

```
% cat new.cfg
qtype batch interactive checkpointing
shell_start_mode unix_behavior
% qconf -Rattr queue new.cfg tcf27-e019.q
```

例 8–4 リソース属性の追加

次のコマンドにより、1000M の値を持つリソース属性 scratch1 と、2 の値を持つリソース属性 long が追加されます。

```
% qconf -rattr exechost complex_values scratch1=1000M,long=2 tcf27-e019
```

例 8–5 リソース属性のホストへの関連付け

次のコマンドにより、リソース属性 short が、値 4 を持つホストに関連付けられます。

```
% qconf -aattr exechost complex_values short=4 tcf27-e019
```

例 8–6 リソース値の変更

次のコマンドにより、scratch1 の値が 500M に変更されますが、そのほかの値は変更されません。

```
% qconf -mattr exechost complex_values scratch-=500M tcf27-e019
```

例 8–7 リソース属性の削除

次のコマンドにより、リソース属性 long が削除されます。

```
% qconf -dattr exechost complex_values long tcf27-e019
```

例 8–8 チェックポイント設定環境用のキューのリストへのキューの追加

次のコマンドにより、チェックポイント設定環境 sph 用のキューのリストに、tcf27–b011.q が追加されます。

```
% qconf -aattr ckpt queue_list tcf27-b011.q sph
```

例 8–9 並列環境でのスロット数の変更

次のコマンドにより、並列環境 make のスロット数が 50 に変更されます。

```
% qconf -mattr pe slots 50 make
```

qselect コマンドを使用したキューインスタンスの対象設定

qselect コマンドにより、キューインスタンスのリストが出力されます。オプションを指定した場合、qselect では、ユーザーが指定した条件に一致するキューインスタンスのみが表示されます。qconf コマンドと qselect を組み合わせて使用すると、変更する特定のキューインスタンスの対象設定を行うことができます。

例 8–10 キューの表示

次のコマンドにより、Linux マシン上のすべてのキューインスタンスが表示されます。

```
% qselect -l arch=glinux
```

次のコマンドにより、CPU を 2 つ搭載したマシン上のすべてのキューインスタンスが表示されます。

```
% qselect -l num_proc=2
```

次のコマンドにより、CPU を 4 つ搭載した 64 ビットの全 Solaris マシン上のすべてのキューインスタンスが表示されます。

```
% qselect -l arch=solaris64,num_proc=4
```

次のコマンドにより、アプリケーションライセンスを提供するキューインスタンスが表示されます。キューインスタンスは、事前に構成されています。

```
% qselect -l app_lic=TRUE
```

qselect と qconf を組み合わせることで、1 つのコマンド行を使用して広範囲の変更を行うことができます。このためには、qselect コマンド全体を逆引用符 (‘ ‘) 内に入れ、それを qconf コマンド行の queue-list 変数の位置で使用します。

例 8–11 qconf コマンドでの qselect の使用法

次のコマンドにより、Solaris マシン上のすべてのキューインスタンスで、prolog スクリプトが sol_prolog.sh に設定されます。

```
% qconf -mattr queue prolog /usr/local/scripts/sol_prolog.sh ‘qselect -l arch=solaris‘
```

次のコマンドにより、2 プロセッサシステム上のすべてのキューインスタンスで、属性 fluent_license が 2 に設定されます。

```
% qconf -mattr queue complex_values fluent_license=2 ‘qselect -l num_proc=2‘
```

キューインスタンスの構成を自動化するもっとも柔軟性が高い方法は、qconf コマンドと qselect コマンドを併用する方法です。これらのコマンドを組み合わせることで、独自のカスタム管理スクリプトを構築できます。

ファイルを使用したグローバル構成またはスケジューラの変更

グローバル構成を変更するには、qconf –mconf コマンドを使用します。スケジューラを変更するには、qconf –msconf コマンドを使用します。

これらのコマンドにより、いずれもエディタで一時ファイルが開かれます。エディタを終了すると、この一時ファイルに保存したすべての変更がシステムにより処理され、ただちに変更が有効になります。一時ファイルを開くのに使用されるエディタは、EDITOR 環境変数により指定されたエディタです。 この変数が定義されていない場合、デフォルトでは vi エディタが使用されます。

EDITOR 環境変数を使用すると、qconf コマンドの動作を自動化できます。この変数の値を変更して、最初の引数により名前が指定されているファイルを変更するエディタプログラムを指定します。エディタで一時ファイルを変更し、エディタを終了すると、システムにより変更が読み込まれ、変更はただちに有効になります。

注 – エディタ操作のあとファイルの変更時間が変更されないと、システムでは誤って、そのファイルが変更されなかったとみなされる場合があります。そのため、変更時間を異なるようにするため、ファイルに書き込む前に sleep 1 命令を挿入する必要があります。

この手法は、あらゆる qconf –m... コマンドに使用できます。特にスケジューラとグローバル構成の管理に便利です。これは、別の方法では手続きを自動化できないためです。

例 8–12 スケジュール間隔の変更

次の例では、スケジューラのスケジュール間隔を変更します。

```
#!/bin/ksh
# sched_int.sh: modify the schedule interval
# usage: sched_int.sh <n>, where <n> is
# the new interval, in seconds. n < 60

TMPFILE=/tmp/sched_int.$$
if [ $MOD_SGE_SCHED_INT ]; then
     grep -v schedule_interval $1 > $TMPFILE
```

例 8–12 スケジュール間隔の変更 *(続き)*

```
     echo "schedule_interval 0:0:$MOD_SGE_SCHED_INT" >> $TMPFILE
# sleep to ensure modification time changes
     sleep 1
     mv $TMPFILE $1
else
     export EDITOR=$0
     export MOD_SGE_SCHED_INT=$1
     qconf -msconf
fi
```

このスクリプトは、自分自身を指定するよう EDITOR 環境を変更します。続いてスクリプトは qconf –msconf コマンドを呼び出します。この 2 番目のスクリプトの入れ子呼び出しは、最初の引数により指定された一時ファイルを変更してから終了します。Grid Engine システムにより自動的に変更が読み込まれ、スクリプトの最初の呼び出しが終了します。

◆ ◆ ◆ 第 9 章

9 チューニング、エラーメッセージ、および障害追跡

この章では、Grid Engine システム環境をチューニングするいくつかの方法を説明します。また、エラーメッセージの通知方法と、よくあるさまざまな問題の解決方法に関するヒントを説明します。

この章で説明する内容は、次のとおりです。

グリッド環境のチューニング

Grid Engine システムは、完全な機能を有する、汎用分散リソース管理ツールです。システムのスケジューラコンポーネントは、幅広いさまざまな計算ファームのシナリオをサポートしています。計算環境から最大限のパフォーマンスを引き出すには、使用可能になっている機能を調べる必要があります。続いて、負荷管理問題を解決するためには実際にどの機能が必要であるかを決定する必要があります。これらの機能の一部を使用不可にすることで、クラスタのスループットのパフォーマンスが向上する可能性があります。

スケジューラ監視

スケジューラ監視は、一部のジョブが振り分けられなかった理由を調べる手助けになります。ただし、すべてのジョブに対して常にこの情報を提供すると、リソースを消費する可能性があります。通常は、この情報はそれほど必要ありません。

スケジューラ監視を使用不可にするには、スケジューラ構成で `schedd_job_info` を false に設定します。128 ページの「QMON を使用したスケジューラ構成の変更」、および `sched_conf` (5) のマニュアルページを参照してください。

完了ジョブ

配列ジョブの場合、qmaster の完了ジョブのリストは非常に大きなサイズになることがあります。qstat は完了ジョブのリストの取得も行うため、完了ジョブのリストを無効化すると、メモリーが節約され、qstat プロセスが高速化します。

完了ジョブのリスト機能を無効化するには、クラスタ構成で finished_jobs をゼロに設定します。43 ページの「QMON を使用したグローバルおよびホスト構成の追加と変更」、および sge_conf(5) のマニュアルページを参照してください。

ジョブの検査

ジョブ発行時に検査を強制することは、振り分け不可能なジョブが永続的に保留状態のままになることを防ぐための、重要な手続きになります。ただし、ジョブの検査は時間を消費するタスクになる可能性もあります。さまざまな実行ノードと消費可能リソースを抱え、すべてのユーザーが独自のジョブプロファイル を有する、異機種システム混在環境においては特に、ジョブの検査が時間を消費する可能性があります。異なるジョブがごく少数しかない同機種システム環境では、一般的なジョブの検査は省略できます。

ジョブの検査を使用不可にするには、クラスタ全体のデフォルト要求で、qsub のオプション –w n を追加します。『Sun N1 Grid Engine 6.1 ユーザーズガイド』の「QMON による高度なジョブの発行」、および sge_request(5) のマニュアルページを参照してください。

負荷しきい値および一時停止しきい値

慎重にマシンへの過剰な予約を行い、また過剰なシステム負荷を防ぐ必要がある場合に、負荷しきい値は必要です。一時停止しきい値も、システムに過剰な負荷をかけることを防止するために使用します。

ノードの過剰な負荷を防止する必要があるもう 1 つのケースは、実行ノードが対話型の負荷に対してオープンなままである場合です。対話型の負荷は、Grid Engine システムの制御下にありません。

計算ファームは、汎用度が低い場合があります。たとえば、計算ノードの各 CPU が唯一のキュースロットによってのみ表され、これらのノードでは対話型負荷が想定されていない場合があります。このような場合、load_thresholds を省略できます。

両方のしきい値を使用不可にするには、load_thresholds を none に設定し、suspend_thresholds を none に設定します。56 ページの「負荷および一時停止しきい値の構成」、および queue_conf(5) のマニュアルページを参照してください。

負荷調整

ジョブが振り分けられたあと、測定された負荷を大きくするために、負荷調整を使用します。この仕組みにより、ジョブの振り分けと対応する負荷の影響との間の遅延が原因である、マシンの過剰な予約が防止されます。必要でない場合は、負荷調整をオフにできます。負荷調整は、ホストと負荷しきい値のソートを伴って、スケジューラに追加の作業を課すことになります。

負荷調整を使用不可にするには、スケジューラ構成で `job_load_adjustments` を `none` に設定し、`load_adjustment_decay_time` をゼロに設定します。128 ページの「QMON を使用したスケジューラ構成の変更」、および `sched_conf` (5) のマニュアルページを参照してください。

直接スケジューリング

Grid Engine システムのデフォルトでは、固定スケジュール間隔でスケジューリング実行を開始します。固定間隔の優れた特徴としては、`qmaster` およびスケジューラの CPU 時間の消費を制限する点があります。好ましくない特徴としては、固定間隔はスケジューラを制限し、人為的にスループットを制限する点があります。多くの計算ファームには、`qmaster` とスケジューラ専用のマシンがあり、このような設定ではスケジューラを制限する理由がありません。`sched_conf`(5) の `schedule_interval` を参照してください。

スケジューラ構成の `flush_submit_sec` および `flush_finish_sec` パラメータを使用することで、直接スケジューリングを構成できます。128 ページの「QMON を使用したスケジューラ構成の変更」、および `sched_conf` (5) のマニュアルページを参照してください。

直接スケジューリングがアクティブである場合、計算ファームのスループットは、`sge_qmaster` とスケジューラをホスティングしているマシンの能力によってのみ制限されます。

緊急度ポリシーおよびリソース予約

緊急度ポリシーを使用すると、リソースに依存するジョブ優先順位方式をカスタマイズできます。このようなジョブ優先順位方式には、次の要素が含まれます。

- 最大の並列ジョブを最初に実行する、一般的な優先
- 高価なライセンスを活用するために、特定のリソースを要求するジョブに対する優先

リソース予約を使用している場合、両方の目標の実現は特に重要です。

DTrace によるパフォーマンスチューニング

数千台のアクティブなコンポーネントにまたがる可能性がある分散システムの障害追跡は、もっとも経験の豊富なシステム管理者にとってさえ難題であることがあります。実際、Grid Engine 管理者には、本番の環境のパフォーマンス低下につながる問題を特定し、再現するための明確な手段はありません。Solaris 10 環境では、DTrace ユーティリティーを使用し、Grid Engine マスターコンポーネントのオンサイトパフォーマンスを監視できます。DTrace は、Solaris 10 環境での動的イベントをトレースするための包括的なフレームワークです。DTrace に関する全般的な情報については、`http://www.sun.com/bigadmin/content/dtrace/` および dtrace のマニュアルページを参照してください。N1 Grid Engine 6.1 ソフトウェア での DTrace の使用の詳細は、`$SGE_ROOT/dtrace/README_dtrace.txt` ファイルを参照してください。

コマンド行からの DTrace によるパフォーマンスチューニング

Solaris 10 DTrace を使用できる場合は、`$SGE_ROOT/dtrace/monitor.sh` スクリプトを使用して、Grid Engine マスターを監視し、パフォーマンス上の問題点を探すことができます。`monitor.sh` スクリプトは次のオプションをサポートしています。

`-interval` *value*	統計間隔を指定します。デフォルトは `15sec` です。間隔が広いほど統計精度は低く、狭いほど高くなります。特に有用な値の範囲は `1sec` から `24hours` です。
`-cell` *cell-name*	$SGE_CELL が「デフォルト」でない場合は必須です。
`-spooling`	統計に加えて `qmaster` スプールのプローブ情報も表示します。このオプションによって、推定されるスプールの問題点に関するより具体的な情報を表示できます。
`-requests`	外部からの `qmaster` 要求のプローブを表示します。このオプションによって、`qmaster` の処理が滞る原因になっているインスタンスを評価するためのより具体的な情報を表示できます。

注 – 重大なメッセージやエラー、警告メッセージがあると、 `monitor.sh` 出力に表示されます。

Grid Engine マスターのパフォーマンス上の問題の分析

効果的なパフォーマンスチューニングを実現するには、分散システムのパフォーマンス上の問題点を理解する必要があります。`$SGE_ROOT/dtrace/monitor.sh` スクリプ

トは、稼働中の Grid Engine マスターのスループット関連データを測定し、そのデータをいくつかのインデックスにまとめて 1 間隔当たり 1 行の形式で出力します。この表示の情報は 4 つの主要カテゴリに分かれます。

- スプーリング — qmaster プロセスにスプールされたオペレーション数と経過時間を示します。
- 要求処理 — レポート、GDI 要求、ACK メッセージなどの、さまざまな種類の送受信メッセージ数を示します。
- スケジューリング — schedd プロセスに送信されたスケジューリング要求数と経過時間を示します。
- Qmaster 処理 — qmaster/schedd 交信、qmaster 要求 I/O 活動、qmaster ロックおよびロック解除要求に関する情報で構成されます。

詳細は、下記の例を参照してください。

問題分析のための **DTrace** 出力例

ここでは、Grid Engine マスターの問題点を検出可能な事例の監視出力例を示します。この例には、次の情報が含まれます。

- qmaster のスプール活動:
  - `#wrt` — spool_write_object() および spool_delete_object() を使用した qmaster の書き込みオペレーション数。重要な書き込みオペレーションのほぼすべてが、この関数を通ります。
  - `wrt/ms` — 各スレッドが spool_write_object() に費やした時間の合計 (ミリ秒単位)。
- qmaster のメッセージ処理:
  - `#rep` — sge_c_report() 経由で qmaster が処理したレポート数。ここには、execd 関数から qmaster に送信された大部分のデータが反映されます。
  - `#gdi` — do_gdi_request() 経由で qmaster 処理した GDI 要求数。クライアントコマンドから送信されたもののほぼすべてが、 GDI 要求として受信されます。ただし、exexd 関数やスケジューラから GDI 要求が送られてくることもあります。
  - `#ack` — do_c_ack() 経由で qmaster が処理した ACK メッセージ数。ACK メッセージ数が多いということは、ジョブシグナリングが考えられます。ただし、ACK メッセージはほかの目的にも使用されます。
- schedd のスケジューリング活動:
  - `#dsp` — schedd での dispatch_jobs() 呼び出し回数。dispatch_jobs() の呼び出し 1 回は、スケジューリングの実行 1 回とみなすことができます。
  - `dsp/ms` — スケジューラが dispatch_jobs() の各呼び出しに費やした時間の合計

- `#sad` — select_assign_debit() の呼び出し回数。select_assign_debit() の呼び出し 1 回は、スケジューラがジョブの割り当てまたは予約を 1 回試み たとみなすことができます。

- qmaster 処理:
  - `#snd` — qmaster が schedd に送信したイベントパッケージ数。この数値が長い時間をかけてゼロにまでなった場合は、どこかに問題があり、qmaster/schedd の同期が失われていることを意味します。
  - `#rcv` — schedd が qmaster から受信したイベントパッケージ数。この数値が長い時間をかけてゼロにまでなった場合は、どこかに問題があり、qmaster/schedd の同期が失われていることを意味します。
  - `#in++` — qmaster 受信メッセージバッファーに追加されたメッセージ数。
  - `#in--` — qmaster 受信メッセージバッファーから削除されたメッセージ数。1 間隔中に追加されたメッセージが削除されたメッセージより多い場合は、未処理メッセージの総数が増えることを意味します。
  - `#out++` — qmaster 送信メッセージバッファーに追加されたメッセージ数。
  - `#out--` — qmaster 送信メッセージバッファーから削除されたメッセージ数。1 間隔中に追加されたメッセージが削除されたメッセージより多い場合は、未配信メッセージの総数が増えることを意味します。
  - `#lck0/#ulck0` — qmaster の “global” ロック関係の sge_lock()/sge_unlock() 呼び出し回数。qmaster の内部リスト (ジョブリスト、キューリストなど) にアクセスする場合は、必ずこのロックを取得する必要があります。
  - `#lck0/#ulck0` — qmaster の “master_config” ロック関係の sge_lock()/sge_unlock() 呼び出し回数。このロックは二次的なロックですが、同じく重要です。

注 – 実際のシステムに表示されるコラムは、下記の例と異なることがあります。

この例では、17:40:32 から 17:41:05 の間にパフォーマンスが低下しています。

```
CPU     ID      FUNCTION:NAME
  0      1            :BEGIN                  Time |   #wrt  wrt/ms |#rep #gdi #ack|   #dsp  dsp/ms    #sad|   #snd
  0  36909         :tick-3sec 2006 Nov 24 17:39:23 |     43       3|   0    8    4|      3     691     121|      4
  0  36909         :tick-3sec 2006 Nov 24 17:39:26 |     83      16|   0   10    3|      3     699     122|      3
  0  36909         :tick-3sec 2006 Nov 24 17:39:29 |    117      24|   0    9    4|      4    1092     198|      4
  0  36909         :tick-3sec 2006 Nov 24 17:39:32 |     19       4|   0    9    3|      3     591     147|      3
  0  36909         :tick-3sec 2006 Nov 24 17:39:35 |    144      28|   0    9    4|      4    1012     173|      4
  0  36909         :tick-3sec 2006 Nov 24 17:39:38 |     46       5|   0    8    3|      3     705     122|      3
  0  36909         :tick-3sec 2006 Nov 24 17:39:41 |    154      31|   0    9    3|      4     894     198|      3
  0  36909         :tick-3sec 2006 Nov 24 17:39:44 |     46       5|   0   10    4|      4     971     162|      4
  0  36909         :tick-3sec 2006 Nov 24 17:39:47 |    154      29|   0    8    3|      3     739     158|      3
  0  36909         :tick-3sec 2006 Nov 24 17:39:50 |     46       5|   0   10    4|      4     815     162|      4
```

```
0  36909         :tick-3sec 2006 Nov 24 17:39:53 |      74      15|   0    8    3|      3     746     136|
0  36909         :tick-3sec 2006 Nov 24 17:39:56 |     116      20|   0   11    4|      4     992     184|
0  36909         :tick-3sec 2006 Nov 24 17:39:59 |      87      18|   0   11    4|      4     851     176|
0  36909          :tick-3sec 2006 Nov 24 17:40:02 |     109      20|   0   12    5|      4     930     184|
 0  36909         :tick-3sec 2006 Nov 24 17:40:05 |      88      15|   0    9    3|      4     995     176|
0  36909          :tick-3sec 2006 Nov 24 17:40:08 |     112      20|   0   12    4|      4     927     184|
0  36909          :tick-3sec 2006 Nov 24 17:40:11 |      32       6|   0    7    4|      3     618     121|
0  36909          :tick-3sec 2006 Nov 24 17:40:14 |     145      30|   0   11    4|      4     988     199|
0  36909          :tick-3sec 2006 Nov 24 17:40:17 |      43       3|   0    7    3|      3     618     121|
0  36909          :tick-3sec 2006 Nov 24 17:40:20 |     157      31|   0   11    4|      4     977     199|
0  36909          :tick-3sec 2006 Nov 24 17:40:23 |      43       4|   0    7    3|      3     701     121|
0  36909          :tick-3sec 2006 Nov 24 17:40:26 |      73      18|   0   11    4|      4     948     171|
0  36909          :tick-3sec 2006 Nov 24 17:40:29 |     127      31|   0   10    4|      4     968     189|
0  36909          :tick-3sec 2006 Nov 24 17:40:32 |      10       3|   0    6    0|      1     203      41|
0  36909          :tick-3sec 2006 Nov 24 17:40:35 |      19       5|   0    5    0|      0       0       0|
0  36909          :tick-3sec 2006 Nov 24 17:40:38 |      16       5|   0    5    1|      0       0       0|
0  36909          :tick-3sec 2006 Nov 24 17:40:41 |       1       0|   0    4    0|      0       0       0|
0  36909          :tick-3sec 2006 Nov 24 17:40:44 |       0       0|   0    4    0|      0       0       0|
0  36909          :tick-3sec 2006 Nov 24 17:40:47 |       0       0|   0    6    0|      1     744      81|
0  36909          :tick-3sec 2006 Nov 24 17:40:50 |       1       0|   0    5    1|      0       0       0|
0  36909          :tick-3sec 2006 Nov 24 17:40:53 |       0       0|   0    4    0|      0       0       0|
0  36909          :tick-3sec 2006 Nov 24 17:40:56 |       0       0|   0    5    0|      0       0       0|
0  36909          :tick-3sec 2006 Nov 24 17:40:59 |       0       0|   0    4    0|      0       0       0|
0  36909          :tick-3sec 2006 Nov 24 17:41:02 |       0       0|   0    4    1|      0       0       0|
0  36909          :tick-3sec 2006 Nov 24 17:41:05 |     165      41|   0   48   60|      0       0       0|
0  36909          :tick-3sec 2006 Nov 24 17:41:08 |     178      28|   0   15   53|      4     965     206|
0  36909          :tick-3sec 2006 Nov 24 17:41:11 |     106      23|   0   27   35|      4     855     166|
0  36909          :tick-3sec 2006 Nov 24 17:41:14 |     198      37|   0   41   70|      4    1189     196|
0  36909          :tick-3sec 2006 Nov 24 17:41:17 |      16       5|   0    9    5|      4     940     161|
0  36909          :tick-3sec 2006 Nov 24 17:41:20 |     162      35|   0   13    8|      4     958     200|
0  36909          :tick-3sec 2006 Nov 24 17:41:23 |      44       6|   0    6    3|      2     544      81|
0  36909          :tick-3sec 2006 Nov 24 17:41:26 |     150      34|   0   13    6|      4     921     199|
0  36909          :tick-3sec 2006 Nov 24 17:41:29 |      43       3|   0    5    2|      2     506      81|
0  36909          :tick-3sec 2006 Nov 24 17:41:32 |     157      37|   0    9    3|      4     978     199|
0  36909          :tick-3sec 2006 Nov 24 17:41:35 |      43       3|   0    7    3|      2     512      85|
0  36909          :tick-3sec 2006 Nov 24 17:41:38 |     127      29|   0    8    3|      4     994     185|
0  36909          :tick-3sec 2006 Nov 24 17:41:41 |      66      11|   0   10    4|      4     973     171|
0  36909          :tick-3sec 2006 Nov 24 17:41:44 |      48      10|   0    8    3|      3     785     128|
0  36909          :tick-3sec 2006 Nov 24 17:41:47 |     142      31|   0   12    4|      4     913     192|
0  36909          :tick-3sec 2006 Nov 24 17:41:50 |      64      13|   0   11    5|      4     853     168|
```

Grid Engine ソフトウェアのエラーレポートの取得方法

Grid Engine ソフトウェアは、特定のファイルにメッセージを記録するか、電子メールを送信する (または両方の手段) でエラーや警告を報告します。ログファイルには、メッセージ ファイルとジョブ STDERR 出力が含まれます。

ジョブが開始されるとただちに、ジョブスクリプトの標準的なエラー (STDERR) 出力がファイルにリダイレクトされます。デフォルトのファイル名と位置が使用されますが、qsub コマンドのある種のオプションを使用してファイル名と位置を指定することもできます。詳細については、Grid Engine システムのマニュアルページを参照してください。

sge_qmaster、sge_schedd、および sge_execd のそれぞれに messages ファイルがあります。各ファイルには同じファイル名、 messages が付けられています。sge_qmaster ログファイルは、マスタースプールディレクトリに存在します。sge_schedd メッセージ ファイルは、スケジューラスプールディレクトリに存在します。実行デーモンのログファイルは、実行デーモンのスプールディレクトリに存在します。スプールディレクトリの詳細については、『Sun N1 Grid Engine 6.1 インストールガイド』の「ルートディレクトリの下のスプールディレクトリ」を参照してください。

各メッセージは、ファイル内の 1 行を使用します。各メッセージは、縦線記号 (|) で区切られた 5 つのコンポーネントに再分割されます。

メッセージのコンポーネントは、次のとおりです。

1. 最初のコンポーネントは、メッセージのタイムスタンプです。
2. 2 つ目のコンポーネントは、メッセージを生成するデーモンを指定します。
3. 3 つ目のコンポーネントは、デーモンが動作しているホスト名です。
4. 4 つ目はメッセージの種類です。メッセージの種類は、次のいずれかです。
   - 通知の N (情報提供が目的)
   - 情報の I (情報提供が目的)
   - 警告の W
   - エラーの E (エラー状態の検出)
   - 重大の C (プログラムの異常終了になる可能性あり)

   クラスタ構成で loglevel パラメータを使用して、グローバルまたはローカルにどのメッセージの種類を記録するかを指定します。
5. 5 つ目のコンポーネントは、メッセージのテキストです。

注 – 何らかの理由でエラーログファイルにアクセスできない場合、Grid Engine システムは、対応するホストのファイル `/tmp/sge_qmaster_messages` 、`/tmp/sge_schedd_messages`、または `/tmp/sge_execd_messages` にエラーメッセージを記録しようとします。

状況によっては、Grid Engine システムは電子メールでユーザーか管理者、またはその両方にエラーイベントを通知します。Grid Engine システムにより送信される電子メールメッセージには、メッセージ本文は含まれません。メッセージテキストは、メールの件名フィールドにすべて含まれます。

さまざまなエラーまたは終了コードの意味

次の表に、ジョブ関連のさまざまエラーコードまたは終了コードの意味を示します。これらのコードは、あらゆる種類のジョブに該当します。

表 9–1 ジョブ関連のエラーまたは終了コード

スクリプト/方法	終了/エラーコード	意味
Job スクリプト	0	正常終了
	99	再度キューに入れる
	Rest	成功。アカウンティングファイルの終了コード
プロローグ/エピローグ	0	正常終了
	99	再度キューに入れる
	Rest	キューのエラー状態。ジョブは再度キューに入れられる

次の表に、並列環境 (PE) 構成関連のジョブのエラーコードまたは終了コードの意味を示します。

表 9–2 並列環境関連のエラーまたは終了コード

スクリプト/方法	終了/エラーコード	意味
pe_start	0	成功
	Rest	キューをエラー状態に設定。ジョブは再度キューに入れられる

表 9–2 並列環境関連のエラーまたは終了コード (続き)

スクリプト/方法	終了/エラーコード	意味
pe_stop	0	成功
	Rest	キューをエラー状態に設定。ジョブは再度キューには入れられない

次の表に、キュー構成関連のジョブのエラーコードまたは終了コードの意味を示します。これらのコードは、対応する方法が書き換えられた場合にのみ該当します。

表 9–3 キュー関連のエラーまたは終了コード

スクリプト/方法	終了/エラーコード	意味
ジョブ開始	0	成功
	Rest	成功。ほかの意味は特になし
一時停止	0	成功
	Rest	成功。ほかの意味は特になし
再開	0	成功
	Rest	成功。ほかの意味は特になし
終了	0	成功
	Rest	成功。ほかの意味は特になし

次の表に、チェックポイント設定関連のジョブのエラーコードまたは終了コードの意味を示します。

表 9–4 チェックポイント設定関連のエラーまたは終了コード

スクリプト/方法	終了/エラーコード	意味
チェックポイント	0	成功
	Rest	成功。ただし、カーネルチェックポイントの場合は、チェックポイントが失敗したことを意味する。

表 9–4 チェックポイント設定関連のエラーまたは終了コード *(続き)*

スクリプト/方法	終了/エラーコード	意味
移行	0	成功
	Rest	成功。ただし、カーネルチェックポイントの場合は、チェックポイントが成功しなかったことを意味する。移行は行われる。
再起動	0	成功
	Rest	成功。ほかの意味は特になし
後処理	0	成功
	Rest	成功。ほかの意味は特になし

デバッグモードでの Grid Engine システムのプログラムの実行

重大なエラー状態が発生した場合に、問題の特定に十分な情報がエラー記録機構によって生成されないことがあります。このため、Grid Engine システムには、ほぼすべての補助プログラムとデーモンをデバッグモードで実行する機能が用意されています。デバッグのレベルは、提供される情報の量および深さに応じて異なります。デバッグのレベルは、0 から 10 の範囲で、10 はもっとも詳細な情報を提供するレベル、0 はデバッグ無効です。

デバッグレベルを設定するため、Grid Engine システムの配布には、ユーザーの `.cshrc` または `.profile` リソースファイルに対する拡張が用意されています。`csh` または `tcsh` のユーザーには、ファイル *sge-root*`/util/dl.csh` が含まれています。`sh` または `ksh` のユーザーに対応するファイルの名前は *sge-root*`/ util/dl.sh` です。標準のリソースファイルに、これらのファイルを取り込む必要があります。`csh` または `tcsh` のユーザーの場合は、`.cshrc` ファイルに次の行を含めます。

```
source sge-root/util/dl.csh
```

`sh` または `ksh` のユーザーの場合は、`.profile` ファイルに次の行を含めます。

```
. sge-root/util/dl.sh
```

いったんログアウトして、ログインし直すと、次のコマンドを使用してデバッグレベルの level を設定できるようになります。

```
% dl level
```

level が 0 より大きい場合、Grid Engine システムのコマンドを開始すると、トレース出力が STDOUT に書き込むようコマンドに強制します。このトレース出力には、警告メッセージ、ステータスメッセージ、エラーメッセージばかりでなく、内部的に呼び出されたプログラムモジュール名が含まれます。メッセージには、ユーザーが指定するデバッグレベルに応じて、エラーの報告に役立つ行番号情報も含まれます。

注 – デバッグトレースを監視するには、大きなサイズのスクロール行バッファーを持つウィンドウを使用する必要があります。たとえば、1000 行のスクロール行バッファーを使用します。

注 – ウィンドウが xterm の場合、 xterm のログ記録機構を使用してあとでトレース出力を調べることができます。

デバッグモードで Grid Engine システムデーモンの 1 つを実行すると、デーモンが端末接続を維持して、トレース出力を書き出します。こうした端末接続は、使用している端末エミュレーションの割り込み文字を入力することによって打ち切ることができます。たとえば、Control-C などを使用します。

デバッグモードを無効にするには、デバッグレベルを 0 に戻します。

dbwriter デバッグレベルの設定

sgedbwriter スクリプトは、dbwriter プログラムを開始します。このスクリプトの位置は、*sge_root* /dbwriter/bin/sgedbwriter です。sgedbwriter スクリプトは、dbwriter の構成ファイルである dbwriter.conf を読み取ります。この構成ファイルの位置は、*sge_root*/*cell* /common/dbwriter.conf です。この構成ファイルは、dbwriter のデバッグレベルを設定します。たとえば、次のように指定します。

```
#
# Debug level
# Valid values: WARNING、INFO、CONFIG、FINE、FINER、FINEST、ALL
#
DBWRITER_DEBUG=INFO
```

dbwriter コマンドの –debug オプションを使用すると、dbwriter により作成されるメッセージの数を変更できます。通常は、デフォルトのデバッグレベル info を使用する必要があります。より詳細なデバッグレベルを使用する場合は、dbwriter により出力されるデータ量を大幅に増やします。

次のデバッグレベルを指定できます。

warning　　重大なエラーと警告のみが表示されます。

`info`	情報メッセージの数が追加されます。`info` はデフォルトのデバッグレベルです。
`config`	たとえば規則の処理に関する、`dbwriter` 構成に関連する追加の情報が得られます。
`fine`	さらに多くの情報が作成されます。このデバッグレベルを選択すると、`dbwriter` により実行されるすべての SQL 文が出力されます。
`finer`	デバッグ用に使用します。
`finest`	デバッグ用に使用します。
`all`	すべてのレベルの情報を表示します。デバッグ用に使用します。

問題の診断

Grid Engine システムには、問題の診断に役立つ、報告手段がいくつか用意されています。次の節では、これらの使用方法を説明します。

保留中のジョブが振り分けられない

保留中のジョブが実行可能な状態であることが明らかであるにもかかわらず、振り分けられないことがあります。Grid Engine システムには、その理由を調べる手段として `qstat -j` *job-id* と `qalter-w v` *job-id* のユーティリティーとオプションのペアがあります。

- `qstat -j` *job-id*

  有効である場合、`qstat -j` *job-id* は、最後のスケジューリング実行で特定のジョブが振り分けられなかった理由のリストを提供します。この監視機能は、有効または無効にすることができます。この監視機能は、`sge_schedd` デーモンと `sge_qmaster` との間の通信で望ましくないオーバーヘッドを生む可能性があるため、無効にすることができます。`sched_conf` (5) のマニュアルページの `schedd_job_info` を参照してください。ID が `242059` のジョブの出力を次の例に示します。

```
% qstat -j 242059
scheduling info: queue "fangorn.q" dropped because it is temporarily not available
queue "lolek.q" dropped because it is temporarily not available
queue "balrog.q" dropped because it is temporarily not available
queue "saruman.q" dropped because it is full
cannot run in queue "bilbur.q" because it is not contained in its hard queuelist (-q)
```

```
cannot run in queue "dwain.q" because it is not contained in its hard queue list (-q)
has no permission for host "ori"
```

この情報は、sge_schedd デーモンによって直接生成されます。この情報の生成には、クラスタの現在の使用量が考慮されます。この情報には、必要な情報が含まれないことがあります。たとえば、ほかのユーザーのジョブによってすべてのキュースロットがすでに占有されている場合、問題のジョブに関する詳細なメッセージは生成されません。

- `qalter -w v` *job-id*

このコマンドは、基本的にジョブが振り分けられない理由を一覧表示します。この目的のため、ドライスケジューリングが実行されます。スロットを含めて消費可能なすべてのリソースが、そのジョブ用に完全に利用可能であるとみなされます。負荷値は変化するため、すべての負荷値は無視されます。

ジョブまたはキューがエラー状態 E と報告される

ジョブまたはキューのエラーが、qstat 出力で、大文字の E で示されます。

ジョブがエラー状態になるのは、Grid Engine システムがジョブを実行しようとして、そのジョブに固有の理由で実行に失敗した場合です。

キューがエラー状態になるのは、Grid Engine システムがジョブを実行しようとして、そのキューに固有の理由で実行に失敗した場合です。

Grid Engine システムには、ジョブ実行エラーが発生した場合に、ユーザーおよび管理者がその診断情報を収集するための一群の機能が用意されています。キューおよびジョブのエラー状態のどちらも、原因はジョブの実行失敗にあります。そのため、診断機能は両方の種類のエラー状態に適用できます。

- ユーザー宛て中止メール。`qsub -m a` コマンドを使用してジョブが発行された場合は、`-M` *user*`[@`*host*`]` オプションで指定されたアドレスに中止メールが送信されます。中止メールには、ジョブのエラーに関する診断情報が含まれており、情報源として利用することを推奨します。
- **qacct** アカウンティング情報。 中止メールが得られない場合、`qacct -j` コマンドを実行できます。このコマンドにより、Grid Engine システムのジョブアカウンティング機能からジョブのエラーに関する情報を入手できます。
- 管理者宛て中止メール。管理者は、適切な電子メールアドレスを指定することによって、ジョブ実行時の問題に関する管理者宛てメールを送信するよう指示できます。sge_conf(5) のマニュアルページの administrator_mail の下を参照してください。管理者宛てのメールには、ユーザー宛ての中止メールよりも詳しい診断情報が含まれ、ジョブ実行エラーがよく発生する場合に利用することを推奨します。

- **Messages** ファイル。管理者宛てのメールが得られない場合は、まず `qmaster` の `messages` ファイルを調べてください。適切なジョブ ID を検索することによって特定のジョブに関するエントリを見つけることができます。デフォルトの設定でインストールした場合、`sge_qmaster messages` ファイルは *sge-root*/*cell*/`spool/qmaster/messages` にあります。

  ジョブの起動元の `sge_execd` デーモンのメッセージに、補足情報が含まれていることもあります。`qacct -j` *job-id* を使用してジョブの起動元のホストを確認し、*sge-root* /*cell*/`spool/`*host*`/messages` でジョブ ID を検索します。

一般的な問題の障害追跡

この節では、一般的な問題の原因と対処に役立つ情報を説明します。

- 問題 — ジョブの出力ファイルに「`Warning: no access to tty; thus no job control in this shell...`」というメッセージが出力される。
  - 原因 — 少なくとも 1 つのログインファイルに `stty` コマンドが含まれていることが考えられます。stty コマンドが役立つのは、端末が存在する場合だけです。
  - 対策 — バッチジョブは端末に関連付けられません。すべての `stty` コマンドをログインファイルから削除するか、`if` 文でstty コマンドを囲う必要があります。`if` 文は、処理前に端末の有無をチェックします。次の例に `if` 文を示します。

```
/bin/csh:
stty -g             # checks terminal status
if ($status == 0)   # succeeds if a
terminal is present
<put all stty commands in here>
endif
```

- 問題 — ジョブの標準エラーログファイルに「`'tty':Ambiguous`」というメッセージが出力される。しかし、ジョブスクリプトで呼び出されるユーザーのシェルには `tty` に対する参照はない。
  - 原因 — デフォルトでは、`shell_start_mode` は `posix_compliant` です。このため、すべてのジョブスクリプトは、キュー定義で指定されたシェルで実行されます。スクリプトは、ジョブスクリプトの先頭行に指定されたシェルでは実行されません。
  - 対策 — `qsub` コマンドに `-S` フラグを使用するか、`shell_start_mode` を `unix_behavior` に変更してください。
- 問題 — コマンド行からジョブスクリプトを実行できるが、`qsub` コマンドを使用して実行しようとすると、ジョブスクリプトが失敗する。

- 原因 — ジョブに対するプロセス数が制限されている可能性があります。制限が設定されているかどうかをテストするには、`limit` と `limit -h` 機能を実行するテストスクリプトを記述します。シェルプロンプトから `qsub` コマンドを使用して両方の関数を対話的に実行し、結果を比較します。
- 対策 — 構成ファイルから、シェルで制限を設けるすべてのコマンドを削除してください。

- 問題 — 実行ホストから負荷 `99.99` が報告される。

  1. 原因 — ホストで `sge_execd` デーモンが動作していません。

     対策 — 実行ホストで root になり、*sge-root*`/` *cell*`/common/sgeexecd` スクリプトを実行することによって `sge_execd` デーモンを起動します。

  2. 原因 — デフォルトドメインの指定に誤りがあります。

     対策 — Grid Engine システムの管理者として、`qconf -mconf` コマンドを実行し、`default_domain` 変数の値を `none` に変更します。

  3. 原因 — `sge_qmaster` ホストが認識している実行ホスト名と、その実行ホストが認識している自身の名前とが異なります。

     対策 — 計算クラスタのホスト名の解決に DNS を使用している場合は、主ホスト名として絶対パスによるドメイン名 (FQDN) が返されるように `/etc/hosts` と NIS を構成します。このように構成しても、当然、`168.0.0.1 myhost.dom.com myhost` というように短い別名を定義、使用することができます。

     DNS を使用していない場合は、`/etc/hosts` のすべてのファイルと NIS テーブルに矛盾がないようにします (例: `168.0.0.1 myhost.corp myhost` または `168.0.0.1 myhost`)。

- 問題 — 次のような警告が 30 秒おきに *cell*`/spool/` *host*`/messages` に出力される。

```
Tue Jan 23 21:20:46 2001|execd|meta|W|local
configuration meta not defined - using global configuration
```

  しかし、*cell*`/common/local_conf` には、各ホスト用のファイルがあり、それぞれに FDQN が存在する。

  - 原因 — 使用しているマシン `meta` では、ホスト名解決でショート名が返されるのに対し、マスターマシンでは、FDQN 付きの `meta` が返されます。
  - 対策 — この点に関して、`/etc/hosts` のすべてのファイルと NIS テーブルの間に矛盾がないようにしてください。この例では、ホスト `meta` の `/etc/hosts` ファイルに、誤って次のようなテキストの行が含まれている可能性があります。

    ```
    168.0.0.1 meta meta.your.domain
    ```

    正しくは、この行は次のようにします。

    ```
    168.0.0.1 meta.your.domain meta.
    ```

- 問題 — デーモンの messages ファイルに CHECKSUM ERROR、WRITE ERROR、または READ ERROR というメッセージが出力されることがある。
  - 原因 — 1 秒間隔で出力されるのでないかぎり、何もする必要はありません。一般にこれらのメッセージは、1 日に 1 回から 30 回、出力されます。
- 問題 — ジョブが特定のキューで完了し、qmaster/messages に次のメッセージを返す。

```
Wed Mar 28 10:57:15 2001|qmaster|masterhost|I|job 490.1
finished on host exechost
```

  しかし、実行ホストの exechost/messages ファイルには次のエラーメッセージが出力される。

```
Wed Mar 28 10:57:15 2001|execd|exechost|E|can't find directory
"active_jobs/490.1" for reaping job 490.1

Wed Mar 28 10:57:15 2001|execd|exechost|E|can't remove directory
"active_jobs/490.1": opendir(active_jobs/490.1) failed:
Input/output error
```

  - 原因 — 自動マウントされる *sge-root* ディレクトリがマウント解除されたために、sge_execd デーモンが現在の作業ディレクトリを失った可能性があります。
  - 対策 — sge_execd ホストにローカルのスプールディレクトリを使用してください。QMON または qconf コマンドを使用して、execd_spool_dir パラメータを設定します。
- 問題 — qrsh ユーティリティーを使用して対話形式のジョブを発行しようとすると、次のエラーメッセージが表示される。

```
% qrsh -l mem_free=1G error: error: no suitable queues
```

  しかし、qsub コマンドを使用したバッチジョブの発行に対してキューを使用できる。これらのキューは、qhost -l mem_free=1G および qstat -f -l mem_free=1G を使用して照会できる。
  - 原因 — 「error: no suitable queues」というメッセージの原因は、qrsh などの対話形式のジョブに対してデフォルトで有効になる submit の -w e オプションにあります。qrsh(1) のマニュアルページの -w e を参照してください。現在のクラスタ構成に従ってジョブが振り分け可能であるかどうかを sge_qmaster が確実に判断できない場合、このオプションがあると、submit コマンドで問題が発生します。この仕組みの意図は、許可できないジョブ要求を事前に拒否することにあります。

- 対策 — この場合は、mem_free が消費可能リソースに設定されているにもかかわらず、ユーザーが各ホストで使用可能にするメモリーサイズを指定しなかったことが原因です。メモリー負荷値はそれぞれに異なるため、この検査では、メモリー負荷値は意図的に考慮されません。このため、クラスタ構成の一部では表示されません。この問題を解決するには、次のいずれかを行います。
  - qrsh のデフォルトオプション -w e を、-w n オプションを使用して明示的に無効にすることでこの検査を全般的に省略する。このコマンドを、*sge-root*/*cell*/common/sge_request に配置することもできます。
  - mem_free を消費可能リソースとして管理する場合は、qconf -me *hostname* を使用して、host_conf の complex_values にあるホストに対し mem_free の容量を指定する。
  - mem_free を消費可能リソースとして管理しない場合は、qconf -me *hostname* を使用することで、complex(5) の consumable カラムで mem_free を消費不可リソースに戻す。

- 問題 — qrsh が、自身が動作しているのと同じノードに振り分けない。このとき qsh シェルから次のメッセージが返される。

```
host2 [49]% qrsh -inherit host2 hostname
error: executing task of job 1 failed:

host2 [50]% qrsh -inherit host4 hostname
host4
```

  - 原因 — gid_range が十分ではないことが考えられます。gid_range には、1 つの数字ではなく範囲として指定してください。Grid Engine システムは、ホスト上の各ジョブに固有の gid を割り当てます。
  - 対策 — qconf -mconf コマンドまたは QMON を使用して gid_range を調整してください。推奨する範囲は次のとおりです。

```
gid_range                  20000-20100
```

- 問題 — 並列ジョブ内で使用すると、qrsh -inherit -V が機能しない。次のメッセージが返される。

```
cannot get connection to "qlogin_starter"
```

  - 原因 — この問題は入れ子にされた qrsh 呼び出しで発生します。原因は -V オプションです。最初の qrsh -inherit 呼び出しでは、環境変数 TASK_ID が設定されます。TASK_ID は、並列ジョブ内の密接に統合されたタスクの ID です。2 つ目の qrsh -inherit 呼び出しでは、このタスクの登録に環境変数を使用します。すでに実行中の最初のタスクと同じ ID を持つタスクを開始しようとするため、このコマンドは失敗します。

- 対策 — qrsh -inherit を呼び出す前に TASK_ID を設定解除するか、-V ではなく -v オプションを使用します。このオプションにより、実際に必要な環境変数だけエクスポートされます。

- 問題 — qrsh がまったく機能していないように見える。次のようなメッセージが返される。

```
host2$ qrsh -verbose hostname
local configuration host2 not defined - using global configuration
waiting for interactive job to be scheduled ...
Your interactive job 88 has been successfully scheduled.
Establishing /share/gridware/utilbin/solaris64/rsh session
to host exehost ...
rcmd: socket: Permission denied
/share/gridware/utilbin/solaris64/rsh exited with exit code 1
reading exit code from shepherd ...
error: error waiting on socket for client to connect:
Interrupted system call
error: error reading return code of remote command
cleaning up after abnormal exit of
/share/gridware/utilbin/solaris64/rsh
host2$
```

  - 原因 — qrsh に対する権限が正しく設定されていない可能性があります。
  - 対策 — *sge-root*/utilbin/ にある次のファイルの権限を調べてください。rlogin および rsh は setuid であり、かつ root により所有されている必要があることに注意してください。

```
-r-s--x--x 1 root      root   28856  Sep 18  06:00  rlogin*
-r-s--x--x 1 root      root   19808  Sep 18  06:00  rsh*
-rwxr-xr-x 1 sgeadmin  adm   128160  Sep 18  06:00  rshd*
```

    注 – *sge-root* ディレクトリも、setuid オプションで NFS マウントされている必要があります。*sge-root* が発行クライアントから nosuid でマウントされている場合、qrsh と、関係するコマンドは機能しません。

- 問題 – 分散 make を起動しようとすると、次のエラーメッセージで qmake が終了する。

```
qrsh_starter: executing child process
qmake failed: No such file or directory
```

- 原因 — Grid Engine システムは、実行ホストで qmake のインスタンスを起動します。Grid Engine システム環境 (特に PATH 変数) がユーザーのシェルリソースファイル (.profile または .cshrc) に設定されていない場合、この qmake の呼び出しは失敗します。
- 対策 — -v オプションを使用して、PATH 環境変数を qmake ジョブにエクスポートしてください。次は、一般的な qmake の呼び出し例です。

```
qmake -v PATH -cwd -pe make 2-10 --
```

- 問題 — qmake ユーティリティーを使用する際、次のエラーメッセージが返される。

```
waiting for interactive job to be scheduled ...timeout (4 s)
expired while waiting on socket fd 5

Your "qrsh" request could not be scheduled, try again later.
```

- 原因 — qmake の呼び出し元であるシェルに ARCH 環境変数が正しく設定されていない可能性があります。
- 対策 – クラスタで使用可能なホストに一致するサポート値を ARCH 変数に設定するか、発行時に適切な値を指定してください (例: qmake -v ARCH=solaris64 ...)。

- 問題 — 次のジョブが SuSE Linux システムに割り当てられると、-cwd オプションが機能しないで、ユーザーのホームディレクトリに cwd-test が作成される。

```
qsub -l arch=lx24-x86 -cwd -b y -o /dev/null -e /dev/null -V "hostname | tee -a
cwd-test"
```

- 原因 — SuSE Linux システムの場合、/etc/csh.login ファイルには、現在の作業ディレクトリをユーザーのホームディレクトリに変更する cd コマンドが含まれています。通常、シェルはシステム全体のプロファイルファイルを参照します。Linux の(t)csh の場合、システム全体のファイルは /etc/csh.login です。
- 対策 – -noshell オプションを使用してください。この場合は、プロファイルファイルを参照することが可能な中間のシェルが存在しないため、~/.login または ~/.cshrc のどちらも参照されません。

  バッチスクリプトを使用するときは、あらゆるプログラムの実行前に、PWD 環境変数を使用して、そのバッチスクリプト内の現在の作業用ディレクトリを変更してください。

  これ以外の回避策は、ユーザーが tcsh または csh のどちらも使用しないことです。cd コマンドは /etc/csh.login にのみ現れるため、問題が起きなくなります。

◆ ◆ ◆ 第 10 章

10 DBWriterの構成

dbwriterコンポーネントは、レポートデータベースのレポートデータの書き込みと削除を行います。dbWriterは次のタスクを実行します。

- レポートファイルからrawデータを読み取り、このrawデータをレポートデータベースに書き込む。
- 派生値を計算する。計算する値と、その計算方法の規則を設定できます。
- 期限切れのデータを削除する。データを保持する期間を設定できます。

sge_qmasterコンポーネントがレポートファイルを生成します。レポートファイルの生成を設定できます。sge_conf(5)のマニュアルページの*reporting_params*属性、およびhost_conf(5)のマニュアルページの*report_variables*属性を参照してください。

設定

インストール手順で、これらのパラメータを設定します。dbwriterを起動するスクリプトは、レポートモジュールで提供されています。詳細は、『N1 Grid Engine 6インストールガイド』を参照してください。dbwriterに対して、次のパラメータを設定する必要があります。

データベースシステム

dbwriterは、各種製造元のデータベースシステムに接続できます(サポートしているシステムは、PostgreSQLおよびOracle)。次のパラメータを設定する必要があります。

- *DRIVER* — 使用するJDBCドライバの名前(たとえばorg.postgresql.Driver)
- *DRIVERJAR* — JDBCドライバを含むJARアーカイブ(たとえばlib/postgres43.jar)。

データベースサーバー

JDBC URL で、どのホストのどのデータベースを使用するかを設定します。URL パラメータに、使用するデータベースの JDBC URL を設定します。使用する構文は、データベースベンダーのガイドラインに従ってください。たとえば、PostgreSQL データベースでは、次のように設定します。`jdbc:postgresql://<hostname>:5432/arco`

レポートファイルのベースディレクトリ

`dbwriter` がレポートファイルを検索するパスは、`REPORTING_FILE` 変数で設定します。通常、ベースディレクトリは `$SGE_ROOT/$SGE_CELL/common` に設定します。

構成

これらのパラメータの設定作業は、インストール手順で行われます。`dbwriter` の動作は、多くのコマンド行パラメータを使用して設定できます。

間隔

-interval パラメータは、`dbwriter` が新しいレポートファイルを探す間隔を設定します。レポートファイルが見つかると、ファイルが読み取られ、データがレポートデータベースに書き込まれます。

Pid

-pid パラメータは、PID ファイルへのパスを定義します。`dbwriter` は起動時に PID ファイルを書き込みます。このファイルには、`dbwriter` のプロセス ID が含まれています。`dbwriter` の停止時に、このファイルは削除されます。

PidCmd

-pidCmd パラメータは、`dbwriter` がそのプロセス ID を特定するために実行するコマンドを定義します。このコマンドは、その親プロセス ID を `stdout` に出力する必要があります。`dbwriter` は Java アプリケーションです。Java 仮想マシンは、自身のプロセス ID を特定することはできません。*pidCmd* のデフォルト値は `$SGE_ROOT/utilbin/$ARCH/checkprog -ppid` です。

連続モード

-continuous パラメータは、連続モードをオンにします。*-continuous* が指定されていないと、 `dbwriter` はタスクを一度だけ実行します。連続モードをオンにすると、*-interval* スイッチで設定された間隔でタスクを繰り返し実行します。

デバッグレベル

-debug オプションを使用して、`dbwriter` が出力するメッセージの量を設定できます。*-debug* オプションのパラメータはデバッグレベルです。通常は、デフォルトのデバッグレベル (`-info`) を使用することを推奨します。より詳細なデバッグレベルを設定すると、`dbwriter` が出力するデータの量が著しく増大します。指定できるのは、次のデバッグレベルです。

- `-warning`: 重大なエラーと警告のみが表示されます。
- info: 多くの情報メッセージが追加されます。-display スイッチを使用しない場合は、このレベルがデフォルトになります。
- `-config`: 規則の処理 (派生値また削除規則) などの、`dbwriter` の設定に関係する追加情報を提供します。
- `-fine`: 詳細情報を出力します。このレベルを選択すると、`dbwriter` が実行するすべての SQL 文が出力されます。
- `-finer`: デバッグ用です。
- `-finest`: デバッグ用です。
- `-all`: すべてのレベルの情報を表示します (デバッグ専用)。

レポートファイル

N1 Grid Engine 6 は、さまざまな種類のデータを含むレポートファイルを記述します。

- ホストの負荷値と消費可能リソース
- キュー消費可能リソース
- ジョブのログ記録
- ジョブのアカウンティング
- 共有ツリーの使用

`dbwriter` コマンドの行 *-reporting* で、レポートファイルのパスを指定します。`dbwriter` コンポーネントは、レポートファイルを自動的に解析します。処理を完了し、すべての情報をデータベースに格納すると、レポートファイルを削除します。

派生値の計算

dbwriter の起動時と連続モードの場合は、1 時間ごとに派生値が計算されます。XML ファイルで、計算する値を設定できます。デフォルトでは、$SGE_ROOT/dbwriter/database/<database_type>/dbwriter.xml が設定されています。<database_type> で、使用するデータベースの種類を定義します。現在、Oracle および Postgres がサポートされています。構成ファイルのパスは、 *-calculation* パラメータを使用して dbwriter に渡されます。

設定ファイルは XML 形式を使用しています。次の節で説明するように、派生値と削除値両方の規則が指定されています。派生値の規則の書式を次に示します。

派生値の書式

1. 最上位の開始タグは <derive> です。次の 3 つの属性を使用して指定します。
   - object — ホスト、キュー、ユーザー、グループ、部署、またはプロジェクトを指定します。この属性に基づいて、派生値は最終的に次のいずれかに格納されます。sge_host_values、sge_queue_values、 sge_user_values、sge_group_values、sge_department_values、sge_project_values。
   - interval — 時間、日、月、または年のいずれかです。
   - variable — 新しい派生値の名前です。
2. 第 2 レベルの開始タグは <sql> または <auto> のいずれかで、値の派生方法を記述します。これらのタグの詳細を次に示します。
3. <sql> – このタグには、派生値の計算に使用される SQL 文を指定します。エントリの正確な構文は、使用するデータベースの種類によって異なります。
4. <auto> - 特定の単純な派生値については、完全な SQL クエリーの代わりにこのタグを使用できます。このタグには 2 つの属性があります。
   - function — 変数に適用する集約関数を指定します。ここでは、使用する種類のデータベースに有効な任意の関数を指定できます。代表的な関数としては、AVG、SUM、VALUE、COUNT、MIN、または MAX があります。
   - variable — 次のテーブルに記録される任意の変数を指定できます。sge_host_values、sge_queue_values、sge_user_values、sge_group_values、sge_department_values、sge_project_values。指定する変数は、<derive> タグで囲まれたオブジェクト属性によって示されるテーブルに存在する必要があります。たとえば、オブジェクトがホスト の場合、変数は sge_host_values に存在する必要があります。
5. 2 つの開始タグに対応する 2 つの終了タグを指定します。

例

<sql> タグを使用した派生規則の例を示します。sge_queue テーブルには、 q_qname と q_hostname からなる複合的な主キーがあります。キュー object_type に指定された

規則の場合、sge_queue テーブルのエントリごとに 1 つのクエリーが作成されます。プレースホルダ __key_0__ は、キューの名前で置き換えられ、__key_1__ はホスト名で置き換えられます。

```
<!--
        average queue utilization per hour
        -->
<derive object="queue" interval="hour" variable="h_utilized">
        <sql>
        SELECT DATE_TRUNC( 'hour', qv_time_start)
                        AS time_start,
                DATE_TRUNC( 'hour', qv_time_start) + INTERVAL '1 hour'
                        AS time_end,
                AVG(qv_dvalue * 100 / qv_dconfig)
                        AS value
                FROM sge_queue_values
                WHERE qv_variable = 'slots' AND
                        qv_parent = (SELECT q_id FROM sge_queue
                                WHERE q_qname = __key_0__
                                        AND q_hostname = __key_1__)
                        AND qv_time_start &lt;= '__time_end__' AND
                        qv_time_end &gt; '__time_start__'
                GROUP BY time_start
</sql>
</derive>
```

<auto> タグを使用した派生規則の例を示します。

```
<!-- average load per hour -->
<derive object="host" interval="hour" variable="h_load">
        <auto function="AVG" variable="np_load_avg" />
</derive>
```

期限切れレコードの削除

dbwriter の起動時と連続モードの場合は、1 時間ごとに期限切れレコードが削除されます。XML ファイルで計算する値を設定できます。デフォルトでは $SGE_ROOT/dbwriter/database/<database_type>/dbwriter.xml が設定されています。<database_type> では、使用するデータベースの種類を定義します。現在、Oracle および Postgres がサポートされています。構成ファイルのパスは、 *-deletion* パラメータを使用して dbwriter に渡されます。

構成ファイルは XML 形式を使用しています。このファイルには、派生値と削除値の両方に対する規則が指定されています。削除規則の書式を次に示します。

- 最上位の開始タグ <delete> には、次の 3 つの属性があります。

- scope — 削除するデータの型です。有効なエントリは *job*、*job_log*、*share_log*、*ip, queue_values*、*ip, department_values*、*user_values*、*group_values* です。この属性に基づいて、`sge_ prepended` と同じ名前のテーブルから値が削除されます。
- time_range — time_amount の単位です。
- time_amount — レコードを保持する単位 (time_range) 数です。

- 第 2 レベルの開始タグ `<sub_scope>` はオプションですが、このタグは削除の追加条件を指定します。サブスコープは、すべての `*_values` スコープおよび `share_log` スコープに対して設定できます。
- 1 つまたは 2 つの開始タグに対応する 2 つの終了タグを指定します。

サブスコープが **_values* 規則に対して設定されている場合、そのサブスコープには、空白で区切られた、削除する変数のリストが指定されています。サブスコープが `share_log` に対して指定されている場合、そのサブスコープには、空白で区切られた、削除する共有ツリーノードのリストが指定されています。

例

次の規則では、サブスコープに指定した 4 つの変数を、`sge_host_values` テーブルから 7 日後に削除します。

```
<delete scope="host_values" time_range="day" time_amount="7">
        <sub_scope>np_load_avg</sub_scope>
        <sub_scope>cpu</sub_scope>
        <sub_scope>mem_free</sub_scope>
        <sub_scope>virtual_free</sub_scope>
</delete>
```

次の規則では、2 年後に `sge_host_values` テーブルからすべての変数を削除します。

```
<delete scope="host_values" time_range="year" time_amount="2"/>
```

次の規則では、ユーザー `fred` のすべてのレコードを 1 か月後に削除します。

```
<delete scope="share_log" time_range="month" time_amount="1">
        <sub_scope>fred</sub_scope>
</delete>
```

索引

Q

R

S

く

け

こ

さ

し

れ

ろ

わ