NEC

NEC Network Queuing System V (NQSV) 利用の手引

[JobManipulator 編]

輸出する際の注意事項

本製品（ソフトウェアを含む）は、外国為替および外国貿易法で規定される規制貨物（または役務）に該当することがあります。

その場合、日本国外へ輸出する場合には日本国政府の輸出許可が必要です。

なお、輸出許可申請手続きにあたり資料等が必要な場合には、お買い上げの販売店またはお近くの当社営業拠点にご相談ください。

はしがき

本書は、NEC Network Queuing System V（NQSV）によるスケジューラ機能であるJobManipulatorについて説明したものです。

備考

本書はNEC Network Queuing System V (NQSV) R1.00以降に対応しています。

(1)　本書に説明しているすべての機能はプログラムプロダクトであり、以下のプロダクト名およびプロダクト番号に対応しています。

プロダクト名	型番
NEC Network Queuing System V (NQSV) /ResourceManager	UWAF00 UWHAF00（サポートパック）
NEC Network Queuing System V (NQSV) /JobServer	UWAG00 UWHAG00（サポートパック）
NEC Network Queuing System V (NQSV) /JobManipulator	UWAH00 UWHAH00（サポートパック）

本書の対象読者

本書は、システム管理者および一般利用者を対象読者として書かれたものです。

とくにシステム管理者がシステムのスケジューリングのポリシーを設計する際、および、一般利用者が構築済みのシステムを利用する際には、本書は有効な説明書となります。

本書は、対象読者が Linux の一般操作および、NQSV に関する一般的な知識を知っていることを前提として記述されています。

本書の読み進め方

本書は、次の構成となっています。章ごとに対象読者の範囲は異なっており、表の一番右の列にその範囲を示しています。記載された対象読者の後に(*)がついている章については、該当する読者は必ずお読みください。

章	タイトル	内容	対象読者
1	JobManipulator の概要	概要について説明	システム管理者 一般利用者
2	環境の構築	JobManipulator のインストールやスケジューリングパラメータの設定について説明	システム管理者(*)
3	運用管理	スケジューリングの基本的な機能について説明	システム管理者(*) 一般利用者
4	高度なスケジューリング機能	スケジューリングの高度な機能について説明	システム管理者(*) 一般利用者
5	SX-Aurora TSUBASA 対応機能	SX-Aurora TSUBASA 対応の機能について説明	システム管理者(*) 一般利用者
6	コマンドリファレンス	コマンドリファレンスについて説明	システム管理者 一般利用者

関連説明書

NEC Network Queuing System V (NQSV)のマニュアルは以下で構成されています。

マニュアル名称	内容
NEC Network Queuing System V (NQSV) 利用の手引［導入編］	システムの全体像および基本的なシステムの構築方法について説明します。
NEC Network Queuing System V (NQSV) 利用の手引［管理編］	NQSV の管理者が実施する各種設定について説明しています。
NEC Network Queuing System V (NQSV) 利用の手引［操作編］	NQSV の一般利用者が使用する各種機能について説明しています。
NEC Network Queuing System V (NQSV) 利用の手引［リファレンス編］	コマンドリファレンスに関して説明しています。
NEC Network Queuing System V (NQSV) 利用の手引［API 編］	NQSV を操作する C 言語のプログラミングインタフェース (API) について説明しています。
NEC Network Queuing System V (NQSV) 利用の手引［JobManipulator 編］	NQSV のスケジューラコンポーネント JobManipulator に関して説明しています。
NEC Network Queuing System V (NQSV) 利用の手引［アカウンティング・予算管理編］	NQSV のアカウンティング機能に関して説明しています。

表記上の約束

本書では次の表記規則を使用しています。

省略記号	…	前述の項目を繰り返すことができることを表しています。ユーザは同様の項目を任意の数だけ入力することができます。
縦棒	|	オプションまたは必須の選択項目を分割します。
中かっこ	{}	１つを選択しなければならない一連パラメータまたはキーワードを表しています。
角かっこ	[]	省略可能な一連パラメータまたはキーワードを表しています。

用語定義・略語

用語・略語	説　明
ベクトルエンジン (VE、Vector Engine)	SX-Aurora TSUBASAの中核であり、ベクトル演算を行う部分です。PCI Expressカードであり、x86サーバに搭載して使用します。
ベクトルホスト (VH、Vector Host)	ベクトルエンジンを保持するサーバ、つまり、ホストコンピュータを指します。
ベクトルアイランド (VI)	VH1台にVEを1枚ないし複数枚組み込んだ単位を指します。
バッチサーバ（BSV）	バッチサーバホスト上に常駐し、NQSVシステム全体を管理します。
ジョブサーバ（JSV）	実行ホスト上に常駐し、ジョブの実行を管理します。
JobManipulator（JM）	NQSVの提供するスケジューラ機能です。計算リソースの空き容量を管理し、ジョブの開始時刻を割り当てます。
アカウンティングサーバ	アカウント情報の収集・管理、予算管理を行います。
リクエスト	NQSVシステムにおけるユーザジョブの単位であり、１つまたは複数のジョブから構成されたものです。バッチサーバによって管理されます。
ジョブ	ユーザジョブの実行単位であり、ジョブサーバによって管理されます。
論理ホスト	実行ホストの資源を論理的（仮想的）に分割し、ジョブに割り当てたものです
キュー	受け取ったリクエストをプールし、管理する機構です。
BMC	Board Management controllerの略語です。IPMI(Intelligent Platform Management Interface)と呼ばれる業界標準のサーバーマネジメントインタフェースに準拠したサーバー管理を行います。
HCA	Host Channel Adapterの略語です。IBネットワークに接続するためにサーバ側に取り付けるPCIeカードです。
IB	InfiniBandの略語です。
MPI	Message Passing Interfaceの略語です。主にノード間で並列コンピューティングを行うための標準規格です。
NIC	Network Interface Cardの略。他ノードと通信するためのハードウェア。

目　次

図目次

第1章　JobManipulator の概要

1.1　はじめに

JobManipulator は大規模クラスタ運用におけるシングルノードジョブ、マルチノードジョブの混在運用に対応するジョブスケジューラです。FIFO(First-In First-Out)を基本とし計算リソース(CPU, メモリ等)の空き容量を管理し、最も早く実行可能な時刻を割り当てるスケジューリングを実現しています。

1.2　JobManipulator の特長

JobManipulator の主な特長は以下の通りです。

- リクエストの実行予定時間（宣言経過時間）とCPU数やメモリ等の必要リソース量をもとに計算リソースの有効活用と高稼働率を実現するバックフィル・スケジューリング
- ユーザ、グループを単位とした計算リソースの配分比とリソース使用実績に基づき、リクエストの優先度制御を行うフェアシェア・スケジューリング
- リクエストの予定前終了やキャンセルによって空きリソースが発生した場合にリクエストのリソース割り当てを最適化する、エスカレーション
- リクエストの実行開始時刻の予約、および実行に必要な計算リソースの事前予約が可能な事前予約機能
- 優先リクエスト（緊急リクエスト, 特別リクエスト）に対する計算リソース割り当てを保障し、即時実行を可能にする割り込みアサイン制御機能
- 一定時間リクエスト実行の予定が無い実行ホストの電源の自動停止や、最大稼働実行ホスト数を設定可能な省電力運用機能
- また、JobManipulatorは以下のような多彩なスケジューリング機能を有し、多種多様なユーザニーズに対応可能です。
- ランリミット設定、アサインリミット設定、リクエスト優先順位設定、ピックアップの追い越し制御、アサインポリシーの設定、JSVアサインポリシーの設定等による柔軟なスケジューリング設定機能
- 10種類以上の項目とそれに対する重み付け設定によるスケジューリングプライオリティの動的設定機能

- リクエストの経過時間制限値に余裕をもたせ、リクエストのリソース占有を防止するElapseマージン機能
- 任意に仮想的なリソースを定義し、スケジューリングに利用可能とするカスタムリソース機能
- ノード障害時においても、正常ノードへの再スケジューリングを行い、計算リソースの稼働率を維持するノード障害対応機能

第2章　　環境の構築

2.1　JobManipulatorの構成

JobManipulator は NQSV 専用のバッチスケジューラです。NQSV/バッチサーバで管理する各ユーザより投入されたリクエストのスケジューリングを行います。

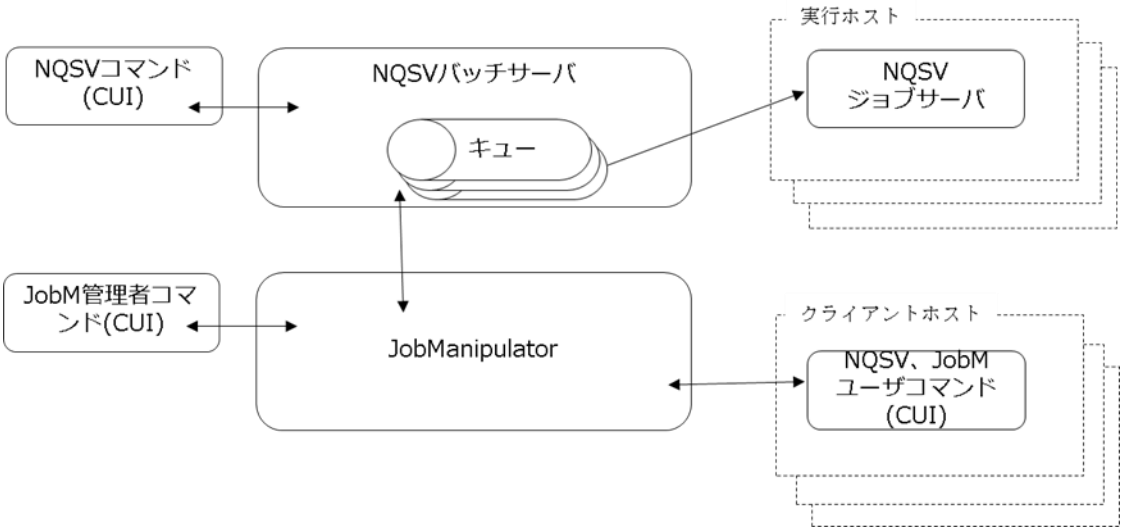


図 2-1：JobManipulator の位置付け

JobManipulator のファイル構成を示します。

ファイル	説明
/opt/nec/nqsv/sbin/nqs_jmd	JobManipulatorスケジューラ
/opt/nec/nqsv/bin/sstat	スケジューラ情報表示用コマンド
/opt/nec/nqsv/sbin/smgr	スケジューラ構成・管理用コマンド
/opt/nec/nqsv/sbin/sushare	ユーザシェア管理用コマンド
標準パス：　/etc/opt/nec/nqsv/nqs_jmd.conf	コンフィグファイル JobManipulatorの動作環境等を定義したファイルです。本ファイルはテキスト形式であり、システム管理者が管理します。
/etc/opt/nec/nqsv/jmtab	JobManipulatorの コンフィグファイル一覧です。本ファイルはテキスト形式であり、システム管理者が管理します。
標準パス：　/etc/opt/nec/nqsv/jm_sharedb.conf	ShareDBファイル 各ユーザのシェア配分や使用実績等を格納するファイルです。 本ファイルはテキスト形式であり、システム管理者が管理します。
標準パス： /var/opt/nec/nqsv/nqs_jmd_<scheduler_id>.log	ログファイル
/etc/opt/nec/nqsv/nqs_jmd_cmdapi.conf	コマンド環境ファイル JobManipulator コマンドが接続するJobManipulatorスケジューラを定義したファイルです。本ファイルはテキスト形式であり、システム管理者が管理します。

2.2　パッケージ構成

JobManipulator のプロダクトは、以下のパッケージで構成されています。

プロダクト名	パッケージと機能内容
NEC Network Queuing System V/ ResourceManager	NQSV-Client-X.XX-X.x86_64.rpm コマンドインタフェース（CUI）
NEC Network Queuing System V/ JobManipulator	NQSV-JobManipulator-X.XX-X.x86_64.rpm NQSV 専用バッチスケジューラ

各パッケージのインストールに関しては、*NQSV* 利用の手引 *[*導入編*]*の内容に従って実施してください。

2.3　基本的な環境の構築

本節では、JobManipulator を起動させるために最低限必要な手順を説明します。

2.3.1　環境

以降の JobManipulator の環境構築手順は、インストール環境として以下のシステム構成の場合を例にとって説明します。

バッチサーバホストに JobManipulator をインストールするものとします。

バッチサーバホスト

ホスト名	IPアドレス	マシンID
bsv.nec.co.jp	192.168.1.1	10

ユーザ

NQSV 管理者ユーザ	一般ユーザ
root （バッチサーバホスト）	user1 （バッチサーバホスト、クライアントホストで同一ユーザ名）

キュー

実行キュー名
execque1

2.3.2　パッケージインストール

(1) バッチサーバホスト

バッチサーバホスト上に、NQSV/JobManipulator パッケージをインストールしてください。

クライアントホスト

スケジューラの情報表示、および、スケジューラの管理操作を行うクライアントホストには、NQSV/ResourceManager の中の NQSV-Client-X.XX-X.x86_64.rpm パッケージをインストールしてください。この中に sstat、smgr、sushare コマンド(以降 JobManipulator コマンド)が含まれています。

2.3.3　JobManipulator の起動

次に、root 権限で下記のコマンドを実行すると JobManipulator が起動します。

```
#systemctl start nqs-jmd
```

初めて JobManipulator を起動したとき、スケジューリング状態は停止状態です。起動後に、smgr(1M)で次のサブコマンドを実行することによりスケジューリングを開始できます。詳細は、2.7.10 スケジューリングの開始/停止設定 を参照してください。

```
#smgr -Po
Smgr: start scheduling
Start Scheduling.
```

2.3.4　キュー設定

NQSV システム上でリクエストを実行するためには、リクエストを投入するための実行キューを作成し、設定する必要があります。NQSV 利用の手引［導入編］を参照して NQSV システムの環境を構築し、手順に従って実行キューを作成・設定してください。

NQSV 利用の手引［導入編］の手順と重複しますが、リクエストを実行するためには、作成した実行キューとスケジューラをバインドする必要があります。スケジューラ ID の既定値は 1 となっているので、scheduler_id には 1 を指定してバインド操作を行ってください。

```
#qmgr -P m
Mgr: bind execution_queue scheduler execque1 scheduler_id=1
```

ここでバインドされた実行キューは、次回の JobManipulator の起動時に自動的にバインドされます。

2.3.5　クライアント環境設定

クライアントホスト上で以降 JobManipulator コマンドを用いて JobManipulator スケジューラの情報表示、および、スケジューラの管理操作をするための設定を行います。

設定はクライアントホスト上の/etc/opt/nec/nqsv/nqs_jmd_cmdapi.conf ファイルで行います。本ファイルの jm_host に JobManipulator を起動したホストのホスト名を指定してください。

root 権限でエディタにて/etc/opt/nec/nqsv/nqs_jmd_cmdapi.conf ファイルを開いて、以下の行を追加してください。

```
jm_host    bsv.nec.co.jp
```

なお、JobManipulator コマンドと man データは以下にインストールされます。

コマンドパス

/opt/nec/nqsv/bin
/opt/nec/nqsv/sbin

man パス

/opt/nec/nqsv/man (English)
/opt/nec/nqsv/man /ja (Japanese)

2.3.6　JobManipulator の停止

root 権限で下記のコマンドを実行すると JobManipulator は停止します。

```
#systemctl stop nqs-jmd
```

2.4　ユニットの管理

NQSV/JobManipulator は以下の 1 つのユニットのみを持ちます。

なお、ユニットの詳細については systemd のマニュアルを、systemctl コマンドの詳細については systemctl のマニュアルを参照してください。

パッケージ名	ターゲットユニット名	サービスユニット名
NQSV/JobManipulator	nqs-jmd.target	nqs-jmd.service

拡張子が.service のユニットはサービスユニットと呼ばれ、デーモンを管理するユニットです。

拡張子が.target のユニットはターゲットユニットと呼ばれ、複数のユニットをまとめて扱うためのユニットです（ただし、NQSV/JobManipulator にはターゲットユニットは 1 つです）。

サービスユニットはターゲットユニットに関連づけられており、NQSV/JobManipulator のインストール直後はこの関連付けは有効となっています。

したがって、次回 OS 起動時に NQSV/JobManipulator は自動的に起動します。OS 起動時に自動起動させないようにするには、root 権限で下記のコマンドを実行することで nqs-jmd.target との関連付けを無効にします。

なお、拡張子が.service の場合は、拡張子を省略可能です。

```
#systemctl disable nqs-jmd
```

再び OS 起動時に NQSV/JobManipulator を自動的に起動したいときは、root 権限で下記のコマンドを実行することでサービスユニットの関連付けを有効にします。

```
#systemctl enable nqs-jmd
```

2.5　JobManipulatorの起動に関する設定

2.5.1　コンフィグファイル

NQSV/JobManipulator をインストールしたホスト上のコンフィグファイルにスケジューラ ID、NQSV バッチサーバ等の設定を行うことができます。

コンフィグファイルの標準パスは/etc/opt/nec/nqsv/nqs_jmd.conf です。

コンフィグファイルは次の形式で行を追加します。

〈指示行〉:〈設定値〉

コンフィグファイルには次の設定ができます。

指示行	設定値	説明
JM_SCHNO	スケジューラID	1番以外のスケジューラIDを使用する場合に指定します。既定値は1となっています。この行の指定は必須です。 0から15の整数を指定できます。
JM_SCHNAME	スケジューラ名	qstat(1)コマンドの-Dオプション等で参照できるスケジューラ名の文字列を指定できます。省略可能です。
BSV_HOST	バッチサーバホスト名	NQSV/JobManipulatorをバッチサーバホストと異なるマシンにインストールする場合、JobManipulatorの接続先バッチサーバホスト名を指定します。省略した場合、localhostのバッチサーバに接続します。
JM_CMDPORT	ポート番号	<JM_CMDPORT>+<JM_SCHNO> が、JobManipulatorコマンドからJobManipulatorに接続するためのポート番号となります。省略した場合、13000になります。

必要に応じて root 権限でエディタにてコンフィグファイルを開いて、設定行を追加してください。

JobManipulator の起動時にコンフィグファイル（/etc/opt/nec/nqsv/nqs_jmd.conf）をロードします。コンフィグファイルの不正があった場合、JobManipulator は終了します。

標準パスと異なるファイルをコンフィグファイルとして指定したい場合、/etc/opt/nec/nqsv/jmtab ファイルに書かれている JobManipulator の コンフィグファイル一覧を編集します。

デフォルトでは、/etc/opt/nec/nqsv/jmtab ファイルには "default" だけが書かれていますが、標準パスと異なるコンフィグファイルを使用する場合は、 root 権限でエディタにて /etc/opt/nec/nqsv/jmtab ファイルを開いて、"default" 行を "#" でコメントアウトするか、 削除した

上で、使用するコンフィグファイルをフルパスで追加してください。

[例] /etc/opt/nec/nqsv/nqs_jmd_001.conf を使用する場合

```
# JobManipulator scheduler startup table
# "default" is /etc/opt/nec/nqsv/nqs_jmd.conf

# default
/etc/opt/nec/nqsv/nqs_jmd_001.conf
```

2.5.2　複数の JobManipulator の起動

バッチリクエスト用と会話リクエスト用等、1 つのバッチサーバにスケジューリング設定の異なる複数の JobManipulator を接続する場合の手順を説明します。

まず、1 つのバッチサーバに接続する各々の JobManipulator のスケジューラ ID は一意となるようコンフィグファイルを設定してください。すなわち、1 つのマシンに複数の JobManipulator を起動する場合は、JobManipulator 毎にコンフィグファイルを作成します。この時、スケジューラ ID が一意となるよう JM_SCHNO の設定値を各々のコンフィグファイルで異なる値に設定してください。

次に、/etc/systemd/system 配下にサービスユニットの定義ファイルを作成してください。その際、サンプルファイル（/etc/opt/nec/nqsv/nqs-jmd0.service.sample）を参考にして作成してください。

以下は設定ファイルの例です。このファイルではスケジューラ ID を 2 にしています。それを指定したコンフィグファイル名は/etc/opt/nec/nqsv/nqs_jmd2.conf にしています。

```
#
# Job scheduler configuration file
#

JM_SCHNO: 2
#JM_SCHNAME:
BSV_HOST: bsv.nec.co.jp
#JM_CMDPORT : 13000
#JM_RERUNWAIT: 600
```

以下は、JobManipulator に与える起動時パラメータを記述したファイルの例です。JobManipulator に起動時パラメータを設定しない場合は、作成の必要はありません。ここでは例として、

/etc/opt/nec/nqsv/nqs_jmd2.env として作成しています。JM_PARAM=の後ろにパラメータを記述してください。

```
# Environment variables for NQSV/JobManipulator
# Parameters to give NQSV/JobManipulator

JM_PARAM="-s ON"
```

以下は、サービスユニットの定義ファイルの例です。ここでは例として、/etc/systemd/system/nqs-jmd2.service というファイル名で作成しています。

```
[Unit]
Description=NQSV JobManipulator - Job Scheduler
PartOf=nqs-jmd.target

[Service]
Type=forking
ExecStart=/opt/nec/nqsv/sbin/nqs_jmd  -f  /etc/opt/nec/nqsv/nqs_jmd2.conf
$JM_PARAM
EnvironmentFile=/etc/opt/nec/nqsv/nqs_jmd2.env
PIDFile=/run/nec/nqsv/nqs_jmd.pid/02/jm_pid_file

[Install]
RequiredBy=nqs-jmd.target
```

ExecStart=には実行ファイル名と起動時パラメータを指定します。-f には作成したコンフィグファイルを記述します。

EnvironmentFile=には起動時パラメータを記述したファイルを指定します。何も指定しない場合はこの行は無くても問題ありません。

PIDFile=は JobManipulator のプロセス ID を記述したファイルです。下記の*[nn]*の部分をスケジューラ ID で置き換えてください。スケジューラ ID が 1 桁の場合でも 2 桁で記述してください。(例：03)

```
PIDFile=/run/nec/nqsv/nqs_jmd.pid/[nn]/jm_pid_file
```

以下の systemctl コマンドの実行はすべて root 権限で行ってください。サービスユニットの定義フ

ァイルを作成後、下記のコマンドを実行してください。

```
# systemctl daemon-reload
```

下記のコマンドを実行すると、スケジューラ ID が 2 の JobManipulator を起動します。

```
# systemctl start nqs-jmd2.service
```

OS 起動時に自動的に JobManipulator を起動したい場合は、下記のコマンドを実行します。

```
# systemctl enable nqs-jmd2.service
```

また、自動起動を設定した JobManipulator は下記のコマンドを実行することにより、まとめて起動できます。

```
# systemctl start nqs-jmd.target
```

下記のコマンドを実行すると、スケジューラ ID が 2 の JobManipulator を停止します。

```
# systemctl stop nqs-jmd2.service
```

また、自動起動を設定した JobManipulator は下記のコマンドを実行することにより、まとめて停止できます。

```
# systemctl stop nqs-jmd.target
```

OS 起動時の自動起動を解除したい場合は、下記のコマンドを実行します。

```
# systemctl disable nqs-jmd2.service
```

2.5.3　NQSV R1.05 以前から R1.06 以降への更新

NQSV R1.05 以前のバージョンと R1.06 以降のバージョンでは複数の JobManipulator を実行する際の管理方法が異なります。このため、R1.05 以前のバージョンで複数の JobManipulator を実行している場合に、R1.06 以降のバージョンへ更新する際には移行作業が必要です。以下はその手順です。なお、以下の手順は root 権限で実施してください。

ここでは例として、/etc/opt/nec/nqsv/jmtab に下記のように記述されている場合を説明します。

```
# JobManipulator scheduler startup table
# "default" is /etc/opt/nec/nqsv/nqs_jmd.conf

/etc/opt/nec/nqsv/nqs_jmd.conf
/etc/opt/nec/nqsv/nqs_jmd2.conf
/etc/opt/nec/nqsv/nqs_jmd3.conf
```

また、各コンフィグファイルの内容は以下の通りとします。

nqs_jmd.conf:

```
#
# Job scheduler configuration file
#

JM_SCHNO: 1
#JM_SCHNAME:
BSV_HOST: bsv.nec.co.jp
#JM_CMDPORT : 13000
#JM_RERUNWAIT: 600
```

nqs_jmd2.conf:

```
#
# Job scheduler configuration file
#

JM_SCHNO: 2
#JM_SCHNAME:
BSV_HOST: bsv.nec.co.jp
#JM_CMDPORT : 13000
#JM_RERUNWAIT: 600
```

nqs_jmd3.conf:

```
#
# Job scheduler configuration file
#

JM_SCHNO: 3
#JM_SCHNAME:
BSV_HOST: bsv.nec.co.jp
#JM_CMDPORT : 13000
#JM_RERUNWAIT: 600
```

まず、JobManipulator を更新します。

```
# rpm -Uvh NQSV-JobManipulator-1.06-xxx.x86_64.rpm
```

次に、/etc/opt/nec/nqsv/nqs-jmd0.service.sample を基にサービスユニットの定義ファイルを作成します。この場合、実行する三つの JobManipulator にあわせて、定義ファイルも三つ作成します。ここではファイル名を nqs-jmd.service、nqs-jmd2.service、nqs-jmd3.service とします。

nqs-jmd.service:

```
[Unit]
Description=NQSV JobManipulator - Job Scheduler
PartOf=nqs-jmd.target

[Service]
Type=forking
ExecStart=/opt/nec/nqsv/sbin/nqs_jmd   -f   /etc/opt/nec/nqsv/nqs_jmd.conf
$JM_PARAM
EnvironmentFile=/etc/opt/nec/nqsv/nqs_jmd.env
PIDFile=/run/nec/nqsv/nqs_jmd.pid/01/jm_pid_file

[Install]
RequiredBy=nqs-jmd.target
```

nqs-jmd2.service:

```
[Unit]
```

```
Description=NQSV JobManipulator - Job Scheduler
PartOf=nqs-jmd.target

[Service]
Type=forking
ExecStart=/opt/nec/nqsv/sbin/nqs_jmd  -f  /etc/opt/nec/nqsv/nqs_jmd2.conf
$JM_PARAM
EnvironmentFile=/etc/opt/nec/nqsv/nqs_jmd2.env
PIDFile=/run/nec/nqsv/nqs_jmd.pid/02/jm_pid_file

[Install]
RequiredBy=nqs-jmd.target
```

nqs-jmd3.service:

```
[Unit]
Description=NQSV JobManipulator - Job Scheduler
PartOf=nqs-jmd.target

[Service]
Type=forking
ExecStart=/opt/nec/nqsv/sbin/nqs_jmd  -f  /etc/opt/nec/nqsv/nqs_jmd3.conf
$JM_PARAM
EnvironmentFile=/etc/opt/nec/nqsv/nqs_jmd3.env
PIDFile=/run/nec/nqsv/nqs_jmd.pid/03/jm_pid_file

[Install]
RequiredBy=nqs-jmd.target
```

定義ファイル作成時のポイントは以下のとおりです。

1. ExecStartで指定するnqs_jmdの-fオプションには、jmtabに記述していたコンフィグファイルの一つをパラメータとして与えます。
2. EnvironmentFileには、nqs_jmdに与える起動時パラメータを記述したファイル名を指定します。
3. コンフィグファイルに記述されたスケジューラIDにあわせて、PIDFileで指定するディレクトリを指定します。

systemctl コマンドで設定を反映してください。

```
# systemctl daemon-reload
```

必要に応じて自動起動を設定します。

```
# systemctl enable nqs-jmd.service nqs-jmd2.service nqs-jmd3.service
```

2.5.4　JobManipulator 起動時のオプション

フェール・オーバ運用時の IP アドレスや、スケジューリング機能の開始・停止、BSV との接続に使用するポート番号を指定して JobManipulator を起動することができます。

フェール・オーバ運用を行う場合は、-a オプションで IP アドレスを指定して起動してください。詳細については、4.11 フェール・オーバ対応を参照してください。

スケジューリング状態を指定して起動する場合は、-s オプションを指定して起動してください。
ON を指定すると、スケジューリング開始状態となります。
OFF を指定すると、スケジューリング停止状態となります。
-s オプションを指定しない場合、前回起動時の設定を引き継ぎます。初めて起動するときに-s オプションを指定しない場合は、スケジューリング停止状態となります。

JobManipulator が BSV との接続に使用するポート番号を既定値（602 番）から変更して起動する場合は、-p オプションにポート番号を指定して起動してください。

JobManipulator 起動時のオプションは、/etc/opt/nec/nqsv/nqs_jmd.env の JM_PARAM に記入してください。

[例] -s ON と -a 192.168.1.1 と-p 12345 を指定して JobManipulator を起動する場合

```
# Environment variables for NQSV/JobManipulator
# Parameters to give NQSV/JobManipulator

JM_PARAM="-s ON -a 192.168.1.1 -p 12345"
```

2.5.5　コマンド環境ファイル

クライアントホスト上の/etc/opt/nec/nqsv/nqs_jmd_cmdapi.conf ファイルで行う JobManipulator コマンドに関する設定について説明します。

/etc/opt/nec/nqsv/nqs_jmd_cmdapi.conf ファイルでは標準では次のようにスケジューラ ID が 1 番の JobManipulator に接続するようになっています。

コンフィグファイルで JM_SCHNO 指示行に 1 以外を指定して JobManipulator を起動したときは、次の行を編集することで JobManipulator コマンドが接続する標準のスケジューラ ID を変更できます。

```
sch_id      1
```

複数の JobManipulator を起動する場合に、標準で接続するスケジューラ ID と異なるスケジューラに接続したい場合は、JobManipulator コマンドの-s オプションでスケジューラ ID を指定します。

詳細については、*NQSV*利用の手引*/*リファレンス編*/*を参照してください。

コンフィグファイルで JM_CMDPORT 指示行を指定して JobManipulator を起動したとき、JobManipulator コマンドが JobManipulator の接続するポート番号の設定は、/etc/opt/nec/nqsv/nqs_jmd_cmdapi.conf ファイルに次の行を追加してください。ここで<jm_base_port>+<スケジューラ ID>が、JobManipulator コマンドから JobManipulator に接続するポート番号となります。

```
jm_base_port <JM_CMDPORTで指定したポート番号>
```

jm_host に JobManipulator の動作ホスト名を指定してください。

```
jm_host < JobManipulatorの動作ホスト名>
```

2.6　スケジューラログ設定

- スケジューラのログ情報をログファイルに出力する設定を行います。
- ログファイルのパス
  スケジューラのログファイルのパス名は任意に指定可能ですが、指定がない場合はデフォルトのパス(/var/opt/nec/nqsv/nqs_jmd_<scheduler_id>.log)に出力します。
  スケジューラ運用中にパス名が変更された場合は元のパス名のファイルはクローズし、新しく設定されたパス名でファイルを作成しログ出力します。
- ログレベル
  ログレベルには、1 から 5 まで指定できます。既定値は1です。通常の運用においては、ログレベル 1で運用することを推奨します。
- ログファイルのサイズ
  ログファイルのサイズも設定可能です。ログファイルサイズの初期設定値は2MBです。サイズの指定がない場合は現在のサイズが引き継がれ、サイズに0が指定された場合は無制限となります。
- ログファイルのバックアップ数
  ログファイルのバックアップ数も設定可能で初期設定値は1です。
  バックアップ数の指定がない場合は現在のバックアップ数が引き継がれ、バックアップ数に0が指定された場合は1となります。
  ログ出力時に指定サイズを超えていた場合、順番に数字がついたバックアップファイルを作成するとともに、新規ファイルにログ出力を行います。

上記の各設定は smgr(1M)コマンドの set logfile サブコマンドにより行います。

```
# smgr -P m
Smgr: set logfile file = /var/opt/nec/nqsv/nqs_jmd_<scheduler_id>.log  size = (1000000)  save = 10
```

2.7　スケジューリングパラメータ設定

本節では、実際に JobManipulator でスケジューリングを行うための各種パラメータの設定方法を説明します。

2.7.1　ランリミット設定

ランリミットは、同時実行可能なリクエスト数の制限値です。

(1)　同時実行リクエスト数制限値

リクエストを同時に実行できる値（同時実行リクエスト数制限値）を設定できます。

同時実行リクエスト数制限値を超えている時間帯には、制限対象のリクエストをアサイン・エスカレーションできません。各項目と内容は以下の通りです。

項目	内容
スケジューラ単位	
スケジューラ単位での同時実行リクエスト数制限 global_run_limit	スケジューラ単位で同時実行リクエスト数を制限します。
スケジューラ単位でのユーザ一律/ユーザ個別の同時実行リクエスト数制限 global_user_run_limit	スケジューラ単位でユーザ一律/ユーザ個別の同時実行リクエスト数を制限します。ユーザ個別の同時実行リクエスト数制限値は、ユーザ指定で設定します。
スケジューラ単位でのグループ一律/グループ個別の同時実行リクエスト数制限 global_group_run_limit	スケジューラ単位でグループ一律/グループ個別の同時実行リクエスト数を制限します。グループ個別の同時実行リクエスト数制限値は、グループ指定で設定します。
キュー単位	
キュー単位での同時実行リクエスト数制限 queue run_limit	キュー単位で同時実行リクエスト数を制限します。
キュー単位でのユーザ一律/ユーザ個別の同時実行リクエスト数制限 queue user_run_limit	キュー単位でユーザ一律/ユーザ個別の同時実行リクエスト数を制限します。ユーザ個別の同時実行リクエスト数制限値は、ユーザ指定で設定します。
キュー内でのグループ一律/グループ個別の同時実行リクエスト数制限 queue group_run_limit	キュー単位でグループ一律/グループ個別の同時実行リクエスト数を制限します。グループ毎の同時実行リクエスト数制限値は、グループ指定で設定します。

コンプレックスキュー単位	
コンプレックスキュー単位での同時実行リクエスト数制限 complex_queue run_limit	コンプレックスキュー単位で同時実行リクエスト数を制限します。
コンプレックスキュー単位でのユーザ一律の同時実行リクエスト数制限 complex_queue user_run_limit	コンプレックスキュー単位でユーザ一律の同時実行リクエスト数を制限します。 なお、ユーザ個別に同時実行リクエスト数制限を設定することはできません。全ユーザに対して同じ値が同時実行リクエスト数制限としてかけられます。
(コンプレックスキュー詳細については、第2章 2.7.5 コンプレックスキュー機能の設定 を参照してください。)	

* デフォルトでは、ユーザ/グループ個別の同時実行リクエスト数制限値は設定していません。
他の制限値のデフォルトは、0(無制限)です。
* ユーザ/グループ個別の制限値を設定している場合は、ユーザ/グループ一律の制限値ではなく、ユーザ/グループ個別の制限値により制限されます。

上記制限値の設定は、**smgr**(1M)コマンドの **set** サブコマンドにより行います。
制限を使用しない場合は、その制限値を 0 にすることで無視することができます。

```
# smgr -P m
Smgr: set global_run_limit = 100
Smgr: set global_user_run_limit = 3
Smgr: set global_user_run_limit = 2 users = (userA,userB)
Smgr: set global_group_run_limit = 20
Smgr: set global_group_run_limit = 10 groups = groupA
Smgr: set queue run_limit = 100 bq1
Smgr: set queue user_run_limit = 2 bq1
Smgr: set queue user_run_limit = 3 users = userA bq1
Smgr: set queue group_run_limit = 15 bq1
Smgr: set queue group_run_limit = 5 groups = groupA bq1

Smgr: set complex_queue run_limit = 100 cq1
Smgr: set complex_queue user_run_limit = 2 cq1
```

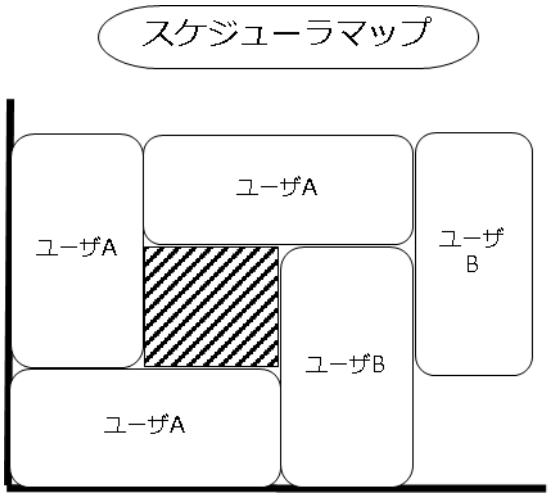


図 2-2：同時実行リクエスト数制限の例

▨の空領域に対して

- 同時実行リクエスト数制限が2の場合
  この領域へアサイン可能なリクエストはありません。
- ユーザ一律の同時実行リクエスト数制限が２の場合
  この領域へはユーザAのリクエストはアサイン不可ですがユーザBのリクエストはアサイン可能です。
- ユーザAとBのユーザ個別の同時実行リクエスト数制限が各々３と２の場合
  ユーザAとユーザBのリクエストがどちらもアサイン可能です。

スケジューラ単位の設定は、**sstat**(1)コマンドの**-S -f** オプションで確認し、キュー単位の設定は、**-Q -f** オプションで確認します。また、ユーザとグループ個別の設定は、**--limit** オプションを併せて指定することで確認します。設定値が 0(無制限)の場合は、UNLIMITED と表示します。

また、ユーザ/グループ個別の同時実行リクエスト数制限は、**smgr**(1M)コマンドの **delete** サブコマンドにより削除できます。

```
# smgr -P m
Smgr: delete global_group_run_limit groups = groupA
Smgr: delete global_user_run_limit users = (userA,userB)
Smgr: delete queue group_run_limit groups = groupA bq1
Smgr: delete queue user_run_limit users = userA bq1
```

同時実行CPU台数制限値

リクエストを同時に実行できる値（同時実行 CPU 台数制限値）を設定できます。同時実行 CPU 台数制限値を超えている時間帯には、制限対象のリクエストをアサイン・エスカレーションできません。同時実行 CPU 台数制限機能で制限される使用 CPU 台数は、リクエストのジョブごとの CPU 台数を用いて求められます。この値は、**qstat(1)**コマンドの**-f** オプションの**"Logical Host Resources"**の**"CPU Number"**の項目で確認できます。詳しくは NQSV 利用の手引［操作編］を参照してください。

各項目と内容は以下の通りです。

項目	内容
スケジューラ単位	
スケジューラ単位でのユーザ一律/ユーザ個別の同時実行 CPU 台数制限 global_user_cpu_run_limit	スケジューラ単位でユーザ一律/ユーザ個別の同時実行CPU台数を制限します。ユーザ個別の同時実行CPU台数制限値は、ユーザ指定で設定します。
スケジューラ単位でのグループ一律/グループ個別の同時実行CPU台数制限 global_group_cpu_run_limit	スケジューラ単位でグループ一律/グループ個別の同時実行CPU台数を制限します。グループ個別の同時実行CPU台数制限値は、グループ指定で設定します。
キュー単位	
キュー単位でのユーザ一律/ユーザ個別の同時実行CPU台数制限 queue user_cpu_run_limit	キュー単位でユーザ一律/ユーザ個別の同時実行CPU台数を制限します。ユーザ個別の同時実行CPU台数制限値は、ユーザ指定で設定します。
キュー単位のグループ一律/グループ個別の同時実行CPU台数制限 queue group_cpu_run_limit	キュー単位でグループ一律/グループ個別の同時実行CPU台数を制限します。グループ個別の同時実行CPU台数制限値は、グループ指定で設定します。

* デフォルトでは、ユーザ/グループ個別の同時実行 CPU 台数制限値は設定していません。他の制限値のデフォルトは 0(無制限)です。
* ユーザ/グループ個別の制限値を設定している場合は、ユーザ/グループ一律の制限値ではなく、ユーザ/グループ個別の制限値により制限されます。

同時実行 CPU 台数制限値の設定は、**smgr**(1M)コマンドの **set** サブコマンドにより行います。
制限を使用しない場合は、その制限値を 0 にすることで無視することができます。

```
# smgr -P m
Smgr: set global_user_cpu_run_limit = 150
Smgr: set global_user_cpu_run_limit = 100 users = (userA,userB)
Smgr: set global_group_cpu_run_limit = 1500
Smgr: set global_group_cpu_run_limit = 1000 groups = groupA
```

```
Smgr: set queue user_cpu_run_limit = 150 bq1
Smgr: set queue user_cpu_run_limit = 100 users = userA bq1
Smgr: set queue group_cpu_run_limit = 1500 bq1
Smgr: set queue group_cpu_run_limit = 1000 groups = groupA bq1
```

スケジューラ単位のユーザ・グループ一律の設定値は、**sstat**(1)コマンドの-S -f オプションで確認し、キュー単位のユーザ・グループ一律の設定値は、-Q -f オプションで確認します。

また、ユーザとグループ個別の設定は、--limit オプションを併せて指定することで確認します。設定値が 0(無制限)の場合は、UNLIMITED と表示します。

また、ユーザ/グループ個別の同時実行 CPU 台数制限は、**smgr**(1M)コマンドの **delete** サブコマンドにより削除できます。

```
# smgr -P m
Smgr: delete global_user_cpu_run_limit users = (userA,userB)
Smgr: delete global_group_cpu_run_limit groups = groupA
Smgr: delete queue user_cpu_run_limit users = userA bq1
Smgr: delete queue group_cpu_run_limit groups = groupA bq1
```

同時実行VEノード数制限値

リクエスト内で使用する VE ノード数について、同時間帯に使用可能な VE ノード数の上限を設定できます。同時実行 VE ノード数制限値を超えている時間帯では、JobManipulator は制限対象のリクエストをアサイン・エスカレーションしません。この機能で制限する使用 VE ノード数は、リクエストの論理ホスト毎の VE ノード数制限を用いて求められます。この値は、**qstat(1)**コマンドの**-f** オプションの**"Logical Host Resources"**の**"VE Node Number"**の項目で確認できます。詳しくは NQSV 利用の手引 [操作編] を参照してください。

なお、本制限機能は、実行ホストが SX-Aurora TSUBASA の場合のみ利用可能です。

各項目と内容は以下の通りです。

項目	内容
スケジューラ単位	
スケジューラ単位でのユーザ一律/ユーザ個別の同時実行 VE ノード数制限 global_user_ve_run_limit	スケジューラ単位でユーザ一律／ユーザ個別の同時実行VEノード数を制限します。ユーザ個別の同時実行VEノード数制限値は、ユーザ指定で設定します。
スケジューラ単位でのグループ一律/グループ個別の同時実	スケジューラ単位でグループ一律／グループ個別の同時実行VEノード数を制限します。グループ個別の同時実行VEノード

行VEノード数制限 global_group_ve_run_limit	数制限値は、グループ指定で設定します。
キュー単位	
キュー単位でのユーザ一律/ユーザ個別の同時実行VEノード数制限 queue user_ve_run_limit	キュー単位でユーザ一律／ユーザ個別の同時実行VEノード数を制限します。ユーザ個別の同時実行VEノード数制限値は、ユーザ指定で設定します。
キュー単位のグループ一律/グループ個別の同時実行VEノード数制限 queue group_ve_run_limit	キュー単位でグループ一律／グループ個別の同時実行VEノード数を制限します。グループ個別の同時実行VEノード数制限値は、グループ指定で設定します。

* デフォルトでは、ユーザ／グループ個別の同時実行 VE ノード数制限値は設定していません。他の制限値のデフォルトは 0（無制限）です。
* あるユーザ／グループに個別の制限値を設定した場合は、ユーザ／グループ一律の制限値より優先されるため、個別の制限値で制限されます。

同時実行 VE ノード数制限値の設定は、**smgr**(1M)コマンドの **set** サブコマンドにより行います。制限を使用しない場合は、値を 0 にしてください。

```
# smgr -P m
Smgr: set global_user_ve_run_limit = 150
Smgr: set global_user_ve_run_limit = 100 users = (userA,userB)
Smgr: set global_group_ve_run_limit = 1500
Smgr: set global_group_ve_run_limit = 1000 groups = groupA
Smgr: set queue user_ve_run_limit = 150 bq1
Smgr: set queue user_ve_run_limit = 100 users = userA bq1
Smgr: set queue group_ve_run_limit = 1500 bq1
Smgr: set queue group_ve_run_limit = 1000 groups = groupA bq1
```

スケジューラ単位のユーザ／グループ一律の設定値は、**sstat**(1)コマンドの-S -f オプションで確認し、キュー単位のユーザ・グループ一律の設定値は、-Q -f オプションで確認します。

また、ユーザとグループ個別の設定は、--limit オプションを併せて指定することで確認します。設定値が 0（無制限）の場合は、UNLIMITED と表示します。

また、ユーザ／グループ個別の同時実行 VE ノード数制限は、**smgr**(1M)コマンドの **delete** サブコマンドにより削除できます。

```
# smgr -P m
Smgr: delete global_user_ve_run_limit users = (userA,userB)
```

```
Smgr: delete global_group_ve_run_limit groups = groupA
Smgr: delete queue user_ve_run_limit users = userA bq1
Smgr: delete queue group_ve_run_limit groups = groupA bq1
```

ランリミットの変更は、アサイン済みリクエストに影響を与えません。それらが変更後の制限値を超えていても、アサイン状態を継続します。変更後の制限値は、次のスケジューリング（インターバル毎のスケジューリングやエスカレーション）より有効になります。

2.7.2　アサインリミット設定

リクエストを同時にアサインできる値を設定できます。各項目と内容は以下の通りです。

この値は、アサインマップ中にアサインされている実行中を含めたリクエストの個数です。

* 以下の制限値間の優先順位はありません。各制限値の単位で制限の照合を行い、制限に抵触した時点でアサイン処理を中断します。

項目	内容
スケジューラ単位	
スケジューラ単位でのユーザ一律の同時アサインリクエスト数制限 global_user_assign_limit	スケジューラ単位でユーザ毎に同時にアサインできるリクエスト数を制限することができます。 ユーザ一律の同時アサインリミットを超えていた場合、アサイン不可とします。 なお、ユーザ別に同時アサインリクエスト数制限を設定することはできません。全ユーザに対して同じ値がアサインリミット制限としてかけられます。 制限値として0が指定された場合は無制限とし、デフォルトは無制限とします。
キュー単位	
キュー内でユーザ一律の同時アサインリクエスト数制限 queue user_assign_limit	キュー内でユーザの同時にアサインできるリクエスト数を制限することができます。 キュー内でユーザの同時アサインリミットを超えていた場合、アサイン不可とします。 なお、ユーザ別に同時アサインリクエスト数制限を設定することはできません。全ユーザに対して同じ値がアサインリミット制限としてかけられます。 制限値として0が指定された場合は無制限とし、デフォルトは無制限とします。
コンプレックスキュー単位	
コンプレックスキュー内でのユーザ一律の同時アサインリクエスト数制限 complex_queue user_assign_limit	コンプレックスキュー内でユーザの同時にアサインできるリクエスト数を制限することができます。 コンプレックスキュー内でユーザの同時アサインリミットを超えていた場合、アサイン不可とします。 なお、ユーザ別に同時アサインリクエスト数制限を設定することはできません。全ユーザに対して同じ値がアサインリミット制限としてかけられます。 制限値として0が指定された場合は無制限とし、デフォルトは無制限とします。
ホスト単位	
ホスト全体での使用可能ＣＰＵ台数制限	ホスト全体で使用可能なＣＰＵ台数を制限することができます。 ホストで実装しているＣＰＵ台数に対して何%まで同時に使用可

executionhost cpunum_limit_ratio	能とするかの制限をかけます。 係数1(100%)の場合は実装ＣＰＵ数までしかジョブをアサインしません。係数2(200%)とするとたとえば実装ＣＰＵ数８のところ１６ＣＰＵまでのジョブをアサインできます。 係数0(0%)とした場合、本制限は無効となりＣＰＵ台数はチェックしません。
ホスト全体での使用可能メモリ制限 executionhost memsz_limit_ratio	ホスト全体で使用可能なメモリ量を制限することができます。 ホストで実装しているメモリ量に対して何%まで同時に使用可能とするかの制限をかけます。係数1(100%)の場合は実装メモリ量までしかジョブをアサインしません。係数2(200%)とすると実装メモリ量10GBのところ20GBまでのジョブをアサインできます。 係数0(0%)とした場合、本制限は無効となりメモリ量はチェックしません。
RSG単位	
RSG毎での使用可能ＣＰＵ台数制限 executionhost rsg_cpunum_limit_ratio	RSG毎で使用可能なＣＰＵ台数を制限することができます。 RSG単位で設定しているＣＰＵ台数(Icpu)に対して何%まで同時に使用可能とするかの制限をかけます。係数1(100%)の場合は設定されたＣＰＵ台数(Icpu)までしかジョブをアサインしません。係数2(200%)とするとIcpu=4の場合、８ＣＰＵまでのジョブをアサインできます。係数0(0%)とした場合、本制限は無効となりＣＰＵ台数はチェックしません。
RSG毎での使用可能メモリ制限 executionhost rsg_memsz_limit_ratio	RSG毎で使用可能なメモリ量を制限することができます。 RSG単位で設定しているメモリ量(Imem)に対して何%まで同時に使用可能とするかの制限をかけます。係数1(100%)の場合は設定されたメモリ量(Imem)までしかジョブをアサインしません。係数2(200%)とするとImem=10GBの場合、20GBまでのジョブをアサインできます。 係数0(0%)とした場合、本制限は無効となりメモリ量はチェックしません。

RSG(Resource Sharing Group)は、CPUSET 機能による実行ホストのリソース分割後の各分割単位の名称です。CPUSET 機能の詳細は、NQSV 利用の手引[管理編] を参照してください。

- RSGを変更する場合は、キューに投入したリクエストを削除し、RSGを変更した後、リクエストを再投入する必要があります。
- CPUレシオ(cpunum_limit_ratio)とソケットスケジューリング機能を併用する場合は、もしCPUレシオが1以外の値に設定されていれば、ジョブにアサインできるCPU台数がホストに実装されている台数と異なるので、CPUコア単位の割り当てができなくなります。そのため、ジョブサーバの環境設定ファイル(/etc/opt/nec/nqsv/nqs_jsv.env)に下記の内容を記載して、ジョブサーバを起動してください。

  JSV_PARAM='--no-cpu-topology'

この場合は、CPUコアの割り当てがソケット単位となります。ソケットスケジューリング機能の詳細は、NQSV 利用の手引き[管理編]の18.1 ソケットスケジューリング機能を参照してください。

上記制限値の設定は、**smgr**(1M)コマンドの **set** サブコマンドにより行います。資源制限を使用しない場合は、その制限値を 0 にすることで無視することができます。

```
# smgr -P m
Smgr: set global_user_assign_limit = 10
Smgr: set queue user_assign_limit = 0 bq1
Smgr: set complex_queue user_assign_limit = 0 cq1
Smgr: set executionhost cpunum_limit_ratio = 2 hostname
Smgr: set executionhost memsz_limit_ratio = 0 hostname
Smgr: set executionhost rsg_cpunum_limit_ratio = 1.5 rsg_number = 0 hostname
Smgr: set executionhost rsg_memsz_limit_ratio = 0 rsg_number = 0 hostname
```

実行ホストの代わりにノードグループを指定することで、ノードグループに属する実行ホストに対して一括設定が可能です。

```
# smgr -P m
Smgr: set executionhost cpunum_limit_ratio = 2 node_group = GrpA
Smgr: set executionhost memsz_limit_ratio = 0 node_group = GrpA
Smgr: set executionhost rsg_cpunum_limit_ratio = 1.5 rsg_number = 0 node_group = GrpA
Smgr: set executionhost rsg_memsz_limit_ratio = 0 rsg_number = 0 node_group = GrpA
```

BSV にてノードグループに実行ホストを追加した際に、追加した実行ホストに CPU 台数／メモリ制限を設定するには、追加した実行ホストに対して個別に設定するか、再度ノードグループに対して設定を実施する必要があります。
また、ノードグループから実行ホストを削除する際も、削除した実行ホストの設定値がノードグループ単位の一括設定時の設定値のままであるため、設定値を変更するには個別に設定する必要があります。

上記の実行ホストに対する設定は、システムに登録（ATTACH）した実行ホストのみに対して実施できます。システムから実行ホストを削除（DETACH)すると、当該ホストに対する設定値も DB か

ら削除されます。

実行ホストに対する設定値は、**sstat**(1)コマンドの-E [-a]オプションで表示します。-E [-a] -g node_group オプションでノードグループに属する実行ホストの使用可能リソース制限値を表示可能です。

```
#sstat -E -g node_groupA
ExecutionHost   CPUNRatio  MemRatio
--------------- -------------------
hostA            2.000000  0.00000
  (RSG 0)        1.500000  0.00000
  (RSG 1)        0.500000  0.00000
hostB            2.000000  0.00000
  (RSG 0)        1.500000  0.00000
  (RSG 1)        0.500000  0.00000
```

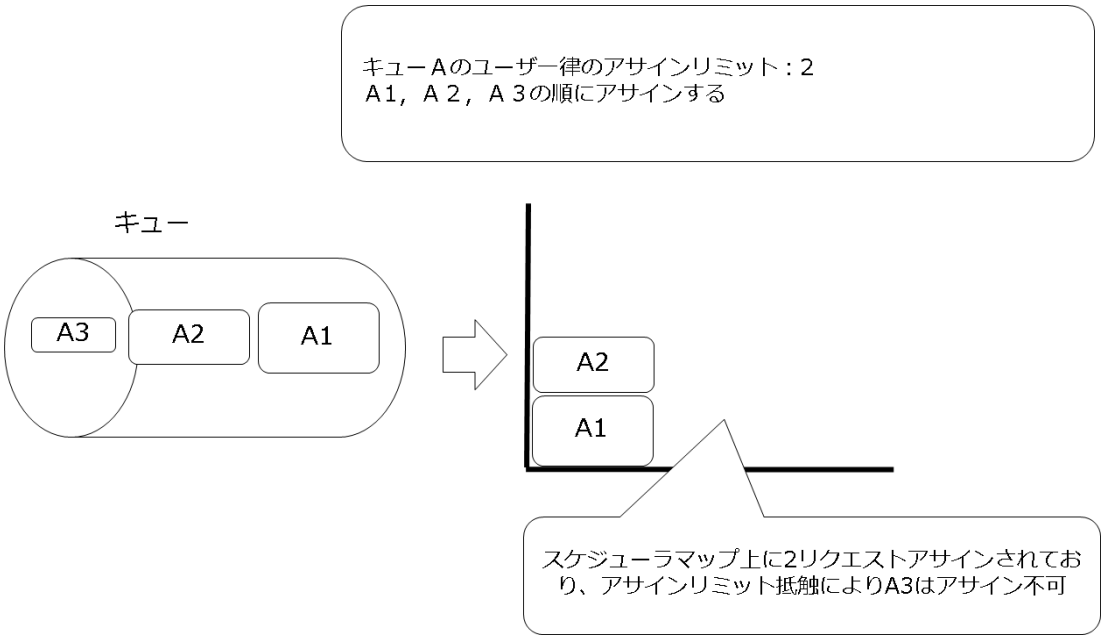


図 2-3：アサインリミットの例

2.7.3　リクエスト優先順位設定

スケジューリング対象リクエストが複数あった場合、どのリクエストからスケジューリングするかの優先順位を調整できるパラメータ（3.1.3. スケジューリングプライオリティ）を設定できます。

重み係数の設定は、**smgr**(1M) コマンドの **set** サブコマンドで行います。使用できるパラメータは以下のとおりです。

パラメータ名	内容
weight_request_priority	リクエストプライオリティに対する重み付け
weight_cpu_number	宣言CPU数に対する重み付け
weight_elapse_time	宣言経過時間に対する重み付け
weight_memory_size	宣言メモリサイズに対する重み付け
weight_job_number	ジョブ数に対する重み付け
weight_ve_number	宣言VE数に対する重み付け
weight_run_wait_time	キューに投入してからの実行待ち時間に対する重み付け
weight_assign_wait_time	アサイン可能になってからの実行待ち時間に対する重み付け
weight_restart_wait_time	SUSPENDされてからの再開待ち時間に対する重み付け
weight_user_share	ユーザシェアに対する重み付け
baseup_interrupted	緊急リクエストによってSUSPENDされたリクエストに対するベースアップ
baseup_reschedule	再スケジューリング対象となったリクエストに対するベースアップ
baseup_user_definition	ユーザ定義に対するベースアップ
pastusage_weight_request_priority	使用実績のリクエストプライオリティに対する重み付け
pastusage_weight_cpu_number	使用実績のCPU数に対する重み付け
pastusage_weight_elapse_time	使用実績の経過時間に対する重み付け
pastusage_weight_memory_size	使用実績のメモリサイズに対する重み付け
pastusage_weight_ve_number	使用実績のVE数に対する重み付け

2.7.4　キュー種別設定

「4.1 緊急リクエスト」「4.2 特別リクエスト」において、実行中のリクエストを止めてまで早急に実行開始したいリクエストを扱うため、実行キューに対して緊急リクエスト用および特別リクエスト用のキューであるという設定を行います。

この設定は、**smgr**(1M)コマンドの **set queue type** サブコマンドで行います。この設定はJobManipulator 内のみ有効であり、実行キュー属性への変更ではありません。

```
# smgr -P m
```

```
Smgr: set queue type = urgent bq1      #bq1キューを緊急キューに設定
Smgr: set queue type = special bq1     #bq1キューを特別キューに設定
Smgr: set queue type = normal bq1      #bq1キューを通常キューに設定
```

但しリクエストが投入されているキューのキュー種別は変更できません。

2.7.5　コンプレックスキュー機能の設定

機能概要

複数のキューをまとめて以下の３つの制限を行うことが可能です。この機能をコンプレックスキュー機能と呼び、複数のキューをまとめたものをコンプレックスキューと呼びます。

- 同時実行リクエスト数制限
- ユーザー一律の同時実行リクエスト数制限
- ユーザー一律の同時アサインリクエスト数制限

これにより、１つのキューの制限だけでなく、コンプレックスキューごとの制限の設定も可能です。複数のコンプレックスキューに同一のキューを所属させることもでき、制限値をより柔軟に設定できます。

* 以下はコンプレックスキューのイメージ図です。

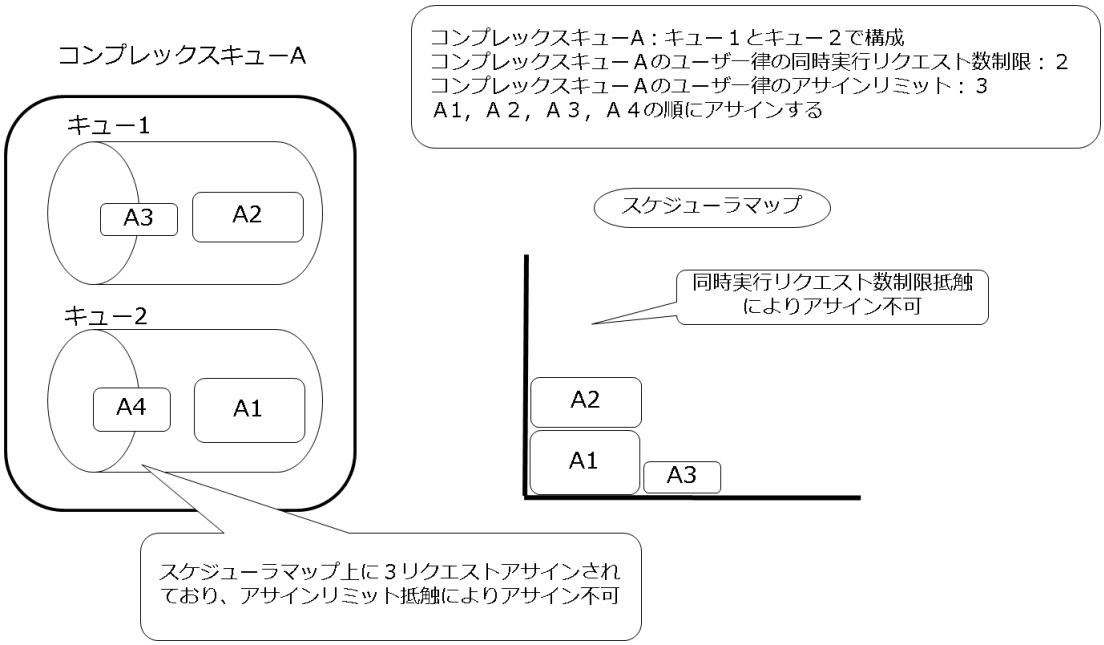


図 2-4：コンプレックスキューの例

コンプレックスキューの設定・実行キューの追加／削除は、smgr(1M)コマンドで行います。またコンプレックスキューの情報表示は、**sstat**(1)コマンドで行います。 コンプレックスキューの設定は、

設定を行った後のスケジューリングから対象となります。

(1)　コンプレックスキューの作成

コンプレックスキューの作成は **smgr**(1M)コマンドの **create complex_queue** サブコマンドで行います。

```
# smgr -P m
Smgr: create complex_queue = complex-queue-name  queue = (queue-name [,queue-name...])
```

- complex-queue-name には、作成するコンプレックスキュー名を指定します。
- 最大コンプレックスキュー名長は、63 文字です。
- コンプレックスキューの作成時には各制限値（同時実行リクエスト数制限／ユーザ一律の同時実行リクエスト数制限／同時アサインリクエスト数制限）は、無制限に設定されます。
- queue-name には、作成したコンプレックスキューに属する実行キュー名を指定します。
- 最大実行キュー名長は、15文字です。
- 実行キューには、以下のような実行キューが指定できます。
- 他のコンプレックスキューに属しているキュー(キューは複数のコンプレックスキューに所属できます。)
- 異なるキュータイプの実行キュー
- JobManipulatorが管理していないキュー
- コンプレックスキューの作成には管理者権限が必要です。

指定されたコンプレックスキュー名、実行キュー名のいずれかに不備がある場合、コンプレックスキューは作成されません。

---

* 以下の場合、エラーとなりコンプレックスキューは作成されません。

- 作成しようとするコンプレックスキューが既に存在する場合
  エラーメッセージ：Specified complex queue already exists.
- 作成しようとするコンプレックスキューの文字数が63文字を超える場合
  エラーメッセージ：Complex queue name too long.
- コンプレックスキューに属する実行キューの文字数が15文字を超える場合

エラーメッセージ：Execution queue name too long.

- コマンドを実行したユーザに管理者権限がない場合
  エラーメッセージ：Operation not permitted.

(2) コンプレックスキューの削除

コンプレックスキューの削除は **smgr**(1M)コマンドの **delete complex_queue** サブコマンドで行います。

```
# smgr -P m
Smgr: delete complex_queue = complex-queue-name
```

- complex-queue-name には、削除するコンプレックスキュー名を指定します。
- コンプレックスキューの削除には管理者権限が必要です。

* 以下の場合、エラーとなり、削除が行われません。

- 削除しようとするコンプレックスキューが存在しない場合
  エラーメッセージ：Specified complex queue doesn't exist.
- コマンドを実行したユーザに管理者権限がない場合
  エラーメッセージ：Operation not permitted.

(3) コンプレックスキューへの実行キューの追加

実行キューの追加は **smgr**(1M)コマンドの **add complex_queue** サブコマンドで行います。

```
 # smgr -P m
Smgr: add complex_queue = (queue-name [,queue-name...])  complex-queue-name
```

- queue-name には、complex-queue-name で指定されたコンプレックスキューに追加する実行キュー名を指定します。
- 実行キュー名長は、最大15文字です。
- 実行キューには、以下のような実行キューが指定できます。
  - 他のコンプレックスキューに属しているキュー(キューは複数のコンプレックスキューに所属できます。)

- 異なるキュータイプの実行キュー
- JobManipulatorが管理していないキュー
  (将来JobManipulatorで管理することを予定しているキューを、あらかじめコンプレックスキューに所属させておくことができます。)

- 実行キューは、複数のコンプレックスキューに所属し、かつすべてのコンプレックスキューを有効できます。
- 実行キューの追加には、管理者権限が必要です。

複数指定された実行キュー名のいずれかが文字数の制限を超えている場合、実行キューはいずれも追加されません。

---

* 以下の場合、エラーとなり実行キューは追加されません。
  - 指定したコンプレックスキューが存在しない場合
    エラーメッセージ：Specified complex queue doesn't exist.
  - コマンドを実行したユーザに管理者権限がない場合
    エラーメッセージ：Not permitted to modify attribute.
  - 指定した実行キューの文字数が15文字を超える場合
    エラーメッセージ：Execution queue name too long.

(4)　コンプレックスキューからの実行キューの削除

実行キューの削除は **smgr**(1M)コマンドの **remove complex_queue** サブコマンドで行います。

```
# smgr -P m
Smgr: remove complex_queue = (queue-name [,queue-name...])  complex-queue-name
```

* 以下の場合はエラーとなり、実行キューは削除されません。
  - 指定したコンプレックスキューが存在しない場合
    エラーメッセージ：Specified complex queue doesn't exist.
  - コンプレックスキューに指定した実行キューが存在しない場合
    エラーメッセージ：Specified execution queue doesn't exist in complex queue.
  - 同じ実行キュー名を複数指定した場合
    エラーメッセージ：Same execution queue name were specified doubly.

- コマンドを実行したユーザに管理者権限がない場合
  エラーメッセージ：Not permitted to modify attribute.

(5) コンプレックスキューの設定

コンプレックスキューの設定は、smgr(1M)コマンドの以下 3 つのサブコマンドによって行います。コンプレックスキューの設定には、操作員権限が必要です。

[同時実行リクエスト数制限]

```
# smgr -P m
Smgr: set complex_queue run_limit = run-limit complex-queue-name
```

- run-limit には、complex-queue-name に指定されたコンプレックスキューに設定する同時実行リクエスト数制限を指定します。
- run-limitに0が指定された場合は、無制限となります。
- 本制限の初期設定値は、無制限です。
- 本制限に指定できる最大値は、2^31までです。

[ユーザー律の同時実行リクエスト数制限]

```
# smgr -P m
Smgr: set complex_queue user_run_limit = run-limit complex-queue-name
```

- run-limit には、complex-queue-name に指定されたコンプレックスキューに設定するユーザー律の同時実行リクエスト数制限を指定します。
- run-limitに0が指定された場合は無制限となります。
- 本制限の初期設定値は無制限です。
- 本制限に指定できる最大値は2^31までです。

[ユーザー律の同時アサインリクエスト数制限]

```
# smgr -P m
Smgr: set complex_queue user_assign_limit = assign-limit complex-queue-name
```

- assign-limit には、complex-queue-name に指定されたコンプレックスキューに設定するユーザ一律の同時アサインリクエスト数制限を指定します。
- assign-limitに0が指定された場合は無制限となります。
- 本制限の初期設定値は、無制限です。
- 本制限に指定できる最大値は、2^31までです。

* 以下の場合エラーとなり、制限値は変更されません。

- 存在しないコンプレックスキューが指定された場合
  エラーメッセージ：Specified complex queue doesn't exist.
- 制限に指定された値が2^31を超える場合
  エラーメッセージ：Assign-limit out of bounds.
  Run-limit out of bounds.
- コマンドを実行したユーザに操作員権限がない場合
  エラーメッセージ：Not permitted to modify attribute.

(6)　コンプレックスキューの情報表示

コンプレックスキューの情報は **sstat**(1)コマンドの**-C** オプションで表示します。

```
# sstat -C

QueueName       Type      RL  URL  UAL   TOT   EXC   QUE   ASG   RUN   EXT   HLD   SUD
--------------- ------- ------------------ ----------------------------------------------
Complex_1       -       ULIM ULIM ULIM     0     0     0     0     0     0     0     0
[jmq0]          Urgent  ULIM ULIM ULIM     0     0     0     0     0     0     0     0
[jmq1]          Special ULIM ULIM ULIM     0     0     0     0     0     0     0     0
Complex_2       -       ULIM ULIM ULIM     0     0     0     0     0     0     0     0
[jmq2]          Normal  ULIM ULIM ULIM     0     0     0     0     0     0     0     0
[jmq4]          -       -    -    -        -     -     -     -     -     -     -     -
```

表示される項目は次のとおりです。

- コンプレックスキュー名
- 所属する実行キュー
- 同時実行リクエスト数制限
- ユーザ一律の同時実行リクエスト数制限

- ユーザ一律の同時アサインリクエスト数制限

* JobManipulator が管理していないキューについては、上記の表示例の jmq4 のようにキュー名だけ表示され、他の項目は"-"になります。

2.7.6　エスカレーション機能の設定

繰り上げ実行

リクエストが予定実行時間よりも早期に終了した場合、ノードリソースの空き領域が発生します。その空き領域を埋めるため、同じノードで後方にアサインされているリクエストが直ちに実行可能であれば、アサインします。 対象となるリクエストは以下の順で選択されます。

1. 繰り上げ実行時のスケジューリングプライオリティが高いリクエスト
2. リクエストの投入が最も早かったリクエスト

繰り上げ実行は JobManipulator の既定動作であり、後述のエスカレーション機能の設定の影響は受けません。

エスカレーション実行間隔の設定

JobManipulator は一定の間隔で定期的にリソースの空き状況をチェックし、リクエストを最適な位置へ移動させる機能をサポートしています。これをエスカレーション機能と呼びます。エスカレーション間隔はスケジューリングインターバル単位で **smgr**(1M)コマンドの **set escalation interval** サブコマンドにより設定可能します。

エスカレーション処理には以下の 2 種類があります。

- 前方エスカレーション
  ノード変更を伴わず、実行開始時刻のみ前方へ移動します。
- 横方向エスカレーション
  ノード変更を行うとともに実行開始時刻も前方へ移動します。

前方向(同一ノード前方)と横方向(別ノード前方)の両方に空きがある場合、前方向へのエスカレーションを優先します。

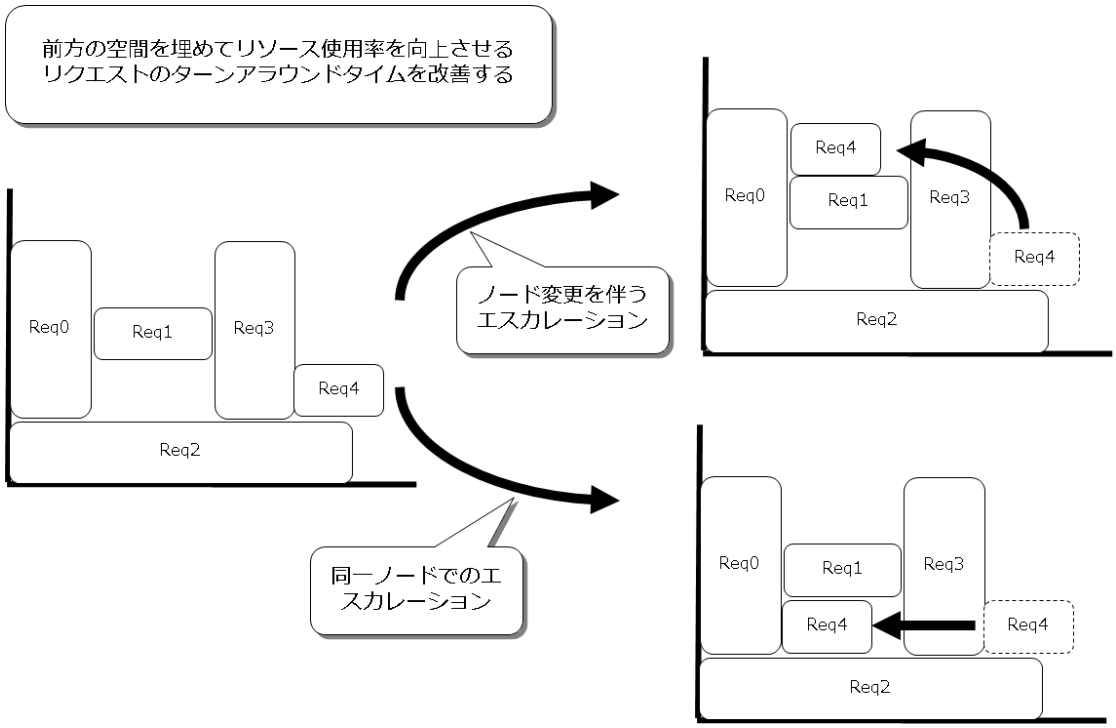


図 2-5：マップ前方にある空領域へのリクエスト移動

SUSPENDED 状態のリクエストに対して、ノード変更を伴うエスカレーションはできません。エスカレーションにより一部のノードが変更となる場合は、変更となったノードへの再ステージングのみ実施します。

エスカレーションについては **smgr**(1M)コマンドの **set use_escalation** サブコマンドにより以下の３つから選択し、設定します。

- Forward :前方エスカレーション
- All： 前方エスカレーション、または、横方向エスカレーション

デフォルトはエスカレーションを行いません(=**off**)。エスカレーション機能を off にした場合でも、リクエスト早期終了時にそのタイミングで後方のリクエストが移動可能であれば、繰り上げ実行は行われます。

	繰り上げ実行	**エスカレーション機能**
実行タイミング	リクエストのEXITING時	ユーザ定義の間隔で定期的に実行
対象	終了リクエストと同一ホストを利用予定のリクエストが対象	アサイン済みの全てのリクエストが対象
ON/OFF 設定	なし	**smgr** (1M)コマンドで設定

以下は、エスカレーション機能を off へ設定する例です。

```
# smgr -P m
Smgr: set use_escalation = off
```

エスカレーション対象リクエストの選択条件指定

横方向エスカレーションは、対象となったリクエストのバッチジョブを一度削除し再度ステージインを実施するため、負荷の高い処理となっています。

JobManipulator では、横方向エスカレーションが頻繁に発生することを防ぐために対象となるリクエストを選択する条件を設定することができます。設定可能な条件は以下のとおりであり、キュー毎に設定することができます。

- 横方向エスカレーション実行前後のリクエストの開始予定時刻の差が一定時間（横方向エスカレーション開始時間差制限値）以下のとき、横方向エスカレーションを禁止する
- リクエストの開始予定時刻と現在時刻の差が一定時間（横方向エスカレーション開始時間制限値）より短く、かつノード変更が発生するジョブ数が一定数（横方向エスカレーションジョブ数制限値）より多いとき、横方向エスカレーションを禁止する

エスカレーション対象リクエストの選択条件は、**smgr**(1M) コマンドの **set queue escalation_limit** サブコマンドで設定します。設定されたエスカレーション対象リクエストの選択条件は、**sstat**(1)コマンドの**-Q -f** オプションで確認できます。

- **Min Forward Time**：横方向エスカレーション開始時間差制限値
- **Max Side Escalation Jobs**：横方向エスカレーションジョブ数制限値
- **No Escalation Period**：横方向エスカレーション開始時間制限値

ステージイン時間の見積もり調整時間の設定

リクエストの横方向エスカレーション時は、当該リクエストのステージインにかかる時間（ステージイン時間）を考慮してエスカレーション可能かを決定します。

考慮されるステージイン時間は、当該リクエストのこれまでの最大ステージイン時間で見積もっていますが、実際のステージイン時間は運用状況によって多少のブレが発生します。このブレの発生によりエスカレーション先の予定実行開始時刻までにステージインが完了しない場合は、横方向エスカレーションが無効になります。この影響を軽減させるため、ステージイン時間の見積もりに一定時間を加える機能を提供します。

設定する値は、システム管理者が当該システムのリクエストのステージイン時間のブレの度合いを

考慮し適切に設定してください。

この時間は、**smgr**(1M)コマンドの **set stage-in_margin** サブコマンドで設定し、**sstat**(1)コマンドの**-S -f** オプションで確認します。

```
#sstat -S -f
JobManipulator Server Host: bsv.nec.co.jp
    JobManipulator Version   = R1.00
    JobManipulator Status    = Active
            :
    Keep Forward Schedule    = OS
    Stage-in Margin = {
      Additional Margin for Escalation = OS
      Stage-in Threshold = OS
      First Stage-in Time = OS
    }
:
```

2.7.7　ピックアップ時の追い越し制御

大規模なリクエストがいつまでも実行されない状態を避けるために、追い越し制御を行うことができます。

追い越し制御はキュー種別毎に **smgr**(1M)コマンドの **set use_overtake_priority** サブコマンドで設定可能です。設定が off の場合は、追い越し制御は行われません。

```
# smgr -P m
Smgr: set use_overtake_priority = on normal # 通常キューの追い越し制御を有効にします。
```

追い越し制御は、スケジューリングプライオリティの閾値（追い越し制御の閾値）を設定することで実現します。あるリクエストのスケジューリングプライオリティが追い越し制御の閾値を超えた場合、そのリクエストのスケジューリングが優先されます。そのリクエストの実行開始時間が決定するまで、他のスケジューリングプライオリティが追い越し制御の閾値よりも低いリクエストは実行開始予定時刻を決定することはできません。

追い越し制御の閾値は、**smgr**(1M) コマンドの **set overtake_priority** サブコマンドで設定します。値は、キュー種別毎に設定します。

以下は、通常キューの追い越しスケジューリングプライオリティ値を 100 に設定する例です。

```
# smgr -P m

Smgr: set overtake_priority = 100 normal
```

追い越し制御の設定は、より緊急度の高いキューに投入されたリクエストには影響しません。例えば、通常キューのリクエストのスケジューリングプライオリティ値が通常キューの追い越し制御の閾値を超えていても、特別キューや緊急キューに投入されたリクエストはこのリクエストを追い越すことが可能です。

追い越し制御はリクエストのスケジューリングプライオリティが追い越し制御の閾値を超えた時点から制御されます。閾値を超える前に追い越して実行開始予定時刻を決定したリクエストは、予定通り実行開始します。

小規模リクエストの追い越し割り当て機能

追い越しを禁止にするスケジューリングプライオリティ値を有効にした場合に、大規模リクエストがそのプライオリティ値を超えた場合は、そのあとに投入されたリクエストが一切アサインされない状況になります。スケジューラマップ上の空き領域で実行可能な小規模リクエストがある場合でも、アサインされずに大規模リクエストのアサインを待つことになります。そこで、スケジューラマップの先頭からスケジューラマップの最後尾までの間にある空き領域に収まるリクエストをアサイン可能とする、小規模リクエストの追い越し割り当て機能を提供します。

この機能により、小規模リクエストが大規模リクエストのアサインを待つことなく、先に隙間の時間で実行できるようになり、システム全体の稼働率の向上やジョブの TAT 時間の改善につながります。

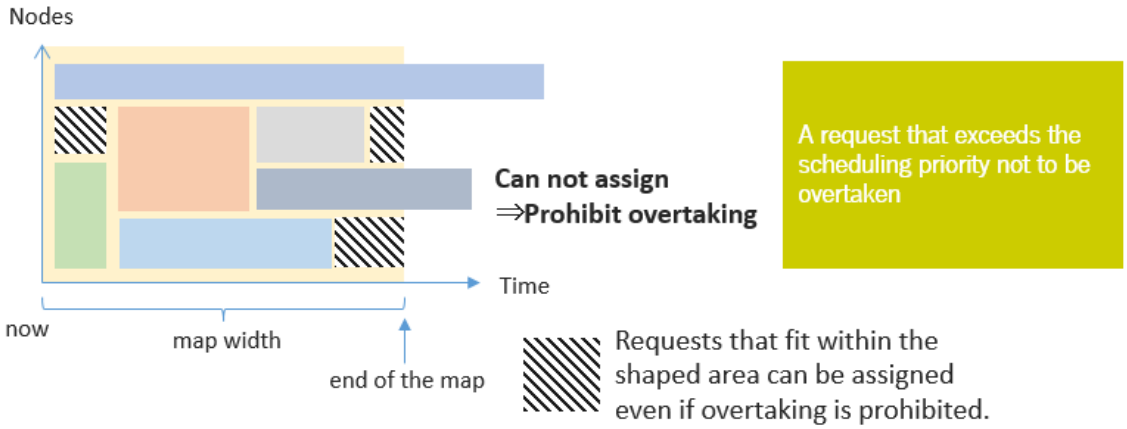


図 2-6：小規模リクエストの追い越し割り当て

この機能は、**smgr**(1M)コマンドの **set overtake allow_small_request** サブコマンドで ON/OFF を設定します。

```
# smgr -P m
Smgr: set overtake allow_small_request = on
```

この機能を ON にした場合は、図 2-6 の斜線領域に収まる小規模リクエストは、追い越してアサインします。

- 大規模リクエストがアサインされた後、前方ジョブの終了によりエスカレーションが発生した場合は、スケジューラマップの隙間に先にアサインした小規模リクエストがエスカレーションの阻害要因になる可能性があります。

2.7.8　アサインポリシーの設定

(1)　CPU台数集中/分散ポリシー

JobManipulator は使用可能 CPU 台数制限まで詰めてジョブをアサインする CPU 台数集中ポリシーおよび、ノードの CPU 使用台数が均一となるようにジョブをアサインするリソース分散ポリシーの 2 種類のリクエストのアサインポリシーをサポートしています。

CPU 台数集中ポリシーでは、空きノードをできるだけ確保し、大規模リクエストを実行しやすくなります。リソース分散ポリシーではノードの負荷分散が可能となります。

ＣＰＵ台数リソース集中アサイン（CPU_concentration）

アサイン時点で使用可能 CPU 台数制限まで詰めて選択します。使用可能 CPU 台数制限を超えるまでは他のノードにアサインしません。

リソース分散アサイン（resource_balance）

アサイン時点で CPU 使用量の一番少ないノードを選択します。

スケジューラ単位のリクエストアサインポリシーの設定は、**smgr**(1M)コマンドの **set assign_policy** サブコマンドで設定します。

デフォルトは **CPU_concentration** です。

```
# smgr -P m
Smgr: set assign_policy=CPU_concentration     # スケジューラ単位のリクエストアサインポリシーをCPU台数リソース集中アサインに設定します。
```

設定には操作員以上の権限が必要です。

リクエストのアサインポリシーは、キュー単位でも設定できます。キュー単位のリクエストアサインポリシーの設定は、**smgr**(1M)コマンドの **set queue assign_policy** サブコマンドで設定します。

```
# smgr -P m
Smgr: set queue assign_policy=CPU_concentration bq1

# bq1 キューのリクエストアサインポリシーをCPU台数リソース集中アサインに設定します。
```

設定には操作員以上の権限が必要です。 キュー単位のアサインポリシーはデフォルトでは設定されていません。その場合、スケジューラ単位のアサインポリシーを適用します。

スケジューラ単位のアサインポリシーとキュー単位のアサインポリシーが異なる場合、キュー単位のアサインポリシーを適用します。

1 ジョブがノードを占有して実行する運用では、どちらのポリシーに設定してもアサインの結果は同じです。その場合、スケジューリング性能が相対的に高い「CPU 台数集中アサイン」に設定することを推奨します。

- キュー単位のリクエストアサインポリシーの設定は、JobManipulatorで管理されているキューに対して可能です。
- リクエストアサインポリシーの設定を変更した場合、再スケジューリングは行われず未アサインのリクエストに対して変更後のリクエストアサインポリシーが有効となります。

CPU 台数リソース集中アサインを設定した場合は、GPU を指定したリクエストに対しては、GPU 集中のスケジューリングを行います。つまり、このようなリクエストを残りの使用可能 GPU 数の少ないノードにアサインします。残りの使用可能 GPU 数が同じ場合は、CPU 台数リソース集中アサインポリシーに従ってアサインします。

(2) リクエストアサインノードの順序設定

JobManipulator ではキュー毎にジョブサーバ単位の優先順位（JSV アサインプライオリティ）を設定することで、リクエストがアサインされるジョブサーバの順序を JSV アサインプライオリティ順に行う機能を提供しています。

JSV アサインプライオリティは、キュー毎にジョブサーバ単位に **smgr**(1M)コマンドの **set queue jsv_assign_priority** サブコマンドで設定します。

```
# smgr -P m
Smgr: set queue jsv_assign_priority = 100 job_server_id = 1 bq1


# キューbq1に対して、ジョブサーバID 1のJSVアサインプライオリティを100に設定します。
```

JobManipulator にバインドしているキューのみに対して設定可能です。ATTACH された実行ホスト上のジョブサーバであれば、バインド状態に関わらず設定可能です。設定には操作員以上の権限が必要です。初期設定値は 0 です。

- 本機能で指定したJSVアサインプライオリティはジョブサーバ選択時にジョブコンディション指定に次いで優先的に使用されます。したがって異なるJSVアサインプライオリティを設定したジョブサーバ間では、下位のアサインポリシーがジョブサーバ選択時に使用されなくなります。
- set queue jsv_assign_priorityサブコマンドにてノードグループを指定することにより、ノードグループに含まれているジョブサーバのJSVアサインプライオリティを一括設定できます。但し、ノードグループに含まれるジョブサーバが増えたり減ったりした場合は、変化が生じたジョブサーバのJSVアサインプライオリティは追随しません。

JobManipulator 全体における JSV アサインプライオリティは、 **sstat**(1)コマンドの **-J** オプションで表示します。

```
# sstat -J
 JSVNO Queue    Priority
 ----- -------- -----------
     0 bq1              200
     1 bq1              100
     1 bq2              100
     2 bq2              200


#bq1とbq2がJSV 1を共有する場合、JSV 1のJSVアサインプライオリティを低く設定します。
```

キュー単位の JSV アサインプライオリティは、**sstat**(1)コマンドの**-Q -f -j** オプションで表示します。キューにバインドしているジョブサーバの JSV アサインプライオリティのみが表示されます。

バインドしていないジョブサーバの JSV アサインプライオリティを表示したい場合は、**sstat**(1)コマンドの**-Q -f -j** オプションに**-a** オプションを付与することで参照可能です。

```
# sstat -Q -f -j bq1
Execution Queue: bq1
     ...中略
     JSV Assign Priority{
         JSV    0 =         200
         JSV    1 =         100
     }
Request Statistical information:
(以下省略)
```

(3)　アサインポリシーの優先度設定と無効化設定

下記のアサインポリシーに関して、優先度と無効化を設定可能です。

- ネットワークトポロジーを考慮したアサイン
  （「3.1.7 (2)ネットワークトポロジーを考慮したアサイン」参照）
- 予定実行開始時刻がクリアされたステージング中ジョブがあるノード以外のノードの優先アサイン
  （「3.1.7 (3)ステージング中ジョブがあるノード以外のノードの優先アサイン」参照）

上記アサインポリシーの優先度設定と無効化設定は、スケジューラ単位に **smgr**(1M)コマンドの **set assign_policy_priority** サブコマンドで設定します。

```
# smgr -P m
Smgr: set assign_policy_priority = <priority> assign_policy = <assign_policy>
```

- assign_policyには、以下のアサインポリシーを設定可能です。
  - network_topology：ネットワークトポロジーを考慮したアサイン
  - staging_job：予定実行開始時刻がクリアされたステージング中ジョブがあるノード以外のノードの優先アサイン
- priorityには、上記アサインポリシーの優先度low /high、あるいは、無効化disableを設定します。
  disable と設定した場合は、本ポリシーは無効となります。運用中に変更した場合は、変更したアサインポリシーをそれ以降のスケジューリングに適用します。

初期設定値は、以下の通りです。

network_topology：high

staging_job ：low

- 操作員権限以上の権限が必要です。

low/high の設定基準、および、設定した時のアサインポリシーの優先順位については、「3.1.7 (1)アサインポリシーの優先順位」を参照してください。

上記アサインポリシーの優先度設定と無効化設定は、**sstat**(1)コマンドの-S -f オプションで表示可能です。

```
# sstat -S -f
JobManipulator Server Host: bsv.nec.co.jp
    JobManipulator Version   = R1.00
JobManipulator Status    = Active
            :
Request Assign Policy    = CPU concentration
Assign Policy Priority = {
Network Topology = high
  Staging Job      = low
}
Global Run Limit         = 10
:
```

2.7.9　再スケジューリングの待ち時間設定

再スケジューリングの待ち時間を設定することにより、リクエストアサイン後のステージイン処理が失敗した場合、または、PRE-RUNNING 処理（実行開始処理）が失敗した場合に、リクエストの再スケジューリングを一定時間待ちます。

本機能により、ステージング/PRE-RUNNING 処理失敗と即座の再スケジューリングの繰り返しを防止できます。

再スケジューリングの待ち時間は、**smgr**(1M)コマンドの **set queue retry_time** サブコマンドにより、キュー単位に設定します。

```
# smgr -P o
Smgr: set queue retry_time staging = 600 pre-running = 300 bq1
```

```
# bq1 キューに対して、ステージイン処理失敗時の再スケジューリングの待ち時間を600秒に、PRE-RUNNING処理失敗時の再スケジューリングの待ち時間を300秒に設定します。
```

設定には操作員以上の権限が必要です。 デフォルトは 0 秒です。また、ステージイン/PRE-RUNNING 処理失敗により、再スケジューリング待ちになったリクエストの再スケジューリング待ちを解除する機能を提供しています。本機能によりリクエストを再スケジューリングの対象に戻すことができます。

再スケジューリング待ちの解除は、**smgr**(1M)コマンドの **stop waiting_retry** サブコマンドにより実施します。

```
# smgr -P o
Smgr: stop waiting_retry request = 123.bsv.nec.co.jp  # リクエスト 123.bsv.nec.co.jpの再スケジューリング待ちを解除します。
```

設定には操作員以上の権限が必要です。

JobManipulator のスケジューリングは、キューを実行可能状態(ACTIVE)にすることにより開始します。しかしキューへの実行可能状態化の設定順によっては、キュープライオリティによるキュー間の優先度付け等のキュー間でのプライオリティ調整を意図したように行うことができません。
このような場合、本機能でスケジューリングを停止したうえで、全キューを実行可能状態にした後、本機能を用いてスケジューリングを一斉に開始することで、意図したプライオリティ調整を行うことができるようになります。

2.7.10 スケジューリングの開始/停止設定

JobManipulator のスケジューリングの開始/停止が設定可能です。

設定は **smgr**(1M) コマンドの start scheduling/stop scheduling サブコマンドにより行います。start scheduling サブコマンドを実行するとスケジューリングを開始します。

stop scheduling サブコマンドを実行するとスケジューリングを停止します。インストール時の設定は stop scheduling です。

```
# smgr -P m
 Smgr: start scheduling
 Smgr: stop scheduling
```

設定には操作員以上の権限が必要です。

JobManipulator のスケジューリングは、キューを実行可能状態(ACTIVE)にすることにより開始します。しかしキューへの実行可能状態化の設定順によっては、キュープライオリティによるキュー間の優先度付け等のキュー間でのプライオリティ調整を意図したように行うことができません。
このような場合、本機能でスケジューリングを停止したうえで、全キューを実行可能状態にした後、本機能を用いてスケジューリングを一斉に開始することで、意図したプライオリティ調整を行うことができるようになります。

第3章　　運用管理

3.1　スケジューリング基本機能

本節では、JobManipulator の基本的な動作について説明します。

3.1.1　スケジューラマップ

JobManipulator は、スケジューラマップを使用し、実行開始時刻とリソースの事前割り当てを行います。そのため、ジョブに対する計画的な計算リソースの分配を実現することができます。

スケジューラマップとは、ジョブサーバ毎に計算資源を時空間分割したもの（セル）の集合体です。セルはスケジューラマップ幅(マップ幅)の最小単位となります。

セルの大きさ（セルサイズ）はスケジューリングインターバルとも呼びます。JobManipulator はスケジューリングインターバルの間隔でスケジューリングを行います。

スケジューリングインターバルの初期値は 60 秒です。マップ幅の初期値は 1 日(86400 秒)です。

JobManipulator はマップにジョブのアサイン処理を行います。セルを未来に渡っていくつまで管理するかは、マップ幅の設定に委ねられます。たとえば、マップ幅が 1 日(86400 秒)でスケジューリングインターバルが 1 分(60 秒)であった場合、ジョブサーバ毎のセル数は 1440 (= 86400/60)　となります。

以下はスケジューラマップの簡単なイメージ図となります。

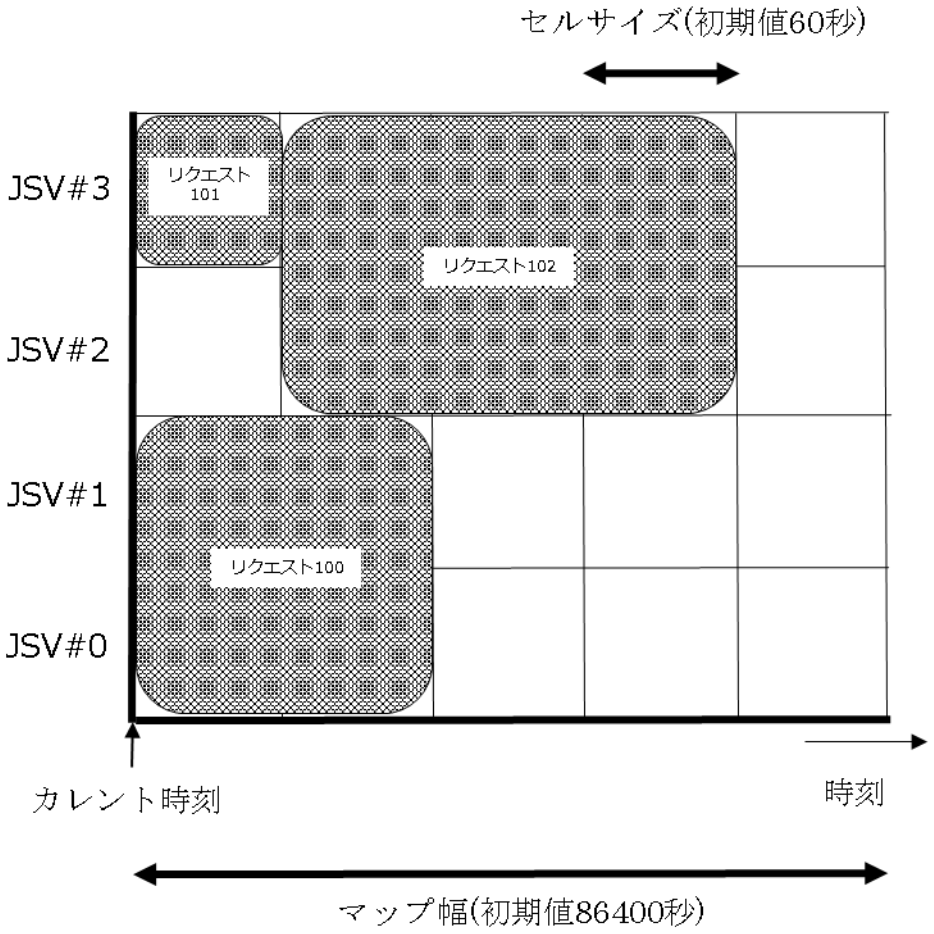


図 3-1：スケジューラマップ

上図では、Request100 と Request101 が実行中です。また、Request101 の実行終了後、Request102 が実行開始されます。

マップ幅を長く設定することでバックフィル特性の強いスケジューリングを実現できます。マップ幅を短く設定することでフェアシェア特性の強いスケジューリングを実現できます。マップ幅を十分（リクエストの宣言経過時間より）短く設定すれば、カレントスケジューリングを実現できます。

(1) マップ幅の設定

スケジューラマップのマップ幅を以下 2 つの単位で設定することができます。

A. スケジューラ単位にマップ幅を設定
B. キュー単位にマップ幅を設定

* 詳細については以下を参照してください。

A. スケジューラ単位

設定方法：

スケジューリングインターバル、およびマップ幅の設定は **smgr**(1M)コマンドの **set mapsize** サブコマンドにより行います。マップ幅に設定できる最小値はスケジューリングインターバルです。

スケジューリングインターバルを変更した場合、それまでのスケジューラマップの情報は再構築され、マップ上にアサインされていたリクエストの開始予定時刻はいったんマップ上からキャンセルされます。

スケジューリングインターバルが同じでマップ幅が大きくなった場合、マップ幅が大きくなった分だけ多くリクエストをアサインすることができるようになります。

またマップ幅が小さくなった場合、マップ幅に入りきらなくなったリクエストは再スケジューリングの対象となりマップ上からキャンセルされます。

マップ幅はスケジューリングインターバルより長く設定する必要があります。

マップ幅とピックアップの関係

以下の図はマップ幅とピックアップのイメージ図です。アサインプール内のリクエストは、スケジューリングプライオリティ （計算された優先度）順に整列しています。

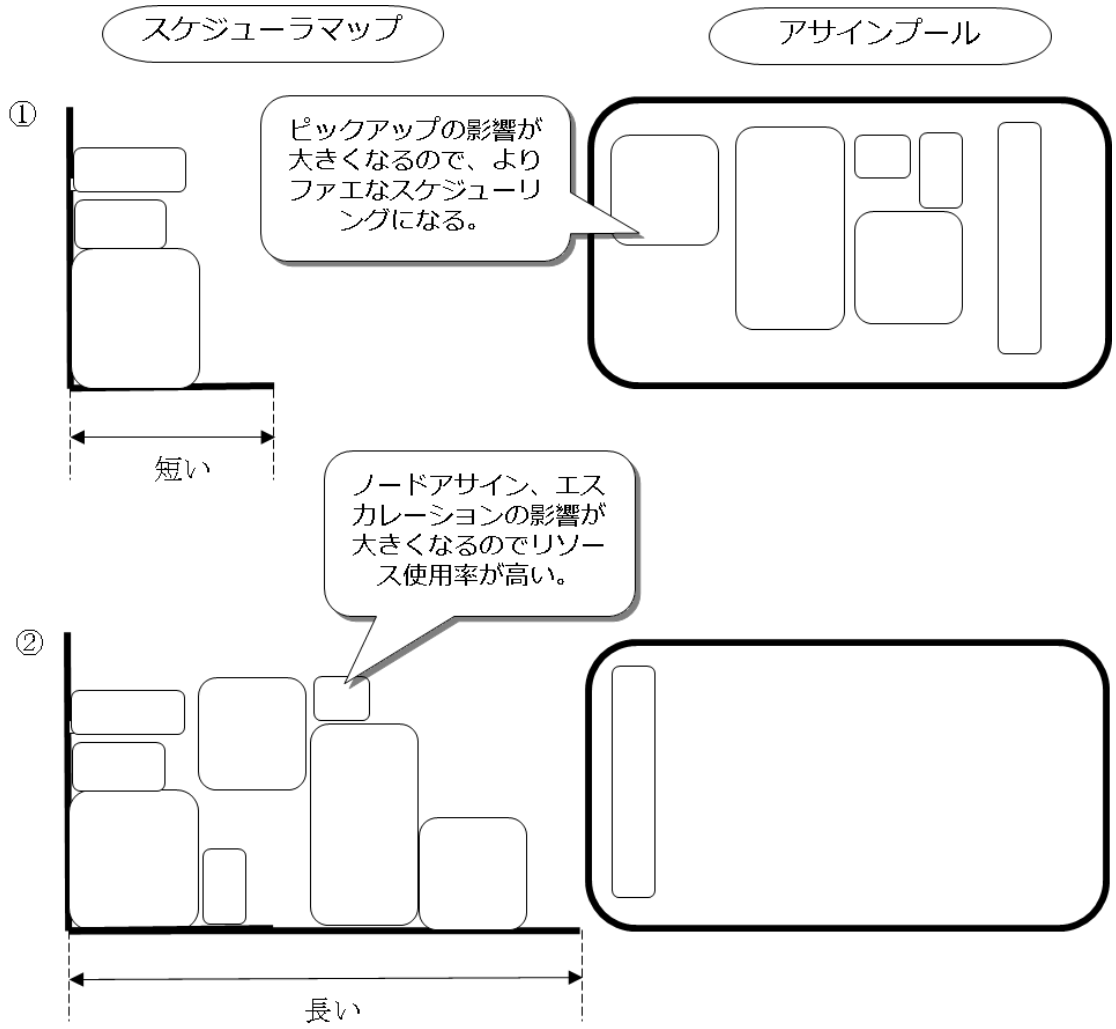


図 3-2：マップ幅とピックアップ

アサインプール：マップ上にアサインされていないリクエスト(まだ実行開始予定時間が決まっていないリクエスト)の集団。

ピックアップ：マップに貼り付ける(アサインする)ために、リクエストを選択すること。

JobManipulator のマップ幅によるスケジューリング特性を変更することができます。

① マップ幅が短い：フェアシェア特性の強いスケジューリング
② マップ幅が長い：バックフィル特性の強いスケジューリング（リソース稼働率の向上）

B. キュー単位

マップ幅はキュー毎に設定することができます。 キュー毎に設定することにより、キュー単位で適切なスケジューリング特性の運用(バックフィルまたはフェアシェア)ができるようになります。スケジューラ単位での設定よりも綿密なスケジューリングが可能になります。

以下の図はキュー毎にマップ幅を指定した場合のイメージ図です。

* 1 つの JobManipulator でキュー毎にスケジューリングの特性を設定できます。
下図では次のような運用を行っています。

- キューAには小規模なジョブが投入されるため、マップ幅を短くしてフェアシェアを効かせています。
- キューBには大規模なジョブが投入されるため、マップ幅を長くして資源使用率を上げる運用にしています。

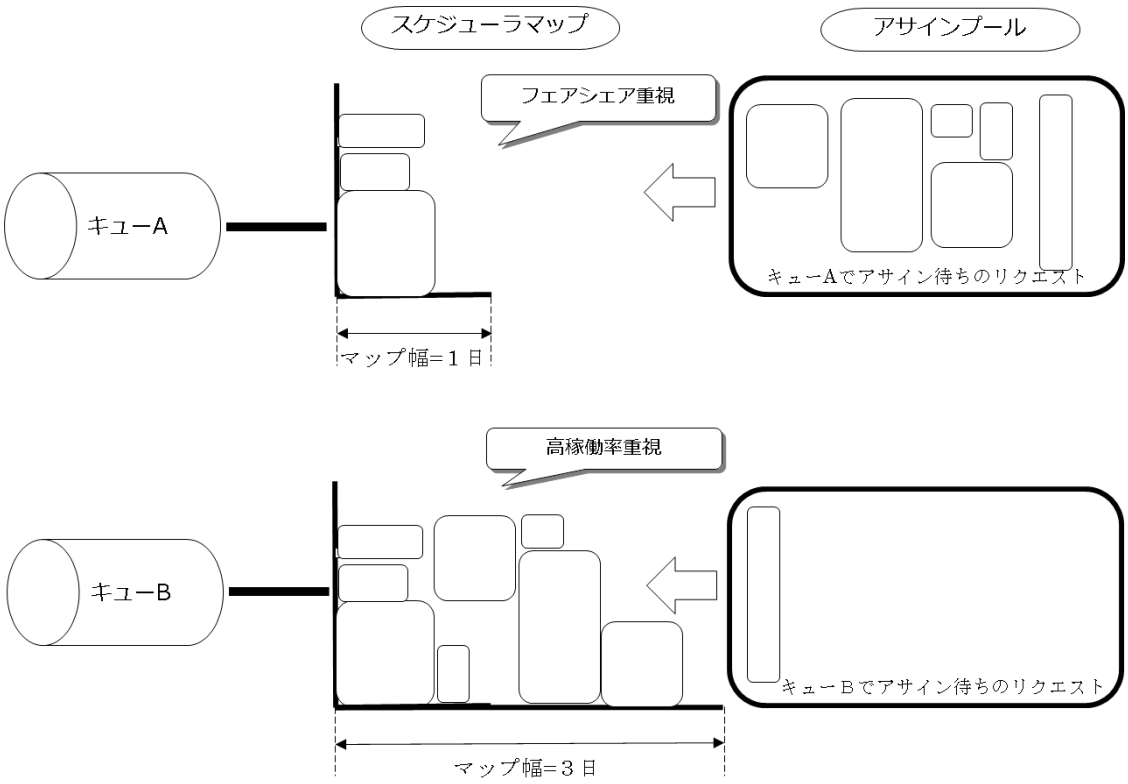


図 3-3：キュー単位のマップ幅の設定

* キューA ではマップ幅を短くし、フェアシェア特性の強いスケジューリングを行います。キューB ではマップ幅を長くし、バックフィル特性の強いスケジューリングを行います。

スケジューリングインターバルはスケジューラ単位に一つだけ設定できます。キュー単位には設定できません。

設定方法：

キュー毎のマップ幅の設定は **smgr**(1M)コマンドの **set queue mapsize sched_time** サブコマンドにより行います。スケジューリングインターバルはキュー単位には設定できません。スケジューラ単位で設定したものが使用されます。

```
#smgr -P m
Smgr: set queue mapsize sched_time = sched_time queue-name

[例] # 以下は、"execqueue1"キューのマップ幅を10000秒に設定しています。
Smgr: set queue mapsize sched_time = 10000 execqueue1
Set queue Mapsize.
```

- sched_timeにはqueue-nameで指定されたキューに設定するマップ幅を指定します。
- マップ幅は秒単位で指定します。設定できる最小値はスケジューリングインターバルです。
- sched_timeに、スケジューラで設定しているマップ幅を超えて指定した場合はエラーとなります。
- キュー毎に設定できるマップ幅の上限はスケジューラで設定しているマップ幅になります。
- マップ幅が設定されていないキューのマップ幅はスケジューラで設定しているマップ幅となります。
- マップ幅を変更した場合、実行中以外のジョブは全て再アサインされます。
  - マップ幅をスケジューラで設定しているマップ幅へ変更したキュー名はログファイル(デフォルトファイル名：/var/opt/nec/nqsv/nqs_jmd_<scheduler_id>.log)に出力されます。
- キュー単位でのマップ幅の設定は、操作員以上の権限が必要です。

スケジューラのマップ幅をキュー毎のマップ幅より小さく指定した場合、該当するキューのマップ幅がスケジューラで設定しているマップ幅に変更されます。

メッセージ：Some queues were changed to the mapsize of the system.

マップ幅が変更されたキュー名は、ログファイル(デフォルトファイル：/var/opt/nec/nqsv/nqs_jmd_<scheduler_id>.log)に出力されます。

---

* 以下(4 つ)の場合エラーとなり、マップ幅は変更されません。

1. 指定されたキュー毎のマップ幅がシステムのマップ幅より大きい場合
   エラーメッセージ：Mapsize too large.(Range of value = xx - xx)
2. JobManipulator で管理されていないキューが指定された場合
   エラーメッセージ：No such queue. (name: <queue-name>).
3. コマンドを実行したユーザに操作員権限以上の権限がない場合
   エラーメッセージ：Not permitted to modify attribute..
4. 指定されたキュー毎のマップ幅がスケジューリングインターバルより小さい場合
   エラーメッセージ：Mapsize too small.(Range of value = xx - xx).

---

2 つ以上のキューが同一ホストを共有する場合、以下の事項に留意してください。
同一ホスト、同一 RSG を利用する 2 つ以上のキューのマップ幅を変更した場合、マップ幅の短いキューにリクエストがアサインされにくい現象が発生します。 この現象を回避するため、以下のような運用を推奨します。

- キュー毎にマップ幅を変更する運用下では、それぞれのキューが異なったホストを管理している状況での運用を推奨します。
- キューが同一ホストを管理する場合は、リソースの競合が起こらないようホストのリソースを RSGによって分割管理することを推奨します。

RSG によって管理しない場合、JobManipulator の CPU リミットレート、メモリリミットレートの設定により、リソース競合を避けることも可能です。

---

(2) マップ幅とスケジューリングインターバルの表示

A.スケジューラ単位

スケジューラごとのマップ幅、およびスケジューリングインターバルは、sstat(1)コマンドの-S,-f オプションで表示します。

```
#sstat -S -f
JobManipulator Server Host: bsv.nec.co.jp
    JobManipulator Version   = R1.00
    JobManipulator Status    = Active
    Scheduler ID             = 1
    Schedule Interval        = 10S
    Schedule Time            = 86400S
:
```

B.キュー単位

キュー毎のマップ幅は、sstat(1)コマンドの-Q,-f オプションで表示します。

```
#sstat -Q -f
Execution Queue: jmq0
    Queue Type           = Normal
    Schedule Time        =  86400S
:
```

(3) 設定に関する留意事項

JobManipulator はスケジューリングインターバルの間隔でスケジューリングを行います。リクエストが大量に存在する場合に、スケジューリングインターバル内でスケジューリングが終了せず、処理を打ち切ることがあります。

運用するサイトのリクエスト量に応じて、スケジューラマップやスケジューリングインターバルを調整することで、開始予定時刻を決めることができないリクエストが滞留しないように調整してください。

3.1.2　リクエストの即時実行

JobManipulator は、リクエストが投入されるとスケジューリングを行い、実行開始の準備が整うとリクエストを実行開始させます。開始予定時刻は、**3.1.1 スケジューラマップ** で述べたとおり、JobManipulator はスケジューリングインターバルの幅を最小単位として決定しますが、リソースに

空きがあればスケジューリングインターバルを待たずに開始することが可能です。

本機能は NQSV/JobManipulator R1.06 にて強化されました。以前のように、スケジューリングインターバルの間隔でリクエストのスケジューリング、および実行開始を実施するには、システム管理者が **smgr**(1M)コマンドの **set realtime_scheduling** サブコマンドで **off** を指定してください。

リクエスト即時スケジューリングモード

スケジューリングインターバルでは、JSV の LINKDOWN や LINKUP 等のイベントを処理する専用の時間帯を設けています。この時間帯は“イベント時間”と呼びます。イベント時間ではイベント処理を集中的に行うため、リクエストの即時スケジューリングを行いません。そのため、イベント時間内に投入されたリクエストは、イベント処理がない場合でも次のインターバル時刻後にスケジューリングします。システム管理者が **smgr**(1M)コマンドの **set realtime_scheduling mode** サブコマンドでリクエスト即時スケジューリングモードを **always** に変更すると、イベント時間内でもリクエストを即時にスケジューリングして実行します。既定値では、イベント時間にリクエスト即時スケジューリングを行いません。**smgr**(1M)コマンドの **set realtime_scheduling mode** サブコマンドで、リクエスト即時スケジューリングモードを default に指定すると、設定は既定値に戻ります。

リクエスト即時スケジューリングモードの設定値は、sstat -Sf の“Realtime Scheduling Mode”欄で確認できます。

3.1.3　過去の使用実績の蓄積と減衰

(1)　過去の使用実績の蓄積

JobManipulator は、各ユーザで実際に使用したシステムリソースを各リクエスト単位に計測し、その累計値をユーザ単位に算出して蓄積します。使用実績を構成するシステムリソースは、以下のとおりです。

CPU台数	CPU台数（ユーザ宣言値）　× Time（経過）	リクエスト単位に計測
経過時間	経過時間（実測値）	リクエスト単位に計測
メモリ使用量	使用メモリ量（実測値）　× Time（経過）	リクエスト単位に計測
リクエスト プライオリティ	リクエストプライオリティ（ユーザ宣言値）　× Time（経過）	リクエスト単位に計測
VE台数	VE台数（ユーザ宣言値）× Time（経過）	リクエスト単位に計測

使用実績として、現時点までの使用実績と合わせて減衰を行った値が蓄積されます。 各リクエストが終了する度にユーザ毎に蓄積した使用実績値を減衰させながら、使用実績値を蓄積します。

半減期（減衰期間）は、**smgr**(1M)コマンドの **set half_reduce_period** サブコマンドで設定するこ

とが可能です。

JobManipulator は、各リクエストが終了する度にユーザ毎に蓄積した使用実績値を減衰させながら、使用実績値を蓄積します。

新使用実績値=使用実績値（蓄積値）＊ 0.5 ^（(現時刻 － 前回時刻）/ 半減期間）+ 今回計測した使用実績値

(2)　使用実績値の減衰

JobManipulator は、各リクエストが終了する度にユーザ毎に蓄積した使用実績値を減衰させながら、使用実績値を蓄積します。

新使用実績値=使用実績値（蓄積値）＊ 0.5 ^（(現時刻 － 前回時刻）/ 半減期間 ）+ 今回計測した使用実績値

(3)　使用実績値によるスケジューリングプライオリティへの反映

使用実績値の要素としては、ＣＰＵ台数、経過時間等を使用し、各要素にはそれぞれの重み付けが行われ、システムで設定したスケールで同一の意味合いに加工し、相対的に比較します。

これらの重み係数の設定は、**smgr**(1M)コマンドの **set** サブコマンドで行います。使用できるパラメータは以下のとおりです。

パラメータ名	内容
pastusage_weight_request_priority	使用実績のリクエストプライオリティに対する重み付け
pastusage_weight_cpu_number	使用実績のCPU数に対する重み付け
pastusage_weight_elapse_time	使用実績の経過時間に対する重み付け
pastusage_weight_memory_size	使用実績のメモリサイズに対する重み付け
pastusage_weight_ve_number	使用実績のVE数に対する重み付け

スケジューリングプライオリティの計算において正規化した使用実績値が使用されます。各要素の正規化式は以下の通りです。

a)　CPU 台数

標準CPU台数のCPUを無制限に使用した（と仮定した）場合を1とします。 正規化式

CPU使用実績値（蓄積値）　/（ 標準CPU台数 / loge2 * 半減期 ）

b)　経過時間

標準ＣＰＵ台数を無制限に使用した（と仮定した）場合を1とします。

正規化式

経過時間実績値（蓄積値）　/（ 標準CPU台数 / loge2 * 半減期 ）

c)　メモリ使用量

標準全実装メモリを無制限に使用した（と仮定した）場合を1とします。

正規化式

メモリ使用実績値（蓄積値）　/（ 標準全メモリサイズ / loge2 * 半減期 ）

d)　リクエストプライオリティ

リクエストプライオリティとして1023を指定したリクエストが無制限に動き続けた（と仮定した）場合を1とします。

正規化式

リクエストプライオリティ使用実績値（蓄積値）　/（ 1023 / loge2 * 半減期 ）

e)　VE 台数

標準VE台数のVEを無制限に使用した（と仮定した）場合を1とします。

正規化式

VE使用実績値（蓄積値）　/（ 標準VE台数 / loge2 * 半減期 ）

(4)　使用実績値の表示

使用実績値は **sushare**(1)コマンドの**-S**（大文字）オプションで表示されます。使用実績値はグループ毎に階層表示され、ユーザ毎の使用実績値とグループ毎の合計使用実績値が出力されます。

グループ毎の合計使用実績値の表示データには、グループ名の先頭に*****が付加されます。表示される項目は次の通りです。

項目名	内容
Group Name	グループ名を表示します。グループの使用実績値を表示する時、最初に出力されます。
User	ユーザ名、またはグループ名を表示します。グループ名の場合は先頭に*が付加されます。
Acctcode	ユーザのアカウントコードを表示します。アカウントコードがない場合noneと表示されます。グループ名の場合は常にnoneと表示されます。
Share	ユーザ、またはグループ毎のシェア配分比を表示します。シェア配分比についてはユーザシェア値を参照してください。
PU_cpunum	ユーザ、またはグループ毎のCPU使用実績値と、システム全体の使用実績率が表示されます。
PU_memsz	ユーザ、またはグループ毎のメモリ使用実績値と、システム全体の使用実績率が表示されます。
PU_elapstim	ユーザ、またはグループ毎の経過時間使用実績値と、システム全体の使用実績率が表示されます。
PU_reqpri	ユーザ、またはグループ毎のリクエストプライオリティ使用実績値と、システム全体の使用実績率が表示されます。
PU_venum	ユーザ、またはグループ毎のVE使用実績値と、システム全体の使用実績率が表示されます。

以下は、実行例です。

```
[Group Name : TOP_GROUP]      ←#グループ名
User     Acctcode Share  PU_cpunum (%)       PU_venum (%)          PU_memsz (%)       PU_elapstim (%)       PU_reqpri (%)
-------------------------------------------------------------------------------------------------------------------------------
*nec       none     0.333     4.190M ( 50.002)    4.190M ( 50.002)    3.996M ( 50.002)   1163:58:00 ( 50.002)    4.190M ( 50.002)
↑#グループ毎の合計使用実績値
*nqs       none     0.667     4.190M ( 49.998)    4.190M ( 50.002)    3.996M ( 49.998)   1163:53:44 ( 49.998)    4.190M ( 49.998)
↑#グループ毎の合計使用実績値

[Group Name : nec]
 User     Acctcode Share  PU_cpunum (%)       PU_venum (%)          PU_memsz (%)       PU_elapstim (%)       PU_reqpri (%)
```

```
--------------------------------------------------------------------------------------------------------------------------------
necusr1  none     0.167     2.095M ( 25.001)   2.095M ( 25.001)     1.998M ( 25.001)    581:59:01 ( 25.001)     2.095M ( 25.001)
↑#ユーザ毎の使用実績値
necusr2  none     0.167     2.095M ( 25.001)   2.095M ( 25.001)     1.998M ( 25.001)    581:58:58 ( 25.001)     2.095M ( 25.001)


[Group Name : nqs]
User     Acctcode Share  PU_cpunum (%)       PU_venum (%)          PU_memsz (%)        PU_elapstim (%)        PU_reqpri (%)
--------------------------------------------------------------------------------------------------------------------------------
nqsusr1  none     0.167     1.048M ( 12.500)   1.048M ( 12.500)  1022.954K ( 12.500)    290:58:24 ( 12.500)     1.048M ( 12.500)
nqsusr2  none     0.167     1.048M ( 12.500)   1.048M ( 12.500)  1022.955K ( 12.500)    290:58:25 ( 12.500)     1.048M ( 12.500)
nqsusr3  none     0.167     1.048M ( 12.500)   1.048M ( 12.500)  1022.956K ( 12.500)    290:58:26 ( 12.500)     1.048M ( 12.500)
nqsusr4  none     0.167     1.048M ( 12.500)   1.048M ( 12.500)  1022.956K ( 12.500)    290:58:26 ( 12.500)     1.048M ( 12.500)
```

3.1.4　スケジューリングプライオリティ

(1)　スケジューリングプライオリティとは

スケジューリングプライオリティは、実行ホストの割り当て時の順番（リクエストのピックアップ）やエスカレーションの実行順に使用されます。

スケジューリングプライオリティに使用される要素は以下に示すとおりとなります。

なお、リクエストの選択の順番は、キュープライオリティが大きい実行キューに投入されたリクエストから選択され、複数リクエストが存在する場合は、各リクエストのスケジューリングプライオリティが大きい順となります。

スケジューリングプライオリティは、以下の要素を元に算出します。

- ユーザシェア値
- 使用実績値
- ユーザランク
- リクエストプライオリティ
- 実行リクエストの要求リソース量
- キューに投入してからの実行待ち時間
- アサイン可能になってからの実行待ち時間

(2)　スケジューリングプライオリティ計算式

スケジューリングプライオリティ

```
=ユーザシェア値                          × 重み係数(ユーザシェア)
+使用実績値（総合）
+ユーザランク(正規化)                    × 重み係数(ユーザランク)
+リクエストプライオリティ(正規化)        × 重み係数(プライオリティ)
+宣言CPU数(正規化)                       × 重み係数(宣言CPU数)
+宣言経過時間(正規化)                    × 重み係数(宣言経過時間)
+宣言メモリサイズ(正規化)                × 重み係数(宣言メモリサイズ)
+ジョブ数(正規化)                        × 重み係数(ジョブ数)
+宣言VE数(正規化 )                       × 重み係数(宣言VE数)
+キューに投入してからの実行待ち時間      × 重み係数(投入からの実行待ち)
+アサイン可能になってからの実行待ち時間× 重み係数(アサイン可能からの実行待ち)
+再開待ち時間                            × 重み係数(再開待ち)
（+ 緊急リクエストによってSUSPEND されたリクエストに対するベースアップ）
（+ 再スケジューリングリクエストに対するベースアップ）
（+ ユーザ定義ベースアップ）
```

以下に個々の項目の詳細を記述します。

ユーザシェア値

ユーザシェア値は、シェア配分比が設定されたファイル(シェア配分比設定ファイル)を使用してスケジューラが算出します。シェア配分比ファイルは **sushare**(1)コマンドで読み込まれます。**sushare**(1)コマンド使用時にファイル指定がない場合、/etc/opt/nec/nqsv/jm_sharedb.conf が対象となります。

シェア配分比設定ファイルの設定形式は、次のとおりです。

```
TOP_GROUP = {
 (G:Group-name | U:User-name[:Account-name]) = Share-distribution-ratio
 (G:Group-name | U:User-name[:Account-name]) = Share-distribution-ratio
 ...
 }
 Group-name = {
 (G:Group-name | U:User-name[:Account-name]) = Share-distribution-ratio
 (G:Group-name | U:User-name[:Account-name]) = Share-distribution-ratio
 ...
 }
 ...
```

ユーザはいずれかの Group-name に属し、各グループはツリー構造で管理されます。 ツリー構造の先頭グループは TOP_GROUP であり、Share-distribution-ratio は、グループ内での配分比を設定します。

Account-name が省略された場合は、アカウントコードによるユーザの区別は行いません。 シェア配分比設定ファイルに存在しないユーザは、ユーザシェア値 = 0 となります。

以下に設定例を示します。

```
/etc/opt/nec/nqsv/jm_sharedb.conf
 TOP_GROUP = {
   U:root=50
   G:GroupA=30
   G:GroupB=20
 }
 GroupA = {
   U:User1=20
   U:User2=10
 }
 GroupB = {
   U:User10=10
   U:User11=10
   U:User12=10
   U:User13=10
   U:User14=10
 }
```

使用実績値（総合）

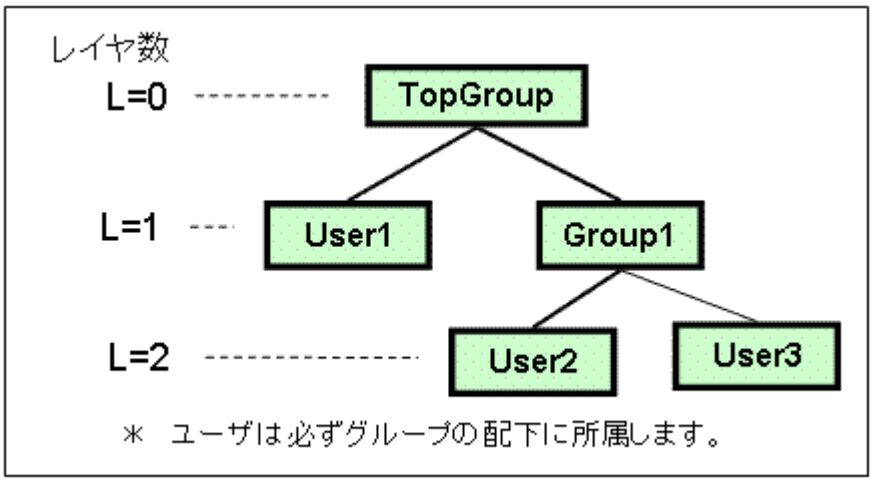
使用実績値（総合）	
= CPU台数（正規化）	× 重み係数(使用実績 CPU台数)
+ 経過時間（正規化）	× 重み係数(使用実績 経過時間)
+ メモリ使用量（正規化）	× 重み係数(使用実績 メモリ使用量)
+ リクエストプライオリティ（正規化）	× 重み係数(使用実績 リクエストプライオリティ)
+ VE台数（正規化）	× 重み係数(使用実績 VE台数)

ユーザランク

ユーザランクとは、あるユーザに割り当てられたシェア配分(予定)と使用実績(実績)の比率をもとに、

階層的に設定されたユーザ間での順位(優先度)を決めるためのものです。

- ユーザランク算出方法
  ユーザは階層化された構造において管理されます。下位層のシェアと使用実績は、そのユーザが属する上位ノードに集計されて管理されます。上位シェアは下位シェアより強い影響度をもって制御されます。特に、最上位層のサイト間のシェアは優先されます。



- 算出方法と公式

1. 各ユーザ単体のランク値を算出します。

(i)　全ユーザのランク値を相対的に直接比較可能とするため、ユーザシェア値を総使用実績で割った値を用います。
(ii)　上記の値を、階層化されたユーザ構造の中で、バランスをとった値に補正するために、ユーザ数とレイヤ数(=階層数) をパラメータとした係数で割ります。つまり、当該ユーザのレイヤまでの総ユーザ数による対数値(log N)に、当該ユーザのレイヤ数(= L :トップを0とする)を乗じた数を係数とします。
log N：ユーザ数Nが大きいほど、大きな値となります。つまりシェアを分け合うユーザがたくさんいるほど、分母が大きくなり、使用率の値((i)の値)の評価値は小さく補正されます。
L：　上位シェアを優先し、かつ最上位サイト間のシェアが特に強い影響度をもつようにするために、レイヤ数を乗じた値を係数とします。

2. 階層化された、ある位置に属する当該ユーザのユーザランクを算出します。すなわち、当該ユーザより直系上位のすべてのユーザについて 1.で求めた値の総和とします。

3. 最小 0、最大 1 となるように正規化を行います。正規化の際の分母は、当該ユーザが位置するレイヤ（階層）によって異なります。

(i)

r = (log R) / (log N * L)

R = ユーザシェア値 / 総使用実績

（0.01 <= R <= 100。範囲外の値は最大値または最小値となります。）

N = ユーザ数（該当ユーザのレイヤまでの総数。トップは含みません。）

L = レイヤ数（トップを0とします。）

(ii)

UserRank = r1 + r2 + r3 + ... + r(L-1)

(iii)

(i)のrの分子の最大値は+2、最小値は-2。

従ってrの最大値は +2/(log N * L)、最小値は -2/(log N * L)となります。

総和の最大値は +2(1/(log N1*1) + 1/(log N2*2) + ... + 1/(log NL * L))

総和の最小値は -2(1/(log N1*1) + 1/(log N2*2) + ... + 1/(log NL * L))

正規化式 = 0.5 + UserRank / (2 * 2 (1 / (log N1 * 1) + 1 / (log N2 * 2) + ... + 1 / (log NL * L)))

リクエストプライオリティ

リクエストプライオリティは、**qsub**(1)コマンドの**-p** オプションによって指定されます。リクエストプライオリティ = 1023 の場合を 1、リクエストプライオリティ= -1024 の場合を 0 とします。

正規化式

（リクエストプライオリティ + 1024 ）/ （ 1023 + 1024 ）

実行リクエストの要求リソース量

要求リソース量には以下のリソース制限値を使用します。

CPU数	論理ホスト毎の同時実行CPU台数（qsub --cpunum-lhost）
経過時間	リクエスト毎の経過時間（qsub -l elapstim_req）
メモリ（オプション）	論理ホスト毎のメモリサイズ（qsub --memsz-lhost）
ジョブ数	ジョブ数（qsub -b）
VE数	論理ホスト毎の同時実行VE台数（qsub --venum-lhost） ※qsub --venodeでVEノード総数を指定した場合、キューの既定搭載VEノード数を使用します。

- CPU数

  ユーザの宣言したCPU台数に従って0（最大実装CPU台数）～ 1(≒ 1台）とします。

正規化式

1 - (CPU数 / 実装量)

- 経過時間

  ユーザの宣言した経過時間に従って、0 (無制限）～ 1 (≒ 1秒）とします。

正規化式

0.5 ^ (経過時間 / 半減期)

- メモリ（オプション）

  ユーザの宣言したメモリサイズに従って0 （最大実装メモリサイズ）～ 1 (≒ 1バイト）とします。

正規化式

1 - (メモリサイズ / 実装量)

- ジョブ数

  ユーザの宣言したジョブ数に従って0 （ジョブ数 = 標準ジョブ数）～ 1 (≒ ジョブ数 = 1) とします。

正規化式

1 - (ジョブ数 / 標準ジョブ数)

- VE数

  ユーザの宣言したVE台数に従って0（最大実装VE台数）～ 1(≒ 1台）とします。

正規化式

1 - (VE数 / 実装量)

キューに投入してからの実行待ち時間

半減期あたりの実行待ち時間を 1 とします。

キューに投入してからの実行待ち時間 / 半減期

アサイン可能になってからの実行待ち時間

半減期あたりの実行待ち時間を 1 とします。

アサイン可能になってからの実行待ち時間 / 半減期

アサイン可能になってからの実行待ち時間の要素は、JobManipulator とキューのバインド、JobManipulator およびバッチサーバの再起動によってリセットされます。

アサイン可能になってからの実行待ち時間の重みづけを大きく設定している場合、JobManipulator とキューのバインド、JobManipulator およびバッチサーバの再起動を経ても優遇したいリクエストに対しては他の要素によってプライオリティを調整してください。

サスペンドされてからの再開待ち時間

半減期あたりの再開待ち時間を 1 とします。

再開待ち時間 / 半減期

緊急リクエストによって SUSPEND されたリクエストに対するベースアップ

緊急リクエストおよび特別リクエストの投入で、サスペンドされたリクエストに対して、スケジューリングプライオリティをベースアップする値を設定します。本ベースアップ値は該当リクエストに対して一律に設定されます。

また、設定値はスケジューラ単位で変更可能です。

再スケジューリングリクエストに対するベースアップ

再スケジューリングされた実行中のリクエスト、予定どおりに実行開始できなかったリクエストに対して、スケジューリングプライオリティをベースアップする値を設定します。本ベースアップ値は該当リクエストに対して一律に設定されます。

また、設定値はスケジューラ単位で変更可能です。

ユーザ定義ベースアップ

本ベースアップ値は、管理者がスケジューリングプライオリティを変更したい場合に設定します。本ベースアップ値は、**smgr**(1M)コマンドで、リクエスト毎に動的に設定することができます。

(3) スケジューリングプライオリティの計算契機

スケジューリングプライオリティが計算されるタイミングを以下に示します。

- リクエスト投入時
- qalter(1)コマンドによるリクエスト属性の変更時

実行待ち時間を含めたスケジューリングプライオリティは、リクエストのピックアップ時に再計算されます。

(4)　スケジューリングプライオリティを使用する処理

スケジューリングプライオリティは次のような処理(実行ホストの割り当てやエスカレーション等)のリクエスト選択等に使用されます。

- 実行ホストの割り当て
- エスカレーション
- 追い越し制御

(5)　設定用サブコマンド

スケジューリングプライオリティの各項目に対する重み付けの値は **smgr**(1M)コマンドの **set** サブコマンドで定義してください。

各項目に対応するサブコマンドを下記に記述します。また、設定には操作員権限が必要です。

項目	smgr(1M)サブコマンド
重み付け	
リクエストプライオリティに対する重み付け	set priority weight request priority
CPU数に対する重み付け	set priority weight cpu number
経過時間に対する重み付け	set priority weight elapse time
メモリサイズに対する重み付け	set priority weight memory size
ジョブ数に対する重み付け	set priority weight job number
VE数に対する重み付け	set priority weight ve number
キューに投入してからの実行待ち時間に対する重み付け	set priority weight run wait time
アサイン可能になってからの実行待ち時間に対する重み付け	set priority weight assign wait time
再開待ち時間に対する重み付け	set priority weight restart wait time
ユーザシェア値に対する重み付け	set priority weight user share
ユーザランクに対する重み付け	set priority weight user rank

ベースアップ値	
緊急リクエストによってSUSPENDされたリクエストに対するベースアップ値	set priority baseup_interrupted
再スケジューリングリクエストに対するベースアップ値	set priority baseup_reschedule
ユーザ定義ベースアップ値　（リクエスト毎に設定）	set request baseup_user_definition
使用実績の重み付け	
使用実績のリクエストプライオリティに対する重み付け	set priority_ pastusage_weight_request_priority
使用実績のCPU数に対する重み付け	set priority pastusage_weight_cpu_number
使用実績の経過時間に対する重み付け	set priority pastusage_weight_elapse_time
使用実績のメモリサイズに対する重み付け	set priority_ pastusage_weight_memory_size
使用実績のVE数に対する重み付け	set priority pastusage_weight_ve_number

以下に、アサイン時のリクエストプライオリティに対する重み付け値を 1 に設定する例を示します。

```
# smgr -Pm

Smgr: set priority weight_request_priority = 1 processing_pattern = assign
```

3.1.5　リクエスト選択アルゴリズム

複数キュー、複数リクエストが存在する状態において、スケジューリング対象となるリクエストは以下のポリシーに従ってピックアップされます。

1. リクエストが投入されたキューの種別が高い(緊急キュー、特別キュー、通常キューの順)
2. 投入されたキューのプライオリティが高い
3. スケジューリングプライオリティが高い
4. キューに投入された時間が早い
5. 4 の条件でリクエストを絞り込んだとき、まだ一つのリクエストに決まらない場合は、スケジューラがそれらのリクエストから一つを決定する

このようにしてリクエストの優先順を決定し、優先順の高いリクエストからスケジューリング対象として扱います。

[注意]

特別なケースとして、マップが緊急リクエストあるいは特別リクエストで一杯になり、次の緊急リク

エストあるいは特別リクエストがアサインできない場合、優先度の低いキュー内のリクエストがスケジューリングの対象となることがあります。その場合、マップ上に一旦アサインできても、その後に他の特別リクエストや緊急リクエストにより実行が止められることがあります。

[例]

あるジョブを以下 2 つのポリシーで投入した場合、"キュー種別：特別キュー / キュープライオリティ：100"の指定を行ったリクエストが優先的にリソースへ割り当てられます。

- キュー種別：特別キュー / キュープライオリティ：100
- キュー種別：通常キュー / キュープライオリティ：100

3.1.6　リクエスト実行開始アルゴリズム

JobManipulator は、リクエスト選択アルゴリズムにより決定されたリクエストに対して、ジョブサーバの割り当て、実行開始時間の設定を行います。

ジョブコンディションが指定されたリクエストの場合は、ジョブコンディションに合致するジョブサーバをジョブアサインの対象とします。ジョブコンディションの指定がない場合は、リクエストが投入された実行キューにバインドされている全てのジョブサーバがジョブアサインの対象となります。

ジョブコンディションでは、condition に条件文を、job_number にジョブコンディションを適用するジョブのジョブ番号を指定します。詳細は *NQSV* 利用の手引*[*操作編*]* を参照してください。

condition の値によるジョブサーバのアサイン方法は以下の通りです。

Condition	job_numberのジョブのアサイン方法
JSV(ジョブサーバ番号)	conditionで指定されたJSV番号のジョブサーバのいずれか
HW(ハードウェア種別)	conditionで指定されたハードウェアのジョブサーバのいずれか
NGRP(ノードグループ名)	conditionで指定されたノードグループ中のジョブサーバのいずれか

バックフィル・スケジューリングを行うためにユーザがリクエスト投入時に指定しなければならない宣言値は次のとおりです。**qsub**(1)コマンドのオプション **-l** にて指定します。

必須オプション

- 経過時間制限値（オプション -l, サブオプション elapstim_req）
  * ただし*4. 8 Elapse無制限機能*をonに設定している場合、経過時間制限値の指定は任意となります

- ジョブ毎の同時実行CPU台数（オプション -l, サブオプション cpunum_job）
  * ただし、qsub(1)コマンドで--exclusiveオプションを指定している場合、cpunum_jobの指定は不要です。
- GPUを使用するリクエストの場合は、ジョブ毎の同時実行GPU台数(オプション -l, サブオプション gpunum_job)を指定する必要があります。
- VEノードを使用するリクエストの場合は、論理ホスト毎VE台数（オプション--venum-lhost）またはVEノード総数（オプション--venode）を指定する必要があります。

本宣言値が設定されていない(unlimited)リクエストはスケジューリング対象にならず、保留(Queued)状態になります。ログファイル（デフォルトファイル名：/var/opt/nec/nqsv/nqs_jmd_<scheduler_id>.log)にその旨出力されます。**qalter**(1)コマンドで本宣言値を設定することによって、保留されているリクエストをスケジューリング対象にする事ができます。

[例] 以下は、経過時間制限値を設定しなかった場合のログメッセージです。

```
Judge_assignable :  Request cannot be scheduled. (Elapse time unlimited) <Request-ID>
```

なお、--exclusive オプションまたは、ジョブ毎の同時実行 CPU 台数に実行ホストの使用可能 CPU 台数を指定することにより、ジョブ毎に１実行ホストを割り当てることも可能です。

また、オプションによりメモリサイズを使用したスケジューリングを行う場合は、ユーザは次の宣言値を指定しなければなりません。

選択オプション

- ジョブ毎のメモリサイズ（オプション -l, サブオプション memsz_job）

メモリサイズを使用したスケジューリングを行う場合、**smgr**(1M)コマンドにて、実行ホストの使用メモリ量制限（memsz_limit_ratio）を設定しておくことが前提となります。メモリサイズを使用するスケジューリングにもかかわらず、本宣言値が設定されていない(unlimited)リクエストは、スケジューリング対象になりません。

ジョブを割り当てる実行ホスト選択時の優先項目は、次のとおりであり、全てが成立する場所を選択します。割り当てる場所がない（実行開始予定時間がマップの範囲外）ときは、次回割り当て処理まで Queued 状態(保留)になります。

1. 計算資源（経過時間、CPU/GPU(GPU数を指定した場合)）が確保できる

2. メモリが確保できる（オプション）
3. 実行開始予定時間が一番早い

なお、バックフィル・スケジューリングではノードの稼働率を最優先にジョブを割り当てるため、ジョブの実行順序は、必ずしも割り当て順序と同様になるとは限りません。

3.1.7 Elapse マージン

Elapse マージンとは、リクエストに設定されている経過時間制限値にスケジューラがマージン時間を付加し、次のリクエストが開始されるまで余裕をもたせる機能です。Elapse マージンを設定している場合は、リクエストに設定されている経過時間制限値と Elapse マージン時間を加算した時間を元にスケジューラマップ上のリソース占有時間を決定します。Elapse マージンを設定していない場合は、リクエストに設定されている経過時間制限値を元にスケジューラマップ上のリソース占有時間を決定します。

Elapse マージンを設定していない場合、リクエストに設定された経過時間制限値がスケジューラマップ上のリソース占有時間となります。しかし、以下に挙げた状態に掛かる時間は、経過時間制限値として加算されません。

- リクエストのステータスがPRE-RUNNING
- リクエストのステータスがPOST-RUNNING

このため、上記状態の処理に掛かる時間と経過時間制限値の合計がリクエストに設定されている経過時間制限値を超えた場合、スケジューラマップに確保したリソース占有時間を越えてリクエストが実行され、他リクエストのリソース占有時間と重なってしまいます。 上記状態に時間の掛かる運用を行っている場合、他リクエストとリソース占有時間が重複しないように Elapse マージンの設定を推奨します。

(1) Elapseマージンの設定

Elapse マージンの設定はキュー単位で設定することができます。キュー単位でリクエストの規模を分けて運用している場合、リクエストの規模に見合った Elapse マージンの設定ができます。

設定方法： Elapse マージンの設定は **smgr**(1M)コマンドの **set queue elapse_margin** サブコマンドで行います。

```
#smgr -P m
Smgr: set queue elapse_margin = elapse_margin queue-name
```

- elapse_marginの初期値は 0 です。
- elapse_marginにはqueue-nameで指定されたキューに設定するElapseマージンを設定します。
- Elapseマージンは、秒単位で指定します。設定できる範囲は、 0 - 2147483647 秒です。
- elapse_marginを変更した場合、実行中以外のジョブは、全て再アサインされます。
- Elapseマージンの設定は、操作員以上の権限が必要です。

* 以下(3 つ)の場合、エラーとなり、Elapse マージンは変更されません。

1. 指定されたElapseマージンが指定可能な値の範囲外であった場合
   エラーメッセージ：Elapse margin value out of bounds.
2. JobManipulatorで管理されていないキューが指定された場合
   エラーメッセージ：No such queue. (name: <queue-name>)
3. コマンドを実行したユーザに操作員権限以上の権限がない場合
   エラーメッセージ：Not permitted to modify attribute.

Elapse マージンの値は、リクエストが PRE-RUNNING/POST-RUNNING に要する時間を考慮する必要があります。これらに要する時間は、システム性能、キューに設定されているユーザ EXIT スクリプト等、運用環境により変化します。Elapse マージン設定値は以下を留意して値を決めてください。

- Elapseマージンを過剰に付加すると、スケジューラマップへアサイン可能なリクエスト数が減少します。
- Elapseマージンの値が少なすぎると、Elapseマージンを使い切り、リクエストのリソース占有が保証できなくなります。

---

(2)　Elapseマージン表示機能

キューに設定されている Elapse マージン

キューに設定されている Elapse マージンの値は、**sstat**(1)コマンドの**-Q,-f** オプションで表示します。

```
#sstat -Q -f
Queue Name: jmq0
    Queue Type            = Normal
```

```
   Schedule Time         = DEFAULTE
   Run Limit             = UNLIMITED
   User Run Limit        = UNLIMITED
   User Assign Limit     = UNLIMITED
   Elapse Margin         = 600S  ←#Elapseマージン
                :
 以下略
```

リクエストの Elapse マージン

リクエストの Elapse マージン値は、sstat(1)コマンドの-f オプションで表示します。

リクエストの Elapse マージン値は、リクエストが投入されたキューの Elapse マージン値が設定されています。

リクエストの Planned End Time、及び Elapse Time には Elapse マージンの値が含まれています。

```
Request ID: 5208.batch_serverhost
   Request Name = test_jm
   User  Name = nqs_user
   User  ID   = 2019
   Group ID   = 500
   Current State           = Running
   Previous State          = Pre-running
   State Transition Time   = 2008-05-01 16:17:55
   State Transition Reason = PRERUN_SUCCESS
   Queue = oseq
   Reservation ID      = -1
   Scheduling Priority (Assign)     = 0.998855
     User Share                     = 0.000000
     User Rank                      = 0.000000
     Request Priority               = 0.000000
     CPU Number                     = 0.000000
     VE Number                      = 0.000000
     Elapse Time                    = 0.998855
     Memory Size                    = 0.000000
     Job Number                     = 0.000000
     Run Wait Time                  = 0.000000
     Restart Wait Time              = 0.000000
     Baseup Interrupted             = 0.000000
     Baseup Reschedule              = 0.000000
     Baseup User Definition         = 0.000000
```

```
    PastUsage Request Priority     = 0.000000
    PastUsage CPU Number           = 0.000000
    PastUsage VE Number            = 0.000000
    PastUsage Elapse Time          = 0.000000
    PastUsage Memory Size          = 0.000000
  Scheduling Priority (Escalation) = 0.500244
    User Share                     = 0.000000
    User Rank                      = 0.000000
    Request Priority               = 0.500244
    CPU Number                     = 0.000000
    VE Number                      = 0.000000
    Elapse Time                    = 0.000000
    Memory Size                    = 0.000000
    Job Number                     = 0.000000
    Run Wait Time                  = 0.000000
    Restart Wait Time              = 0.000000
    Baseup Interrupted             = 0.000000
    Baseup Reschedule              = 0.000000
    Baseup User Definition         = 0.000000
    PastUsage Request Priority     = 0.000000
    PastUsage CPU Number           = 0.000000
    PastUsage VE Number            = 0.000000
    PastUsage Elapse Time          = 0.000000
    PastUsage Memory Size          = 0.000000
  Planned Start Time = (Already Running...)
  Planned End Time   = 2008-05-01 16:44:35     ←#Elapseマージンが付加された予定終了時刻
  Elapse Margin      = 600S                    ←#Elapseマージン
Job Server a Job belongs to (Job No.:JSV No.):
    0:500
Resources Limits:
  Elapse Time = 1600S                          ←#Elapseマージンと経過時間制限値の和
  CPU Number  = 8
  Memory Size = 256MB
```

3.1.8　アサインポリシー

(1)　アサインポリシーの優先順位

通常アサイン時は、以下の優先順位でアサインポリシーを適用し、ノードを選択します。なお、一

部のアサインポリシーは、優先度の設定により優先順位が変化します。詳細については、「2.7.8 (3) アサインポリシーの優先度設定と無効化設定」を参照してください。

1. リクエストを最も早い時間にアサインできるノードの優先アサイン
2. JSV アサインプライオリティが高いノードの優先アサイン
3. 予定実行開始時刻がクリアされたステージング中ジョブがあるノード以外のノードの優先アサイン（優先度設定が high の場合）
   （「3.1.7 (3) ステージング中ジョブがあるノード以外のノードの優先アサイン」参照）
4. ネットワークトポロジーを考慮したアサイン（優先度設定が high の場合）
   （「3.1.7 (2) ネットワークトポロジーを考慮したアサイン」参照）
5. トポロジー性能を考慮したスケジューリング機能を考慮したアサイン（実行ホストが SX-Aurora TSUBASA の場合）
6. CPU 台数による集中/分散ポリシー
7. 前後検査アサインポリシー
8. 予定実行開始時刻がクリアされたステージング中ジョブがあるノード以外のノードの優先アサイン（優先度設定が low の場合）
9. バインドしているキューが少ないノードの優先アサイン
10. ネットワークトポロジーを考慮したアサイン（優先度設定が low の場合）

割り込みアサイン時のポリシーは、上記優先順位に加えて以下のポリシーを考慮します。

1. Dynamic JSV Priorityが高いノード（実行ホストがSX-Aurora TSUBASAの場合）
2. RUNNING状態のリクエストがないノード
3. 割り込みにより再スケジューリングされるリクエストが少ないノード

Dynamic JSV Priority については、第 5 章 SX-Aurora TSUBASA 対応機能 の 5.7 Dynamic JSV Priority を参照してください。

上記の優先度設定は、**smgr**(1M)コマンドの **set assign_policy_priority** サブコマンドにより設定します。無効化設定も可能です。詳細については、「2.7.8 (3)アサインポリシーの優先度設定と無効化設定」を参照してください。

(2) ネットワークトポロジーを考慮したアサイン

複数のノード群を多段のネットワークスイッチ(NW-SW)で接続するシステム構成の時、複数ノードを使用してノード間通信を行うリクエストにノードをアサインする場合、ノード間通信を速くするために、できるだけ同一のネットワークスイッチ（NW-SW）に接続されているノード群をアサインする

機能を提供します。この機能を、ネットワークトポロジーを考慮したアサイン機能と言います。

このネットワークトポロジーを考慮したアサイン機能を利用するには、JobManipulator を起動する前に、通信レイテンシの少ないノード群をノードグループでグループ化する必要があります。ノードグループの操作については、qmgr(1M)コマンドを利用します。詳細は、NQSV 利用の手引[リファレンス編] を参照してください。

本アサインポリシーは、**smgr**(1M)コマンドの **set assign_policy_priority** サブコマンドの network_topology アサインポリシーに対する設定により、優先度の変更が可能です。稼働率を重視する運用では、ポリシーの優先度を既定値の high から low に変更することを推奨します。

ネットワークトポロジーを考慮したアサイン機能の利用

通信レイテンシの少ないノード群をグループ化します。

1. ノードグループの作成

ネットワークトポロジーのためのノードグループを作成します。なお、ノードグループのタイプを nw_topo とします

```
#qmgr -Pm
Mgr: create node_group = <ノードグループ名>  type = nw_topo [switch_layer = <layer>]
```

- node_group　：任意のノードグループ名
- type　：nw_topo （固定）
- switch_layer　ネットワークスイッチの段数。2段までスケジューリングが可能。

2. ノードグループへのノード登録

ノードグループに対し、通信レイテンシの少ないノード群を登録します。同一ノードが複数のノードグループに属しないようにしてください。本機能の場合ノードグループをネストさせることはできません。

```
#qmgr -Pm
Mgr: edit node_group add job_server_id = <jsvid>-<jsvid> <ノードグループ名>
Mgr: edit node_group add job_server_id = (<jsvid>,<jsvid>,<jsvid>...) <ノードグループ名>
```

ノードをグループ化するイメージは下記のとおりです。

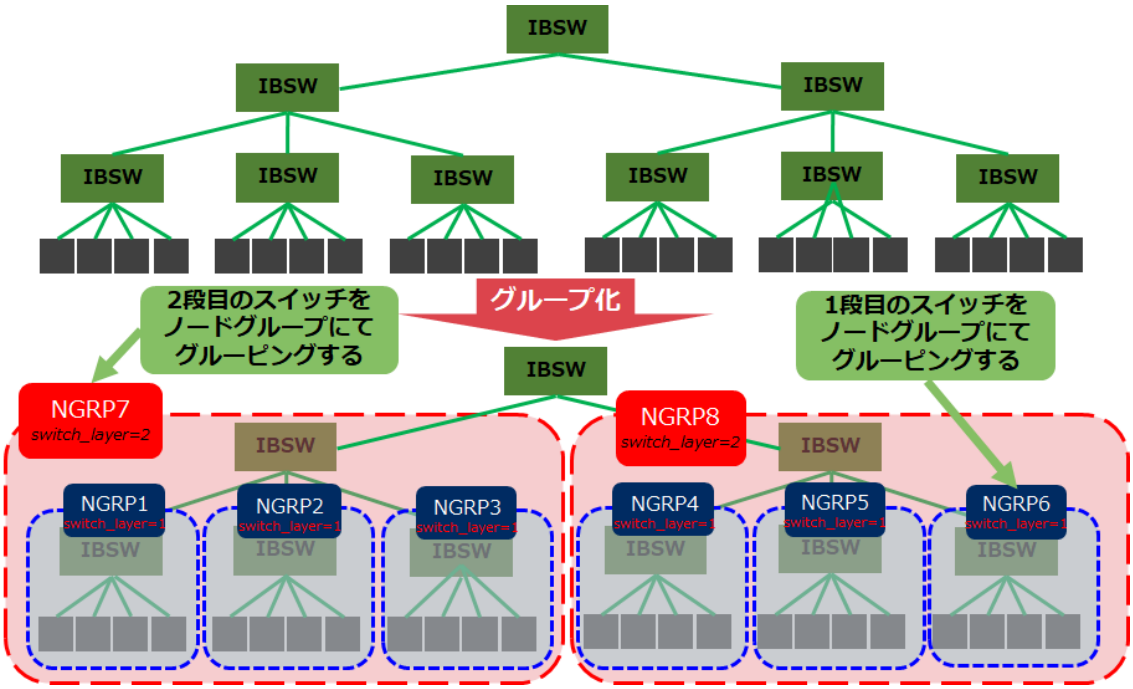


図 3-4：ネットワークトポロジーノードグループを定義するイメージ

ネットワークトポロジーを考慮したアサイン機能の中止

ネットワークトポロジーを考慮したアサイン機能を中止する場合には、**smgr**(1M)コマンドの **set assign_policy_priority** サブコマンドで network_topology アサインポリシーに対して、disable と設定するか、(1)で作成したネットワークトポロジーのためのノードグループ（type が nw_topo となっているノードグループ）を削除します。

```
#qmgr -Pm
Mgr: delete node_group = <ノードグループ名>
```

(3)　ステージング中ジョブがあるノード以外のノードの優先アサイン

予定実行開始時刻にステージングが間に合わない場合は、当該リクエストの予定開始時刻をクリアし、ステージング完了後、元のノードにおける予定実行開始時刻を再決定します。他のリクエストをアサインする時、同じ時刻にアサインできる別のノードを選択することにより、当該ノードを予定実行開始時刻がクリアされたリクエストに残せます。

ステージングを実施し、ステージングが実行に間に合わない場合に予定実行開始時刻がクリアされたリクエストの TAT を重視する運用では、ポリシーの優先度を既定値の low から high に変更することを推奨します。

3.1.9　サスペンドされたリクエスト

- **qsig**(1)コマンドによりリクエストをサスペンドした場合
  JobManipulatorでは、そのリクエストについては特に操作は行わず、サスペンドされたリクエストが確保したリソースは保持されたままとなります。この場合、リクエストがサスペンド中の間も経過時間は経過し、経過時間に達するとバッチサーバから強制終了されます。そのため、後続リクエストのスケジューリングには影響はありません。この動作は、**qsig**(1)コマンドの実行者が一般利用者/管理者にかかわらず一定です。**qsig**(1)コマンドによりサスペンドされたリクエストは**sstat**(1)コマンドの**-f**オプションで確認でき、**Suspend Reason**に**SIGSTOP**と表示されます。

- **smgr**(1M)コマンドによりリクエストをサスペンドした場合
  実際にはメモリを使用したままですが、そのリクエストが確保していたリソースが全て解放されたものとみなします。リクエストの経過時間が一旦停止され未アサイン状態となります。**smgr**(1M)コマンドによるリクエストのサスペンドは管理者/操作員が実行可能であり、**suspend request** サブコマンドを使用します。 **smgr**(1M)コマンドによりサスペンドされたリクエストは**sstat**(1)コマンドの**-f**オプションで確認でき、**Suspend Reason**に**SMGR_SUSPEND**と表示されます。このサスペンドされたリクエストは、**smgr**(1M)コマンドの**resume request** サブコマンドによりリジューム要求することで宣言経過時間から実行済み時間を引いた時間をもとにしてリジューム可能な領域をアサインし、予定再開時刻に達したらスケジューラからリジュームを行います。**smgr**(1M)コマンドによりリジューム要求されたリクエストは**sstat**(1)コマンドの**-f**オプションで確認でき、**Suspend Reason**に**SMGR_RESUME**と表示されます。

- 緊急/特別リクエストの割り込みによりスケジューラがサスペンドした場合
  実際にはメモリを使用したままですが、そのリクエストが確保していたリソースが全て解放されたものとみなします。リクエストの経過時間は一旦停止し、宣言経過時間から実行済み時間を引いた時間をもとにしてリジューム可能な領域をアサインし、予定再開時刻に達したらスケジューラからリジュームを行います。緊急/特別リクエストの割り込みによりサスペンドされたリクエストは**sstat**(1)コマンドの**-f**オプションで確認でき、**Suspend Reason**に**INTERRUPT**と表示されます。

実行ホストが SX-Aurora TSUBASA システムの場合は、**5.6 VE ジョブのサスペンド**の節も参照してください。

smgr(1M)コマンドもしくは緊急/特別リクエストの割り込みによりサスペンドされたリクエストについて

- CPUは解放されますが、プロセスは残存しているためメモリは確保したままとなります。したがってサスペンド中に他のリクエストが実行されてもメモリもしくはスワップは十分に確保できる状態にしておく必要があります。他のリクエスト実行時にメモリ不足が発生した場合、ジョブがアボートする原因となります。
- サスペンドされたリクエストに対して、管理者は**qsig**(1)コマンドでリジュームすることができます。このときリジュームされたリクエストは直ぐに実行を再開するため、既に実行されているリクエストとリソースの競合が発生する可能性があります。

3.1.10 ジョブコンディション

ユーザの投入したリクエストのジョブをどの実行ホスト（ジョブサーバ）で実行するかは JobManipulator が決定します。しかし、ユーザリクエストの種類やサイトでの運用ポリシーによっては、特定のジョブを指定の実行ホスト（ジョブサーバ）上で運用したいケースも考えられます。このような場合には、ジョブに対してジョブコンディションを指定してください。

ジョブコンディションは、あるジョブをどういった条件の実行ホスト、ジョブサーバで実行させるのかといった実行条件をリクエスト投入時に指定するものであり、JobManipulator はここで指定されたジョブコンディションをもとにスケジューリングします。

ジョブコンディションは、qsub(1)コマンド、qlogin(1)コマンド、あるいは、qrsh(1)コマンドの -B オプションで指定します。ジョブコンディションの指定形式の詳細は、NQSV 利用の手引[リファレンス編]の qsub(1)、qlogin(1)、または qrsh(1)のコマンドリファレンスを参照してください。

3.1.11 終了遅延スケジューリング機能

I/O 障害等によりジョブが終了せず、リクエストの実行時間がスケジューラマップ上のリソース占有時間を越えることがあります。

本機能は、この状況下でリソースの重複を防ぐために、終了が遅延したリクエストに割り当てているノードをスケジューリング対象外とします。また、その影響を受けたリクエストを再スケジューリングします。終了遅延リクエストが終了した後、当該ノードはスケジューリング対象に戻ります。

本機能を有効にするには、下記のように NQSV/JobManipulator のコンフィグファイル（/etc/opt/nec/nqsv/nqs_jmd.conf）に EXITDELAY_SCHEDULING 指示行を設定します。設定後、

NQSV/JobManipulator を再起動してください。

```
EXITDELAY_SCHEDULING: ON
```

qalter(1)コマンドにて経過時間制限値をリソース占有時間以上に拡大した実行中リクエストは本機能の対象となり、後続のリクエストが再スケジューリングされる場合があります。再スケジューリングを防ぐためには、**qalter**(1)コマンドと併せて **qrerun**(1)コマンドを実行してください。

3.2　システムの情報の表示

JobManipulator の情報を確認するには、sstat(1)コマンドを実行してください。

sstat(1)コマンドに次のオプションを付けて実行すると、各情報を表示します。

表示される情報	指定オプション
バッチリクエスト情報	オプションなし
アサインマップ情報	**-A**
予約区間情報	**-B**
コンプレックスキュー情報	**-C**
計画省電力区間情報	**-D**
実行ホスト情報	**-E**
ジョブサーバ情報	**-J**
スケジューリングプライオリティ情報	**-M**
キュー情報	**-Q**
スケジューラサーバ情報	**-S**

バッチリクエスト情報、スケジューラサーバ情報、キュー情報は詳細情報の表示が可能です。詳細情報を知りたい場合は、各オプションを**-f** オプション付きで実行してください。

第4章　　高度なスケジューリング機能

4.1　緊急/特別リクエスト

緊急リクエストは、通常のリクエストより優先してアサインし、実行するリクエストです。

キューには urgent、special、normal の 3 つのレベルがあり、キュー単位で設定可能です。urgent タイプのキューに投入されたリクエストを緊急リクエスト、special タイプのキューに投入されたリクエストを特別リクエスト、normal タイプのキューに投入されたリクエストを通常リクエストと呼びます。

緊急リクエストは、緊急キューに投入されたリクエストです。緊急リクエストは、特別リクエストや通常リクエストより優先してアサインされて実行を開始します。

特別リクエストは、特別キューに投入されたリクエストです。特別リクエストは、通常リクエストより優先してアサインされて実行を開始します。

キューのタイプは、下記の **smgr**(1M)コマンドのサブコマンドで設定してください。

smgr(1M)
set queue type = urgent | special | normal <*queue-name***>**

緊急/特別リクエストは、実行中の通常リクエストに割り込む位置を指定できます。以下の 2 つの位置を指定できます。

- 通常リクエストを中断して、緊急/特別リクエストを実行する
- 通常リクエストの実行完了後に、緊急/特別リクエストを実行する

スケジューラ単位、およびキュー単位に設定が可能です。キュー単位の設定がない場合はスケジューラ単位の設定に従います。

通常リクエストの実行を中断する場合、割り込み位置を current に設定します。通常リクエストの実行完了後に緊急/特別リクエストを実行する場合、割り込み位置を next_run に設定します。

割込み位置を current に設定している場合でも、緊急/特別リクエスト投入時に実行中のリクエストが、投入リクエストと同一レベル以上であった場合は、実行中リクエストの後方にアサインされます。

通常リクエストの実行を中断させる方法は、サスペンド、もしくはリランを設定できます。この中断方法は、スケジューラ単位で設定が可能です。割込み位置の設定が current の場合のみ、本設定が有効になります。

通常リクエストを中断させる方法にサスペンドを選択した場合、サスペンドされたリクエストは緊急/特別リクエストの後方にアサインされ、緊急/特別リクエストの終了後にリジュームします。

通常リクエストを中断させる方法にリランを選択した場合、割り込まれたリクエストはリランされ

ます。リランされたリクエストを優先して再実行するために、JobManipulator はスケジューリングプライオリティに再スケジューリングリクエストに対するベースアップを付加して再スケジューリングします。なお、割り込まれたリクエストがリラン禁止の設定でも強制的にリランします。

割込み位置および通常のリクエストの実行を中断させる方法は、下記の **smgr**(1M)コマンドのサブコマンドで設定してください。

smgr(1M)
- **set interrupt_to_where**　　：割込み位置（スケジューラ単位）の設定
- **set queue interrupt_to_where**　　：割込み位置（キュー単位）の設定
- **set interruption_method**　　：通常のリクエストの実行を中断させる方法

実行ホストが SX-Aurora TSUBASA システムの場合は、**5.6 VE ジョブのサスペンド**の節も参照してください。

リクエストの実行を中断させる方法が **suspend** の場合の注意事項：
smgr(1M)コマンドの **suspend request** サブコマンド、あるいは **qsig(1)**コマンドを使用して、サスペンドされたリクエストが存在する場合、緊急/特別リクエストを利用することはできません。緊急/特別リクエストを実行した場合、サスペンドされたリクエストのリソース管理は保証されません。

4.1.1　緊急リクエストによるアサイン禁止

緊急リクエストは、リソース不足によりすぐに実行することができずに、実行待ちとなる場合があります。このとき、緊急リクエストがアサインされている時刻よりも前方に小さい空きリソースが生じることがあります。このような空きリソースに、新規投入された下位タイプリクエストのアサインを禁止することが可能です。この設定を有効にした場合、実行待ちの緊急リクエストがアサインされている実行ホスト全体がアサイン禁止となります。アサイン禁止は緊急リクエストが実行開始することで解除されます。

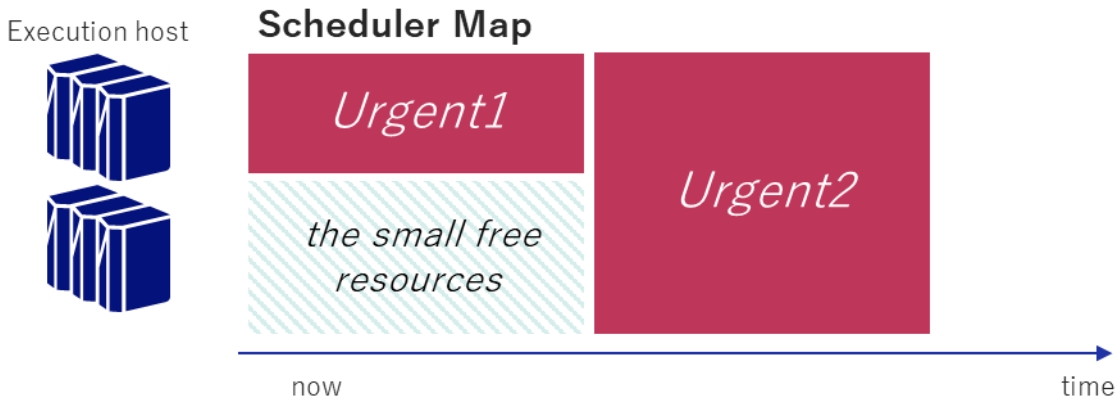


図 4-1 緊急リクエストがアサインされている時刻よりも前方に生じる小さい空きリソース

この機能により、実行待ちの緊急リクエストに必要なリソースが空き次第、速やかに緊急リクエストを開始することが可能になります。緊急リクエストの実行を最優先とするため、この動作が既定値となります。

緊急リクエストの投入量が多い場合は、後から投入された下位タイプリクエストを緊急リクエストよりも先に、追い越してアサイン・実行することを許容する運用が必要かもしれません。その場合はこの機能を off にすることで、後から投入された下位タイプリクエストを緊急リクエストよりも先に、追い越してアサイン・実行することを許容することが可能です。下記の **smgr**(1M)コマンドの **set interrupt_assign_block** サブコマンドで設定してください。

smgr(1M)
　set interrupt_assign_block = *on* | *off*

実行ホストが SX-Aurora TSUBASA システムで、かつ、リクエストの実行を中断させる方法が **suspend** の場合、本機能を off に設定してもサスペンドされた VE ジョブがリジュームするまでは、新規投入された下位タイプリクエストのアサインは禁止します。これはリジュームするジョブの VE のリソースを確保するための動作です。

4.2　会話リクエスト

会話リクエストを即時に投入順に実行するには、スケジューリングインターバルとスケジューラマップ幅を短く設定します。スケジューリングインターバルの目安は 2 秒で、スケジューラマップ幅の目安は 3 秒です。なお、会話リクエストに対して、バッチリクエストと同様にバックフィル・スケジューリング機能もサポートしています。

- スケジューラマップ幅は、スケジューリングインターバルより1秒以上大きく設定する必要があります。
- 会話リクエストとバッチリクエストを別々のスケジューリングインターバルでスケジューリングする場合は、会話キューとバッチキューをそれぞれ別々のJobManipulatorで管理する必要があります。
- 会話リクエストをスケジューリングするための指定必須パラメータは、バッチリクエストと同様です。 詳細は、*3.1.5* リクエスト実行開始アルゴリズムを参照してください。

バッチリクエストと同様に、会話リクエストは実行ホストの使用可能 CPU 台数制限、使用可能 GPU 台数制限と使用可能メモリ制限を超過しないようにスケジューリングされます。

複数キュー、複数リクエストが存在する状態において、スケジューリング対象となる会話リクエストは以下のポリシーに従ってピックアップされます。

1. 投入されたキューのプライオリティが高い
2. スケジューリングプライオリティが高い
3. キューに投入された時間が早い
4. 上記 3 つの条件で決まらない場合は、上記条件が同じリクエストの中から任意の一つを選択します。

スケジューリングプライオリティは、次の通りです。

スケジューリングプライオリティ　＝　ユーザ定義ベースアップ

即時実行できない場合の動作は、

qmgr(1M)の**"set interactive_queue real_time_scheduling"**により *submit_cancel* に設定するか

wait に設定するかによって異なります。

- *submit_cancel*に設定した場合
  リクエストの投入がキャンセルされます。
- *wait*に設定した場合
  実行ホストの割り当てを待ちあわせます。アサインできた場合は、予定実行開始時刻に実行されます。アサインされない場合は、スケジューリングインターバルおきにスケジューリングされます。

会話キューの情報は、**sstat**(1)コマンドの**-Q** オプションで表示します。**-Q -i** オプションで会話キューのみの情報を表示します。**-f** オプションを付けることでキューの詳細情報を表示します。

```
#sstat -Q -i
[INTERACTIVE QUEUE]
 ==================
QueueName          RL  URL  UAL   TOT   EXC   QUE   ASG   RUN   EXT   SUD
---------------- ---------------- ----------------------------------------
iq               ULIM ULIM ULIM     0     0     0     0     1     0     0
```

バッチリクエストと同様に、会話リクエストの情報は、**sstat**(1)で表示します。**-f** オプションを付けることで会話リクエストの詳細情報を表示します。

会話リクエストには、バッチリクエストに対する基本的なスケジューリング機能をサポートしていますが、緊急タイプと特別タイプの会話キューおよびデッドラインスケジューリング機能はサポートしていません。

4.3　パラメトリックリクエスト

パラメトリックリクエストのサブリクエストを通常のバッチリクエストと同等に扱いスケジューリングします。下記の操作において、パラメトリックリクエストのサブリクエストを指定することによりサブリクエストの表示および操作が可能です。

また、パラメットリックリクエストを指定することによりパラメトリック内のサブリクエストに対して一括表示あるいは一括操作が可能です。指定方法については各コマンドを参照してください。

- パラメットリックリクエスト内のサブリクエストの表示（**sstat**(1)）
- スケジューリングプライオリティのユーザ定義ベースアップ値の設定（**smgr**(1M)コマンド

のサブコマンド**set request baseup_user_definition**）

- 再スケジューリング待ちの解除（**smgr**(1M) コマンドの**stop waiting_retry request**サブコマンド）
- 管理者によるサスペンド（**smgr**(1M) コマンドの**suspend request**サブコマンド）
- 管理者によるリジューム（**smgr**(1M) コマンドの**resume request**サブコマンド）

パラメトリックリクエストの詳細は、*NQSV* 利用の手引*[*操作編*]* を参照してください。

4.4　ワークフロー

ワークフロー内のリクエスト間の実行順序の時間的関係性(*)を維持してワークフロー内のリクエストをアサインします。 ワークフロー内のリクエストの再スケジューリング、エスカレーション、繰上げ実行に関しても同様です。

実行順序の時間的関係性には、下記の 2 種類があります。

- 前後実行関係
  リクエスト投入時に**qsub(1)**コマンドの**--after**オプションで指定する前後関係です。先行リクエストを実行終了してから後続リクエストを実行します。この関係を維持するために、先行リクエストをアサインした後、続いて（先行リクエストの実行を待たずに）後続リクエストをアサインします。
- 同時実行関係
  リクエスト投入時に**qsub(1)**コマンドの**--parallel**オプションで指定する同時実行関係です。複数のリクエストを同時に実行します（これらのリクエストを同時実行リクエストと呼ぶ）。この関係を維持するために、同時実行リクエストを同時刻にアサインします。

ワークフロー内のリクエストが再スケジューリングされる場合は、当該リクエストの同時実行リクエスト内のリクエストおよび後続リクエストも再スケジューリングされます。ワークフロー内のリクエストに対するスケジューリング（アサイン時/エスカレーション時）の優先度は、以下の点を除いて通常のバッチリクエストと同様です。

- 先行リクエストより後続リクエストのスケジューリング優先度が高い場合でも、先行リクエストを先にスケジューリングし、先行リクエストをアサインした直後に後方リクエストをスケジューリングします。
- 同時実行リクエストのスケジューリング優先度は、同時実行リクエスト内のリクエストにお

いて最も高いスケジューリング優先度とします。

- 後続リクエストの実行は、先行リクエストの実行結果ファイルを参照するため、前後リクエスト間では共有ファイルシステムによりファイル連携をする必要があります。
- 先行リクエストの実行結果ファイルを直接共有ファイルシステムではなく、ローカルDISKに書き、共有ファイルシステム上へステージアウトする場合は、一定の時間がかかる可能性があります。そのため、後続リクエストを先行リクエストの直後にアサインする場合は、後続リクエストが実行開始時に先行リクエストのステージアウトがまだ完了しておらず、先行リクエストの実行結果を参照できない可能性があります。後続リクエストが予定実行開始時刻において確実に実行するために、ステージアウト待ち時間[1]を設定することを推奨します。ステージアウト待ち時間は、**smgr**(1M) コマンドの**set queue wait_stageout**サブコマンドにより設定します。
- 同時実行リクエストは、同じスケジューリング優先度を持つため、同一タイプのキューに投入する必要があります。同時実行リクエストを異なるタイプのキューに投入した場合は、当該同時実行リクエストは、スケジューリングの対象外となります。
- 先行リクエストとしてパラメトリックリクエストを指定した場合、全サブリクエストの実行終了後後続リクエストをアサインします。先行リクエストとしてサブリクエストを指定した場合、サブリクエストのアサインに続いて、後続リクエストをアサインします。
- ハイブリッドリクエスト（**qsub**(1)コマンドでのリクエスト投入時に**--job-separator**あるいは**---**オプションを用いてジョブ毎に異なる資源を要求して投入したリクエスト）を同時実行リクエストとして投入した場合は、当該リクエストはスケジューリングされません。

4.5　実行開始時刻指定機能

4.5.1　実行開始時刻指定

qsub(1)コマンドのオプション **-s** にリクエストの実行開始予定時刻を指定します。これによりユーザが指定した時刻(時刻指定)にリクエストの実行を開始します。時刻指定で予約したリクエストは、その時刻で必ず実行される必要があるため、前方へのエスカレーションが可能であってもエスカレーション対象としません。

1.ステージアウト待ち時間…後方リクエストを前方リクエストの予定実行終了時刻の直後ではなく、一定の間隔を空けた時間にアサインします。この間隔をステージアウト待ち時間と呼びます。

時刻指定で予約したリクエストは、より上位のキューに投入されたリクエストによって割り込まれます。つまり通常リクエストは、緊急、または特別リクエストの割り込み対象となり、特別リクエストは、緊急リクエストの割り込み対象となります。

4.5.2　実行開始時刻指定できない場合の対応

時刻指定によりリクエストを投入したが、指定した時間にアサインできない場合の対応として、以下から選択することができます。

- 指定した時間にアサインできなかった旨をメッセージ出力しリクエストを削除する。
- 指定した時間帯に最も近い時刻でアサインする。

この設定は、**smgr**(1M) コマンドの **set treat_unbookable_request** サブコマンドにより設定します。

4.6　事前予約機能

本機能は、事前に指定区間のリソースを予約することができます。

事前予約機能には、保守用予約区間と、ジョブ実行用予約区間の 2 種類があります。

種別	説明
保守用予約区間	ハードウェアなどのメンテナンス時に使用します。保守用予約区間ではジョブは実行されません。 実行ホスト名、あるいは、ノードグループを指定して作成します。 この予約区間は、実行ホスト指定予約区間とも呼びます。
ジョブ実行用予約区間	ユーザのリクエストを確実に実行させるために使用します。 実行キュー指定で作成します。この予約区間は、実行キュー指定の予約区間とも呼びます。また、テンプレート指定の予約区間も作成可能です。

4.6.1　予約区間の作成

リソースと区間指定で予約します。予約区間には、予約時に予約区間 ID（0～9999）が付けられ、予約区間へのジョブ投入、や予約区間の削除等に使用されます。予約区間は、ATTACH された実行ホ

ストのみに対して予約でき、スケジューラマップ外でも予約可能です。

予約区間は、**smgr**(1M) コマンドの **create resource_reservation** サブコマンドで以下の項目を指定することにより作成します。

- 予約区間開始時刻（必須）
- 予約区間の期間　（必須）
- 実行キュー 又は　実行ホスト　（いずれかが必須）
- 実行ホスト数　（実行キュー指定時のオプション）
- 実行ホスト毎のCPU台数（実行キュー指定時のオプション）
- グループ名（実行キューと実行ホスト数指定時のオプション）
- 利用可能グループを指定します。グループ指定なしの場合は、全てのユーザが当該予約区間を利用可能です。

予約要求するには、NQSV 権限の操作員権限以上が必要です。

- 実行ホスト（**hostname**）を指定した場合、保守用予約区間となりジョブは実行できません。ジョブを実行するには、実行キュー（**queue**）を指定してください。
- 実行キューへのアクセス権がないユーザー／グループは予約区間を作成することができません。キューのアクセス制限の詳細については、*NQSV*利用の手引き *[*管理編*]* を参照してください。

予約可能条件

予約区間は、以下を除いた場所に作成できます。

- 実行ホスト上にジョブが既にアサインされている時間帯
- 実行ホスト上に既に予約区間が設定されている時間帯
- 実行ホスト上に既に省電力計画が設定されている時間帯

実行キュー指定の予約区間はリクエストを実行可能なホストおよび時間帯に作成可能です。
実行キュー指定の予約区間は、以下の２種類があります。

- 実行ホスト数指定で作成
- 実行ホスト数指定なしで作成（実行キューにバインドしている全実行ホストを予約する）

予約しようとする区間にリソースの空きがない場合は、エラーとなり予約区間は作成できません。

実行ホスト数指定なしで作成した場合、予約区間を作成した後に実行キューに追加された実行ホストも予約の対象となります。

バッチサーバで、リソース予約区間のアカウンティング機能を ON にしている場合は、予約区間作成時に実行ホスト数指定が必須です。さらに、アカウンティングサーバで予算管理を ON にし、当該キューに対して課金レートを設定している場合は、予約区間作成時に実行ホスト数指定に加えてグループ名の指定が必須です。必須の指定を省略した場合、エラーとなり予約区間は作成できません。

予約区間を作成した後、予約した実行ホストが障害やアンバインドなどにより運用から除外された場合は、当該実行ホストに対する予約は無効となります。残りの実行ホストは引き続き有効です。

実行キュー指定の予約区間ではリクエストを実行することができます。予約区間の期間は、Elapse マージンやステージアウト待ち時間を考慮して決定してください。

4.6.2　予約区間へのリクエスト投入

qsub(1)コマンドの**-y** オプションに予約区間 ID を指定することでキュー指定の予約区間にて予約したリソースを使ってリクエストを実行することができます。予約区間にジョブを投入するキューに対してアクセス権が必要です。予約区間の開始時刻が過ぎた場合でも、予約区間が終了するまでは新規リクエストを投入し実行することができます。予約区間では予約したリソースが空いている限りリクエストを複数実行可能です。また、**qsub**(1)コマンドの**-s** オプションでの開始時刻を指定したリクエスト投入も可能です。

テンプレート指定のリクエスト投入については、コンテナテンプレートの場合は、実行キュー指定の予約区間への投入は可能です。OpenStack テンプレートの場合は、投入できません。Docker を使った VE ジョブのプロビジョニングを行う場合は、テンプレート指定の予約区間ではなく、キュー指定の予約区間を使ってください。

予約区間でリクエストを実行するためには、Elapse 無制限機能を利用している場合でも経過時間(-l elapstim_req オプション)の指定が必要です。

qsub(1)コマンド実行時には以下のような、投入されたリクエストが予約区間で実行可能かどうかのチェックは行いません。リクエストを投入後、JobManipulator が予約区間にアサイン可能かどうか判断します。判断する観点は、計算リソース、予約区間指定キューとリクエスト投入キューが合致しているかどうか、**qsub**(1)コマンドの**-s** オプションやワークフローの指定による時刻の確認などです。

予約区間にリクエストがアサインされたかどうかは **sstat**(1)コマンドの **-f** オプションおよび **-B -f** オプションによって確認できます。アサインされた場合、Assigned と表示され、アサインされなかっ

た場合、Queued と表示されます。

4.6.3　予約区間の削除

予約区間の削除には **smgr**(1M)コマンドによる削除と、予約区間内のリクエスト終了による自動削除の 2 種類があります。

smgr(1M)コマンドによる削除

smgr(1M)コマンドの **delete resource_reservation** サブコマンドで予約区間 ID を指定して予約区間を削除することができます。

qsub (1)の**-y** オプションで指定して投入されたリクエストが存在する場合は予約区間を削除しません。ただし、強制削除オプション(**force**) を指定した場合は、その予約区間内に存在するリクエストを全て削除し、その上で予約区間を削除します。その際、JobManipulator は削除したリクエストのオーナに対してリクエストが削除されたことを通知するメールを送付します。

予約区間を削除するには、NQSV 権限の操作員権限以上が必要です。ただし、ジョブが存在する予約区間を削除するには、NQSV 権限の管理者権限が必要です。

予約区間内のリクエスト終了による自動削除

既定値ではキュー指定の予約区間に対して **qsub** (1)の**-y** オプションで指定して投入されたリクエストが存在しなくても予約区間は継続して有効です。

予約区間の開始時刻以降に予約区間内のリクエストが全て存在しなくなった時点で予約区間を削除する設定が可能です。設定は **smgr**(1M)コマンドの **set auto_delete_resource_reservation** サブコマンドで **on** を指定します。既定値は **off** です。

予約区間は終了時刻になると削除されます。JobManipulator は予約区間内に存在するリクエストを全て削除してから予約区間を削除します。その際、JobManipulator は削除したリクエストのオーナに対してリクエストが削除されたことを通知するメールを送付します。

また、実行ホスト指定の予約区間に対して、予約している実行ホストを DETACH するとき、当該実行ホストに関する予約情報を削除しますが、これによって予約区間内の実行ホストの予約情報が全てなくなる場合も、予約区間を削除します。

4.6.4　予約区間へのリクエストのスケジューリング

予約区間に投入されたリクエストは、予約区間内で一番早く実行可能な時刻に開始予定時刻を決定します。投入時刻がすでに予約区間内のときも、投入時刻から一番早く実行可能な時刻にアサインします。投入されたリクエストが予約区間内にアサインできない場合、そのリクエストは Queued 状態のままとなります。

4.6.5　予約区間情報表示機能

予約区間に関する情報を表示します。表示される情報は次のとおりです。

- 予約区間ID
- 予約区間名（詳細表示）
- グループ名(実行キュー指定の予約区間のみ)
- 実行キュー（実行キュー指定の予約区間のみ）
- 予約区間開始時刻
- 予約区間期間
- 予約区間の要求実行ホスト数
- 予約区間の実行ホスト当たりの要求CPU台数
- 予約区間が使用する実行ホスト名と実行ホストの予約状態（詳細表示）
- 予約区間を使用するリクエスト情報（詳細表示）

対応コマンド

予約区間の情報は **sstat**(1)コマンドの **-B** オプションで参照します。

```
# sstat -B
[Queue or Host Resource Reservations]
RES ID Start Time          End Time            NodeNum CPUNum  Queue
------ ------------------- ------------------- ------- ------- --------
    27 2007-10-12 13:00:00 2007-10-12 13:20:00       1       0 execque1
```

グループ指定の予約区間を表示するには、**-B** オプションと合わせて **--group** オプションを指定します。グループ指定の予約区間情報のみを表示します。グループ名も表示されます。

```
# sstat -B --group
[Queue or Host Resource Reservations]
RES ID Start Time          End Time            NodeNum CPUNum  Queue    GrpName
------ ------------------- ------------------- ------- ------- -------- --------
```

```
    28 2007-10-12 13:00:00 2007-10-12 13:20:00       1       0 execque1 groupA
```

実行キュー指定の予約区間の表示範囲は、下表に示すとおりです。

--group指定	ユーザ権限	表示範囲
あり	一般利用者権限 特別利用者権限	自グループの予約区間のみ
	グループ管理者権限	管理グループの予約区間のみ
	操作員権限 管理者権限	全てのグループ指定ありの予約区間
なし	一般利用者権限 特別利用者権限	グループ指定なしの予約区間
	グループ管理者権限	管理グループの予約区間のみ
	操作員権限 管理者権限	全ての予約区間

ただし、アクセス許可のない実行キューの予約区間は表示されません。実行ホスト指定予約区間は、全てのユーザに表示しますが、--group 指定の場合は、表示しません。

予約区間の詳細情報は **sstat**(1)コマンドの**-B,-f** オプションにより表示します。

```
# sstat -Bf
Resource Reservation ID: 27
   Resource Reservation Name = (none)
   Group Name = groupA
   Queue Name = execque1
   Reserve Start Time = 2007-10-12 13:00:00
   Reserve End Time   = 2007-10-12 13:20:00
   Execution Host Number = 1
   Reserve CPU Number by Host = ALL_CPU
   Reserved Hosts (HOST_NAME : STATUS):
     hostserver : ACTIVE
   Requests uses this reservation area:
     None
```

4.6.6　実行キュー指定の予約区間に対する課金

バッチサーバおよびアカウンティングサーバで、予約区間に対する課金を有効に設定している場合、

実行キューおよび実行ホスト数指定の予約区間に対して、予約作成時に予算超過チェックが行われ、予約区間の終了または削除時に予約アカウントファイルの生成とそれに基づく課金が行われます。

この場合、予約区間の作成は、**smgr**(1M) コマンドの **create resource_reservation** サブコマンドに **hostnum** と **group** の指定が必要です。予約区間に対する課金の設定方法については、*NQSV* 利用の手引/アカウンティング・予算管理編/ を参照してください。

予約区間に対する課金を有効にする場合、予約区間が存在しないことを確認したうえで実施してください。予約区間が存在する場合、当該予約区間を削除するか、当該予約区間が終了してから課金の設定を変更してください。

4.6.7　ヘルスチェックとクリーンアップ区間の設定

予約区間の前後にそれぞれ実行ホストの健全性チェック（ヘルスチェック）とクリーンアップ（スクラッチ領域の削除など）を実施する運用のために、実行キューと実行ホスト数指定予約区間の前後にそれぞれヘルスチェックを行う区間（PRE-MARGIN）とクリーンアップを行う区間（POST-MARGIN）（予約区間マージン）を設定する機能を提供します。

予約区間を作成するときに、「PRE-MARGIN + 予約区間長 + POST-MARGIN」の領域を確保します。PRE-MARGIN と POST-MARGIN の開始時刻においてそれぞれヘルスチェックとクリーンアップの実施を BSV 経由で要求を出し、実行ホスト側においてシステム管理者が事前に用意したヘルスチェックとクリーンアップ用スクリプトを実行します。

なお、ジョブは PRE-MARGIN と POST-MARGIN にはアサインできません。

PRE-MARGIN と POST-MARGIN の設定

PRE-MARGIN と POST-MARGIN は、smgr(1M) コマンドの **set queue resource_reservation** サブコマンドによりキュー単位に設定します。

```
# smgr -P m
Smgr : set queue resource_reservation pre-margin = <seconds> | post-margin = <seconds> <queue_name>
```

- 本値の単位は秒です。
- 既定値は、0（ヘルスチェック/クリーンアップを実施しない設定）です。
- 該当キューの予約区間が存在する場合は、現在の設定値より大きい値に変更できません。現在の設定値より小さい値に変更した場合は、すでに作成した予約に対しても適用します。

- 操作員権限以上の権限が必要です。

設定値は、sstat(1)コマンドの-Q -f オプションにより確認可能です。

```
#sstat -Q -f
Execution Queue: testq
    Queue Type            = Normal
    Schedule Time         = DEFAULT
     〈中略〉
    Wait_Stageout = OS
    Min Operation Hosts = 10240
    Reservation Margin = {
      Pre-margin  = OS
      Post-margin = OS
    }
:
```

ヘルスチェックとクリーンアップ用スクリプトの配置

本機能を利用する際、運用開始前に、実行ホスト上の/opt/nec/nqsv/sbin/extscr/に下記のスクリプトを配置しておいてください。

ヘルスチェック用スクリプト：/opt/nec/nqsv/sbin/extscr/HealthCheck
クリーンアップ用スクリプト：/opt/nec/nqsv/sbin/extscr/CleanUp

スクリプトを呼び出すとき、第一引数でキュー名を、第二引数でキューのタイプ(“batch”、あるいは、“interactive”)を渡すため、キューごとに処理を切り分ける場合は、これらの引数によりスクリプト内で切り分けてください。

- JobManipulatorを再起動するとき、「PRE-MARGIN + 予約区間 + POST-MARGIN」を再確保します。既に開始した予約を再確保できません。
- ヘルスチェックに失敗した実行ホストに対して、予約区間未開始であれば、実行キューにバインドしている実行ホストから代替の実行ホストを確保し、ヘルスチェックを実施しますが、予約開始時に、予約した実行ホストの中にヘルスチェックが成功していない実行ホストがある場合は、この予約は利用できないため、自動的に削除されます。

4.6.8　テンプレート指定の予約区間作成機能

本機能は実行ホストが SX-Aurora TSUBASA システムの場合は利用できません。実行キュー指定の予約区間作成機能において、テンプレートとテンプレートで起動するマシン数を指定した予約区間の作成が可能です。ここでマシン数とは、仮想マシン（VM）の台数、ベアメタルサーバの台数、またはコンテナ数です。JobManipulator は、テンプレート名およびマシン数で指定されたリソース量を指定された開始時刻に予約します。

単一予約区間内で同一実行ホスト上に複数の仮想マシン（VM）とコンテナを起動する予約区間を作成可能です。この場合、アサインポリシーに従って予約する実行ホストを選択します。詳しくは 2.3.8 アサインポリシーの設定 を参照してください。

テンプレート指定のリクエストを確実に実行させるための予約区間は、実行キュー指定の予約区間にテンプレートを指定して作成します。テンプレート指定の予約区間に投入するリクエストは、**qsub**(1) の**-y** オプションで予約区間の ID を指定して投入します。この場合、リクエストに指定するテンプレートが予約区間の指定テンプレートと一致する必要があります。それらが一致していない場合、リクエストはスケジューリングされません。

- プロビジョニングの場合、リクエストの実行毎にOSやコンテナの起動・停止を行うため、テンプレート指定の予約区間においては、ヘルスチェックとクリーンアップを行いません。予約作成時に指定したキューに対するPRE-MARGINやPOST-MARGINの設定は、テンプレート指定の予約区間においては無視されます。
- PRERUNNINGおよびPOSTRUNNING時のユーザEXITスクリプト（仮想マシン（VM）またはコンテナの場合）あるいは起動・停止スクリプト（ベアメタルサーバの場合）に、ヘルスチェック・クリーンアップ処理を組み込むことにより、リクエスト実行毎のヘルスチェック・クリーンアップが可能となります。その場合、Elapseマージン（仮想マシン（VM）またはコンテナの場合）あるいは起動/停止タイムアウト時間（ベアメタルサーバの場合）にヘルスチェック・クリーンアップにかかる時間を見積もり適切に設定してください。
- VEが定義されているコンテナテンプレートを指定したテンプレート指定の予約区間は作成できません。
- 予約区間作成時に指定されたコンテナテンプレートのVE数を**qmgr**(1M)にて1以上に変更した場合は、当該テンプレートで作成された予約区間が削除されます。変更後のテンプレートでジョブを予約区間に投入する場合は、実行キュー指定の予約区間を利用してください。

(1)　テンプレート指定の予約区間の作成

テンプレート指定の予約区間は、**smgr**(1M)コマンドの **create resource_reservation** サブコマンドで開始時刻、期間、キュー名、テンプレート名、テンプレートで起動するマシン数を指定して作成します。

```
#smgr -P m
Smgr: create resource_reservation starttime = <start_time> blocktime = <block_time> queue = <queue_name> template =
<template_name> machinenum = <machine_num> [name = <resource_reservation_name>] [group = <group_name>]
```

- template_nameにテンプレート名を指定します。
- machine_numに指定されたテンプレートで起動するマシン数を1から10240の値で指定します。
- template_nameとmachine_num の指定と実行ホスト数（hostnum）あるいは実行ホスト毎のCPU数（cpunum）との同時指定はできません。
- group_nameを指定した場合、指定グループに属するユーザのみが当該予約区間を利用できます。
- 操作員権限以上の権限が必要です。

予約しようとする区間にリソースの空きがない場合は、予約時にエラーとなります。

(2) テンプレート指定の予約区間の表示

テンプレート指定の予約区間のサマリ情報は、**sstat**(1)コマンドの-B オプションで表示します。

```
$sstat -B
[Template Resource Reservations]
RES ID Start Time          End Time            Template MacNum Queue
------ ------------------- ------------------- -------- ------ --------
     2 2016-03-30 18:00:00 2016-03-30 19:00:00 ostmp1        6 tmp_que
```

-B オプションと合わせて、--group オプションを指定した場合は、グループ指定の予約区間情報のみを表示します。

```
$sstat -B --group
[Template Resource Reservations]
RES ID Start Time          End Time            Template MacNum Queue    GrpName
------ ------------------- ------------------- -------- ------ -------- --------
     3 2016-03-30 18:00:00 2016-03-30 19:00:00 ostmp2        2 tmp_que  group1
```

-B オプションは、テンプレート指定でない予約区間情報も表示します。**-B** オプションと合わせて、**--template** オプションを指定した場合は、テンプレート指定の予約区間情報のみ表示可能です。

-B -f オプションを指定した場合は、テンプレート指定の予約区間の詳細情報を表示します。

```
$sstat -B -f
Resource Reservation ID: 1
:
Reserve End Time   = 2016-07-27 15:00:00
    Reserve Template = vm_1
    Reserve Machine Number = 6
  Reserved Machines (HOST_NAME : STATUS):
      192.168.0.1 : ACTIVE
      192.168.0.1 : ACTIVE
      192.168.0.2 : ACTIVE
      192.168.0.2 : ACTIVE
      192.168.0.3 : ACTIVE
      192.168.0.4 : ACTIVE
  Requests uses this reservation area:
:
```

(3)　テンプレート指定の予約区間へのリクエストの投入

テンプレート指定の予約区間へのリクエストを投入は、**qsub**(1)コマンドの-y オプションで予約 ID を指定し、**--template** オプションで予約作成時に指定したテンプレートを指定して行います。JobManipulator はリクエストの 1 ジョブに予約区間内の 1 マシンを割り当てます。

```
$qsub -y <予約ID> --template=<テンプレート>
```

テンプレート指定の予約区間に投入されたリクエストは、**sstat**(1)コマンドのほかに **sstat**(1)コマンドの**-B -f** オプションでも表示します。

```
$sstat -B -f
Resource Reservation ID: 1
:
Requests uses this reservation area:
  RequestID       ReqName  UserName Queue                Pri STT PlannedStartTime
  --------------- -------- -------- -------- ---------- --- -------------------
```

```
1               sleep    user      vmque     500.2443/  0.5002 QUE -:
```

(4)　テンプレート指定の予約区間に対する課金

バッチサーバおよびアカウンティングサーバで、テンプレート指定の予約区間に対する課金を有効に設定している場合、テンプレート指定の予約区間に対して、予約作成時に予算超過チェックが行われ、予約区間の終了または削除時に予約アカウントファイルの生成とそれに基づく課金が行われます。

この場合、テンプレート指定の予約区間の作成は、**smgr**(1M) コマンドの **create resource_reservation** サブコマンドに group の指定が必要です。テンプレート指定の予約区間に対する課金の設定方法については、*NQSV* 利用の手引[アカウンティング・予算管理編] を参照してください。

4.7　ShareDBマージ機能

実行ホストの種類ごとにジョブの運用ポリシーが異なる場合など、複数の JobManipulator を運用することがあります。このとき、全ての実行ホストで一貫してフェアシェア・スケジューリングするため、ShareDB マージ機能の利用を推奨します。

ShareDB マージ機能とは、JobManipulator 単位に保持された ShareDB（各ユーザのシェア配分や使用実績等を格納するファイル)の使用実績値をマージし､マージ後のデータを利用する機能です。この機能を使うと、全 JobManipulator で同じマージ後のデータを保持します。例えば、あるクラスタの使用実績と別のクラスタの使用実績をマージした ShareDB のデータをスケジューリングプライオリティ計算に反映させることなどが行えます。

4.7.1　ShareDB マージ機能概要

JobManipulator 毎に蓄積されている使用実績値(ShareDB)を収集してユーザ毎にマージします。マージ後の実績値は、今後のプライオリティ計算に利用する使用実績値として各 JobManipulator の ShareDB へ格納します。マージされる使用実績は、ShareDB 内に蓄積される全ての使用実績値（以下の 4 項目）が対象となります。

- 経過時間
- CPU台数
- メモリ使用量
- リクエストプライオリティ
- VE台数

この実績値の集計には、JobManipulator 単位にマージ率（掛け率）を指定可能です。例えばマージ処理時に各スケジューラの使用実績値を 10：1 の比率でマージするといった運用が可能となります。

[例] ある1つのシステム環境下で、各スケジューラの使用実績値を以下の 2 つの比率でマージした場合の違いを示します。

A. クラスタ1：クラスタ2 = 5：1
B. クラスタ1：クラスタ2 = 2：1

クラスタ1のリソースを使用するほど、Aでの運用下の方がBでの運用下よりも、マージ後の使用実績値へ大きい値を反映させるなどの重み付けができます。

マージ処理では、以下のような運用下でのスケジューラを多様に指定することができます。(以下の場合以外でも指定可能です。)

- 1つのホスト上で複数のスケジューラを運用している場合
- 1つのホスト上で1つのスケジューラを運用し、別のホストで複数のスケジューラを運用している場合
- 複数のホスト上で複数のスケジューラを運用している場合

下記は ShareDB マージ処理のイメージ図です。

マージ処理例

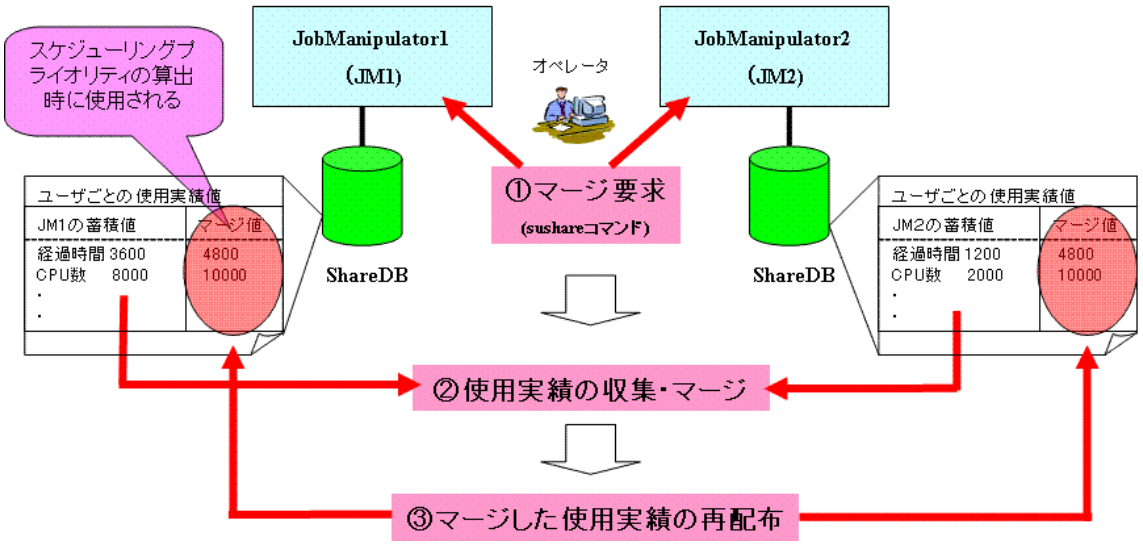


図 4-2：ShareDB マージ処理のイメージ図

* **sushare**(1)コマンドの**-M**オプションを一度実行すると、①～③が一括で行われます。
① オペレータは**sushare**(1)コマンドを使用してマージ対象となるJobManipulatorマージ要求を行います。 各JobManipulatorで蓄積されたShareDBの実績値が収集されマージされます。
② マージされた値はマージ値として各JobManipulatorに渡され、ShareDBに登録されます。(ローカル値とマージ後実績値の両方に実績値を加算します。)
③ 各JobManipulatorはスケジューリングプライオリティを算出する際、マージ値を使用します。

- **sushare**(1)コマンドの**-M**オプションを実行すると、各スケジューラのローカル実績値が収集されます。
- 設定ファイル（詳細は*4.7.4 ShareDBマージ設定ファイル* 参照）に従った実績値の計算、マージ後のアップデートを一括で行います。
- ローカルとマージ後の値は両方とも、JobManipulatorを起動しているホスト上のデータベース/var/opt/nec/nqsv/nqs_jmd/database /<scheduler_id>/pu_dbに保存されています。

- マージ後、各クラスタではローカル値とマージ後実績値の両方に実績値を加算していきます。

4.7.2　ShareDB マージ設定方法

使用実績値のマージ処理は、**sushare**(1)コマンドの**-M** オプションで設定します。

```
# sushare -Pm -M [マージ設定ファイル名] [-l[ログファイル名]]
```

- ログファイルを指定する際に-lオプションの後にログファイル名を指定します。
- ログファイルは、**sushare**(1)コマンド**-M**オプションを実行するホスト上に蓄積されます。
- **sushare**(1)コマンド**-M**オプションは、設定ファイルで指定された各JobManipulatorとTCP/IPで接続し、ShareDBファイル上のデータをマージします。
- **sushare**(1)コマンドがインストールされている任意のホストで**sushare**(1)コマンド**-M**オプションの実行が可能です。
- 設定ファイル（デフォルトファイル名：/etc/opt/nec/nqsv/jm_merge_sharedb.conf）で指定された内容に従ってShareDBファイル上のデータのマージを実行します。
- **sushare**(1)コマンドが直接ファイルを読み込むため、設定ファイルは**sushare**(1)を実行するホストに置く必要があります。
- マージ率を運用中に変更した場合は、次回の**sushare**(1)コマンド**-M**オプションの実行に反映されます(変更時点でのマージ後実績値には反映しません)。
- マージした実績値をShareDBに書き戻す際、ShareDB内に該当ユーザが存在しない場合は無視されます(該当ユーザが新たに作成される事はありません)。
- 使用実績値のマージ処理の設定には操作員権限が必要です。

マージ処理は、**sushare**(1)コマンドの**-M** オプションが実行されなければ行われません。定期的にマージ処理を行う場合は、cron 等を利用してください。

[例] 以下は **sushare**(1)コマンドの実行例です。**-M** オプションと、-l オプションの後にログファイル名"test2.log"を指定し、最後に設定ファイル名"test2"を指定しています。マージの実施時に下記のようなイメージが標準出力として出力されます。

```
#sushare -P m -M -ltest2.log test2
sushare : 7 records were acquired from host1(sch_id=2)
```

```
↑サーバ"host1"スケジューラID2番から7レコードを読み込んだことを表します。
sushare : 5 records were acquired from host1(sch_id=1)
sushare : 7 records is transmitted to all hosts.
sushare : Completed.  ←マージ処理が完了
```

マージ処理の詳細はログファイル(デフォルトファイル名：nqs_jmd_sharedb_merge.log）に出力されます。

以下はログファイルの出力イメージです。赤字は説明書きになります。

```
Tue Dec 11 20:44:27 2007 sushare：  host1(2), user1[(none)], CPU=0.000000, VE=0.000000, MEMORY=0.000000,
ELAPSE=0.000000, PRIOLITY=0.000000
→上記はサーバ"host1" スケジューラID2番から　user1（アカウントコード無し）のデータを取得したことを意味します。値はローカル値です。

Tue Dec 11 20:44:27 2007 sushare：7 records were acquired from host1(sch_id=2)
→上記はサーバ"host1" スケジューラID2番からトータル7レコードを読み込んだことを意味します。

Tue Dec 11 20:44:27 2007 sushare：  user1[(none)], CPU=6722.278397, VE=5213.331234, MEMORY=6083.999465,
ELAPSE=6722.278397,PRIOLITY=6083.999465
→上記はuser1(アカウントコード無し）のデータがマージされたことを意味します。値はマージ後の値です。
```

* 以下の場合、エラーとなり、**sushare**(1)コマンドによるマージ処理が実行されません。

- 設定ファイルに存在しないファイル名が指定された場合
- 設定ファイル中に指定されたホストが起動されていない場合
- 設定ファイル中に指定されたホストは起動しているがJobManipulatorが起動されていない場合

sushare(1)コマンドによるマージ処理はスケジューリングを止めること無く JobManipulator 運用中に実施可能です。システム障害などによりマージ処理が途中で終了した場合は、次回の sushare -M 実行時に前回未処理だった部分も含めて、マージが行われます。

4.7.3　ShareDB の使用実績値の表示

ShareDB の実績値は **sushare**(1)コマンドの以下 2 つのオプションを使用して表示します。

- **-S** オプション　（マージ後実績値）

- **-L** オプション （マージ後実績値とローカル実績値）

以下は各実行例です。**sushare**(1)コマンドの**-S**（大文字）オプションでは、スケジューラの各実績値が、マージ後実績値で表示されます。

```
[Group Name : TOP_GROUP]
 User      Acctcode Share  PU_cpunum (%)       PU_venum (%)          PU_memsz (%)        PU_elapstim (%)        PU_reqpri (%)
-------------------------------------------------------------------------------------------------------------------------------
*nec       none     0.333     4.190M (50.002)   4.190M (50.002)     3.996M (50.002)   1163:58:00 (50.002)     4.190M (50.002)
*nqs       none     0.667     4.190M (49.998)   4.190M (49.998)     3.996M (49.998)   1163:53:44 (49.998)     4.190M (49.998)


[Group Name : nec]
 User      Acctcode Share  PU_cpunum (%)       PU_venum (%)          PU_memsz (%)        PU_elapstim (%)        PU_reqpri (%)
-------------------------------------------------------------------------------------------------------------------------------
 necusr1   none     0.167     2.095M (25.001)   2.095M (25.001)     1.998M (25.001)   581:59:01 (25.001)      2.095M (25.001)
 necusr2   none     0.167     2.095M (25.001)   2.095M (25.001)     1.998M (25.001)   581:58:58 (25.001)      2.095M (25.001)


[Group Name : nqs]
 User      Acctcode Share  PU_cpunum (%)       PU_venum (%)          PU_memsz (%)        PU_elapstim (%)        PU_reqpri (%)
-------------------------------------------------------------------------------------------------------------------------------
 nqsusr1   none     0.167     1.048M (12.500)   1.048M (12.500)  1022.954K (12.500)   290:58:24 (12.500)      1.048M (12.500)
 nqsusr2   none     0.167     1.048M (12.500)   1.048M (12.500)  1022.955K (12.500)   290:58:25 (12.500)      1.048M (12.500)
 nqsusr3   none     0.167     1.048M (12.500)   1.048M (12.500)  1022.956K (12.500)   290:58:26 (12.500)      1.048M (12.500)
 nqsusr4   none     0.167     1.048M (12.500)   1.048M (12.500)  1022.956K (12.500)   290:58:26 (12.500)      1.048M (12.500)
```

sushare(1)コマンドの**-L** オプションでは、スケジューラの各実績値が、マージ後実績値/ローカル実績値で表示されます。

```
[Group Name : TOP_GROUP]
 User      Acctcode Share          PU_cpunum (%)                  PU_venum (%)                   PU_memsz (%)                   PU_elapstim (%)                  PU_reqpri (%)
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
*nec       none     0.333    4.190M/   4.190M (50.002/50.002)   4.190M/   4.190M (50.002/50.002)   3.996M/   3.996M (50.002/ 50.002)  1163:58:19/ 1163:58:19 (50.002/50.002)   4.190M/   4.190M (50.002/50.002)
*nqs       none     0.667    4.190M/   4.190M ( 49.998/ 49.998) 4.190M/   4.190M ( 49.998/ 49.998) 3.996M/   3.996M ( 49.998/ 49.998) 1163:54:03/ 1163:54:03 ( 49.998/ 49.998) 4.190M/   4.190M ( 49.998/ 49.998)


 [Group Name : nec]
  User      Acctcode Share          PU_cpunum (%)                  PU_venum (%)                   PU_memsz (%)                   PU_elapstim (%)                  PU_reqpri (%)
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
necusr1   none     0.167    2.095M/   2.095M ( 25.001/ 25.001)   2.095M/   2.095M ( 25.001/ 25.001)   1.998M/   1.998M ( 25.001/ 25.001)  581:59:10/  581:59:10 ( 25.001/ 25.001)   2.095M/   2.095M ( 25.001/ 25.001)
necusr2   none     0.167    2.095M/   2.095M ( 25.001/ 25.001)   2.095M/   2.095M ( 25.001/ 25.001)   1.998M/   1.998M ( 25.001/ 25.001)  581:59:08/  581:59:08 ( 25.001/ 25.001)   2.095M/   2.095M ( 25.001/ 25.001)
```

```
[Group Name : nqs]

User     Acctcode Share          PU_cpunum (%)                PU_venum (%)                PU_memsz (%)                          PU_elapstim (%)                PU_reqpri (%)

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

nqsusr1  none     0.167    1.048M/   1.048M ( 12.500/ 12.500)   1.048M/   1.048M ( 12.500/ 12.500)  1022.958K/ 1022.958K ( 12.500/ 12.500)   290:58:29/   290:58:29 ( 12.500/ 12.500)   1.048M/   1.048M ( 12.500/ 12.500)

nqsusr2  none     0.167    1.048M/   1.048M ( 12.500/ 12.500)   1.048M/   1.048M ( 12.500/ 12.500)1022.960K/ 1022.960K ( 12.500/ 12.500)   290:58:30/   290:58:30 ( 12.500/ 12.500)   1.048M/   1.048M ( 12.500/ 12.500)

nqsusr3  none     0.167    1.048M/   1.048M ( 12.500/ 12.500)   1.048M/   1.048M ( 12.500/ 12.500) 1022.961K/ 1022.961K ( 12.500/ 12.500)   290:58:31/   290:58:31 ( 12.500/ 12.500)   1.048M/   1.048M ( 12.500/ 12.500)

nqsusr4  none     0.167    1.048M/   1.048M ( 12.500/ 12.500)   1.048M/   1.048M ( 12.500/ 12.500)  1022.961K/ 1022.961K ( 12.500/ 12.500)   290:58:31/   290:58:31 ( 12.500/ 12.500)   1.048M/   1.048M ( 12.500/ 12.500)
```

* **-s**(小文字)オプションで、JobManipulator のスケジューラ ID を指定します(-s オプションで指定しない場合はデフォルトのスケジューラになります)。

（詳細は NQSV 利用の手引[リファレンス編]の **sushare**(1)コマンド**-s** オプションを参照してください。）

4.7.4　ShareDB マージ設定ファイル

マージ処理は **sushare**(1)コマンドの設定ファイル（デフォルトファイル名：/etc/opt/nec/nqsv/jm_merge_sharedb.conf）に従って実行されます。

設定ファイルに指定可能な項目は以下のとおりです。

- **コメント行**

  '#'で始まる行は、コメントです。

- **HOST行**

  JobManipulatorを実行しているホストのホスト名またはIPアドレスを指定します。HOST行は、スケジューラIDやマージ率より前に指定する必要があります。

- **SCH_ID行**

  JobManipulatorのスケジューラID

- **MERGE_RATE行**

  マージ率。ローカル実績値にマージ率を掛けた値がマージされます。

- **CPU行**

  CPUマージ率。CPUのローカル実績値にマージ率を掛けた値がマージされます。本項目は省略可能であり、省略された場合は、MERGE_RATEが使用されます。

- **ELAPSE行**

  ELAPSEマージ率。ELAPSEのローカル実績値にマージ率を掛けた値がマージされます。本項目は省略可能であり、省略された場合は、MERGE_RATEが使用されます。

- **MEMORY行**

MEMORYマージ率。MEMORYのローカル実績値にマージ率を掛けた値がマージされます。本項目は省略可能であり、省略された場合は、MERGE_RATEが使用されます。

- **PRIORITY行**
  PRIORITYマージ率。PRIORITYのローカル実績値にマージ率を掛けた値がマージされます。本項目は省略可能であり、省略された場合は、MERGE_RATEが使用されます。
- **VE行**
  VEマージ率。VEのローカル実績値にマージ率を掛けた値がマージされます。本項目は省略可能であり、省略された場合は、MERGE_RATEが使用されます。

マージ率は掛け率です。リソース毎のマージ率も全て掛け率となります。
[例] 以下は掛け率を使用した CPU 実績値の計算例です。

スケジューラ A		スケジューラ B	
CPU 実績値	100	CPU 実績値	150
CPU マージ率	2	CPU マージ率	3

上記の場合、100 * 2 + 150 * 3 となり、マージ後の値は 650 になります。

[例] 以下はクラスタ 1 およびクラスタ 2 を管理する 2 つの JobManipulator をマージ対象とした設定ファイルの例です。

前半がクラスタ 1 用設定、後半がクラスタ 2 用設定で、マージ比率は 10:1 です。

```
# JobM for cluster1
HOST=hostA
SCH_ID=1
MERGE_RATE=10

# JobM for cluster2
HOST=hostB
SCH_ID=11
```

sushare(1)コマンドが直接ファイルを参照するため、設定ファイルは **sushare**(1)コマンドを実行するホストに置く必要があります。

* 以下の場合、エラーとなり、エラー行の内容、エラー種別を出力しマージ処理は行われません。

- HOST行が無い場合又はHOST行がスケジューラIDやマージ率より後に指定された場合
  エラーメッセージ："HOST" is not specified.
- SCH_ID行が無い場合
  エラーメッセージ："SCH_ID" is not specified.
- MERGE_RATE未設定でCPU行が無い場合
  エラーメッセージ："CPU" is not specified.
- MERGE_RATE未設定でELAPSE行が無い場合
  エラーメッセージ："ELAPSE" is not specified.
- MERGE_RATE未設定でPRIORITY行が無い場合
  エラーメッセージ："PRIORITY" is not specified.
- MERGE_RATE未設定でMEMORY行が無い場合
  エラーメッセージ：" MEMORY" is not specified.
- MERGE_RATE未設定でVE行が無い場合
  エラーメッセージ："VE" is not specified.
- SCH_IDに数値以外の値が指定された場合
  エラーメッセージ：Only the numerical value can be specified for "SCH_ID"
- CPUに数値以外の値が指定された場合
  エラーメッセージ：Only the numerical value can be specified for "CPU"
- ELAPSEに数値以外の値が指定された場合
  エラーメッセージ：Only the numerical value can be specified for " ELAPSE"
- PRIORITYに数値以外の値が指定された場合
  エラーメッセージ：Only the numerical value can be specified for "PRIORITY"
- MEMORYに数値以外の値が指定された場合
  エラーメッセージ：Only the numerical value can be specified for " MEMORY"
- MERGE_RATEに数値以外の値が指定された場合
  エラーメッセージ：Only the numerical value can be specified for "MERGE_RATE"
- VEに数値以外の値が指定された場合
  エラーメッセージ：Only the numerical value can be specified for "VE"
- 無効な行が書かれていた場合
  エラーメッセージ：Unknown key word.

- HOST行に指定したHOSTのアドレス変換ができなかった場合
  エラーメッセージ：Unknown host name.
- 複数の同一ホストが指定されていた場合
  エラーメッセージ：Host is doubly specified.
- SCH_ID行が二重定義されていた場合
  エラーメッセージ："SCH_ID" is doubly specified.
- CPU行が二重定義されていた場合
  エラーメッセージ："CPU" is doubly specified.
- ELAPSE行が二重定義されていた場合
  エラーメッセージ：" ELAPSE" is doubly specified.
- PRIORITY行が二重定義されていた場合
  エラーメッセージ：" PRIORITY" is doubly specified.
- MEMORY行が二重定義されていた場合
  エラーメッセージ：" MEMORY" is doubly specified.
- VE行が二重定義されていた場合
  エラーメッセージ："VE" is doubly specified.

4.8　Elapse無制限機能

Elapse 無制限機能とは、経過時間制限値を指定しない(=無制限(Unlimited)が指定された)リクエストのスケジューリングを可能にする機能です。

* 経過時間制限を指定する場合については、第 *3*章　*3.1.5* リクエスト実行開始アルゴリズムを参照してください。
* ジョブ毎の同時実行 CPU 台数(qsub コマンドの-l オプション、サブオプション cpunum_job)の指定は 必須です。

Elapse 無制限機能を有効にしたスケジューリングでは、以下の運用ポリシーになります。

- 経過時間制限値が指定されたリクエストもスケジューリングされます。
- 経過時間制限値が指定されたリクエストの後方には経過時間制限値指定あり/無制限のリクエストをアサイン可能です。
- 経過時間制限=無制限のリクエストの後方にはリクエストはアサインされません。

- 経過時間無制限が指定されたリクエストの実行が完了した場合、リソースは開放され、他のリクエストがアサインされます。

4.8.1　Elapse 無制限機能設定方法

Elapse 無制限機能を有効にする(経過時間制限=無制限のリクエストをスケジューリングする)には、**smgr**(1M)コマンドの **set use_elapstim_unlimited** サブコマンドにて設定します。

```
# smgr -P m
Smgr : set  use_elapstim_unlimited = on | off
```

- onが指定された場合は、Elapse資源制限に無制限が指定されたリクエストのスケジューリングが可能となります。
- 初期設定値は、off（Elapse制限有り）です。
- スケジューラ単位で設定が可能です。

Elapse 無制限機能の設定には、操作員権限が必要です。

本設定により Elapse 資源制限を on にした場合、既に投入されている Elapse=無制限のリクエストについてもスケジューリングされます。Elapse 資源制限に無制限を off にした場合には、既に投入されている Elapse=無制限のリクエストでアサインされていないものはスケジューリングされなくなります。既にアサインされているリクエストについては、そのままアサインされた状態となりスケジューリングされた通りに実行されます。

予約区間の設定がされているホストには、予約区間機能が優先され、無制限リクエストはアサインされません。

4.8.2　Elapse 無制限設定の表示方法

Elapse 無制限の on/off(可能/不可能)の設定値は、**sstat**(1)コマンドの**-S, -f** オプションにより表示します。

```
$ sstat -S -f
JobManipulator Server Host: bsv.nec.co.jp
    JobManipulator Version   = R1.00
    JobManipulator Status    = Active
    Scheduler ID             = 5
    Schedule Interval        = 60S
```

```
    Schedule Time            = 604800S
    Use Elapse Unlimited     = ON
:
```

4.9　実装CPU/GPU数追随機能

CPU/GPU 障害や復旧等、実行ホストの実装 CPU/GPU 数が変化した場合、JobManipulator は、最新の実装 CPU/GPU に基づいてスケジューリングを行います。実装 CPU/GPU 数が変化した場合、スケジューラマップへアサインされていたリクエストは再スケジューリングされ、最新の実装 CPU/GPU 数に基づいてスケジューリングが実施されます。再スケジューリングの対象となるリクエストは以下のとおりです。

- 実装 CPU/GPU 数に変化のあった実行ホストにアサインされているリクエストのうち、ジョブが実行中以外の全てのリクエスト
- 実装 CPU/GPU 数に変化のあった実行ホストにアサインされているリクエストがマルチノードジョブであった場合、その後方にアサインされている全てのリクエスト

再アサイン対象となったリクエストは、再アサイン前にアサインされていた時間が早いリクエストからスケジューラマップへアサインされます。

本機能はバッチサーバ設定の Load Interval に依存します。
Load Interval が 0 に設定されていた場合、本機能は動作しません。
実装 CPU 数追随機能を有効にするには、バッチサーバの Load Interval を 0 以上に設定する必要があります。バッチサーバの Load Interval は実装 CPU/GPU 数の更新タイミングとなります。
従って、Load Interval を大きな値にすると、実装 CPU/GPU 数の更新が遅くなり、実装 CPU/GPU 数変更からスケジューリングに反映されるまで時間が掛かることになります。
バッチサーバの Load Interval については、 *NQSV* 利用の手引*[*管理編*]* を参照してください。

4.10 フェール・オーバ対応

JobManipulator は、CLUSTERPRO に対応しています。CLUSTERPRO によって JobManipulator サーバホストを二重化することによって、スケジューラをダウンさせることなく、継続スケジューリングすることができます。

CLUSTERPRO が提供する仮想 IP アドレスを指定するには、JobManipulator(nqs_jmd)起動時に **-a** オプションを使用します。このオプションによって IP アドレスが指定された場合、JobManipulator は以下の動作を行います。

sstat(1)コマンドで表示される JobManipulator サーバホスト名がこの IP アドレスに対応するホスト名となります。

フェール・オーバ時、実行中リクエストは実行継続し、実行開始待ちのリクエストは一旦開始予定時刻がクリアされ、再スケジューリングされます。

4.11 ノード障害時の対応

ジョブサーバにジョブがアサインされた状態でノード障害（ジョブサーバのリンクダウン）が発生した場合、その障害ノードにおけるジョブをクリアし、再スケジューリングできるようにします。

4.11.1 ノード障害時の再スケジューリング

ノード上に存在するリクエスト種別には以下のものがあります。

1. 実行中のリクエスト
2. 開始待ちのリクエスト
3. ステージイン中のリクエスト

これらのリクエストがノード上に存在する状態で、ノード障害によりノードダウンした場合、これらのリクエストは以下のように再スケジューリングされます。

実行中のリクエストについては「4.11.2 実行中ジョブの強制リラン機能」を参照してください。

実行開始待ちのリクエストは、ジョブのパージを行った後 QUEUED 状態となり、再スケジューリングされます。

障害発生後のマップ維持機能を使用することにより、ノード障害発生時に一定時間後に実行開始するリクエストの開始予定時刻を維持し、開始予定時刻が変更となるリクエストを最小限に抑えること

も可能です。障害発生後のマップ維持機能については「4.11.4 障害発生後のマップ維持機能」を参照してください。

4.11.2 実行中ジョブの強制リラン機能

実行中のジョブが存在するノードでノード障害が発生した場合、ジョブはストール状態になります。このストールしたジョブを、スケジューラの設定により強制的にリラン可能です。ストールした実行中ジョブはリランを行うことで、再スケジューリングされます。

実行中ジョブの強制リランの設定は、**smgr**(1M)コマンドの **set forced_rescheduling** サブコマンドで設定します。 設定には操作員以上の権限が必要です。 デフォルトは OFF であり、ノードの復旧を待って再スケジューリングされます。

強制リランの対象となるリクエストの状態は以下のとおりです。

・RUNNING
・SUSPENDING
・SUSPENDED
・RESUMING
・POST-RUNNING

4.11.3 BSV との接続時の強制リランウエイト機能

JobManipulator とバッチサーバとの接続時に実行中のストールジョブが存在した場合、JobManipulator はストールジョブの強制リランウエイト時間(デフォルト 10 分)が経過するまで待ち合わせます。

ウエイト時間中にジョブのストール状態が解除されると、ジョブの強制リランを行いません。ウエイト時間経過後、ジョブのストール状態が継続していると、実行中ジョブの強制リラン機能が有効な場合、ジョブのリランを実施します。

ウエイト時間は、コンフィグファイルでカスタマイズします。カスタマイズを実施する場合には、コンフィグファイル(/etc/opt/nec/nqsv/nqs_jmd.conf)に以下の行を追加してください。

```
JM_RERUNWAIT: 600 #起動時の強制リランウエイト時間(指定単位は秒)
```

本機能は JobManipulator とバッチサーバとの接続時のみ有効です。接続完了後の運用中にストールを検出したジョブは、実行中ジョブの強制リラン機能が有効な場合、直ちに強制リランが実施されます。

4.11.4 障害発生後のマップ維持機能

(1) 機能概要

ノード障害発生後、早急にメンテナンスを実施し、障害ノードを運用へ復帰させる場合、メンテナンスに要する時間後のマップを維持する運用が考えられます。そのために、ノード障害発生時に一定時間以降のマップを維持する機能を提供します。その時間までの実行中リクエストは再スケジューリングされますが、それ以降に実行開始するリクエストの開始予定時刻を維持し、開始予定時刻が変更となるリクエストを最小限に抑えます。

(2) マップ維持の設定

障害発生後のマップ維持機能は、**smgr**(1M) コマンドの **set keep_forward_schedule** サブコマンドによりスケジューラ単位に設定可能です。

```
#smgr -P m
Smgr: set keep_forward_schedule = <second>
```

- 操作員権限以上の権限が必要です。
- 障害発生時刻からsecond秒後の時刻以降のマップを維持します。
- secondが0の場合は、マップを維持せず、該当ノードにアサインされているリクエストを再スケジューリングします。
- 本設定の初期値は0とします。

本設定時間を経過しても障害が発生したノードが ACTIVE 状態にならない場合は、そのノードにアサインされているリクエストを再スケジューリングします。

マップ維持の設定の表示方法

マップ維持の設定状況は、**sstat**(1)コマンドの**-S -f** オプションで表示可能です。

```
$ sstat -S -f
JobManipulator Server Host: bsv.nec.co.jp
    JobManipulator Version   = R1.00
    JobManipulator Status    = Active
    Scheduler ID             = 1
            :
    Keep Forward Schedule    = 3600S
 :
```

4.11.5 障害遭遇リクエストの最優先実行機能

本機能は、多くのリクエストの実行やシステム全体の稼働率に大きく影響を与えるため、利用することは推奨いたしません。利用する場合は、必ず注意事項を十分理解した上で、利用してください。

(1) 機能概要

実行ノードの障害発生時に、障害に遭遇した実行中のリクエストをリランし、後方にアサインして再実行することができます。しかし、多くのリクエストを実行している運用では、障害遭遇リクエストが再び実行を開始するまでに長い時間を要する可能性があります。そこで、本機能を利用することで、障害遭遇した実行中リクエストを最優先で再スケジューリングし、再実行することができます。本機能を利用すると、障害遭遇したリクエストが再実行するまでの間、他の実行予定のリクエストは開始予定時刻が取り消され、実行ホストに空きがある場合でも開始されません。本機能は、多くのリクエストの実行、および、システム全体の稼働率に大きく影響を与えますので利用することは推奨いたしません。

(2) 機能の設定

障害遭遇リクエストの最優先実行機能を利用するには、JobManipulator のコンフィグファイル（既定値：/etc/opt/nec/nqsv/nqs_jmd.conf）に下記の行を記載する必要があります。記載後、JobManipulator を再起動することで有効になります。

```
HWFAILURE_OVERTAKE:ON
```

上記の記述がない場合、または値が OFF の場合は、障害遭遇リクエストの最優先実行機能が無効になります。

また、本機能を利用するには、4.11.2 実行中ジョブの強制リラン機能を有効にする必要があります。

(3) 本機能利用時の注意事項

本機能を利用することは強く推奨しません。本機能を利用する場合は以下の注意事項があります。

- 障害に遭遇したリクエストの再実行開始予定時刻が決まるまで、他のリクエストの開始予定時刻は全て取り消され、実行ホストに空きがあっても開始されません。したがって、システム全体の稼働率に大きく影響を与えます。
- リクエストが障害に遭遇すると、他のリクエストの開始予定時刻は全て取り消されます。他のリクエストは、既に決定した開始予定時刻に実行できなくなります。
- 障害の発生により、障害に遭遇したリクエストが再実行するための実行ホストやリソースが不

足する場合、障害を復旧しない限り、全てのジョブ実行が停止することになります。

4.12 デッドラインスケジューリング

4.12.1 デッドラインスケジューリング概要

JobManipulator のバックフィル・スケジューリングは、リクエストの開始予定時刻が最も早くなるようにアサインします。デッドラインスケジューリングは、リクエストを可能な限りユーザが指定したデッドライン時刻に終了する様にアサインし、スケジューラマップ前方の空きリソースを優先して他のリクエストにアサイン可能にするスケジューリングです。このデッドライン時刻を設定したリクエストをデッドラインリクエストと呼びます。

デッドラインリクエストは、エスカレーション動作順などにおいて非デッドラインリクエストに比べ優先度が低いためスケジューリング上の不利益が生じます。このため JobManipulator では、スケジューリングプライオリティの決定要素であるリソースの使用実績値を減免する機能を提供し、デッドラインリクエストを利用するユーザに対してインセンティブを与えることができます。

デッドラインスケジューリングは、通常キューに投入されたデッドラインリクエストのみに有効となり、緊急/特別リクエストはデッドラインスケジューリングされません。また、アサイン待ちリクエストがなく現在時刻に空きリソースがあり、かつデッドラインリクエストが実行可能な場合、そのデッドラインリクエストは直ぐに実行開始され、ノードの稼働率低下を防ぎます。

4.12.2 デッドラインスケジューリングの設定

デッドラインスケジューリングによるスケジュールを行うには、キュー設定のデッドラインスケジューリングを **on** にします。この設定は、**smgr**(1M)コ マンドの **set queue deadline mode** サブコマンドで実施します。設定が **off** の場合は、デッド ラインリクエストに設定されているデッドライン時刻は無視され、バックフィル・スケジューリングとなります。

運用中にデッドラインスケジューリングを変更した場合、デッドラインリクエストは次のように処理されます。

- **OFFからONにした場合**

  qstat(1)/**sstat**(1)コマンドで、デッドライン時刻が表示されるようになり、次のスケジューリングインターバルからデッドラインスケジューリングを実施します。アサイン済みのデッドラインリクエストは再スケジューリングされませんが、その後のエスカレーションのインタ

ーバルでデッドラインスケジューリングが実施されます。

- **ONからOFFにした場合**

  qstat(1)/sstat(1)コマンドでデッドライン時刻が(none)と表示されるようになります。スケジューラマップ内にアサイン済みのデッドラインリクエストは再スケジューリングされません。スケジューラマップ外にアサイン済みのデッドラインリクエストはジョブが削除され、QUEUEDリクエストとして再スケジューリングされます。

4.12.3 デッドラインリクエストの投入

デッドラインリクエストは、デッドライン時刻を指定してリクエスト投入（qsub）します。指定方法は、以下のとおりです。

```
$ qsub -Y deadline_time script


  deadline_time : [[[[CC]YY]MM]DD]hhmm[.SS]


          CC：西暦年のはじめの2桁
          YY：西暦年の後ろの2桁
          MM：月(01-12)
          DD：日(01-31)
          hh：時(00-23)
          mm：分(00-59)
          SS：秒(00-61)


deadline_timeにデッドライン時刻を指定します。
```

デッドラインリクエスト投入時に現在時刻とデッドライン時刻の整合性はチェックされません。デッドライン時刻が既に過去の時刻となっていた場合は通常のリクエストとして扱いスケジューリングします。

デッドライン時刻を設定したリクエストは、qstat(1) コマンドの-f オプションもしくは sstat (1)コマンドの-f オプションで確認でき、デッドライン時刻が未設定のリクエストは、(none)と表示されます。

4.12.4 デッドラインリクエストのスケジューリング

JobManipulator は、キュー設定のデッドラインスケジューリングが **on** でかつデッドラインリクエストが投入されたとき、そのデッドラインリクエストに設定されたデッドライン時刻に可能な限り終了するようにスケジューリングを行います。

デッドラインリクエストが投入され実行開始されるまでの処理は以下のとおりです。

アサインプールからのピックアップ

デッドラインリクエストをアサインプールからピックアップする順序は通常リクエストと同様に扱われ、スケジューリングプライオリティに従ってピックアップされます。

アサイン

デッドラインリクエストは、指定したデッドライン時刻とリクエストの終了予定時刻が最も近くなるように、スケジューラマップへアサインします。

アサイン時刻の候補は、以下の順に決定します。

1. デッドライン時刻と終了予定時刻が同じ
2. デッドライン時刻より前で、かつデッドライン時刻に最も終了予定時刻が近い時刻
3. デッドライン時刻より後で、かつデッドライン時刻に最も終了予定時刻が近い時刻

スケジューラマップにリソースの空きが常に無い状態で運用されている場合、デッドラインリクエストをアサインすることができず、恒常的にデッドラインを超過することが考えられるため、デッドラインリクエストは、スケジューラマップ外でもアサインします。

アサイン場所の変更

エスカレーション機能（2.7.6(3) エスカレーション機能の設定」参照）が ON の場合、アサイン済みのデッドラインリクエストのアサイン場所を変更可能かどうか、エスカレーションインターバルごとにチェックします。その際、既にアサインしたノード以外もアサイン場所の候補となります。

1. スケジューラマップ先頭に空き（デッドラインリクエストが直ちに実行できるリソース）が存在し、アサインプールにアサイン可能なリクエストがない場合、
   デッドラインリクエストのエスカレーションを実施し、スケジューラマップ先頭にアサインして実行を開始します。

2. デッドラインリクエストの終了予定時刻が、よりデッドライン時刻に近い位置に変更可能である場合、再アサインします。

実行

デッドラインリクエストが開始予定時刻に達すると、実行を開始します。デッドラインリクエストは、一度実行状態(RUNNING)になると、通常リクエストとして扱われるようになり、バッチジョブが存在する間はデッドラインスケジューリングの対象となりません。

但し、実行中にリランとなった場合は、再びデッドラインリクエストとして、デッドラインスケジューリングの対象となります。

- 緊急/特別リクエストによって割り込まれた場合
  通常リクエストと同様に扱います。
  （詳細は「4.1緊急/特別リクエスト」を参照してください。）
- **qhold**(1)コマンドでホールドを実施した場合
  通常リクエストと同様にリクエストはアサインプールに戻り、リリース後再アサインします。
  リリース後、デッドラインスケジューリングは行いません。
- **smgr**(1M)コマンドでサスペンドを実施した場合
  通常リクエストと同様にリクエストはアサインプールに戻り、**smgr**(1M)コマンドでリジューム要求後再アサインします。リジューム要求後、デッドラインスケジューリングは行いません。
- **qsig**(1)コマンドでサスペンドを実施した場合
  通常リクエスト同様にリクエストはスケジューラマップ上でリソースの占有を続けます。

デッドラインリクエストは、デッドライン時刻までにリクエストが終了するようにスケジューリングされますが、以下に挙げる要因により終了予定時刻がデッドライン時刻を超過する場合があります。

1. デッドライン時刻までのリソースに空きが無い場合
   - アサイン時にリソース不足
   - 以下の要因により再スケジューリング時にリソース不足
   - ステージイン完了の遅延
   - ノード障害
   - 緊急/特別リクエストによる割り込み
   - **qrerun**(1)コマンドによるリラン
   - **qrls**(1)コマンドによるリリース
   - **smgr**(1M)コマンドによるリジューム要求

- スケジューラマップ長の変更
- 実装CPU/GPU数追随機能

2. スケジューリングができない状態の場合

- スケジューラマップ先頭にアサインしてもデッドライン時刻を超過する場合
- 実行キューの停止
- 実行キューにジョブサーバがバインドされていない
- スケジューリングを停止中にデッドライン時刻を超過した場合
- 他リクエストが追い越し制御に抵触した場合
- リクエストのリソース指定過多、またはノードダウンによるリソース不足

デッドライン時刻を超過する場合、デッドラインリクエストはデッドライン時刻に最も近く終了するようにスケジューリングされます。

4.12.5　デッドラインリクエストのリソース使用実績値

デッドラインリクエストは、エスカレーション動作順などにおいて非デッドラインリクエストに比べ優先度が低くなりスケジューリング上の不利益が生じるため、JobManipulator ではスケジューリングプライオリティの決定要素であるリソースの使用実績値について、デッドラインリクエストの使用実績値を管理者の設定条件に従った調整（リソース使用実績値の減免率）を可能にします。

使用実績値の調整はデッドラインリクエストに関する個々のバッチジョブが終了した時点で実施されます。実際の実績値から減免率分を差し引いた値を調整後の実績値とし、その値を ShareDB に反映します。

使用実績値としては 5 種類（経過時間、CPU 台数、メモリ使用量、リクエストプライオリティ、VE 台数）が存在し、全て同じ減免率を用いられます。使用実績値の減免率は一律ではなく、デッドライン時刻と該当リクエストの実行終了予定時間（実行開始予定時間に宣言経過時間と ELAPSE マージン時間を加えた時間）との差に比例した値となります。

具体的にはデッドライン時刻ちょうどに終了した場合を基準として、それより早く終了した場合は減免率が少なく、遅く終了した場合（デッドライン時刻を超過した場合）は減免率が多くなるよう調整可能とします。

この調整は減免率調整パラメータで定義され、オペレータ権限以上を有する管理者により、キュー単位に **smgr**(1M)コマンドの **set queue deadline reduce** サブコマンドで設定します。キューに設定されている減免率調整パラメータは、一般利用者以上の権限を有する利用者であれば、**sstat**(1)コマン

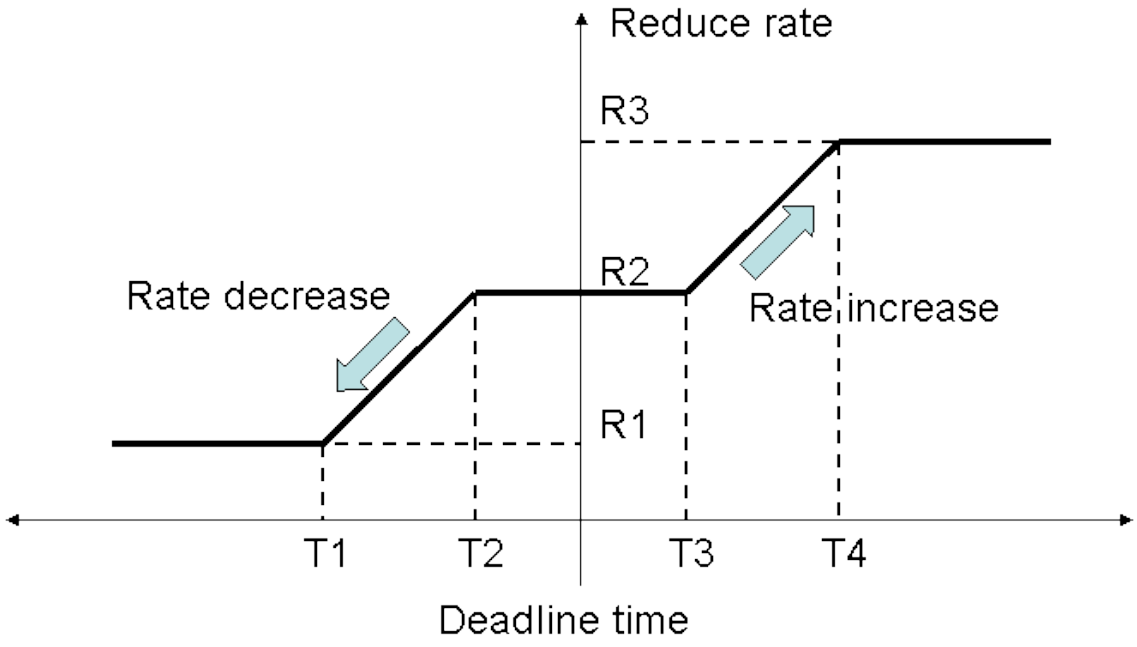
ドの-Q -f オプションで確認できます。

減免率調整パラメータ

減免率は、以下の 7 つの減免率調整パラメータによって定義されます（【】内は略称）。

- 【R3】 減免率上限（maximum reduce rate）
- 【R2】 オンタイム減免率（ontime reduce rate）
- 【R1】 減免率下限（minimum reduce rate）
- 【T3】 減免率増加開始時間（start time of rate increase）
- 【T4】 減免率増加終了時間（end time of rate increase）
- 【T2】 減免率減少開始時間（start time of rate decrease）
- 【T1】 減免率減少終了時間（end time of rate decrease）
-

T1 から T4 の時間は、デッドライン時刻を起点とした相対時間（0 以上の整数値、単位は秒）で設定します。 R1 から R3 の減免率は、0 から 1.0 までの実数値で設定します。



適用される減免率の算出方法について上記グラフを使って以下に説明します。なお、以下の算出式では適用される減免率を Rd、リクエストの実行終了予定時間（デッドライン時刻からの相対時間で表す）を Tr とします。

- T1より早い時刻にリクエストが終了した場合

Rdは下限であるR1が一律に適用されます。

```
Rd = R1 [ T1 < Tr ]
```

- T1からT2の間でリクエストが終了した場合
  Trが大きいほど（T1に近いほど）、それに比例してRdは減少します。ただし、T1とT2が同値の場合はRdをR1とします。

```
Rd =  ((R2 − R1) ／ (T1 − T2)) ＊ Tr + ((T1 ＊ R1 − T2 ＊ R2) ／ (T1 − T2))    [ T1 ＞ T2, T1 ≧ Tr ＞ T2 ]

Rd = R1    [ T1 = T2, T1 ≧ Tr ＞ T2 ]
```

- T2からT3の間（デッドライン時刻を含む）で終了した場合
  Rdはオンタイム減免率であるR2が一律適用されます。

```
Rd = R2 [ T2 ≧ Tr, Tr < T3 ]
```

- T3からT4の間でリクエストが終了した場合
  Trが大きいほど（T4に近いほど）、それに比例してRdは増加します。ただし、T3とT4が同値の場合はRdをR3とします。

```
Rd =  ((R3 − R2) ／ (T4 − T3))  ＊ Tr +  ((T4 ＊ R2 − T3 ＊ R3) ／ (T4 − T3))    [ T4 ＞ T3, T4 ≧ Tr ＞ T3 ]
Rd = R3    [ T4 = T3, T4 ≧ Tr ＞ T3 ]
```

- T4より遅い時刻にリクエストが終了した場合
  Rdは上限であるR3が一律に適用されます。

```
Rd = R3 [ T4 < Tr ]
```

減免率調整パラメータの初期値、取り得る値の範囲、および制約事項は以下の通りです。

パラメータ名		初期値	最大値	最小値	制約事項
R3	減免率上限	1.0	1.0	0	R3≧R2
R2	オンタイム減免率	1.0	1.0	0	なし
R1	減免率下限	1.0	1.0	0	R1≦R2
T3	減免率増加開始時間	0	2^31-1	0	T3≦T4

T4	減免率増加終了時間	0	2^31-1	0	なし
T2	減免率減少開始時間	0	2^31-1	0	T2≦T1
T1	減免率減少終了時間	0	2^31-1	0	なし

減免率調整パラメータは、オペレータ権限以上をもつ管理者によって **smgr**(1M)コマンドにより実行キュー毎に設定されます。運用中にパラメータ値が変更された場合、値を変更した後に終了したバッチジョブから新しいパラメータで算出した減免率が適用されます。

減免率の使用実績への適用

減免率の適用は、バッチジョブの終了を契機に行われます。適用の対象となるリソースは以下の 5 つであり、これら全てに同じ減免率が適用されます。

- 経過時間
- CPU台数
- メモリ使用量
- リクエストプライオリティ
- VE台数

上記 5 つのリソースの使用実績に減免率を適用した値を使用実績値として､該当ユーザの ShareDB に足し込まれます。

4.13 外部ポリシー組込機能

4.13.1 外部ポリシー組込機能の概要

外部ポリシー組込機能は、JobManipulator のスケジューリングをサイト独自のポリシー（以下、外部ポリシーと呼ぶ）にカスタマイズすることができます。JobManipulator は、サイトが独自に作成した下表の API を使用して外部ポリシーに従ってスケジューリングします。

以下の 3 つの外部ポリシーを JobManipulator に組み込むことができます。

1. 投入時の外部ポリシー
   リクエスト投入時に外部ポリシー（ユーザ、グループ個別に資源使用量を制限する等）に従ってリクエストの投入を制御できます。
2. リクエストプライオリティの外部ポリシー
   リクエスト投入時に外部ポリシーに従ったリクエストプライオリティを設定することで、リクエストの優先度を制御できます。
3. アサイン時の外部ポリシー

アサイン時に外部ポリシー（ユーザ、グループ個別に同時アサイン CPU 数を制限する等）に従ってリクエストのアサインを制御できます。

外部ポリシー組込機能の API は、以下のとおりです。

外部ポリシー組込機能用API	機能内容
RLIM_connect	接続処理
RLIM_disconnect	切断処理
RLIM_chkresource	投入時の外部ポリシーチェック処理
RLIM_getpriority	外部ポリシーに従ったリクエストプライオリティ値の取得処理
RLIM_chkrunlimit	アサイン時の外部ポリシーチェック処理
RLIM_relrunlimit	アサイン時の外部ポリシーチェックの解除処理

4.13.2 外部ポリシー組込機能の設定

コンフィグファイル（/etc/opt/nec/nqsv/nqs_jmd.conf 、もしくは JobManipulator 起動時に、'-f'で指定するファイル）に以下のパラメータを設定し、nqs_jmd を再起動してください。上記 3 つの機能ごとに有効・無効および対象となるリクエストタイプを設定します。

(1) 外部ポリシー組込機能の有効化

この 3 つの機能ごとに有効・無効を設定します。

- **API_SUBREQ_CHK:** *ON*| *OFF*
  投入時の外部ポリシーの有効・無効を設定します。
- **API_SET_PRI:** *ON*| *OFF*
  リクエストプライオリティの外部ポリシーの有効・無効を設定します。
- **API_ASSIGN_CHK:** *ON*| *OFF*
  アサイン時の外部ポリシーの有効・無効を設定します。

本パラメータを省略した場合は OFF が適用されます。ON/OFF 以外を設定した場合、または、":"のあとに何も設定していない場合、nqs_jmd は標準出力にエラーメッセージを出力し、起動を中止します。ON を設定した場合、後述の外部ポリシー組込機能用 API 共有ライブラリのパス指定が必須となります。

(2) 対象リクエストタイプ指定

機能ごとに対象となるリクエストタイプを設定します。

- **API_SUBREQ_CHK_TYPE:** *request_type* [,*request_type*...]

投入時の外部ポリシーの対象リクエストタイプを設定します。

- **API_SET_PRI_TYPE:** *request_type* [,*request_type*...]
  リクエストプライオリティの外部ポリシーの対象リクエストタイプを設定します。
- **API_ASSIGN_CHK_TYPE:** *request_type* [,*request_type*...]
  アサイン時の外部ポリシーの対象リクエストタイプを設定します。

*request_type*に、以下のリクエストタイプを設定できます。

normal: 通常キューに投入されたリクエスト

special: 特別キューに投入されたリクエスト

urgent: 緊急キューに投入されたリクエスト

all: normal, special, urgentの全てのリクエスト

本パラメータを省略した場合は、通常キューに投入されたリクエストのみ該当機能の対象となります。本パラメータを指定する場合、1 個以上のタイプ設定が必要となります。

複数個のタイプを設定する場合、","で区切って設定してください。*request_type* に **normal, special, urgent, all** 以外が設定されていた場合、または、":"のあとに何も指定していない場合、nqs_jmd は標準出力にエラーメッセージを出力し、起動を中止します。

(3) **外部ポリシー組込機能API共有ライブラリのパス指定**

外部ポリシー組込機能 API 共有ライブラリのパスを設定します。

API_LIB_PATH: *library_path*
library_path に、外部ポリシー組込機能 API 共有ライブラリのパスを設定します。

この 3 つの機能（API_SUBREQ_CHK/API_ASSIGN_CHK/API_SET_PRI）のうち、1 つでも有効に設定した場合、本設定は必須となります。本パラメータを省略した場合、nqs_jmd は標準エラー出力にエラーメッセージを出力し、起動を中止します。

4.13.3　外部ポリシーデーモン接続

外部ポリシー組込機能を有効に設定した場合は、RLIM_connect 関数を使用して外部ポリシーデーモンと接続します。接続に失敗した場合はスケジューリング間隔ごとにリトライします。接続が成功するまで外部ポリシーはスケジューリングに反映されません。JobManipulator 停止時に、RLIM_disconnect 関数を使用して外部ポリシーデーモンとの接続を切断します。

4.13.4　投入時の外部ポリシー

API_SUBREQ_CHK が ON の場合、**API_SUBREQ_CHK_TYPE** で指定されたタイプのキューに

投入されたリクエストに対して、RLIM_chkresource 関数を使用して外部ポリシー（ユーザ、グループ個別に資源使用量を制限する等）に従ってリクエストの投入を制御できます。

下表のとおり、**RLIM_chkresource** 関数の返り値に応じた処理を実行します。

返り値の意味：返り値	処理
投入可： 0	該当リクエストをスケジューリング対象とします。
投入不可： -3 システムエラー： -1	該当リクエストを削除し、リクエストに設定されたメールアドレスにメールを送信します。 メールの内容は、以下のとおりです。 Subject: NQSV request: *request_id.machine_id* is deleted. 本文： Reason: RLIM_chkresource error Detail: *RLIM_chkresource*関数から返却されたメッセージ
接続エラー： -2	チェック対象となるASSIGNED状態のリクエストをスケジューラマップからクリアし、チェック対象のリクエストのスケジューリングを停止します。 外部ポリシーデーモンへの再接続をスケジューリング間隔ごとに実施します。

リクエスト投入時の外部ポリシーチェックのタイミングは以下のとおりです。

- リクエスト投入時
- スケジューラ起動時
- 外部ポリシーデーモンへ再接続時
- 実行キューとスケジューラのバインド時

なお、下記状態のリクエストに対して、チェックを行いません。

- 連携リクエストでHELD状態のリクエスト
- qsub時に-h指定で投入されたHELD状態のリクエスト
- qsub時に-a指定で投入されたWAITING状態のリクエスト

4.13.5　リクエストプライオリティの外部ポリシー

API_SET_PRI が ON の場合、**API_SET_PRI_TYPE** で指定されたタイプのキューに投入したリクエストに対して RLIM_getpriority 関数を使用して外部ポリシーに従ったリクエストプライオリティを設定することで、リクエストの優先度を制御します。

JobManipulator は、RLIM_getpriority 関数の戻り値に従って以下の処理を行います。

返り値の意味：返り値	処理
取得成功： 0	取得したリクエストプライオリティ値をリクエストに設定します。
システムエラー： -1	リクエストを削除し、リクエストに設定されたメールアドレスにメールを送信します。 メールのsubjectおよび本文は、以下のとおりです。 Subject: NQSV request: *request_id.machine_id* is deleted. 本文： Reason: RLIM_getpriority error Detail: *RLIM_getpriority*関数から返却されたメッセージ
接続エラー： -2	JobManipulatorのアサイン時のチェック対象のASSIGNED状態のリクエストをマップからクリアし、リクエストプライオリティ設定の対象リクエストのスケジューリングを停止します。外部ポリシーデーモンへの再接続を1スケジューリング間隔ごとに実施します。

RLIM_getpriority 関数が呼び出されるタイミングは、以下のとおりです。

- リクエスト投入時
- スケジューラの起動時
- 外部ポリシーデーモン再接続時
- 実行キューとスケジューラのバインド時

なお、下記状態のリクエストに対して、リクエストプライオリティ値の取得・設定を行いません。

- 連携リクエストでHELD状態のリクエスト
- qsub時に-h指定で投入されたHELD状態のリクエスト
- qsub時に-a指定で投入されたWAITING状態のリクエスト

以下の場合、リクエストを削除しリクエストに設定されたメールアドレスにメールを送信します。

- 取得したリクエストプライオリティ値が設定可能な値の範囲（-1024 ～ 1023）を超過した場合

  送信するメールは、以下のとおりです。

  Subject: NQSV request: *request_id.machine_id* is deleted.
  本文：Reason: Request priority exceeds limit.

- プライオリティの設定に失敗した場合

  送信するメールは、以下のとおりです。

  Subject: NQSV request: *request_id.machine_id* is deleted.

本文：Reason: Request priority cannot be set.

4.13.6 アサイン時の外部ポリシー

(1)　アサイン時の外部ポリシーチェック

API_ASSIGN_CHK が ON の場合、**API_ASSIGN_CHK_TYPE** で指定されたタイプのキューに投入されたリクエストをアサインする際に、RLIM_chkrunlimit 関数を使用して外部ポリシー（ユーザ、グループ個別に同時アサイン CPU 数を制限する等）によって、リクエストのアサインを制御します。

下表のとおり、**RLIM_chkrunlimit** 関数の返り値に応じた処理を実行します。アサイン時のチェックのタイミングは、リクエストをステージングする直前で、アサイン場所が決定された後です。

返り値の意味：返り値	処理
アサイン可： 0	該当リクエストをアサインします。
アサイン不可： -3	アサイン可能になるまでスケジューリング間隔ごとにアサイン処理をリトライします。
システムエラー： -1	該当リクエストを削除し、リクエストに設定されたメールアドレスにメールを送信します。 メールの内容は、以下のとおりです。 Subject: NQSV request: *request_id.machine_id* is deleted. 本文： Reason: RLIM_chkrunlimit error Detail: *RLIM_chkrunlimit*関数から返却されたメッセージ
接続エラー： -2	チェック対象となるASSIGNED状態のリクエストをスケジューラマップからクリアし、チェック対象のリクエストのスケジューリングを停止します。外部ポリシーデーモンへの再接続をスケジューリング間隔ごとに実施します。

(2)　アサイン時の外部ポリシーチェックの解除

アサイン時の外部ポリシーチェックの対象となったリクエストのジョブが終了、削除されたとき RLIM_relrunlimit 関数が実行されます。外部ポリシーデーモンにおいて、リクエスト状態の管理処理などを実施します。アサイン時の外部ポリシーチェックを解除するタイミングは以下のとおりです。

- リクエスト終了時
- ジョブサーバのアンバインドなどによりジョブがキャンセルされたとき

JobManipulator は、下表のとおり、**RLIM_relrunlimit** 関数の戻り値に応じた処理を実行します。

返り値意味：返り値	処理

解除処理が成功： 0	なし
システムエラー： -1	リクエストを削除し、リクエストに設定されたメールアドレスにメールを送信します。 メールの内容は、以下のとおりです。 Subject: NQSV request: *request_id.machine_id* is deleted. 本文： Reason: RLIM_relrunlimit error Detail: *RLIM_relrunlimit*関数から返却されたメッセージ
接続エラー： -2	JobManipulatorのアサイン時のチェック対象のASSIGNED状態のリクエストをスケジューラマップからクリアし、アサイン時のチェック対象となるリクエストのスケジューリングを停止します。外部ポリシーデーモンへの再接続をスケジューリング間隔ごとに実施します。

4.13.7　API 関数

JobManipulator は、サイト独自に定義した下記の API 関数を呼び出すことにより、外部ポリシー組込機能を実現します。

(1)　RLIM_connect

形式

int RLIM_connect(char *msg)

機能概要

外部ポリシーデーモンとの接続を行います。

異常終了の場合、その原因の説明を *msg* に設定します。

引数

char *msg<OUT>：エラー発生時の詳細メッセージ（128 文字以内）の格納用領域

戻り値

接続成功	：0
システムエラー	：-1
接続エラー	：-2

(2)　RLIM_disconnect

形式

int RLIM_disconnect(char *msg)

機能概要

外部ポリシーデーモンとの接続を切断します。

異常終了の場合、その原因の説明を *msg* に設定します。

引数

char *msg<OUT>：エラー発生時の詳細メッセージ（128 文字以内）の格納用領域

戻り値

切断成功	：0
システムエラー	：-1
接続エラー	：-2

(3) RLIM_chkresource

形式

int RLIM_chkresource(ReqID *reqid, uid_t uid, gid_t gid, char *qname, Resources *resources, char *msg)

機能概要

引数で指定したリクエストに対して、投入時の外部ポリシーをチェックし、その結果を返却します。

異常終了の場合、その原因の説明を *msg* に設定します。

引数

ReqID *reqid<IN>　：チェック対象のリクエストID

```
typedef struct {
int    mid;          /* Machine ID */
int    seqno;        /* Sequential number */
int    subreq_no; /*Subrequest number */
} ReqID;
```

uid_t uid<IN>　：チェック対象リクエストのオーナのユーザID

gid_t gid<IN>　：チェック対象リクエストのオーナのグループID

char *qname<IN>　：チェック対象リクエストのキュー名

Resources *resources<IN>　：チェック対象リクエストに宣言された要求リソース量

```
typedef struct {
int job_num;
int elapse;
int cputime_per_job;
long long disk_per_job;
int cpunum_per_job;
} Resources;
```

job_num：	チェック対象リクエストに宣言されたジョブ数
elapse：	チェック対象リクエストに宣言された経過時間(秒)
cputime_per_job：	チェック対象リクエストに宣言されたジョ

ブ毎のCPU時間(秒)
disk_per_job：　任意の整数
cpunum_per_job：　チェック対象リクエストに宣言されたジョブ毎のCPU台数

char *msg<OUT>　：エラー発生時の詳細メッセージ（128文字以内）の格納用領域

戻り値

投入可　：0（外部ポリシー制限に抵触していない場合、「投入可」を返却します。）
システムエラー　：-1
接続エラー　：-2
投入不可　：-3（外部ポリシー制限に抵触している場合、「投入不可」を返却します。）

ハイブリッドリクエスト（**qsub**(1)コマンドでのリクエスト投入時に**--job-separator** あるいは**---**オプションを用いてジョブ毎に異なる資源を要求して投入したリクエスト）の場合、最初のジョブグループの CPU 時間、CPU 台数が設定されます。

（4） RLIM_getpriority

形式

int RLIM_getpriority(ReqID *reqid, uid_t uid, gid_t gid, int *pri, char *msg)

機能概要

引数で指定したリクエストに対して、外部ポリシーに従ったリクエストプライオリティを取得し、処理結果を返却します。

異常終了の場合、その原因の説明を *msg* に設定します。

引数

ReqID *reqid<IN>　：チェック対象のリクエストID
uid_t uid<IN>　：チェック対象リクエストのオーナのユーザID
gid_t gid<IN>　：チェック対象リクエストのオーナのグループID
int *pri<IN/OUT>　：リクエストプライオリティ値を格納する領域のアドレス
char *msg<OUT>　：エラー発生時の詳細メッセージ（128文字以内）の格納用領域

戻り値

正常終了　：0（外部ポリシーに従ったリクエストプライオリティは**pri**に設定します。
システムエラー　：-1
接続エラー　：-2

（5） RLIM_chkrunlimit

形式

int RLIM_chkrunlimit(ReqID *reqid, uid_t uid, gid_t gid, char *qname, char *msg)

機能概要

引数で指定したリクエストに対して、アサイン時の外部ポリシーのチェックを行い、その結果を返却します。

異常終了の場合、その原因の説明を *msg* に設定します。

引数

ReqID *reqid<IN>	：チェック対象のリクエストID
uid_t uid<IN>	：チェック対象リクエストのオーナのユーザID
gid_t gid<IN>	：チェック対象リクエストのオーナのグループID
char *qname<IN>	：チェック対象リクエストのキュー名
char *msg<OUT>	：エラー発生時の詳細メッセージ（128文字以内）の格納用領域

戻り値

アサイン可	：0
システムエラー	：-1
接続エラー	：-2
アサイン不可	：-3

(6) RLIM_relrunlimit

形式

int RLIM_relrunlimit(ReqID *reqid, uid_t uid, gid_t gid, char *msg)

機能概要

外部ポリシーデーモン側において、引数で指定されたリクエストの状態管理などを実施し、その結果を返します。

例えば、該当リクエストを外部アサインポリシーチェックの対象外とします。

異常終了の場合、その原因の説明を *msg* に設定します。

引数

ReqID *reqid<IN>	：チェック対象のリクエストID
uid_t uid<IN>	：チェック対象リクエストのオーナのユーザID
gid_t gid<IN>	：チェック対象リクエストのオーナのグループID
char *msg<OUT>	：エラー発生時の詳細メッセージ（128文字以内）の格納用領域

戻り値

正常終了	：0
システムエラー	：-1
接続エラー	：-2

4.14 省電力運用機能

4.14.1 機能概要

省電力運用機能として、以下の２つの機能を提供します。

- リクエストの実行状況に応じて稼働ノードを最適に制御する動的省電力運用機能
- あらかじめノードを停止する時間帯を登録することで、計画的にノードを制御する計画省電力運用機能

これらの機能により、計算ノードの稼働状況に応じて電源供給を制御することが可能となり、不必要な消費電力を削減することができます。

省電力運用機能は、以下の条件をすべて満たす実行ホストに対して利用可能です。

- BMC (Baseboard Management Controller)を搭載している実行ホスト（ただし、OpenStackのベアメタルサーバを除く）
- JobManipulatorとバインドしているキューとバインドしているJSVの実行ホスト
- 障害に遭遇していない実行ホスト
- 運用開始後、リンクアップかつJobManipulatorとバインドしているキューとバインドしている状態になったことのあるJSVの実行ホスト

【実行ホストの設定】

- 実行ホストに搭載されているBMC（Baseboard Management Controller）が利用できるように設定
- ノード管理エージェントホストにipmitoolをインストール
- ノード管理エージェントを起動
- ノード管理エージェントの詳細については*NQSV* 利用の手引*[*管理編*]* を参照してください。

実行ホスト上のジョブサーバが実行ホストの起動後に自動的に起動されるように設定する必要があります。ジョブサーバの自動起動については、*NQSV* 利用の手引 *[*管理編*]* の *3.5* ジョブサーバの起動を参照してください。なお、NQSV のランチャーデーモンによるジョブサーバの起動と systemd によるジョブサーバの起動のいずれかだけ設定してください。両方設定すると省電力運用機能による実行ホストの起動時にジョブサーバの起動が失敗する可能性があります。

省電力運用機能によるノードの起動・停止の状況は、sstat(1)コマンドの-E --eco-status オプションで表示可能です。

```
$ sstat -E --eco-status
ExecutionHost   EcoStatus  StateTransitionTime OFF(D) ACCUM
--------------- --------- ------------------- ------ -----
Host1           PEAKCUT    2015-05-26 16:30:00      1   100
Host2           EXCLUDED   2015-06-30 12:00:00      1   101
Host3           -          -                        1    98
```

省電力対象外状態(EXCLUDED)となった理由は、sstat(1)コマンドの-E --eco-status -f オプションで表示可能です。

```
$ sstat -E --eco-status -f Host2
Execution Host: Host2
    Eco Status            = EXCLUDED
    State Transition Time = 2015-06-30 12:00:00
    Exclude Reason        = START_FAIL
    DC-OFF Times (Day)    = 1
    DC-OFF Times (ACCUM)  = 101
```

4.14.2 動的省電力運用機能

動的省電力運用機能は、計算ノードの稼働状況に応じて、計算ノードの DC 電源を OFF/ON する機能です。電力消費のピークカットを実現するため、スケジューラ全体の設定として「最大稼働ノード数」を設け、稼働ノード が「最大稼働ノード数」以下となるように適切にノードを停止します。電力消費の「ピークカット緊急度の設定」により、ノードを即時停止するモードと実行中リクエストの終了を待ってノードを停止するモードを選択可能です。

動的省電力運用機能では、一定期間リクエストが利用しないノードを自動的に運用から切り離す（ノード停止する）ことにより消費電力を削減します。ただし、ノードが一時的に利用されていないことにより極端に多くのノードが停止されジョブ運用に影響することを防ぐために、キュー単位のパラメータとして「キューの最小稼働ノード数」を設け、ノード停止は、稼働ノード数が「キューの最小稼働ノード数」を下回らないようにノードを停止します。アサイン待ちのリクエストが出現した時にノードを、各キューの合計稼働ノード数が「最大稼働ノード数」を超えない範囲で起動し運用に戻しま

す。緊急リクエストの緊急実行を保証するため、緊急リクエストの投入によりノードを起動する場合は、この制限を無視してノードを起動します。

動的省電力運用機能を ON に設定しているとき、JobManipulator とバインドしているキューにバインドしている JSV が動作しているノードがすべて電源制御の対象となるため、メンテナンス時など、ノードを省電力運用の対象から除外したい場合は、当該 JobManipulator とバインドしているすべてのキューから JSV をアンバインドする必要があります。

(1)　動的省電力運用機能の設定

動的省電力運用機能は、**smgr**(1M)コマンドの **set dynamic_dc_control** サブコマンドによりスケジューラ単位で設定可能です。

```
#smgr -P m
Smgr: set dynamic_dc_control = off | on
```

- 既定値はoffです。
- offからonに変更した場合は、動的電源制御を開始します。
- onからoffに変更した場合は、動的電源制御を停止します。
  JobManipulatorとバインドしているキューにバインドしているすべてのノードを即時に起動します。ただし、HW障害ノードと計画省電力停止中のノードを除きます。
- 操作員権限以上の権限が必要です。

設定値は、 **sstat**(1)コマンドの-S -f オプションにより確認可能です。

```
#sstat -S -f
JobManipulator Server Host: bsv.nec.co.jp
    JobManipulator Version   = R1.00
    JobManipulator Status    = Active
    Scheduler ID             = 1
            :
    Auto Delete Resource Reservation = OFF
    Forced Re-Scheduling     = OFF
    Dynamic DC Control       = OFF
:
```

(2)　最大稼働ノード数

最大稼働ノード数は、**smgr**(1M) コマンドの **set max_operation_hosts** サブコマンドによりスケジューラ単位に設定可能です。

```
#smgr -Pm
Smgr: set max_operation_hosts = <number of hosts>
```

- 既定値は10240です。
- 設定範囲は、0~10240です。
- 操作員権限以上の権限が必要です。

設定値は、 **sstat**(1)コマンドの**-S -f** オプションにより確認可能です。

```
#sstat -S -f
JobManipulator Server Host: bsv.nec.co.jp
    JobManipulator Version   = R1.00
    JobManipulator Status    = Active
    Scheduler ID             = 1
            :
    Auto Delete Resource Reservation = OFF
    Forced Re-Scheduling     = OFF
    Dynamic DC Control       = OFF
    Max Operation Hosts      = 10240
:
```

(3) ピークカット緊急度の設定

ピークカット緊急度は、**smgr**(1M)コマンドの **set peak_cut_urgency** サブコマンドによりスケジューラ単位に設定可能です。

```
#smgr -P m
Smgr: set peak_cut_urgency = wait_run | right_now
```

- 既定値は、wait_run（実行中リクエスト終了待ちモード）です。
  このとき、実行中リクエストの終了を待ち合わせてノードを停止します。
- right now（即時停止モード）は、実行中リクエストをリランし、ノードを即時停止します 。
- 操作員権限以上の権限が必要です。

設定値は、**sstat**(1)コマンドの**-S -f** オプションにより確認可能です。

```
$sstat -S -f
JobManipulator Server Host: bsv.nec.co.jp
    JobManipulator Version   = R1.00
            :
    Auto Delete Resource Reservation = OFF
    Forced Re-Scheduling     = OFF
    Dynamic DC Control       = OFF
    Max Operation Hosts      = 10240
    Peak Cut Urgency         = wait_run
:
```

(4)　キューの最小稼働ノード数

最小稼働ノード数は、smgr(1M)コマンドの set queue min_operation_hosts サブコマンドによりキュー単位に設定可能です。

```
#smgr -P m
Smgr: set queue min_operation_hosts = <number of hosts>  <queue_name>
```

- 既定値は10240です。
- 設定可能な値の範囲は0～10240です。
- 操作員権限以上の権限が必要です。

設定値は、**sstat**(1)コマンドの-**Q**-**f** オプションにより確認可能です。

```
#sstat -Q -f
Execution Queue: jmq0
    Queue Type            = Normal
    Schedule Time         = DEFAULT
              :
    Wait_Stageout = OS
    Min Operation Hosts = 10240
:
```

(5)　省電力停止回数制限

頻繁なノード停止・起動は、HW 故障の原因となるため、1 日あたりの動的省電力運用機能によるノード停止回数を制限する機能を提供します。動的省電力により 1 日にノード停止する回数は、下記

smgr(1M)コマンドにより設定した回数以内に制限されます。

なお、「ピークカット」や計画省電力停止によるノード停止も停止回数としてカウントはしますが、ノードは停止可能です。

省電力停止回数制限値は、**smgr**(1M)コマンドの **set dc-off_limit** サブコマンドによりスケジューラ単位に設定可能です。

```
#smgr -P m
Smgr: set dc-off_limit = <number_of_times>
```

- 設定可能な値の範囲は1～200です。
- 既定値は5です。
- 操作員権限以上の権限が必要です。

設定値は、**sstat**(1)コマンドの**-S -f** オプションにより確認可能です。

```
# sstat -S -f
JobManipulator Server Host: bsv.nec.co.jp
    JobManipulator Version   = R1.00
    JobManipulator Status    = Active
    Scheduler ID             = 1
            :
    Dynamic DC Control       = OFF
    Max Operation Hosts      = 10240
    Peak Cut Urgency         = wait_run
    Min Idle Time            = 300S
    Estimated DC-OFF Time    = 3600S
    DC-OFF Limit             = 5
    Use Overtake Priority = {
      Normal  = OFF
      Special = OFF
    }
:
```

(6)　最低アイドル時間

本機能は、ノードを運用に組み込んだ直後やジョブの実行終了直後にノードが停止されることを防止するため、下記最低アイドル時間の計測開始から一定時間(最低アイドル時間)経過後にノードを停

止させる機能です。最低アイドル時間内にジョブが実行された場合は、ノードを停止させません。

最低アイドル時間の計測開始は、下記の中で最も遅いタイミングです。

- ノード上の実行中ジョブがなくなったとき
- ノードを起動したとき
- JobManipulatorを起動したとき
- **smgr**(1M)コマンドにより動的省電力機能を有効に設定したとき
- JSVをJobManipulatorとバインドしているキューにバインドしたとき
- JSVをバインドしているキューにJobManipulatorをバインドしたとき

最低アイドル時間は、**smgr**(1M)コマンドの **set min_idle_time** サブコマンドによりスケジューラ単位に設定可能です。

```
#smgr -P m
Smgr: set min_idle_time = <seconds>
```

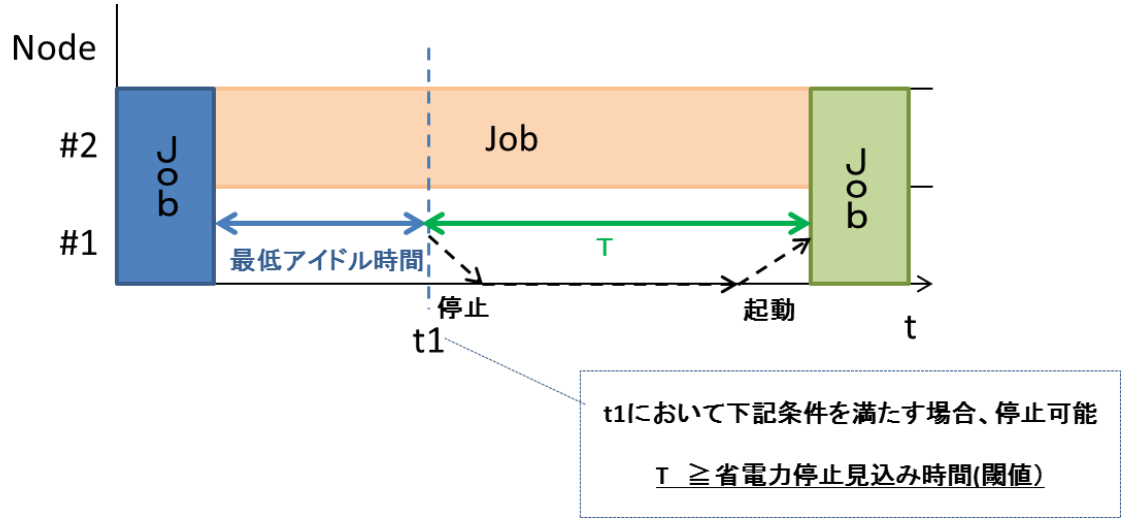
- 単位は、秒です。
- 設定可能な値の範囲は0秒～2147483647秒です。
- 既定値は300秒です。
- 操作員権限以上の権限が必要です。

設定値は、**sstat**(1)コマンドの**-S -f** オプションにより確認可能です。

```
# sstat -S -f
JobManipulator Server Host: bsv.nec.co.jp
    JobManipulator Version   = R1.00
    JobManipulator Status    = Active
    Scheduler ID             = 1
            :
    Dynamic DC Control       = OFF
    Max Operation Hosts      = 10240
    Peak Cut Urgency         = wait_run
    Min Idle Time            = 300S
    Estimated DC-OFF Time    = 3600S
:
```

(7)　省電力停止見込み時間

本機能は、下図に示しているように動的省電力機能によりノードが一定時間（省電力停止見込み時間）以上停止可能な場合、つまり、現在時刻から「省電力停止見込み時間（閾値）」の間に実行開始予定のジョブがない場合のみ、ノードを停止させる機能です。本機能により、ノード停止後にすぐ起動するといった無駄なノード停止を抑止することが可能です。



省電力停止見込み時間（閾値）は、**smgr**(1)コマンドの **set estimated_dc-off_time** サブコマンドによりスケジューラ単位に設定可能です。

```
#smgr -P m
Smgr: set estimated_dc-off_time = <seconds>
```

- 単位は、秒です。
- 設定可能な値の範囲は2秒～2147483647秒です。
- 設定値は「ノード停止マージン ＋ ノード起動マージン」以上に設定してください。「ノード停止マージン」と「ノード起動マージン」については、(8)ノード停止マージンとノード起動マージンを参照してください。
- 既定値は3600秒です。
- 操作員権限以上の権限が必要です。

設定値は、**sstat**(1)コマンドの**-S -f** オプションにより確認可能です。

```
# sstat -S -f
JobManipulator Server Host: bsv.nec.co.jp
    JobManipulator Version   = R1.00
    JobManipulator Status    = Active
```

```
    Scheduler ID              = 1
            :
    Dynamic DC Control        = OFF
    Max Operation Hosts       = 10240
    Peak Cut Urgency          = wait_run
    Min Idle Time             = 300S
    Estimated DC-OFF Time     = 3600S
:
```

(8)　ノード停止マージンとノード起動マージン

ノードの停止と起動には数分程度の時間がかかるため、ノードの停止／起動処理にかかる見込み時間として、「ノード停止マージン」と「ノード起動マージン」を設けます。省電力運用のためにノードを停止して次に起動して利用可能となるまでの時間は、最低、「ノード停止マージン」＋「ノード起動マージン」となります。

ノード停止マージンとノード起動マージンは、コンフィグファイル(/etc/opt/nec/nqsv/nqs_jmd.conf)に指定します。指定形式は、以下の通りです。

```
MARGIN_FOR_STOP_HOST:300
MARGIN_FOR_START_HOST:600
```

- MARGIN_FOR_STOP_HOSTに、ノード停止マージンを指定します。
- MARGIN_FOR_START_HOSTに、ノード起動マージンを指定します。
- 単位は秒です。
- 設定可能範囲は、1秒～2147483647秒です。
- 設定を省略した場合は、既定値となります。
  MARGIN_FOR_STOP_HOSTの既定値は300、MARGIN_FOR_START_HOSTの既定値は600です。

なお、これらのパラメータは、計画省電力運用機能にも適用します。

- 動的省電力機能では、ノード停止要求後、ノード停止マージンの時間が経過しても電源 OFF に至らない場合、ノードは省電力制御対象外となります。
- 省電力制御対象外となった場合、JSV を LINKUP することで対象に復帰します。
- ノードの状況は、**sstat**(1)コマンドの**-E --eco-status** オプション、あるいは、**-E -f** オプションの

EcoStatus で確認できます。

4.14.3 計画省電力運用機能

計画省電力運用機能は、計算ノード稼働率に高低傾向（ex. 平日の稼働率は高いが週末は低い、季節によって稼働率に変動がある等）がある場合に、運用管理者によって電源制御のスケジュール（計画省電力区間）を策定し、スケジュール通りに実行ホストの DC 電源を OFF/ON する機能です。

計画省電力運用では、計画省電力区間の開始時刻以降に実行ホストの停止を開始し、計画省電力区間の終了時刻にジョブ運用の再開が可能となるよう実行ホストを起動します。ただし、動的省電力運用機能が ON の場合、実行ホストを起動するか否かは、動的省電力運用機能により決定します。

計画省電力区間の時間帯には、リクエストをアサインすることはできません。但し、緊急リクエスト（Urgent リクエスト）については、省電力計画により停止している実行ホストを起動することで実行可能となる場合に、省電力計画を削除し実行ホストを起動します。

(1) 計画省電力区間の設定

計画省電力区間の作成は、**smgr**(1M) コマンドの **create eco_schedule** サブコマンドにより行います。作成には、操作員以上の権限が必要です。

```
create eco_schedule starttime = 〈開始時刻〉 endtime= 〈終了時刻〉
hostname = 〈ホスト名〉
```

- starttimeには、計画省電力区間の開始時刻を指定します。
- endtimeには、計画省電力区間の終了時刻を指定します。
- hostnameには、対象となるホスト名を指定します。

計画省電力区間には、計画省電力 ID（0～9999）が付与されます。この計画省電力 ID は、計画省電力区間の削除に使用します。

starttime と endtime の間隔が「ノード停止マージン ＋ノード起動マージン」以上となるように設定してください。

計画省電力区間を複数設定することができますが、同一実行ホストに対して時間帯が重複するような設定を行うことはできません。

また、以下の場合は、計画省電力区間を設定できません。

- 指定した計画省電力区間にアサイン済みリクエストが存在する場合
- 指定した計画省電力区間にキュー指定で作成した予約区間が設定されている場合

(2) 計画省電力区間の削除

計画省電力区間の削除は、**smgr**(1M) コマンドの **delete eco_schedule** サブコマンドにより行います。削除には、操作員以上の権限が必要です。

```
delete eco_schedule = <計画省電力ID>
```

(3) 計画省電力区間の表示

sstat(1)コマンドの-D オプションで計画省電力 ID、計画省電力区間の開始時刻、終了時刻、対象実行ホストを表示することができます。

```
$sstat -D
EcoID EcoStartTime        EcoEndTime          ExecutionHost
------ ------------------- ------------------- ---------------
     0 2014-12-06 18:00:00 2014-12-06 23:00:00 host1
     1 2014-12-06 18:00:00 2014-12-06 23:00:00 host2
```

また **sstat**(1)コマンドの-**D -f** オプションにより詳細情報を表示することもできます。

```
$sstat -Df
Eco Schedule ID: 0
    Scheduled Start Time   = 2014-12-06 18:00:00
    Scheduled End Time     = 2014-12-06 23:00:00
    Number of Scheduled Hosts = 1
  Scheduled Hosts:
      host1

Eco Schedule ID: 1
    Scheduled Start Time   = 2014-12-06 18:00:00
    Scheduled End Time     = 2014-12-06 23:00:00
    Number of Scheduled Hosts = 1
  Scheduled Hosts:
      host2
```

4.15 カスタムリソース機能

4.15.1 機能概要

カスタムリソース機能とは、定義されたカスタムリソース情報に基づき、スケジューリングにて同時に使用するカスタムリソースの利用量を制御する機能です。システム管理者は任意に仮想的なリソースを定義します。これをカスタムリソース情報と呼びます。カスタムリソース情報には、カスタムリソース名、リソースを消費する単位、同時に利用するリソース量を制御する対象の範囲と上限値を設定します。

ユーザは、リクエスト投入時、投入コマンド（**qsub**(1), **qlogin**(1), **qrsh**(1)）で、**--custom** オプションにて各カスタムリソース名と利用量を指定します。JobManipulator は、この値を参照して同時に使用するカスタムリソースの利用量を合計し、それが定義されたカスタムリソースの上限値を超えない様にスケジューリングします。

カスタムリソース機能、カスタムリソース情報の設定方法、キューの設定方法の詳細については、*NQSV* 利用の手引 *[管理編]* を参照してください。カスタムリソース機能を使用したリクエスト投入方法の詳細については、*NQSV* 利用の手引 *[操作編]* を参照してください。

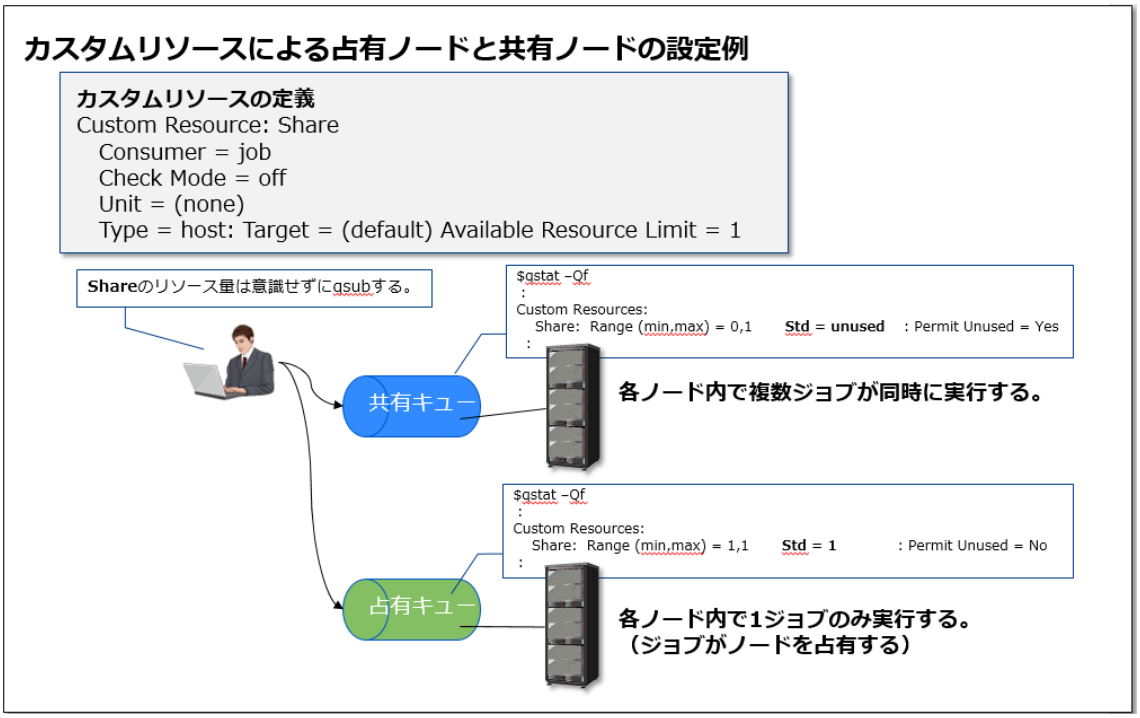
4.15.2 カスタムリソース情報を用いたスケジューリング

リクエストに指定したカスタムリソースの利用量は **qstat**(1)コマンドの**-f** オプションの"Custom Resources"の項目で確認できます。詳しくは *NQSV* 利用の手引 *[操作編]* を参照してください。

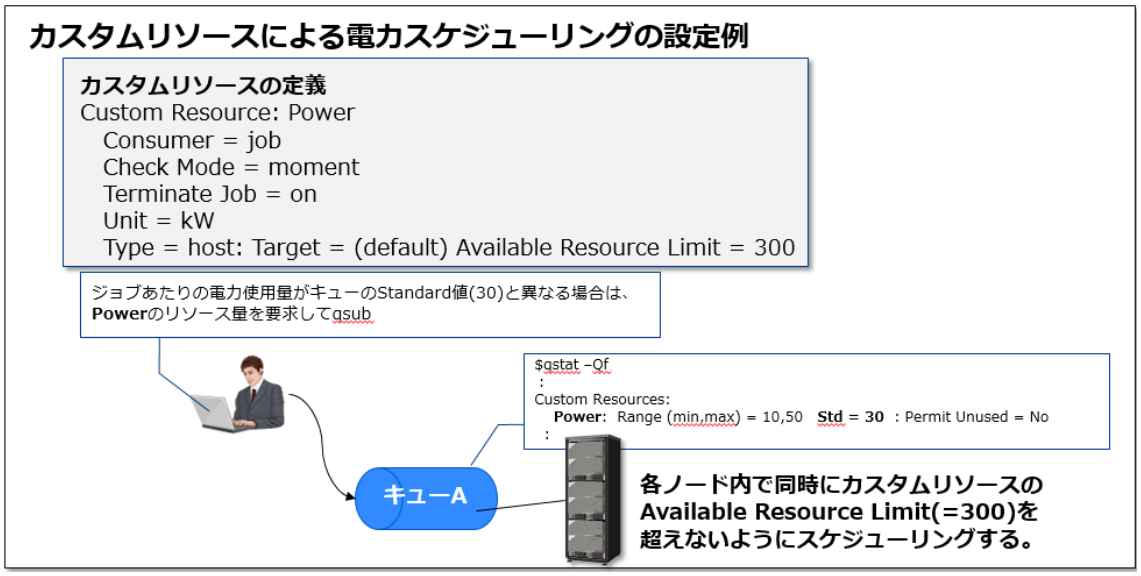
カスタムリソースの利用量を指定してリクエストが投入された場合は、JobManipulator がリクエストのカスタムリソースの利用量を当該カスタムリソースの消費単位でカウントし、利用量制御の対象種別(BSV/実行ホスト)の範囲において同時利用可能リソースの最大値を、スケジューラマップ上のどの時刻においても超えないようにジョブをアサインします。

4.15.3 カスタムリソース機能の使用例

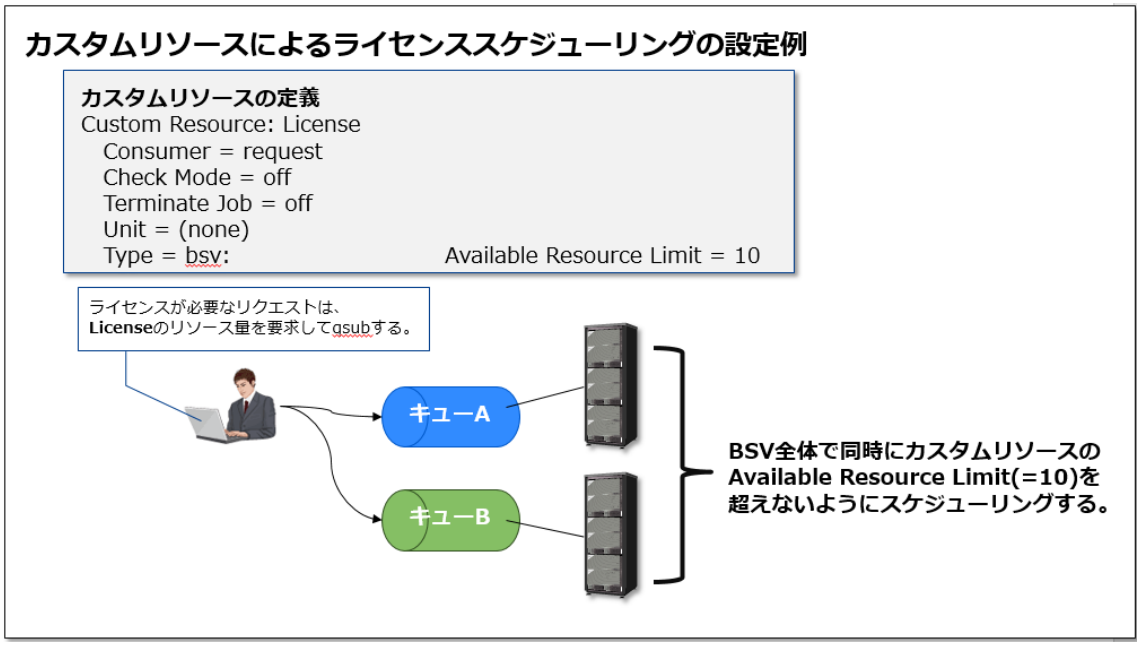
(1)　占有ノードと共有ノードの設定



(2)　瞬間電力スケジューリング



(3)　ISVソフトライセンススケジューリング


カスタムリソースによるライセンススケジューリングの設定例
カスタムリソースの定義
Custom Resource: License
Consumer = request
Check Mode = off
Terminate Job = off
Unit = (none)
Type = bsv:
Available Resource Limit = 10
ライセンスが必要なリクエストは、
Licenseのリソース量を要求してqsubする。
キューA
キューB
BSV全体で同時にカスタムリソースの
Available Resource Limit(=10)を
超えないようにスケジューリングする。

4.16 OpenStackと連携したプロビジョニング

4.16.1 機能概要

OpenStack と連携したプロビジョニング環境として、仮想マシン（VM）によるプロビジョニングとベアメタルサーバによるプロビジョニングをサポートしています。OpenStack と連携したプロビジョニングの設定方法の詳細については、*NQSV* 利用の手引*[*管理編*]* を参照してください。OpenStack と連携したプロビジョニングのリクエスト投入方法の詳細については、*NQSV* 利用の手引*[*操作編*]* を参照してください。

JobManipulator は、仮想マシン（VM）やベアメタルサーバのスケジューリングを行います。本機能は実行ホストが SX-Aurora TSUBASA システムの場合は利用できません。

4.16.2 実行ホスト起動失敗時の再スケジューリング待ち時間設定

OpenStack と連携したプロビジョニング環境により起動された、仮想マシン（VM）やベアメタルサーバの起動失敗時には、当該実行ホストにアサインしているリクエストを再スケジューリングし、再スケジューリング後のアサイン状況に応じてリクエストの実行開始のために再度起動をリトライします。

再スケジューリングの待ち時間を設定し、起動失敗したプロビジョニングの実行ホストは起動に失敗したテンプレートでの再スケジューリングを一定時間待ちます。この時間を、実行ホスト起動失敗時の再スケジューリング待ち時間といいます。このようなテンプレートを指定した起動の失敗の原因として、テンプレートの不正が考えられます。また、その場合、起動のリトライが再度失敗する可能性があります。

本機能により、起動失敗したテンプレートによる再スケジューリング待ち時間を設定し、その間にメンテナンスを実施し実行ホストが起動失敗を繰り返すことを防止することが可能になります。

実行ホスト起動失敗時の再スケジューリング待ち時間は、**smgr**(1M) コマンドの **provisioning_start_retry_time** サブコマンドにより設定します。

```
#smgr -P m
Smgr: set provisioning_start_retry_time = <seconds>
```

- 単位は、秒です。
- 既定値は0です。値0はすぐに再スケジューリングすることを意味します。
- 設定変更時には、設定変更前から再スケジューリング待ちをしている実行ホストを含めて変更後の値が有効になります。

- 操作員権限以上の権限が必要です。

設定値は、**sstat**(1)コマンドの**-S -f** オプションにより確認可能です。

```
$ sstat -S -f
:
     Stage-in Margin = {
        Additional Margin for Escalation = OS
        Stage-in Threshold               = OS
        First Stage-in Time              = OS
    }
    Provisioning Start Retry Time = OS  ←  実行ホスト起動失敗時の再スケジューリング待ち時間
  Request Statistical Information:
:
```

smgr(1M)コマンドの **stop waiting_retry** サブコマンドを実行することにより、再スケジューリング待ちが解除されます。

```
#smgr -P m
Smgr: stop waiting_retry executionhost = <hostname>
```

- hostnameにはプロビジョニングの実行ホスト名を指定します。
- 操作員権限以上の権限が必要です。

hostname に指定したプロビジョニングの実行ホストに対するテンプレート指定でのリクエストのスケジューリングを再開します。

4.16.3 プロビジョニング時の実行ホストのスケジューリング

OpenStack と連携したプロビジョニング環境の、仮想マシン（VM）によるプロビジョニングとベアメタルサーバによるプロビジョニング時、リクエストはテンプレートを指定して投入します。プロビジョニング時の実行ホストのスケジューリングは、通常の実行ホストと同じです

この際の仮想マシン（VM）の起動・停止にかかる時間は Elapse マージンに含まれるように設定し、ベアメタルサーバの起動・停止にかかる時間は、テンプレートの起動見込み時間と停止見込み時間で設定します。

テンプレートの起動見込み時間と停止見込み時間の設定の詳細については、*NQSV* 利用の手引*[*管理編*]* を参照してください。

リクエストを仮想マシン（VM）で実行する場合、仮想マシン（VM）の起動後リクエストが実行さ

れ、リクエストの実行終了後、仮想マシン（VM）を停止します。リクエストをベアメタルサーバで実行する場合、ベアメタルサーバの起動後リクエストが実行され、リクエストの実行後ベアメタルサーバを停止します。

OpenStack と連携したプロビジョニング環境により起動された仮想マシン（VM）やベアメタルサーバの停止失敗時には、該当ホストをスケジューリングから除外します。スケジューリングから除外した実行ホストは、 **sstat**(1)コマンドの**-E --hw-failure** オプションで確認可能です。

```
$ sstat -E --hw-failure
ExecutionHost Status             V
--------------- ---------------- -
executionhost1 EXCLUDED          -
```

スケジューリングから除外された実行ホストは、問題解決後、当該ノード上の JSV を（JobManipulator とバインドしている）全てのキューからアンバインドした後、再度いずれかのキューにバインドすることによりアサイン対象に戻すことができます。

仮想マシン（VM）の実行ホストは、省電力機能の対象ですが、ベアメタルサーバの場合、省電力機能の対象外です。ベアメタルサーバは省電力機能とは無関係にその実行ホストにアサインされたリクエストの実行開始/終了時に、各々起動/停止されるからです。

ベアメタルサーバが省電力機能の対象外になっていることは **sstat**(1)コマンドの**-E --eco-status** オプションで確認可能です。

```
$ sstat -E --eco-status
ExecutionHost   EcoStatus StateTransitionTime OFF(D) ACCUM
--------------- --------- ------------------- ------ -----
BareMetalhost   EXCLUDED  2016-07-13 09:07:30      0     0
```

仮想マシン（VM）やベアメタルサーバの場合、優先リクエストの割込み位置として next_run のみサポートします。
仮想マシン（VM）やベアメタルサーバで実行されるリクエストは、**smgr**(1M)コマンドによるサスペンドはできません。

4.16.4 ベアメタルサーバ上のリクエストのステージアウト待ち時間

実行ホストがベアメタルサーバの場合、ステージアウトは他のリクエストの実行開始と並行して実

施せず、リクエストのステージアウトが完了したあと、ベアメタルサーバを再起動し、後続のリクエストを実行開始します。そのため、リクエストのステージアウトにかかる時間（ステージアウト待ち時間)を設定することにより､ステージアウト時間を考慮してスケジューリングすることができます。

ステージアウト待ち時間は、**smgr**(1M)コマンドの **set queue wait_stageout** サブコマンド により設定します。

```
#smgr -P m
Smgr: set queue wait_stageout = <second> <queue-name>
```

- *second*にステージアウト待ち時間を設定します。
- 単位は秒です。
- 既定値は、0 です。
- 操作員権限が必要です。

ベアメタルサーバで実行するリクエストの実行終了予定時刻からステージアウト待ち時間を空けた時刻以降にベアメタルサーバを再起動するものとして後続リクエストをアサインします。設定したステージアウト待ち時間は、**sstat** (1)コマンドの**-Q -f** オプションで確認できます。

4.17 Dockerと連携したプロビジョニング

4.17.1 機能概要

Docker と連携したプロビジョニング環境として、コンテナによるプロビジョニングをサポートしています。Docker と連携したプロビジョニングの設定方法の詳細については、*NQSV* 利用の手引*[*管理編*]* を参照してください。Docker と連携したプロビジョニングのリクエスト投入方法の詳細については、*NQSV* 利用の手引*[*操作編*]* を参照してください。

JobManipulator は、プロビジョニングのコンテナに対するスケジューリングを行います。

4.17.2 実行ホスト起動失敗時の再スケジューリング待ち時間設定

Docker と連携したプロビジョニング環境により起動された、コンテナの起動失敗時には、当該実行ホストにアサインしているリクエストを再スケジューリングし、再スケジューリング後のアサイン状況に応じてリクエストの実行開始のために再度起動をリトライします。

再スケジューリングの待ち時間を設定し、起動失敗したプロビジョニングの実行ホストは起動に失敗したテンプレートでの再スケジューリングを一定時間待ちます。この時間を、実行ホスト起動失敗

時の再スケジューリング待ち時間といいます。この再スケジューリング待ち時間は、*4.17 OpenStack と連携したプロビジョニング*の場合と同様です。

このようなテンプレートを指定した起動の失敗の原因として､テンプレートの不正が考えられます。また、その場合、起動のリトライが再度失敗する可能性があります。

本機能により、起動失敗したテンプレートによる再スケジューリング待ち時間を設定し、その間にメンテナンスを実施し実行ホストが起動失敗を繰り返すことを防止することが可能になります。

実行ホスト起動失敗時の再スケジューリング待ち時間の設定、設定値の確認、および解除コマンドの詳細説明は、*4.17.2 実行ホスト起動失敗時の再スケジューリング待ち時間設定*を参照してください。

4.17.3 プロビジョニング時の実行ホストのスケジューリング

Docker と連携したプロビジョニング環境の、コンテナによるプロビジョニング時、リクエストはテンプレートを指定して投入します。プロビジョニング時の実行ホストのスケジューリングは、通常の実行ホストと同じです

この際のコンテナの起動・停止にかかる時間は Elapse マージンに含まれるように設定します。テンプレートの起動見込み時間と停止見込み時間の設定の詳細については、*NQSV* 利用の手引*[*管理編*]* を参照してください。

リクエストをコンテナで実行する場合、コンテナの起動後リクエストが実行され、リクエストの実行終了後、コンテナを停止します。

Docker と連携したプロビジョニング環境により起動されたコンテナの停止失敗時には、該当ホストをスケジューリングから除外します。スケジューリングから除外した実行ホストは、 **sstat**(1)コマンドの**-E --hw-failure** オプションで確認可能です。

```
$ sstat -E --hw-failure
ExecutionHost Status            V
--------------- ---------------- -
executionhost1 EXCLUDED         -
```

スケジューリングから除外された実行ホストは、問題解決後、当該ノード上の JSV を（JobManipulator とバインドしている）全てのキューからアンバインドした後、再度いずれかのキューにバインドすることによりアサイン対象に戻すことができます。

コンテナの実行ホストは、省電力機能の対象です。

4.18 初回ステージイン時間の設定機能

ファイルステージングを行うリクエストがスケジューラマップの先頭付近にアサインされた場合、ステージングが間に合わず開始予定時刻が取り消される可能性があります。そこで、初回ステージングにかかる時間として初回ステージイン時間をスケジューラ単位で設定可能とします。JobManipulator はその時間分ステージングがかかるものとしたアサインを行います。

初回ステージイン時間の間にステージングが終われば、開始予定時刻は取り消されません。初回ステージイン時間は、**smgr**(1M)コマンドの **set stage-in_margin first_stage-in_time** サブコマンドにより設定します。

```
#smgr -P m
Smgr: set stage-in_margin first_stage-in_time = <value>
```

- valueに初回ステージイン時間を設定します。
- 単位は、秒です。
- 既定値は0です。
- 操作員権限以上の権限が必要です。
- 設定変更時には、設定変更以降のアサインで有効になります。

設定値は、**sstat**(1)コマンドの**-S -f** オプションにより確認可能です。

```
$ sstat -S -f
:
    JobManipulator Version   = R1.00
            :
    Keep Forward Schedule    = OS
    Stage-in Margin = {
        Additional Margin for Escalation = OS
        Stage-in Threshold = OS
        First Stage-in Time = OS
    }
:
```

4.19 プレステージング機能

4.19.1 機能概要

リクエストのアサインやエスカレーション時に多数のリクエストで同時にステージングが発生することによるファイルシステムへの負荷を低減するための機能です。また、リクエストがアサインされてから実行開始までのステージング回数を減らすために、ステージングなしにリクエストのアサインができる機能をサポートします。

リクエストアサイン後、ステージングは、**smgr**(1M)コマンドの **set stage-in_margin stage-in_threshold** サブコマンドで設定されたステージイン開始時間閾値に入ったら実施されます。

4.19.2 ステージイン開始時間閾値の設定

ステージイン開始時間閾値は、**smgr**(1M)コマンドの **set stage-in_margin stage-in_threshold** サブコマンドにより設定します。

```
#smgr -P m
Smgr: set stage-in_margin stage-in_threshold = <value>
```

- valueにステージイン開始時間閾値を設定します。単位は、秒です。
- 既定値は0です。0の場合は、リクエストがスケジューラマップにアサインされると直ちにステージングが開始されます。
- 操作員権限以上の権限が必要です。
- 設定変更時には、設定変更以降のアサインで有効になります。

設定値は、**sstat**(1)コマンドの**-S -f** オプションにより確認可能です。

```
$ sstat -S -f
:
    JobManipulator Version   = R1.00
    :
    Keep Forward Schedule    = 0S
    Stage-in Margin = {
        Additional Margin for Escalation = 0S
        Stage-in Threshold = 0S
        First Stage-in Time = 0S
    }
:
```

4.20 実行ホストの詳細情報表示機能

sstat(1)コマンドの-E -f オプションで実行ホストの詳細情報が表示可能です。sstat(1) コマンドの-E、-E --eco-status -f、-E --hw-failure オプションで各々表示される情報が一括表示されます。

sstat -E -f を実行したイメージは以下のようになります。

```
$sstat -E -f
Execution Host: Host1
    CPU Number Ratio  = 1.000000
    CPU Number Ratio of RSG = {
        RSG 0 = 1.000000
    }
    Memory Size Ratio = 0.000000
    Memory Size Ratio of RSG = {
        RSG 0 = 0.000000
    }
    Eco Status = {
        Status                = EXCLUDED
        State Transition Time = 2017-06-20 10:49:36
        Exclude Reason        = HW_FAILURE
        DC-OFF Times (Day)    = 0
        DC-OFF Times (ACCUM)  = 0
    }
    Hardware Failure = {
        Status = CPUERR
    }

Execution Host: Host2
    CPU Number Ratio = 1.000000
    CPU Number Ratio of RSG = {
        RSG 0 = 1.000000
    }
    Memory Size Ratio = 0.000000
    Memory Size Ratio of RSG = {
        RSG 0 = 0.000000
    }
    Eco Status = {
        DC-OFF Times (Day)    = 0
        DC-OFF Times (ACCUM)  = 0
    }
```

```
    Hardware Failure = {
        Status = EXCLUDED
        Exclude Reason = VE_DEGRADATION
        VE Degradation = YES
    }
```

sstat -E -f -a を実行したイメージは下記のようになります。アンバインドされているホストに対しては、Hardware Failure 欄は表示されません。この例では Host3 がアンバインドされているホスト、Host4 はバインドされているホストです。

```
$sstat -E -f -a
Execution Host: Host3
    CPU Number Ratio  = 1.000000
    CPU Number Ratio of RSG  = {
        RSG 0 =  1.000000
        ..................
        RSG 31 =  1.000000
    }
    Memory Size Ratio = 0.000000
    Memory Size Ratio of RSG = {
        RSG 0 =  0.000000
        ..................
        RSG 31 =  0.000000
    }
    Eco Status = {
        Status                = EXCLUDED
        State Transition Time = 2017-06-20 10:49:36
        Exclude Reason        = UNBIND
        DC-OFF Times (Day)    = 0
        DC-OFF Times (ACCUM)  = 0
    }
Execution Host: Host4
    CPU Number Ratio  = 1.000000
    CPU Number Ratio of RSG  = {
        RSG 0 =  1.000000
        ..................
        RSG 31 =  1.000000
    }
    Memory Size Ratio = 0.000000
```

```
    Memory Size Ratio of RSG = {
        RSG 0 =  0.000000
        ..................
        RSG 31 =  0.000000
    }
    Eco Status = {
        DC-OFF Times (Day)    = 0
        DC-OFF Times (ACCUM)  = 0
    }
    Hardware Failure = {
        Status = EXCLUDED
        Exclude Reason = VE_DEGRADATION
        VE Degradation = YES
    }
```

4.21 最小ネットワークトポロジーノードグループ選択機能

4.21.1 機能概要

MPI ジョブの通信をより高速に行うことを目的として、ノードが属するネットワークトポロジーノードグループ数が最小となるようにノードを選択します。

JobManipulator のアサインポリシーは、リクエストが最も早い時間に実行できるノードを優先して選択します。そのため、ネットワークトポロジーを考慮したアサイン設定でも、ネットワークスイッチを跨いでアサインされる可能性があります。

例えば、下記のように Req1→Req2→Req3→Req4→Req5 の順でリクエストを投入した場合、Req3 は、2 つのネットワークスイッチを跨いでスケジューリングされます。JobManipulator の既定のアサインポリシーでは、ジョブを空きノードでより早く実行することを優先します。

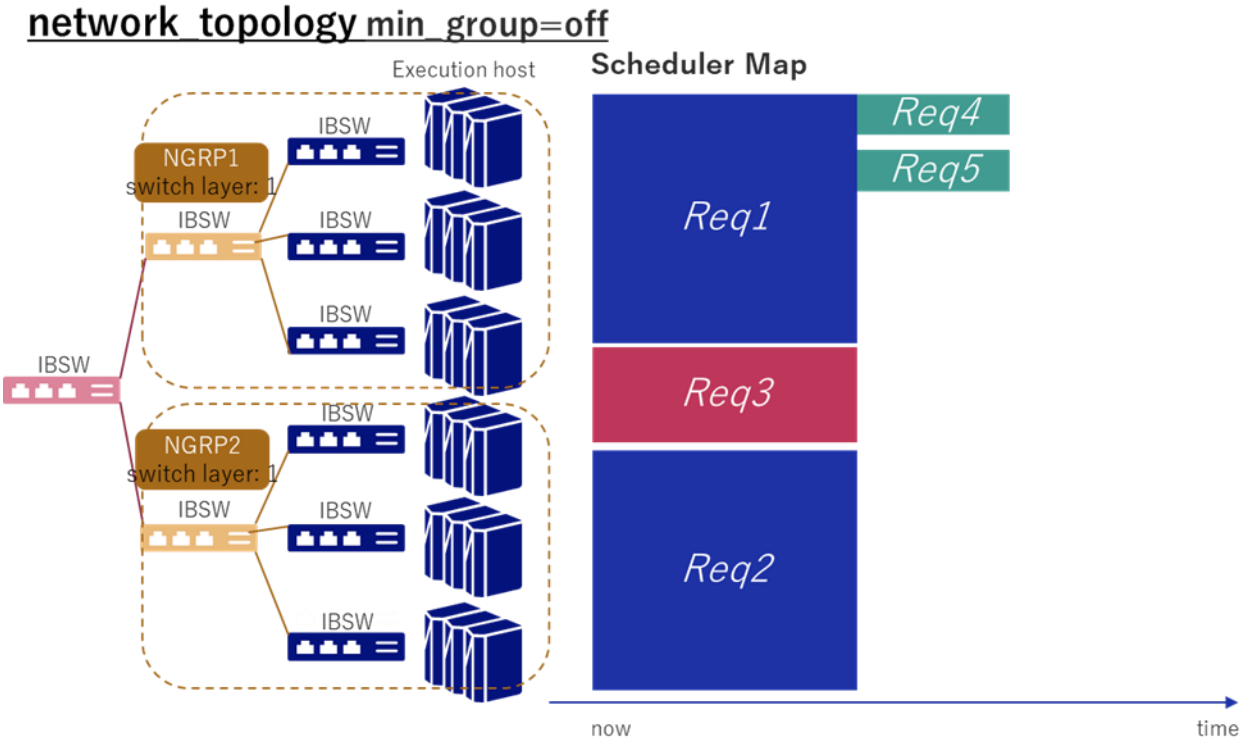


図 4-3：ジョブ実行を優先したスケジューリングの例

最小ネットワークトポロジーグループ選択機能では、開始時刻を遅らせても最小のネットワークトポロジーグループにおさめて選択します。

たとえより早い時刻にリクエストをアサイン可能なノードがあったとしても、各リクエストにとって最小となるネットワークトポロジーグループ数以内となるようにノードを選択します。

前述の例で本機能を適用した場合は、下記のようにスケジューリングされます。Req3 は 1 グループが最小となるため、1 グループ内での選択となるようにノードを選択し、後方の時刻へアサインします。Req4 と Req5 は前方の空き領域にスケジューリングされます。

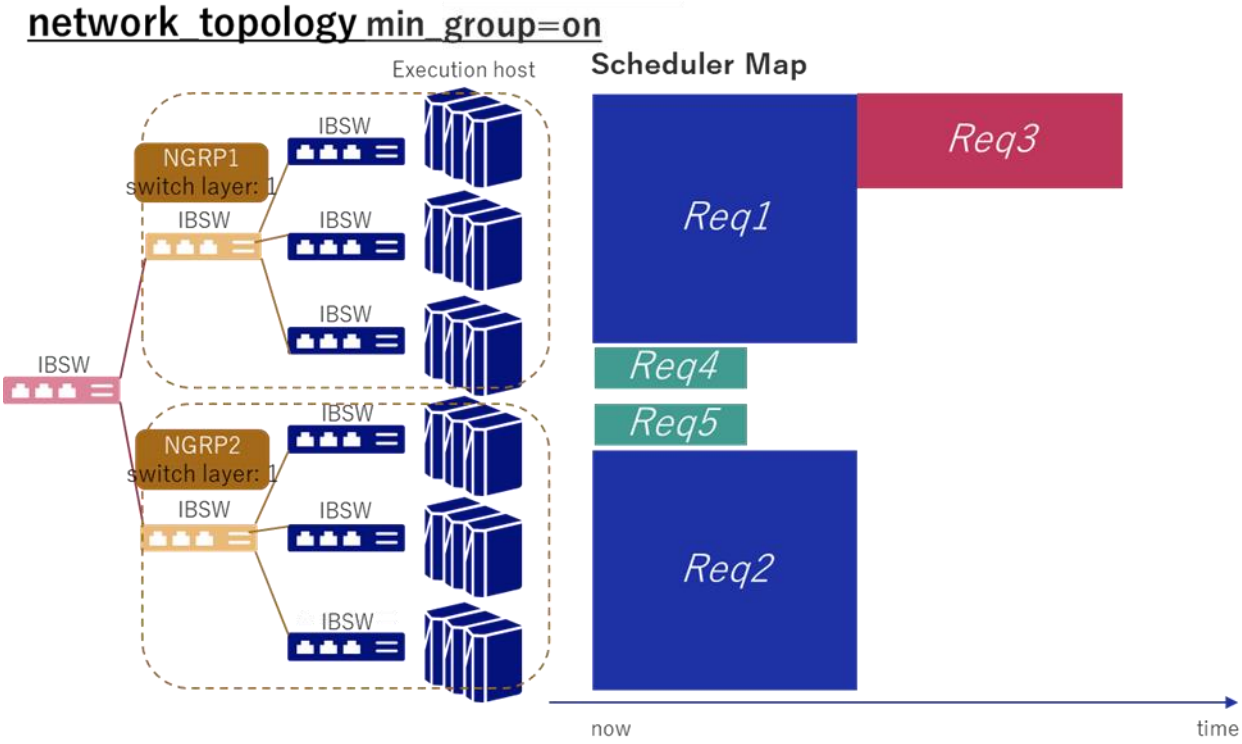


図 4-4：ネットワークトポロジーを優先したスケジューリングの例

- 本機能は、MPI ジョブの通信をより高速に行うことを目的としています。したがって、ジョブトポロジーが Distribute のリクエストは対象となりません。
- 本機能は **qmgr**(1M) の **create node_group switch_layer** あるいは **set node_group switch_layer** サブコマンドで指定する スイッチレイヤーのレベルの最も小さいネットワークトポロジーノードグループに適用されます。
- ジョブコンディションを使ってジョブがアサインされるジョブサーバが指定された場合は、ジョブコンディションを優先するため、当該リクエストが最小ネットワークトポロジーノードグループを選択することは保証しません。
- ジョブの順序を重視する場合、本機能を ON にする際はジョブの実行順序について十分検討の上設定してください。特定の条件下において投入したジョブをあとから投入したジョブが

追い越して実行しやすくなることがあります。例えば 1 グループを 8 ノードで構成し、大半が 4 ノードジョブの実行の場合、ノードの空きはジョブが終わる毎に 4 ノード単位で生じます。したがって、5 ノード以上のジョブを追い越してあとから投入した 4 ノードジョブを実行することがあります。ただし、5 ノード以上のジョブは最初に決定した開始予定時刻より実行開始が遅くなることはありません。

4.21.2 設定

最小ネットワークトポロジーノードグループ機能は、**smgr**(1M)コマンドの **set queue network_topology min_group** サブコマンドにより、キュー単位に設定します。設定には操作員以上の権限が必要です。 キューのデフォルトの設定は off となります。

設定例)

```
# smgr -P o
Smgr: set queue network_topology min_group = on bq1
```

上記では、bq1 に投入されたリクエストはすべて最小ネットワークトポロジーノードグループを選択するようにスケジューリングされます。

リクエストの最小グループ数を求めるために、ノードグループあたりの実行ホスト数を設定します。ノードグループあたりの実行ホスト数は、**smgr**(1M)コマンドの **set queue network_topology host_per_group** サブコマンドにより、キュー単位に設定します。設定には操作員以上の権限が必要です。

設定例)

```
# smgr -P o
Smgr: set queue network_topology host_per_group = 512 execution_queue
```

たとえばノードグループあたり 512 ノードに設定した場合、512 ノード以下を使用するリクエストは必ず 1 グループ内に収まるようにアサインします。512 ノード未満のグループが存在する場合や、障害や省電力の要因で 512 ノードを下回る場合でも 1 グループを 512 ノード 1 グループとしてスケジューリングを行います。

ノードグループあたりの実行ホスト数を明示的に設定することを推奨します。
設定しない場合、既定値ではスケジューリング毎に自動的に選択して使用します。
既定値において自動的に選択する場合、リクエストの投入キューに BIND されているスケジューリング対象となる実行ホストの、ネットワークトポロジーノードグループ毎の実行ホスト数のうち、最も出現回数が多い実行ホスト数を選択します。
障害遭遇やキュー構成によって実行ホスト数が変わる影響を避けたい場合は、ノードグループあたりの実行ホスト数を明示的に設定してください。

設定値は、**sstat**(1)コマンドの**-Q -f** オプションにより確認可能です。

```
#sstat -Q -f
Execution Queue: jmq0
    Queue Type            = Normal
    Schedule Time         = DEFAULT
              :
    Network Topology Control = {
      Network Topology Minimum Scheduling = ON
      Hosts per group = 512
    }
:
```

また、**sstat**(1)コマンド -f オプションにてリクエスト単位の値が確認可能です。

```
#sstat -f
Request ID: 1467.bsv0
    Request Name = batch job 1
    User  Name = user1
              :
    Network Topology Control:
      Network Topology Minimum Scheduling = ON
      Hosts per group = 512
      Jobs per host   = 1 (Default)
:
```

4.22 FIFOスケジューリング

FIFO スケジューリングとは、リクエストを投入した順番通りに実行するスケジューリングです。利用するには、システム管理者が **smgr**(1M)コマンドの **set queue schedule_type** サブコマンドで **fifo** に設定することにより利用できます。FIFO スケジューリングの設定はキュー単位です。

FIFO スケジューリングに設定したキューでは、投入されたリクエストは、は、その時点で実行に必要なリソースが確保できれば、すぐに開始時刻が決定します。また、先に投入されたリクエストの開始時刻が決定していれば、次のリクエストはスケジューリングされ、実行に必要なリソースが確保できれば開始時刻が決定します。

予約区間や、時刻指定、ワークフローにより、すぐに開始ができないリクエストは、リソースが確

保でき、実行開始が可能な時刻となるまで実行開始時刻は表示されません。

例外としては、連携リクエストのために HELD 状態となっているリクエスト、リクエストを **qhold**(1)コマンドによりホールドしたリクエスト、**qalter**(1)コマンドの **-a** オプションにより WAITING 状態となっているリクエストは、後から投入したリクエストが追い越して実行します。

FIFO スケジューリングを適用すると、ピックアップ時の追い越し制御は無効となります。また、FIFO スケジューリングでは、初回ステージイン時間との併用はできません。

FIFO スケジューリングを行う際は、FIFO スケジューリングを行うキューと、Backfill スケジューリングを行うキューで対象となる実行ホストが重複しないようにキューを設計してください。

4.23 クラウドバースティング機能

4.23.1 機能概要

クラウドバースティング機能とは、HPC クラスタにおいて実行待ちのジョブが一定以上存在するとき、クラウドのコンピューティング資源へ当該ジョブをバースティングして実行させ、ジョブの TAT の向上を図る機能です。機能イメージは下記のようになります。

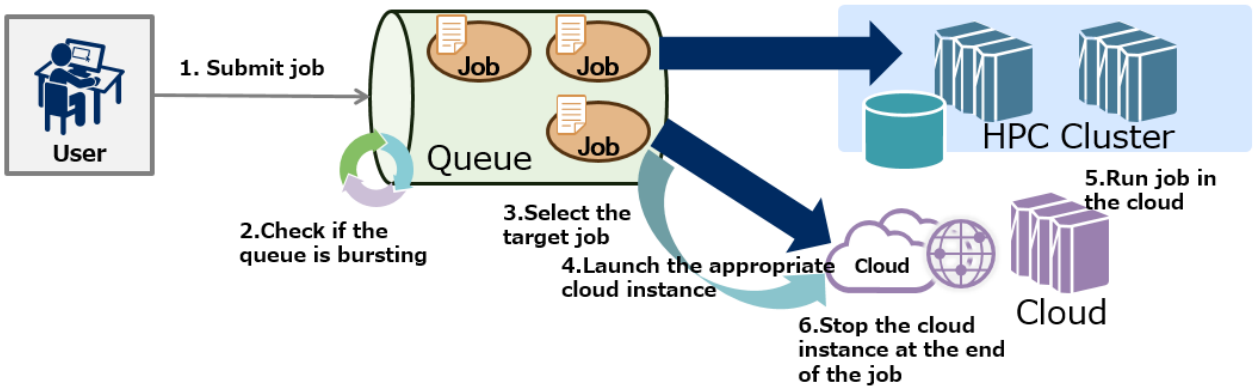


図 4-5：クラウドバースティングの機能イメージ

クラウドバースティングの全体の流れは次のとおりです。

1. ユーザは「バースティング対象可」を指定しキューにリクエストを投入します。
2. NQSV は、ポリシーに従ってバースティングを行うかをスケジューリングインターバルごとに判断します。
3. 各リクエストのバースティングプライオリティを計算し、バースティング対象リクエストを決定します。
4. NQSV は、バースティング対象リクエストの要求量に近いイメージを選択し、クラウドを起動します。
5. バースティング対象リクエストは、起動したクラウド環境上で実行します。
6. NQSV は、バースティング対象リクエストの終了に応じて順次クラウド環境を停止します。

クラウドの起動イメージは下図のようになります。

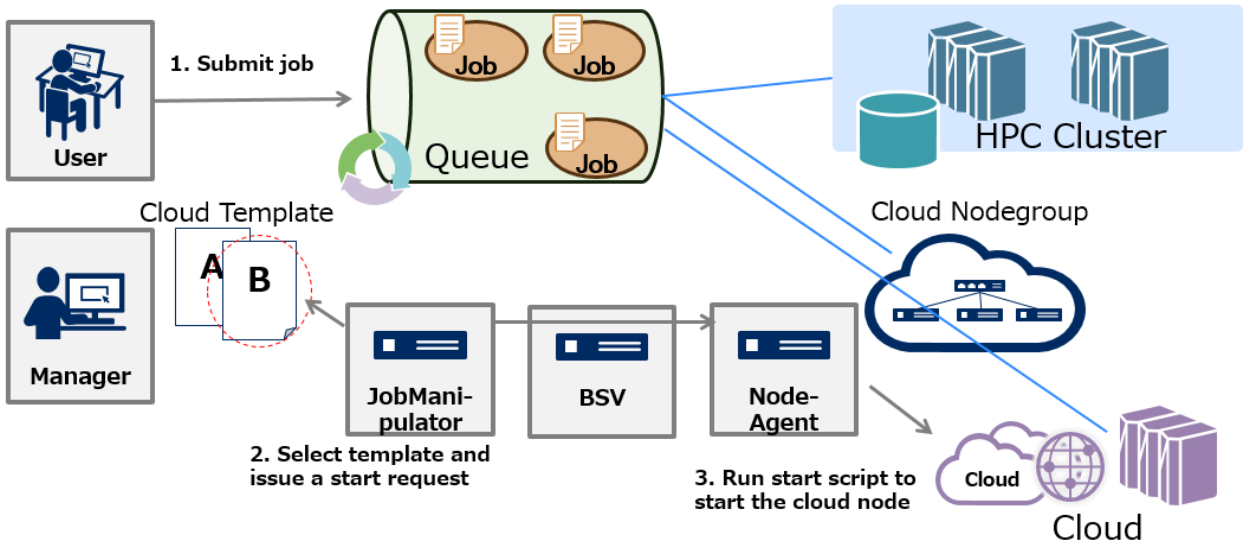


図 4-6：クラウド起動のイメージ

クラウド起動の流れの詳細は下記の通りです。

1. JobManipulator はポリシーに従ってリクエストごとにクラウドテンプレートとクラウドノードグループを選択し、必要な量のクラウドの起動を要求します。
2. ノード管理エージェントは要求されたクラウドテンプレートとクラウドノードグループの情報に従ってシェルスクリプト経由でクラウドを起動します。
3. 起動が成功すると、BSV 上でクラウド 1 ノードに対して 1 つ JSV 番号が払い出され、実行ホストがアタッチされます。
4. 予め JSV が自動的に起動するようにクラウドの OS イメージが作成されているため、クラウドインスタンスが起動すると、JSV が自動的に起動します。
5. JSV と BSV の TCP リンクが確立する（JSV が LINKUP する）と、指定されたクラウドバースティングノードグループとバインドされているキューと JSV が自動的にバインドされます。
6. リクエストは選択されたテンプレートで起動したクラウドにスケジューリングされて実行します。
7. リクエストが終了し次第、JobManipulator はクラウドに対して停止を要求します。ノード管理エージェントは要求された情報に従ってシェルスクリプト経由でクラウドを停止します。
8. 停止が完了すると、BSV 上でのクラウドノードおよび JSV は自動的にキューからアンバインドされて、実行ホストがデタッチされて管理対象外となります。

NQSV では以下のクラウド環境をサポートしています。

・Amazon 社 AWS(Amazon Web Service)

・Microsoft 社 Azure

・Oracle 社 OCI(Oracle Cloud Infrastructure)

4.23.2 クラウドバースティング 設定の概要

クラウドバースティングを利用するための設定の流れを説明します。

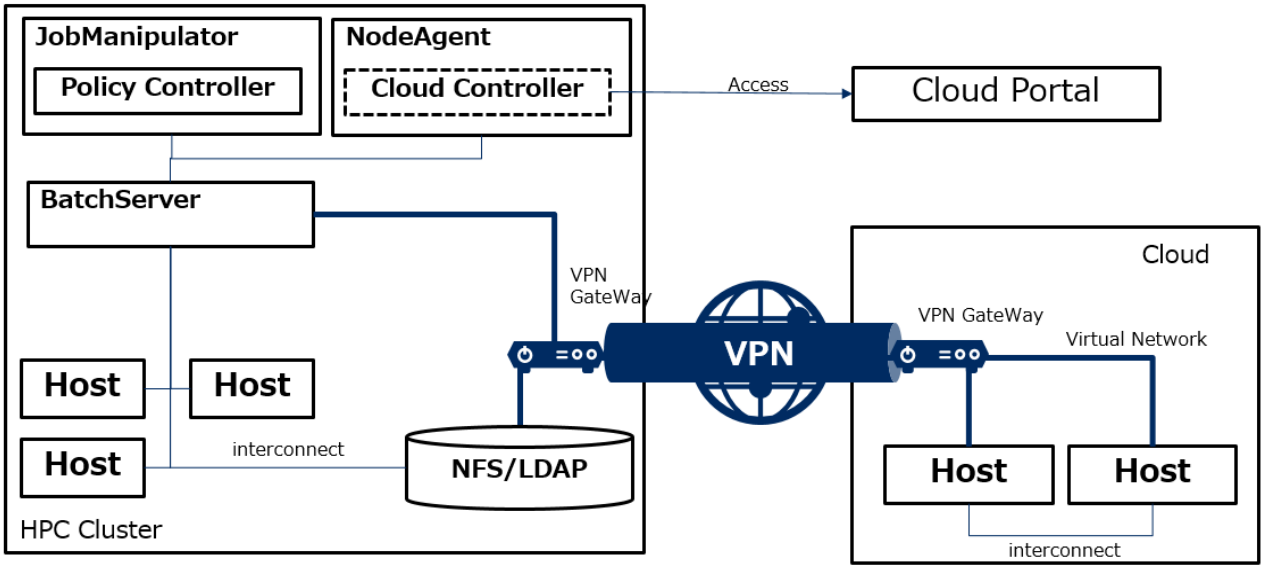


図 4-7：クラウドバースティングの構成

クラウドバースティングは、ノード管理エージェントを経由してクラウドポータルにアクセスし、クラウドを起動します。ノード管理エージェントの CloudController ではシェルスクリプトにてクラウドの起動・監視・停止を行うことができますので、管理者はクラウド環境にあわせてシェルスクリプトをカスタマイズすることにより、クラウドの起動・監視・停止を行います。

クラウドバースティングを行うために、まず、下図のようにクラウドバースティング用テンプレートとクラウドノードグループを作成する必要があります。

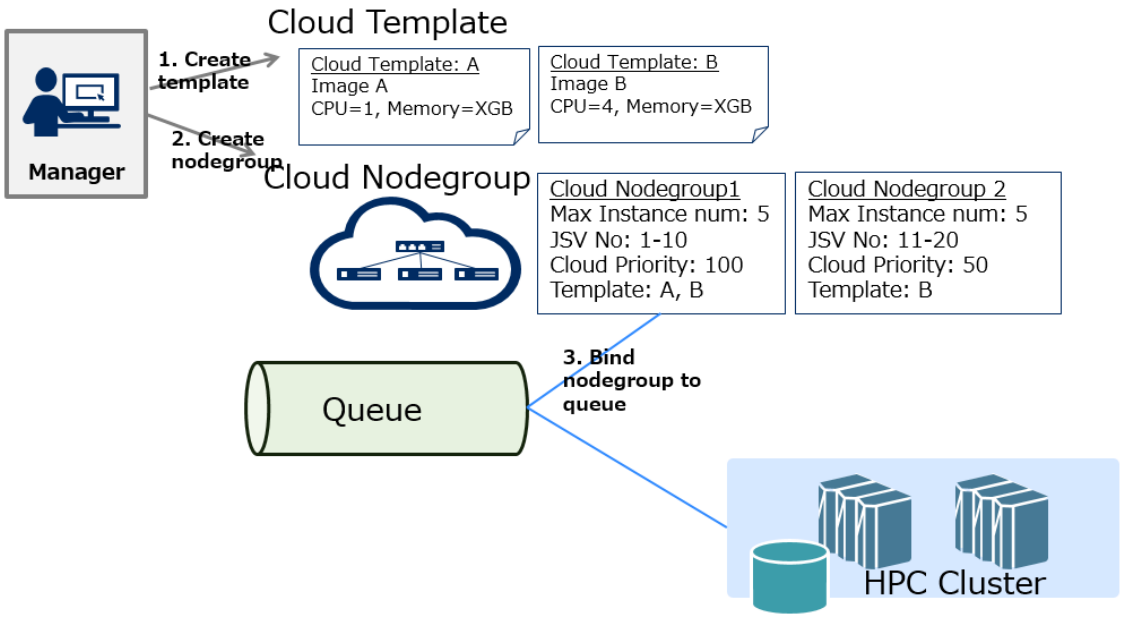


図 4-8：テンプレートとノードグループの設定

1. [管理者] クラウド環境で利用可能なイメージを**クラウドバースティング用テンプレート**として作成します。
2. [管理者] クラウド環境を定義する**クラウドバースティングノードグループ**を作成します。
3. [管理者] クラウドバースティングノードグループとキューをバインドします。

クラウドバースティングノードグループとバインドしたキューにリクエストを投入することで、JobManipulator によってポリシーに従ってバースティングを行うか判断が行われ、クラウドバースティングノードグループに追加されたテンプレートから適切なイメージが選択されます。

キューは、バッチキューあるいは会話キューが利用可能です。

以降の節ではクラウドバースティング機能を利用するための詳細な設定方法やジョブの投入方法等を説明します。

4.23.3 クラウドバースティング用テンプレートの設定

クラウド環境で起動するインスタンスの OS イメージや資源の情報を、クラウドバースティング用テンプレート（以降テンプレートと呼ぶ）として定義します。

テンプレートをクラウドバースティングノードグループに登録することにより、NQSV は自動的にジョブ要求リソース量と最も近いテンプレートを選択して、そのテンプレートが示すクラウドの環境下でジョブを実行します。ユーザはリクエスト投入時にテンプレートを指定する必要はありません。

テンプレートの定義

テンプレートはシステム管理者があらかじめ定義します。テンプレートは複数定義することができ

ます。各テンプレートには、クラウド環境の情報として、以下の要素が定義できます。

要素名	定義
テンプレート名	テンプレートの名前。 最大 47 文字までの任意の名前を指定できます。 名前に、空白、並びに、「"(ダブルクォーテーション)」、「@」を含むことはできません。
イメージ名 (image)	起動するクラウドインスタンスの OS イメージ名。 クラウドインスタンスの起動に利用可能なイメージ名を指定します。 最大 127 文字まで定義可能です。
CPU数 (cpu)	割り当てる CPU の数。 １以上の整数を指定します。 この値は、このテンプレートが指定されたリクエストのジョブ毎の CPU 数制限値としても扱われます。
メモリサイズ (memsz)	割り当てるメモリサイズ。 １以上の整数で、単位（B, KB, MB, GB, TB, PB,EB）を付けて指定します。 この値は、このテンプレートが指定されたリクエストのジョブ毎のメモリサイズ制限値としても扱われます。
GPU数 (gpu)	割り当てる GPU の数。 0 以上の整数を指定します。GPU の割り当てを行わない場合は 0 を指定します。既定値は 0 です。 この値は、このテンプレートが指定されたリクエストのジョブ毎の GPU 数制限値としても扱われます。
VE数 （ve）	割り当てる VE の数 0 以上の整数を指定します。VE の割り当てを行わない場合は 0 を指定します。既定値は 0 です。 この値は、このテンプレートが指定されたリクエストのジョブ毎の VE 数制限値としても扱われます。
起動見込み時間 (boot_timeout)	クラウドインスタンスを起動して JSV が LINKUP してくるまでのタイムアウト時間。 秒単位で、1～2147483647 の整数で指定します。既定値は 300 秒です。
停止見込み時間 (stop_timeout)	クラウドインスタンスを停止するタイムアウト時間。 秒単位で、1～2147483647 の整数で指定します。既定値は 300 秒です。
独自定義 (custom)	起動環境として独自に定義した情報が有る場合は記載します。

	最大 400 文字まで指定できます。
コメント (comment)	テンプレートについてのコメントを記載できます。 最大 255 文字まで指定できます。

テンプレートの操作

(1) テンプレートの作成

テンプレートを新たに作成するには、qmgr(1M) を管理者権限で起動し、以下の"create cloud_template"サブコマンドを使用します。

```
create cloud_template=<template_name> image=<OS_image> cpu=<cpunum> memsz=<memory_size> [gpu=<gpunum>] [ve= <venum>]
 [custom="<custom_define>"] [comment="<comment>"] [boot_timeout=<timeout>] [stop_timeout=<timeout>]
```

AWS_MEDIUM_RHEL8 という名前で、以下の設定で、テンプレートを作成する例を挙げます。

- 起動するインスタンスのイメージ： ami-12ab34cd56ef78gh9
- 割り当てCPU数： 8
- 割り当てメモリ ： 64GB
- コメント： For AWS medim instance

```
$ /opt/nec/nqsv/bin/qmgr -Pm
Mgr: create cloud_template=AWS_MEDIUM_RHEL8 image=ami-12ab34cd56ef78gh9 cpu=8 memsz=64gb comment="For AWS medim
instance"
Cloud_Template AWS_MEDIUM_RHEL8 created.
```

(2) テンプレートの削除

作成済みのテンプレートを削除するには、qmgr(1M) を管理者権限で起動し、以下の"delete cloud_template"サブコマンドを使用します。ただし、対象のテンプレートを使用しているリクエストが存在する場合は削除することができません。

```
delete cloud_template =<template_name>
```

(3) テンプレートの編集

作成済みのテンプレートの内容を編集（上書き）するには、qmgr(1M) を管理者権限で起動し、以下の"set cloud_template"サブコマンドを使用します。

```
set cloud_template image=<OS_image> <template_name>
set cloud_template cpu=<cpunum> <template_name>
set cloud_template memsz=<memory_size> <template_name>
set cloud _template gpu=<gpunum> <template_name>
set cloud _template ve=<venum> <template_name>
set cloud_template custom="<custom_define>" <template_name>
set cloud_template comment="<comment>" <template_name>
set cloud_template boot_timeout=<timeout> <template_name>
set cloud_template stop_timeout=<timeout> <template_name>
```

(4) テンプレートのロック・ロック解除

上記の"delete cloud_template"および"set cloud_template"サブコマンドを実行するにあたり、一時的にテンプレートを使用できないようにロックすることができます。

ロックしたテンプレートはリクエストをクラウド環境で実行するときに選択されなくなります。既にクラウド環境で実行しているリクエストについては何も影響しません。

例えば、テンプレートを編集したい場合、まず対象のテンプレートをロックし、新たなリクエストのテンプレートの使用を禁止した上で、テンプレートを編集します。編集終了後、テンプレートのロックを解除し、使用可能とします。

テンプレートの使用をロックするには、qmgr(1M) を管理者権限で起動し、以下の"lock cloud_template"サブコマンドを使用します。

```
lock cloud_template =<template_name>
```

また、テンプレートのロックを解除するには、qmgr(1M) を管理者権限で起動し、以下の"unlock cloud_template"サブコマンドを使用します

```
unlock cloud_template =<template_name>
```

テンプレートの表示

システムに定義されたテンプレートの情報を参照するには qstat --template を使用します。

【表示例】

```
$qstat --template
[Cloud Template]
================================================================================
```

```
Template   L Image      CPU  Memory GPU  VE  Custom              Comment
---------- - ---------- ---- ------ ---- --- ------------------- -------------------
AWS_MEDIUM - ami-12ab34    8  64.0G    0   0 (none)              For AWS medim insta*
```

より詳細なテンプレートに関する情報を参照するには qstat --template -f を使用します。

【表示例】

```
$qstat --template -f
Cloud Template: AWS_MEDIUM_RHEL8
  Lock State   = UNLOCK
  OS Image     = ami-12ab34cd56ef78gh9
  CPU Number   = 8
  Memory Size  = 64GB
  GPU Number   = 0
  VE Number    = 0
  Boot Timeout = 300
  Stop Timeout = 300
  Custom       = (none)
  Comment      = For AWS medim instance
```

4.23.4 クラウドバースティングノードグループの設定

クラウドバースティングノードグループ(以降、ノードグループと呼ぶ)はクラウドインスタンスを起動するために必要なノードグループであり、起動するインスタンス数(JSV 番号の範囲にて設定)、インスタンス起動用のテンプレート、クラウド環境のネットワーク名等の情報を定義します。このノードグループをキューにバインドすることで、当該キューに投入されたリクエストがクラウドバースティング対象可であれば、当該ノードグループで定義されたテンプレートと JSV 番号でクラウドインスタンスを起動します。そのインスタンス上で JSV を起動してジョブを実行します。

ノードグループの作成

ノードグループの作成は、qmgr(1M)コマンドの"create node_group"サブコマンドにより type オプションに cloud を指定して行います。

以下に、"cl_ng1"という名前でノードグループを作成する場合の例を示します。

このとき、qmgr(1M)コマンドは管理者権限で起動してください。

```
$ qmgr -Pm
Mgr: create node_group = cl_ng1 type=cloud
Node_group cl_ng1 created.
```

ノードグループへのテンプレートの追加

ノードグループにクラウドバースティング用テンプレートを設定することで、クラウドのインスタンスを起動するときにテンプレートで指定された OS イメージでインスタンスを起動することができます。テンプレートは、ノードグループの作成時に設定することもできますが、qmgr の"edit node_group add template"サブコマンドによって追加することができます。同一ノードグループには最大 20 個のテンプレートが追加できます。テンプレートを追加するとき、qmgr コマンドは管理者権限で起動してください。

```
$ qmgr -Pm
Mgr: edit node_group add template = AWS_MEDIUM_RHEL8 cl_ng1
Add template to Node_group (AWS_MEDIUM_RHEL8).
```

下記のように複数のテンプレートをまとめて追加することもできます。

```
$ qmgr -Pm
Mgr: edit node_group add template =(AWS_MEDIUM_RHEL8.1,AWS_MEDIUM_RHEL8.2) cl_ng1
Add template to Node_group (AWS_MEDIUM_RHEL8.1).
Add template to Node_group (AWS_MEDIUM_RHEL8.2).
```

ノードグループからのテンプレートの削除

ノードグループに追加したテンプレートは qmgr の"edit node_group delete template"サブコマンドによって削除することができます。このとき、qmgr コマンドは管理者権限で起動してください。

```
$ qmgr -Pm
Mgr: edit node_group delete template = AWS_MEDIUM_RHEL8 cl_ng1
Delete template to Node_group (AWS_MEDIUM_RHEL8).
```

下記のように複数のテンプレートをまとめて削除することもできます。

```
$ qmgr -Pm
Mgr: edit node_group delete template =(AWS_MEDIUM_RHEL8.1,AWS_MEDIUM_RHEL8.2) cl_ng1
Delete template to Node_group (AWS_MEDIUM_RHEL8.1).
Delete template to Node_group (AWS_MEDIUM_RHEL8.2).
```

ノードグループへのインスタンスの追加

ノードグループに実行ホストを追加することで、当該ノードグループにて起動できるクラウドイン

スタンスが追加されます。追加する実行ホストは、これからクラウド環境で起動するインスタンスのため、以下の実行ホストを追加することはできません。

・他のノードグループに属している実行ホスト

・キューにバインドされている実行ホスト

・ジョブサーバにアタッチされている実行ホスト

また、他のクラウドバースティングノードグループをノードグループ単位で追加することもできません。

ノードグループへのインスタンスの追加には、qmgr(1M)コマンドの"edit node_group add job_server_id"サブコマンドを使用し、追加する実行ホストのジョブサーバ番号を指定します。ジョブサーバ番号の指定方法は従来の common タイプのノードグループと同様です。

ノードグループからのインスタンスの削除

ノードグループから実行ホストを削除することで、当該ノードグループにて起動できるクラウドインスタンスが削除されます。以下の場合は、削除エラーとなります。

・ジョブが存在する実行ホスト

・クラウドインスタンスの起動中または停止中である実行ホスト

複数の実行ホストをまとめて削除する場合は、1 台の実行ホストの削除に失敗すると、指定された全実行ホストは削除されません。

ノードグループからインスタンスの削除には、qmgr(1M)コマンドの"edit node_group delete job_server_id"サブコマンドを使用し、削除する実行ホストのジョブサーバ番号を指定します。ジョブサーバ番号の指定方法は従来の common タイプのノードグループと同様です。

ノードグループのプライオリティの設定

同一クラウドバースティング用のキューに複数のノードグループをバインドできます。リクエストをクラウドバースティングの対象として実行させる際、どのノードグループを優先して選択するかは、ノードグループのプライオリティで設定できます。ノードグループのプライオリティを設定するには、qmgr(1M)コマンドを操作員以上の権限で起動して、"set node_group priority"サブコマンドを使用します。プライオリティの値は 0～63 の数字で指定します。既定の場合はプライオリティの値が 1 となります。

```
$ qmgr -Po
Mgr: set node_group priority = 10 cl_ng1
Set priority to Node_group (cl_ng1).
```

ノードグループのネットワーク名の設定

ノードグループで設定されているテンプレートで起動したクラウドインスタンスが配置されているネットワーク名をノードグループ単位で設定できます。ネットワーク名の設定は任意です。設定された場合は、当該情報がクラウドインスタンス起動時に起動用スクリプトに環境変数にて渡されます。この設定によって、同一クラウドにおいて異なるネットワーク(例：異なるアベイラビリティゾーンや異なるサブネット)環境でクラウドインスタンスを起動することができます。

ノードグループのネットワーク名を設定するには、qmgr(1M)コマンドを操作員以上の権限で起動して、"set node_group network"サブコマンドを使用します。255 文字までの文字列で指定します。

```
$ qmgr -Po
Mgr: set node_group network = subnet-A cl_ng1
Set network to Node_group (cl_ng1).
```

ノードグループの表示

ノードグループの状態は、qstat(1) コマンドの -G オプションで表示することができます。Type に cloud と表示します。

```
$ qstat -G
NodeGroup       Type      BatchServer     Comment          JSVs BindQueue
--------------- --------- --------------- ---------------- ---- --------------------
cl_ng1          cloud     bsv1.example.co (none)              1 cloud_bq1
```

また、qstat -G に -f オプションつけることにより、各ノードグループの詳細情報を表示することができます。

```
$ qstat -G -f
Node Group: cl_ng1
    Type = cloud
    Comment = (none)
    Bind Queue list = {
      cloud_bq1
    }
    Job Server number list = {
        100
    }
    Lock State = UNLOCK
    Priority = 1
    Instances = Live: 0 / Max: 1
```

```
  Network Name = (none)
  Template = {
    (none)
  }
```

以下は、クラウドバースティングノードグループの独自の情報になります。

Priority　：クラウド環境で実行する際に選択されるノードグループのプライオリティ。

Instances：クラウドインスタンス数。Live は現在起動中のインスタンス数、Max は起動可能な最大インスタンス数です。Max は、ノードグループに追加している実行ホスト数になります。

Network Name：クラウド環境のネットワーク名。

Template：クラウドインスタンスを起動するためのクラウドバースティング用テンプレート。

ノードグループのロック・ロック解除

設定等のミスによりキューにバインドされたノードグループによってクラウドインスタンスを起動できない場合があります。当該ノードグループを一時的に使用不可にするために、ノードグループをロックすることはできます。ロックされたノードグループは、スケジューリングを行う際に選択対象外となります。当該ノードグループにて起動したクラウドインスタンス上で実行しているジョブについては、ロックの影響を受けません。また、クラウドインスタンスの起動や停止、実行中に異常が発生した場合は、起動用スクリプトや停止用スクリプト、監視用スクリプトの返却値によってノードグループがロックされることもあります。スクリプトの詳細は **4.23.5 節**を参照してください。

ノードグループを手動でロックするには、**qmgr(1M)**コマンドを管理者権限で起動し、**"lock node_group"**サブコマンドを使用します。

```
qmgr -Pm
Mgr: lock node_group = cl_ng1
node_grpup cl_ng1 locked.
```

ロックされたノードグループはロック状態が自動的に解除されません。ロック解除するには、**qmgr(1M)**コマンドを管理者権限で起動し、**"unlock node_group"**サブコマンドを使用します。

```
qmgr -Pm
Mgr: unlock node_group = cl_ng1
node_grpup cl_ng1 unlocked.
```

ノードグループのロック・ロック解除は、クラウドバースティングノードグループに対してのみ行え

ます。common タイプのノードグループとネットワークトポロジーノードグループに対しては操作できません。

ノードグループの削除

ノードグループを削除する場合、ノードグループがキューにバインドされていない状態で、**qmgr(1M)** コマンドを管理者権限で起動し、"**delete node_group**"サブコマンドを使用します。

```
$ qmgr -Pm
Mgr: delete node_group = cl_ng1
Node_group cl_ng1 deleted.
```

4.23.5 ノード管理エージェントの設定

管理対象ノードグループの設定

クラウドバースティングノードグループをノード管理エージェントの設定ファイル(/etc/opt/nec/nqsv/nag.cof)に記載することで、当該ノードグループがノード管理エージェントの管理対象となり、当該ノードグループの設定によってクラウドインスタンスの起動、停止および監視を行うことができます。設定方法は下記のようになります。カンマ区切りで複数設定が可能です。

```
CLOUD_NGRP:<cloud_ngrp_name1>,<cloud_ngrp_name2>…
```

設定後、ノード管理エージェントを再起動してください。運用中に上記の設定を変更して systemctl reload を行っても、上記の設定に限っては、ノードエージェントを再起動しなければ、新しい値は適用されません。

クラウドインスタンス起動用スクリプト

インスタンスの起動処理は、システム管理者が作成したシェルスクリプトによって行います。本シェルスクリプトは、以下のパスで配置します。

起動用スクリプト：/opt/nec/nqsv/sbin/cloud_prog/cloud_start.sh

起動用スクリプトで以下の処理を実施する必要があります。

1)クラウドインスタンス起動

スクリプトに指定された環境変数に従って、利用されるクラウド環境の CLI コマンドを実行してクラウドインスタンスを作成して起動します。1 回の起動では 1 インスタンスしか起動できません。

2)インスタンス起動待ち受け

インスタンスが起動されるまで待ち受けます。待ち受けるタイムアウト時間はスクリプトで自由に設

定できますが、クラウドバースティング用テンプレートに設定されている起動見込み時間以内にしてください。

3)起動結果返却

インスタンスの起動結果を標準出力に出力します。

起動用スクリプト実行時には以下の環境変数を設定して起動します。

環境変数	値
NQSV_CLOUD_IMAGE	クラウドインスタンス起動用イメージ名。 クラウドバースティング時に選択したクラウドノードグループのバースティング用テンプレートに設定された OS イメージが設定されます。
NQSV_CLOUD_NETWORK_NAME	クラウドのネットワーク名。 クラウドバースティング時に選択したクラウドノードグループに設定されたネットワーク名が設定されます。
NQSV_CLOUD_HOSTNAME	クラウドインスタンスホスト名。 クラウドノードグループに追加されている実行ホストのジョブサーバ番号の範囲内の数字が設定されます。この値をクラウドインスタンスのホスト名として使用することが可能です。
NQSV_CLOUD_NAME	クラウド名。 クラウドバースティング時に選択したクラウドノードグループ名が設定されます。

インスタンスの起動結果は以下の形式で標準出力に出力する必要があります。

起動成功時：

```
NQSV_INSTANCE_UP_RESULT:OK
NQSV_INSTANCE_ID:<instance-id>
NQSV_INSTANCE_IP:<ip-address>
```

上記の情報を標準出力に 1 行ずつ出力してください。

<instance-id>に起動されたクラウドインスタンスの ID を指定してください。この情報を使ってジョブ終了時に当該インスタンスを停止します。

<public-ip-address>に起動されたクラウドインスタンスの IP アドレスを指定してください。バッチサーバホストからアクセスできる IP アドレスであれば、パブリック IP でもプライベート IP でも問題はありません。

起動失敗時：

```
NQSV_INSTANCE_UP_RESULT:NG
NQSV_INSTANCE_UP_FAILSTOP:YES|NO
NQSV_INSTANCE_UP_FAILLOCK: YES|NO
```

上記の情報を標準出力に 1 行ずつ出力してください。

NQSV_INSTANCE_UP_FAILSTOP で起動したインスタンスを停止する必要があるかどうかを示します。停止が必要な場合は、YES、停止が不要な場合は、NO と指定してください。通常、インスタンス起動前に何らかのエラーが発生した場合は、NO、インスタンス起動後に何等かのエラーが発生した場合や起動タイムアウトが発生した場合に YES で指定します。
YES を指定する場合、停止したいインスタンスの以下の情報を 1 行ずつ出力してください。以下の情報の出力がない場合、インスタンスの停止を行うことができません。IP アドレスは有効な範囲内の値を設定してください。

```
NQSV_INSTANCE_ID:<instance-id>
NQSV_INSTANCE_IP:<ip-address>
```

NQSV_INSTANCE_UP_FAILLOCK でクラウドバースティングノードグループをロックするかどうかを示します。ロックが必要な場合は、YES、ロックが不要な場合は、NO と指定してください。

クラウドインスタンス監視用スクリプト

インスタンスの監視処理は、システム管理者が作成したシェルスクリプトによって行います。本シェルスクリプトは、以下のパスで配置します。
監視用スクリプト：/opt/nec/nqsv/sbin/cloud_prog/cloud_watch.sh

監視シェルスクリプトは、NQSV から一定間隔で呼び出されます。デフォルトの間隔は 60 秒です。監視間隔は、NQSV_CLOUD_WATCH_INTERVAL 環境変数で指定できます。

監視用スクリプトで以下の処理を実施する必要があります。
1)クラウドインスタンス監視
スクリプトに指定された環境変数に従って、利用されるクラウド環境の CLI コマンドを実行して複数のクラウドインスタンスをチェックします。

2)監視結果返却

各インスタンスの監視結果を標準出力に出力します。

監視用スクリプト実行時には以下の環境変数を設定して起動します。

環境変数	値
NQSV_CLOUD_WATCH_INSTANCES	監視されるクラウドインスタンス ID。 複数のインスタンスを起動して監視する必要があった場合に、複数のインスタンス ID をカンマ―で区切って設定します。

インスタンスの監視結果は以下の形式で標準出力に出力する必要があります。
全てのインスタンスで異常が発生していない時：

```
NQSV_INSTANCE_WATCH_RESULT:OK
```

上記の情報を標準出力に出力してください。

いずれかのインスタンスで異常が発生した時：

```
NQSV_INSTANCE_WATCH_RESULT:NG
NQSV_INSTANCE_WATCH_FAILLOCK:YES|NO
NQSV_INSTANCE_ID:<instance-id>
```

上記の情報を標準出力に 1 行ずつ出力してください。
NQSV_INSTANCE_WATCH_FAILLOCK でクラウドバースティングノードグループをロックするかどうかを示します。ロックが必要な場合は、YES、ロックが不要な場合は、NO と指定してください。
<instance-id>に異常が発生した 1 つのインスタンスのインスタンス ID を指定してください。複数のインスタンスで異常が発生した場合は、NQSV_INSTANCE_ID:<instance-id>行を、発生した数分出力してください。

クラウドインスタンス停止用スクリプト

インスタンスの停止処理は、システム管理者が作成したシェルスクリプトによって行います。本シェルスクリプトは、以下のパスで配置します。
停止用スクリプト：/opt/nec/nqsv/sbin/cloud_prog/cloud_stop.sh

停止用スクリプトで以下の処理を実施する必要があります。
1)クラウドインスタンス停止および終了
スクリプトに指定された環境変数に従って、利用されるクラウド環境の CLI コマンドを実行してクラ

ウドインスタンスを停止して終了します。1 回の処理では 1 インスタンスしか停止できません。また、インスタンスを通常停止と強制停止の 2 つのモードで停止できるようにしてください。NQSV から通常停止モードでインスタンスを停止できなかった場合は強制停止モードでインスタンスを停止します。

2)インスタンス停止待ち受け
インスタンスが停止および終了されるまで待ち受けます。待ち受けるタイムアウト時間はスクリプトで自由に設定できますが、クラウドバースティング用テンプレートに設定されている停止見込み時間以内にしてください。

3)停止結果返却
インスタンスの停止結果を標準出力に出力します。

停止用スクリプトを以下の形式で実行します。強制停止の場合、引数に"force"を付加します。

```
/opt/nec/nqsv/sbin/cloud_prog/cloud_stop.sh [force]
```

実行時には以下の環境変数を設定して起動します。

環境変数	値
NQSV_CLOUD_STOP_INSTANCE	停止されるクラウドインスタンスID。

インスタンスの停止結果は以下の形式で標準出力に出力する必要があります。
通常停止成功時：

```
NQSV_INSTANCE_DOWN_RESULT:OK
```

上記の情報を標準出力に出力してください。

通常停止失敗時：

```
NQSV_INSTANCE_DOWN_RESULT:NG
NQSV_INSTANCE_DOWN_FAILSTOP:YES|NO
NQSV_INSTANCE_DOWN_FAILLOCK:YES|NO
```

上記の情報を標準出力に 1 行ずつ出力してください。
NQSV_INSTANCE_DOWN_FAILSTOP で停止しようとするインスタンスを強制停止する必要があるかどうかを示します。停止が必要な場合は、YES、停止が不要な場合は、NO と指定してください。
NQSV_INSTANCE_DOWN_FAILLOCK でクラウドバースティングノードグループをロックするかどうかを示します。ロックが必要な場合は、YES、ロックが不要な場合は、NO と指定してください。

強制停止成功時：

```
NQSV_INSTANCE_FORCEDOWN_RESULT:OK
```

上記の情報を標準出力に出力してください。

強制停止失敗時：

```
NQSV_INSTANCE_FORCEDOWN_RESULT:NG
NQSV_INSTANCE_FORCEDOWN_FAILLOCK:YES|NO
```

上記の情報を標準出力に 1 行ずつ出力してください。
NQSV_INSTANCE_FORCEDOWN_FAILLOCK でクラウドバースティングノードグループをロックするかどうかを示します。ロックが必要な場合は、YES、ロックが不要な場合は、NO と指定してください。

クラウドインスタンス起動・監視・停止用サンプルスクリプト

クラウドインスタンス起動・監視・停止用スクリプトのサンプルが AWS、OCI、Azure の三種類でバッチサーバホストにインストールされています。本サンプルを参考に、前述の各処理を実装したクラウドインスタンス起動用スクリプト、クラウドインスタンス監視用スクリプトおよびクラウドインスタンス停止用スクリプトを作成し、適切なパスに配置してください。

本サンプルは最低限の機能の実装イメージを示すために提供するものであり、全てのクラウド環境での動作を保証するものではありません。環境構築にあたっては、実環境に合わせて適切な処理を実装してください。

各サンプルスクリプトの概要は以下の通りです。

(1)　クラウドインスタンス起動用スクリプトのサンプル

インストールパス：

/opt/nec/nqsv/sbin/cloud_prog/aws_start.sh.sample
/opt/nec/nqsv/sbin/cloud_prog/azure_start.sh.sample
/opt/nec/nqsv/sbin/cloud_prog/oci_start.sh.sample

処理概要：

・　AWS の場合：

環境変数 NQSV_CLOUD_IMAGE で指定されたイメージ ID でクラウドインスタンスを起動します。

スクリプト起動時に設定した環境変数は下記のように設定します。
NQSV_CLOUD_HOSTNAME：インスタンスの Name タグの値
NQSV_CLOUD_NETWORKNAME：サブネット ID

インスタンスのタイプは t2.micro、アベイラビリティゾーンは ap-northeast-1a です。

インスタンスが running 状態になるまで、5 秒間隔で状態監視を行います。全体の監視時間は 300 秒です。300 秒を超えて running にならなかった場合は起動失敗とします。
正常に起動できた場合は、インスタンス ID とインスタンスのパブリック IP アドレスを標準出力に出力します。正常に起動できなかった場合は、異常の結果を標準出力に出力します。

以下の情報は未指定です。実際の環境に合わせて適切な値を指定してください。
キーペア
セキュリティグループ ID

・　Azure の場合：
環境変数 NQSV_CLOUD_IMAGE で指定されたイメージ ID でクラウドインスタンスを起動します。
スクリプト起動時に設定した環境変数は下記のように設定します。
NQSV_CLOUD_HOSTNAME：インスタンスのコンピュータ名と VM 名
NQSV_CLOUD_NETWORKNAME：サブネット ID

インスタンスのタイプは Standard_DS1_v2 です。

インスタンスが VM running 状態になるまで、5 秒間隔で状態監視を行います。全体の監視時間は 300 秒です。300 秒を超えて VM running にならなかった場合は起動失敗とします。
正常に起動できた場合は、インスタンス ID とインスタンスのパブリック IP アドレスを標準出力に出力します。正常に起動できなかった場合は、異常の結果を標準出力に出力します。

以下の情報は未指定です。実際の環境に合わせて適切な値を指定してください。
リソースグループ

・　OCI の場合：
環境変数 NQSV_CLOUD_IMAGE で指定されたイメージ ID でクラウドインスタンスを起動します。
スクリプト起動時に設定した環境変数は下記のように設定します。
NQSV_CLOUD_HOSTNAME：ホスト名および表示名
NQSV_CLOUD_NETWORKNAME：サブネット ID

インスタンスのタイプは VM.Standard.E2.1.Micro、アベイラビリティドメインは NTMs:AP-OSAKA-1-AD-1 です。

インスタンスが RUNNING 状態になるまで、5 秒間隔で状態監視を行います。全体の監視時間は 300 秒です。300 秒を超えて RUNNING にならなかった場合は起動失敗とします。
正常に起動できた場合は、インスタンス ID とインスタンスのパブリック IP アドレスを標準出力に出力します。正常に起動できなかった場合は、異常の結果を標準出力に出力します。

以下の情報は未指定です。実際の環境に合わせて適切な値を指定してください。
コンパートメント ID

(2)　監視用スクリプトのサンプル

インストールパス：

/opt/nec/nqsv/sbin/cloud_prog/aws_watch.sh.sample
/opt/nec/nqsv/sbin/cloud_prog/azure_watch.sh.sample
/opt/nec/nqsv/sbin/cloud_prog/oci_watch.sh.sample

処理概要：

・AWS の場合

環境変数 NQSV_CLOUD_INSTANCES で指定されたインスタンス ID でクラウドインスタンスの状態をチェックします。複数のインスタンス ID が指定された場合は、全インスタンスの状態を 1 つずつチェックします。インスタンスの状態が running であれば正常と判断し、それ以外の場合は異常と判断します。
監視結果を標準出力に出力します。異常と判断したインスタンスのインスタンス ID を標準出力に出力します。

・Azure の場合

環境変数 NQSV_CLOUD_INSTANCES で指定された仮想マシン名でクラウドインスタンスの状態をチェックします。複数の仮想マシン名が指定された場合は、全インスタンスの状態を 1 つずつチェックします。インスタンスの状態が VM running であれば正常と判断し、それ以外の場合は異常と判断します。
監視結果を標準出力に出力します。異常と判断したインスタンスのインスタンス ID を標準出力に出力します。

・OCI の場合

環境変数 NQSV_CLOUD_INSTANCES で指定されたインスタンス ID でクラウドインスタンスの状態をチェックします。複数のインスタンス ID が指定された場合は、全インスタンスの状態を 1 つず

つチェックします。インスタンスの状態が RUNNING であれば正常と判断し、それ以外の場合は異常と判断します。
監視結果を標準出力に出力します。異常と判断したインスタンスのインスタンス ID を標準出力に出力します。

(3)　停止用スクリプトのサンプル

インストールパス：

/opt/nec/nqsv/sbin/cloud_prog/aws_stop.sh.sample
/opt/nec/nqsv/sbin/cloud_prog/azure_stop.sh.sample
/opt/nec/nqsv/sbin/cloud_prog/oci_stop.sh.sample

処理概要：

・AWS の場合

環境変数 NQSV_CLOUD_STOP_INSTANCE で指定されたインスタンス ID でクラウドインスタンスを停止(stop)して終了(terminate)します。スクリプト実行時のコマンドラインオプションに force が指定された場合は、停止処理を行わずに、直接に終了(terminate)を行います。
停止(stop)処理を行うときに、インスタンスが stopped 状態になるまで、5 秒間隔で状態監視を行います。全体の監視時間は 180 秒です。180 秒を超えて stopped にならなかった場合は停止失敗とします。すでに stopped、stopping、shutting-down、terminated 状態になっているインスタンスは停止処理をスキップします。
終了(terminate)処理を行うときに、インスタンスが terminated 状態になるまで、5 秒間隔で状態監視を行います。全体の監視時間は 180 秒です。180 秒を超えて terminated にならなかった場合は停止失敗とします。
停止処理の結果を標準出力に出力します。

・Azure の場合

環境変数 NQSV_CLOUD_STOP_INSTANCE で指定された VM 名でクラウドインスタンスを停止(stop)して削除(delete)します。スクリプト実行時のコマンドラインオプションに force が指定された場合は、停止処理を行わずに、直接に削除(delete)を行います。
停止(stop)処理を行うときに、インスタンスが VM stopped 状態になるまで、5 秒間隔で状態監視を行います。全体の監視時間は 180 秒です。180 秒を超えて VM stopped にならなかった場合は停止失敗とします。すでに VM stopped、VM stopping、VM deallocating、VM deallocated 状態になっているインスタンスは停止処理をスキップします。
削除(delete)処理を行うときに、インスタンスが VM deallocated 状態になるまで、5 秒間隔で状態監視を行います。全体の監視時間は 180 秒です。180 秒を超えて VM deallocated にならなかった場合は停止失敗とします。

停止処理の結果を標準出力に出力します。

以下の情報は未指定です。実際の環境に合わせて適切な値を指定してください。
リソースグループ

・OCI の場合
環境変数 NQSV_CLOUD_STOP_INSTANCE で指定されたインスタンス ID でクラウドインスタンスを停止(stop)して終了(terminate)します。スクリプト実行時のコマンドラインオプションに force が指定された場合は、停止処理を行わずに、直接に終了(terminate)を行います。
停止(stop)処理を行うときに、インスタンスが STOPPED 状態になるまで、5 秒間隔で状態監視を行います。全体の監視時間は 180 秒です。180 秒を超えて STOPPED にならなかった場合は停止失敗とします。すでに STOPPED、STOPPING、TERMINATING、TERMINATED 状態になっているインスタンスは停止処理をスキップします。
終了(terminate)処理を行うときに、インスタンスが TERMINATED 状態になるまで、5 秒間隔で状態監視を行います。全体の監視時間は 180 秒です。180 秒を超えて TERMINATED にならなかった場合は停止失敗とします。
停止処理の結果を標準出力に出力します。

4.23.6 クラウドインスタンス上のジョブサーバの設定

クラウドインスタンス上でジョブサーバが実行されるために、クラウドインスタンスを起動するために OS イメージを作成するときに NQSV/JobServer を事前にインストールして作成する必要があります。
クラウドインスタンス上で動作するジョブサーバは、ジョブサーバ番号が BSV により自動的に払い出されます。そのため、ジョブサーバ番号(オプション-n)を指定せずに nqs_shpd を起動します。下記のようにジョブサーバ起動スクリプト(/opt/nec/nqsv/sbin/systemd_prog/nqs_jsv.sh)を修正することで、クラウドインスタンス上でジョブサーバがジョブサーバ番号なしで起動されます。なお、NQSV/JobServer のインストール時の nqs_jsv.sh は下記の修正が行われていないため、修正して配置してください。

修正前

```
#!/bin/sh
# NQSV JobServer Startup Script.

# Auto start config file (Don't change!)
JSV_START_CONF=/etc/opt/nec/nqsv/.jsv_start
```

```
PATH=/sbin:/usr/sbin:/bin:/usr/bin
export PATH

case $1 in
start)
    if [ "$JSV_NUMBER" = "AUTO" ]
    then
        if [ -f $JSV_START_CONF ]
        then
            cat $JSV_START_CONF | ¥
            while read line
            do
                set $line
                jsvno2=`printf "%04d" $1`
                /opt/nec/nqsv/sbin/nqs_shpd -h $2 -n $1 $JSV_PARAM
                if [ $? = 0 ]
                then
途中省略
    else
        for jsvno in $JSV_NUMBER
        do
            jsvno2=`printf "%04d" $jsvno`
            /opt/nec/nqsv/sbin/nqs_shpd -h $BSV_HOST_NAME -n $jsvno $JSV_PARAM > /dev/null
            if [ $? = 0 ]
            then
                RCNT=0
                while [ $RCNT -lt 5 ]
                do
```

修正後

```
#!/bin/sh
# NQSV JobServer Startup Script.

# Auto start config file (Don't change!)
JSV_START_CONF=/etc/opt/nec/nqsv/.jsv_start

PATH=/sbin:/usr/sbin:/bin:/usr/bin
export PATH
```

```
case $1 in
start)
    if [ "$JSV_NUMBER" = "AUTO" ]
    then
        if [ -f $JSV_START_CONF ]
        then
            cat $JSV_START_CONF | ¥
            while read line
            do
                set $line
                jsvno2=`printf "%04d" $1`
                /opt/nec/nqsv/sbin/nqs_shpd -h $2 $JSV_PARAM
                if [ $? = 0 ]
                then
途中省略
    else
        for jsvno in $JSV_NUMBER
        do
            jsvno2=`printf "%04d" $jsvno`
            /opt/nec/nqsv/sbin/nqs_shpd -h $BSV_HOST_NAME $JSV_PARAM > /dev/null
            if [ $? = 0 ]
            then
                RCNT=0
                while [ $RCNT -lt 5 ]
                do
```

実行ホストがクラウドか否かは qstat(1)コマンド –E オプションで確認することができます。クラウドであれば、ExecutionHost の項目に[C]が付加されます。また、-f オプションを指定した詳細表示では Execution Host の項目に[Cloud]が付加されます。

```
$ qstat -E host1
ExecutionHost   BatchServer     OS          Release    Hardware     VE  Load   Cpu
--------------- --------------- ----------- ---------- ---------- ---- ----- -----
[C]host1        host0           Linux       4.18.0-477 x86_64        0   0.0   0.8

$ qstat -Ef host1
Execution Host: host1 [Cloud]
    Batch Server = host0
    Operating System = Linux (Rocky Linux release 8.8 (Green Obsidian))
    Version =
    Release = 4.18.0-477.15.1.el8_8.x86_64
    Hardware = x86_64
    Cloud Template = cl_tmpl
以下省略
```

また、qstat(1)コマンドの-S オプションでも確認することができます。クラウドであれば、

ExecutionHost の項目に[C]が付加されます。また、-f オプションを指定した詳細表示では Execution Host の項目に[Cloud]が付加されます。

```
$ qstat -S 1000
JSVNO JobServerName   BatchServer     ExecutionHost   LINK BIND Queue    Jobs  Load   Cpu
----- --------------- --------------- --------------- ---- ---- -------- ---- ----- -----
 1000 JobServer1000   host0           [C]host1        UP   Y    bq,iq       1   0.0   0.0

$ qstat -Sf 1000
Job Server Name: JobServer1000
    Job Server Number  = 1000
    Job Server Version = R1.08 (linux)
    Batch Server   = host0
    Execution Host = host1 [Cloud]
    LINK Batch Server = UP
    BIND Queue = BIND
以下省略
```

クラウド実行用の実行ホスト及びジョブサーバは NQSV から自動的に操作されます。直接 qmgr(1M) コマンドで下記の操作を行うことはできません。

- 実行ホストとジョブサーバのアタッチ（attach execution_host）
- ジョブサーバの起動（start job_server）
- ジョブサーバの停止（stop job_server）
- ジョブサーバのキューへのバインド（bind execution_queue job_server / bind interactive_queue job_server）
- ジョブサーバのキューからのアンバインド（unbind execution_queue job_server / unbind interactive_queue job_server）

例外的に実行ホストとジョブサーバのデタッチは行うことができます。この操作を行うことで、クラウド上で起動しているインスタンスを停止することができます。

```
$ qmgr -Pm
Mgr: detach execution_host host=host1
```

4.23.7 クラウドバースティングのポリシーの概要

JobManipulator では以下のようなポリシーでバースティング対象リクエストとクラウドノードグループを選択します。

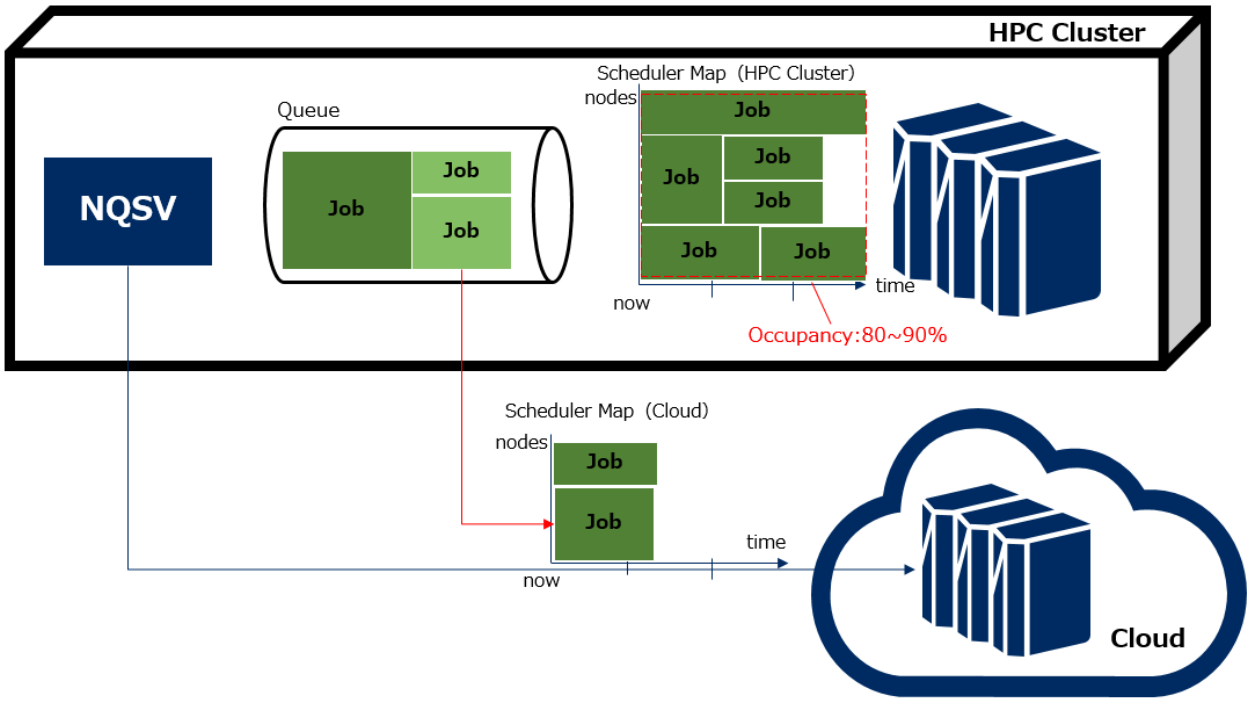


図 4-9：クラウドバースティングのポリシー

1. キューのスケジューラマップ（HPC クラスタ）の占有率を計算し、閾値を超えているかどうかをスケジューリングインターバルごとにチェックします。
2. キューのスケジューラマップ（HPC クラスタ）の占有率が閾値を超えている場合、各リクエストのバースティングプライオリティを計算し、バースティング対象リクエストを決定します。
3. キューにバインドされているクラウドバースティングノードグループのプライオリティに従ってそれぞれのノードグループに設定されたクラウドバースティング用テンプレートのリソース量をチェックし、バースティング対象リクエストに最適なノードグループとテンプレートを選択して、そのテンプレートの OS イメージでクラウドインスタンスを起動要求します。
4. バースティング対象リクエストは選択した OS イメージで起動されたクラウド上にスケジューリングし、実行します。
5. クラウド上でのリクエストの実行が全て終了すると、クラウドインスタンスの停止要求を行います。すべてのクラウドインスタンスが停止すると、ふたたび 1 から実施します。

占有率の計算はスケジューリングインターバル毎に行います。リクエストの実行が終了し、全てのクラウドインスタンスが停止してから次のクラウドバースティングを行います。

4.23.8 スケジューリングの設定

投入されたリクエストをバースティングしてクラウド上で実行するために、以下のようにスケジューリングの設定が必要です。

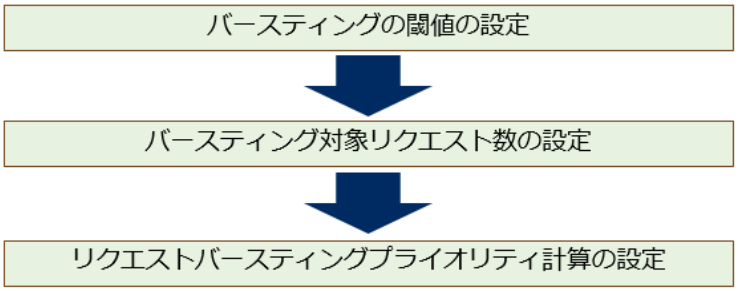


図 4-10：スケジューリングの設定

バースティングの閾値の設定

スケジューラマップ占有率が何%を超えたらクラウドバースティングを行うかの閾値は **smgr(1M)**コマンドの **set queue cloud_bursting occupancy_ratio** サブコマンドでキュー毎に設定します。設定値は 0 から 1 の範囲の小数で指定します。0 の場合は、常にクラウドバースティングを行います。1 の場合はバースティングを行いません。既定値は 1 です。

スケジューラマップ占有率とは、キューとバインドしている実行ホスト（NQSV の管理上のクラウドインスタンス上の実行ホスト除く）上のリソース量×マップ長に対する、そのノード上にアサインしているジョブ、予約区間、計画省電力のリソース量×占有時間の総和が占める割合です。カレントのスケジューラ占有率がバースティングの閾値を超えた場合は、当該キューに投入されたリクエストがクラウド上にバースティングできるようになります。
バースティングの閾値、バースティングかどうかの状態、カレントのスケジューラマップ占有率は、以下のように **sstat -Qf** で確認できます。

```
$ sstat -Qf
Execution Queue: bq
<中略>
    Elapse Margin          = OS
    Cloud Bursting Occupancy Ratio = 1.000000
    Current Cloud Bursting Occupancy Ratio = 0.000000
    Current Cloud Bursting Status = DISABLE
    Display Cloud Bursting Priority = ON
<以降省略>
```

バースティング対象リクエスト数の設定

一度にクラウドバースティングの対象として選択するリクエスト数を **smgr(1M)**コマンドの **set cloud_bursting request_number** で設定します。既定値は 0（選択されない）です。クラウドノード

グループで定義したインスタンス数が実際に一度に利用できる最大のノード数となります。そのノード数の範囲内で実行できるリクエスト数を指定してください。
現在の設定値は **sstat -Sf** コマンドで確認できます。

```
$ sstat -Sf
JobManipulator Server Host: bsv
<中略>
    Device Group Topology = ON
    Cloud Bursting Request Number = 10
    Cloud Bursting Priority Weight = {
<以降省略>
```

リクエストバースティングプライオリティ計算の設定

キューのスケジューラマップの占有率がバースティングの閾値を超えてバースティング可能な状態になっている場合、当該キューに投入された各リクエストのバースティングプライオリティが計算されます。バースティングプライオリティの高い順にバースティング対象リクエスト数だけ、バースティング対象リクエストを選択します。

バースティングプライオリティは開始予定時刻が決められていないスケジューリング対象のリクエストに対して計算されます。開始予定時刻が決定済のリクエスト、実行中リクエスト、HPC クラスタのノードにジョブを持つリクエストに対しては計算されません。

バースティングプライオリティはスケジューラマップ占有率の閾値を超える場合に計算されますが、**smgr(1M)**コマンドの **set queue cloud_bursting display_bursting_priority** を **on** に設定すると常に計算されます。この設定の既定値は **off** です。**off** の場合でも、キューのスケジューラマップ占有率がバースティングの閾値を超えている場合は計算が行われます。バースティングプライオリティ計算はスケジューリングインターバル毎に行います。

バースティングプライオリティは、要求資源量、待ち時間、スケジューリングプライオリティ、カスタムリソースの４つの要素からなります。各要素には重みづけを設定できます。既定値は待ち時間のみでバースティングプライオリティが計算され､待ち時間が長いほどプライオリティが高くなります。具体的な計算式は、次のとおりです。

バースティングプライオリティ ＝（要求資源量の要素（正規化）＋
待ち時間の要素+
スケジューリングプライオリティの要素+

カスタムリソース 1 の要素（正規化）+
[カスタムリソース 2 の要素（正規化）…]）×バースティング可否

バースティングプライオリティが 0 以下の場合、バースティング不可を示します。スケジューリングプライオリティやカスタムリソースの値が負の値であることにより、リクエストがバースティング対象とならない場合が生じますので、注意してください。

バースティングプライオリティの各要素について説明します。

1)要求資源量の要素
要求資源量の要素の計算式は、次のとおりです。

要求資源量の要素　＝　（リクエストあたりの CPU 台数×宣言 Elapse 時間（正規化）+
リクエストあたりの GPU 台数×宣言 Elapse 時間（正規化）+
リクエスト当たりのメモリ量×宣言 Elapse 時間（正規化））
×要求資源量の重みづけ

要求資源量の要素の重みづけは **smgr(1M)**コマンドの **set cloud_bursting priority_weight_resource** で設定します。既定値は 0 です。

2)待ち時間の要素
待ち時間の要素の計算式は、次のとおりです。

待ち時間の要素　＝　（現在時刻－リクエスト投入時刻）×待ち時間の要素の重みづけ

待ち時間の要素の重みづけは **smgr(1M)**コマンドの **set cloud_bursting priority_weight_wait_time** で設定します。既定値は 1 です。

3)スケジューリングプライオリティの要素
スケジューリングプライオリティの要素の計算式は、次のとおりです。

スケジューリングプライオリティの要素　＝
スケジューリングプライオリティ×スケジューリングプライオリティの要素の重みづけ

スケジューリングプライオリティの要素の重みづけは **smgr(1M)**コマンドの **set cloud_bursting priority_weight_scheduling_priority** で設定します。既定値は 0 です。

4)カスタムリソースの要素
カスタムリソースの要素は、定義したカスタムリソース毎に計算します。重みづけの設定もカスタム

リソース毎に行います。

消費単位がリクエストのカスタムリソースは、次のように計算します。

カスタムリソース１の要素　＝
リクエストにおけるカスタムリソース１の値×カスタムソース１の要素の重みづけ

消費単位がジョブのカスタムリソースは、次のように計算します。

カスタムリソース２の要素　＝
ジョブにおけるカスタムリソース２の値×リクエストのジョブ数
×カスタムソース２の要素の重みづけ

カスタムリソースの要素の重みづけは **smgr(1M)** コマンドの **set cloud_bursting priority_weight_custom_resource** でカスタムリソース毎に設定します。既定値は０です。

5)バースティング可否の要素
バースティング不可の場合、バースティングプライオリティは０となります。

リクエスト投入時に **qsub(1)**、**qlogin(1)**、または、**qrsh(1)**で**--enable-cloud-bursting=yes** を指定した場合、全体に１をかけます。**--enable-cloud-bursting=no** を指定した場合、あるいは、指定しなかった場合、全体に０をかけます。

リクエストバースティングプライオリティを計算するための各要素の重みづけの設定値は **sstat –Sf** で確認できます。

```
$ sstat -Sf
JobManipulator Server Host: bsv
<中略>
    Device Group Topology = ON
    Cloud Bursting Request Number = 10
    Cloud Bursting Priority Weight = {
      Resource            = 20
      Wait Time           = 20
      Scheduling Priority = 30
      Custom Resource = {
        cr1 = 10
        cr2 = 20
      }
    }
```

```
<以降省略>
```

4.23.9 リクエストの投入

バースティング可としてリクエストを投入するにはバッチリクエストならば qsub(1)、会話リクエストならば qlogin(1)または qrsh(1)の各コマンドにオプション--enable-cloud-bursting=yes を指定します。このオプションを指定して投入されたリクエストはバースティング可能な状態になったらクラウド上にバースティングされます。

```
qsub --enable-cloud-bursting=yes
qlogin --enable-cloud-bursting=yes
qrsh --enable-cloud-bursting=yes
```

一方でオプション--enable-cloud-bursting=no を指定した、またはオプション--enable-cloud-bursting を指定しない場合、そのリクエストは常にバースティングされません。
既に投入済みのリクエストのバースティング可否を qalter(1)コマンドで変更することもできます。

```
qalter --enable-cloud-bursting={yes|no} <rid>
```

yes を指定すると、そのリクエストは必要に応じてクラウド上にバースティングされるように変更されます。no を指定すると、そのリクエストは決してバースティングされないように変更されます。
*<rid>*はリクエスト ID です。
なお、既に実行を開始したリクエストに対しては効力を持ちません。
クラウドバースティングの可否は qstat -f で確認することができます。Enable Cloud Bursting という項目にその可否が表示されます。

```
$ qstat -f 1.bsv
Request ID: 1.bsv
    Request Name = STDIN
    User  Name = user
    Group Name = group
    User  ID   = 851
    Group ID   = 701
    Current State           = Running
    Previous State          = Pre-running
    State Transition Time   = Wed Apr 21 11:02:21 2021
    State Transition Reason = PRERUN_SUCCESS
    Queue = cloud@host0 (Execution Queue)
    Job Topology = Distribute Job
```

```
   Request Priority  = 0
   Request Loglevel  = 0
   Rerunable    = Yes
   Holdable     = Yes
   Hold Type    = (none)
   Migratable   = Yes
   Suspend Type = (none)
   Account Code = (none)
   Stdout = host0:/home/user/STDIN.o1
   Stderr = host0:/home/user/STDIN.e1
   Reqlog = (none)
   Shell = (none)
   Mail Address = user@host0
   Mail Option  = (none)
   Job Condition:
       Job NO: 0 ""
   Number of Jobs = 1
   Created Request Time = Wed Apr 21 11:01:03 2021
   Entered Queue Time   = Wed Apr 21 11:01:03 2021
   Planned Start Time   = Wed Apr 21 11:02:21 2021
   Execute Request Time = (none)
   Started Request Time = Wed Apr 21 11:02:21 2021
   Ended Request Time   = (none)
   Requested Start Time = (none)
   Deadline Time        = (none)
   UMASK = 022
   Checkpoint Interval = 0
   Restart File Directory = (none)
   Reservation ID      = (none)
   qattach command = Enable
   Attach = No
   Cluster Type Select = NONE
   Cloud Template = host1
   UserPP Script = (none)
   Exclusive = (none)
   HCA Number = (none)
   Accept Sigterm = No
   Enable Cloud Bursting = Yes
 Execution Hosts(JSVNO):
   host1(1000)
```

```
以下省略
```

実際にジョブがクラウド上で実行されているか否かは **qstat -J** で確認できます。クラウド上で実行されているのであれば、ExecutionHost の項目に[C]が付加されて表示されます。また、**-f** を指定した詳細表示では Execution Host の項目に[Cloud]が付加されます。

```
$ qstat -J 1.bsv
JNO RequestID        EJID   Memory      CPU JSVNO ExecutionHost   UserName Exit
---- --------------- ----- -------- -------- ----- --------------- -------- ----
   0 1.bsv           14714    5.58M     0.00  1000 [C]host0        user        -

$ qstat -Jf 1.bsv
Request ID: 1.bsv
    Batch Job Number = 0
    Execution Job ID = 14714
    User  Name = user
    User  ID   = 851
    Group ID   = 701
    Job Server Number = 1000
    Job Server Name = JobServer1000
    Execution Host = host1 [Cloud]
    Exit Code = (none)
  Resources Information:
以下省略
```

4.23.10 クラウド選択のポリシー

スケジューラマップの占有率が閾値を超えている場合、各リクエストのバースティングプライオリティを計算し、バースティングプライオリティの高い順にバースティング対象リクエスト数だけ、バースティング対象リクエストを選択します。

1 リクエストに対して、そのリクエストが実行するクラウドをリクエスト投入キューとバインドしているクラウドノードグループから選択します。クラウドノードグループが複数バインドしている場合、クラウドノードグループのプライオリティの高い順に選択します。
クラウドノードグループで利用できるテンプレートが複数ある場合、そのリクエストのジョブと資源量の近いテンプレートを選択します。

下記に挙げるリクエストはバースティングプライオリティの値と関係なくバースティング対象として選択されません。

- 予約区間指定（qsub -y）のリクエスト
- 開始時刻指定（qsub -s）のリクエスト
- デッドライン時刻指定（qsub -Y）のリクエスト
- qsub --after 指定に指定した先行リクエストが終了していないリクエスト
- qsub --parallel で 1 度に投入したリクエスト数が「バースティング対象リクエスト数」を超える同時実行リクエスト
- OpenStack およびコンテナのテンプレート指定のリクエスト

4.23.11 クラウドバースティングしたリクエストの強制リラン機能

クラウド環境ではネットワークの瞬断により JSV が LINKDOWN となることがあります。実行中ジョブの強制リラン機能（詳しくは、4.11.2 実行中ジョブ強制リラン機能を参照）を on に設定にするとネットワーク瞬断のたびにリクエストはリランし、クラウド環境は一旦停止し、再度クラウド起動を繰り返すことがあります。

そのため、クラウドバースティングしたリクエストに対する実行中ジョブの強制リラン機能は、HPC クラスタ環境での実行中ジョブの強制リラン機能とは別の設定にしています。smgr(1M)コマンドの set forced_rescheduling コマンドで、set forced_rescheduling = on cloud のように末尾に cloud を付けることでクラウドバースティングしたリクエストに対してのみ有効な設定となります。既定値は off で JSV の LINKDOWN によってクラウドバースティングしたリクエストはリランされません。

4.24 リクエストのアサインモード

3.1.1 **スケジューラマップ（3）設定に関する留意事項** に記載したとおり、リクエストが大量に存在する場合には、リクエストが滞留しないようにスケジューラマップやスケジューリングインターバルを調整することが望ましいです。

上記パラメータでの調整が難しい場合のために本機能を提供します。本機能は、スケジューリングインターバル時刻での処理を打ち切った場合に、次のスケジューリングインターバルで継続して行うかどうかを選択する機能です。この際、ポリシーを考慮したスケジューリングにはならないことに注意ください。

smgr(1M)コマンドの set scheduling_method assign_mode サブコマンドで設定できます。既定値は reset です。reset に設定した場合は、スケジューリングに時間がかかる場合、一定の時間で処理を

打ち切ります。**continue** に設定した場合は、この処理を打ち切らず、継続して行います。

この設定は、sstat -Sf の"Scheduling Method"の"Assign Mode"で確認できます。

---

continue に設定した場合、現在時刻時点でのスケジューリングプライオリティ等のポリシーを考慮したスケジューリングとなりません。最初に中断の起きたスケジューリングインターバル時点でのスケジューリングプライオリティなどのポリシーに従ったリクエストのスケジューリングを継続する動作になります。

4.25 スケジューリング中の実行開始予定時刻の表示

リクエストが大量に存在する場合に、スケジューリングに一定の時間がかかることがあります。

この間、**sstat**(1)コマンドでの実行開始予定時刻の参照に時間がかかることがあります。

その場合、**qstat**(1)コマンドの --planned-start-time オプションで開始予定時刻を参照してください。

開始予定時刻を **qstat**(1)コマンドで参照することが難しい場合には、以下の設定を行うことでスケジューリング処理中でも直近の実行開始予定時刻を **sstat**(1)コマンドで参照できます。本機能は、オプションなしの **sstat**(1)コマンドが対象です。

本機能を有効にするには、NQSV/JobManipulator のコンフィグファイル（/etc/opt/nec/nqsv/nqs_jmd.conf）に FORK_WHILE_SCHEDULING の指示行を設定します。その上、スケジューリング処理開始後、実行開始予定時刻を sstat(1)コマンドでの参照が可能になるまでの秒数を設定値に指定します。

設定値は 0 からスケジューリングインターバル-5 秒まで指定可能です。

設定後、NQSV/JobManipulator を再起動してください。

以下は、スケジューリング処理が 30 秒以上かかった場合に実行開始予定時刻を sstat(1)コマンドでの参照を可能にする例です。

```
FORK_WHILE_SCHEDULING: 30
```

- 本機能は JM と JM が接続するバッチサーバが同一ホスト上で運用されている場合にのみ利用可能です。

- オプションありの **sstat**(1)コマンド、**smgr**(1M)コマンド、および、**sushare**(1M)コマンドをスケジューリング処理中に実行した場合には、エラーとなります。
- 本機能を利用する場合、**smgr**(1M)コマンドおよび **sushare**(1M)コマンドは **qmgr**(1M)コマンドですべてのキューを停止した状態で実行してください。

4.26 スケジューリング不可リクエストのキャッシュ機能

3.1.1 **スケジューラマップ（3）設定に関する留意事項** に記載したとおり、リクエストが大量に存在する場合に、スケジューリングインターバル内でスケジューリングが終了せず、処理を打ち切ることがあります。上記の様な状況の中でも、同一ユーザーが同一条件で連続投入しかつ開始予定時刻が決まらないリクエストに対して、本機能は効果があります。

本機能はそのスケジューリング結果をキャッシュすることにより、スケジューリングインターバル内でのスケジューリング対象のリクエストを増やすことができます。

本機能を有効にするには、操作員権限以上で **smgr**(1M)コマンドの下記のサブコマンドを実行してください。

```
Smgr: set scheduling_method non_scheduled_request_cache = on
```

開始予定時刻を決めることができている場合、効果はありません。

条件の異なるリクエストや、連続して投入していない場合、効果はありません。

第5章　SX-Aurora TSUBASA 対応機能

5.1　概要

この章では、JobManipulator の SX-Aurora TSUBASA 対応機能について説明します。本機能は実行ホストが SX-Aurora TSUBASA システムの場合のみ有効です。

5.2　VE割り当て機能

VE を使用する場合は、**qsub**(1)コマンド、**qlogin**(1)コマンド、あるいは、**qrsh**(1)コマンドの--venum-lhost オプションまたは--venode オプションで VE ノード要求数を指定します。VE ノードを要求するリクエストに対して JobManipulator は指定された VE ノード数が確保できる実行ホスト(VI)を選択します。

5.3　VE障害機能

5.3.1　機能概要

VE の障害や復旧によって実行ホストの VE ノード数が変化した場合（以降、VE ノード縮退)、次のスケジューリング方式を選択できます。

1. 最新の VE ノード数に追随してスケジューリングを行う。
2. VE ノード縮退の発生した VI をスケジューリングから除外する。
3. VE ノード 1 個以上が縮退した場合に VI をスケジューリングから除外する。縮退した全 VE ノードが復旧した際に自動的にスケジューリング対象に戻す。

5.3.2　VE ノード縮退時のスケジューリング方式の設定

VE ノード縮退時のスケジューリング方式は、**smgr**(1M)コマンドの set scheduling_method ve_degradation でスケジューラ単位に設定します。設定には操作員以上の権限が必要です。既定値は、continue であり、VE ノードが縮退しても、最新の VE ノード数に追随してスケジューリングを行います。exclude を指定した場合、VE ノード縮退の発生した VI をスケジューリング対象から除外します。auto を指定した場合、1 台以上の VE ノードが縮退すると、当該 VI をスケジューリング対象から除外します。縮退した VE が全て復旧した場合は、当該 VI が直ちにスケジューリング対象に戻ります。

設定値を変更した場合の動作は下表の通りです。

表 5-1 VE ノード縮退時のスケジューリング方式の動作

変更前	変更後	動作
continue	exclude	VE ノード縮退の発生した VI を直ちにスケジューリング対象から除外します。
continue	auto	VE ノード縮退の発生した VI を直ちにスケジューリング対象から除外します。縮退した VE ノードが全て復旧した場合は、当該 VI が直ちにスケジューリング対象に戻ります。
exclude	continue	VE ノード縮退によりスケジューリング除外されている VI は直ちにスケジューリング対象に戻ります。
exclude	auto	VE ノード縮退によりスケジューリング除外されている VI は除外のままです。縮退した VE ノードが全て復旧した場合は、当該 VI が直ちにスケジューリング対象に戻ります。
auto	continue	VE ノード縮退によりスケジューリング除外されている VI は直ちにスケジューリング対象に戻ります。
auto	exclude	VE ノード縮退によりスケジューリング除外されている VI は除外のままです。縮退した VE ノードが全て復旧しても当該 VI がスケジューリング対象に戻りません。

- 本機能はバッチサーバ設定の Load Interval に依存します。
  Load Interval が 0 に設定されていた場合、本機能は動作しません。
  VE ノード数の増減を検知にするには、バッチサーバの Load Interval を 1 以上に設定する必要があります。バッチサーバの Load Interval は VE ノード数の更新タイミングとなります。
  従って、Load Interval を大きな値にすると、VE ノード数の更新が遅くなり、VE ノード数の変化からスケジューリングに反映されるまで時間が掛かることになります。
  バッチサーバの Load Interval については、 ***NQSV* 利用の手引[管理編]**を参照してください。
- auto を指定した場合、デバイスリソース定義ファイル（VI 上の /etc/opt/nec/nqsv/resource.def）での定義と比較して VE ノード数の縮退と復旧を判断します。デバイスリソース定義ファイルが存在しない VI では、continue と同等の動作になります。デバイスリソース定義ファイルについては、**5.4 HCA 割り当て機能** を参照

してください。

5.3.3　sstat による表示

設定値は **sstat**(1)コマンドの-S -f オプションにより確認可能です。

```
$sstat -S -f
JobManipulator Server Host: bsv.nec.co.jp
    JobManipulator Version   = R1.00
            :
    Stage-in Margin = {
      Additional Margin for Escalation = 0S
      Stage-in Threshold = 0S
      First Stage-in Time = 0S
    }
    Provisioning Start Retry Time = 0S
    Scheduling Method = {
      VE Degradation = Continue
    }
:
```

VI 毎の縮退状況は、**sstat**(1)コマンドの**-E --hw-failure** オプションで表示可能です。「Status」欄は VI の状態を、「V」欄は過去に VE ノード縮退の発生があったか否かを示します。

continue 設定の場合、VE ノード縮退が発生したノードは、縮退した VE ノード数に追随してスケジューリングされていれば、「Status」欄に「DEGRADED」が表示され、「V」欄に「D」が表示されます。

```
$sstat -E --hw-failure
ExecutionHost   Status          V
--------------- --------------- -
executionhost1  DEGRADED        D   ←VEノードが縮退し運用されている
```

exclude また auto 設定の場合、VE ノード縮退が発生したノードは、VI がスケジューリング除外状態であれば、「Status」欄に「EXCLUDED」が表示され、「V」欄に「D」が表示されます。

```
$sstat -E --hw-failure
ExecutionHost   Status          V
--------------- --------------- -
```

```
executionhost1   EXCLUDED        D     ←VEノードが縮退し運用除外されている
```

VI 縮退状況は、JobManipulator とバインドしているすべてのキューからアンバインドし、バインドし直すことによってリセットされます。

5.4　HCA割り当て機能

5.4.1　機能概要

本節では、実行ホストとして使用する SX-Aurora TSUBASA システムについて、以下の構成を例にとり説明します。

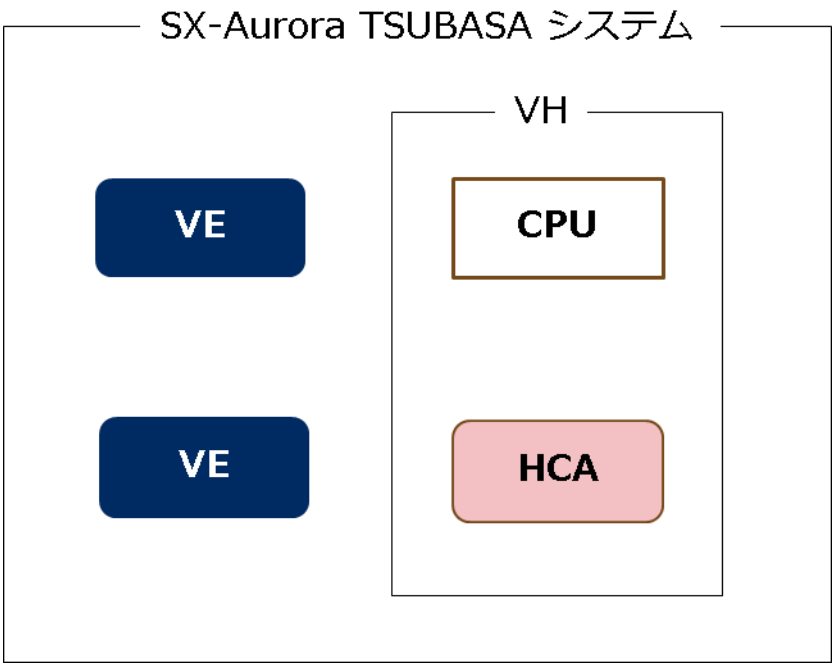


図 5-1：SX-Aurora TSUBASA システム

ベクトルエンジン(VE)とは、SX-Aurora TSUBASA の中核であり、ベクトル演算を行う部分です。VE は PCI Express カードであり、x86 サーバに搭載して使用します。ベクトルホスト(VH)とは、VE を搭載している x86 サーバ(ホストコンピューター)を指します。VH のモデルによっては、複数の VE および VE 間の通信のための InfiniBand NIC(HCA)が搭載される場合があります。

VE を搭載しているホストコンピューター、VE、HCA を総括して、ベクトルアイランド（VI）と呼びますが、VI と VH は、実行ホストとしては同じものを指していると言えます。

NQSV では、VH 上にジョブサーバを起動し、VH 上でジョブを実行します。VH 上で起動したジョブスクリプト内から、VE 用のプログラムを実行します。VE 用のプログラムを実行するための VE や HCA は NQSV が割り当てます。（NQSV では資源としてジョブに割り当てる VE のことを VE ノードと呼びます。）VE 用のプログラムは NQSV が割り当てた VE ノードを使用してプログラムを実行します。

以下は、VH 上での VE プログラムの実行イメージを表します。

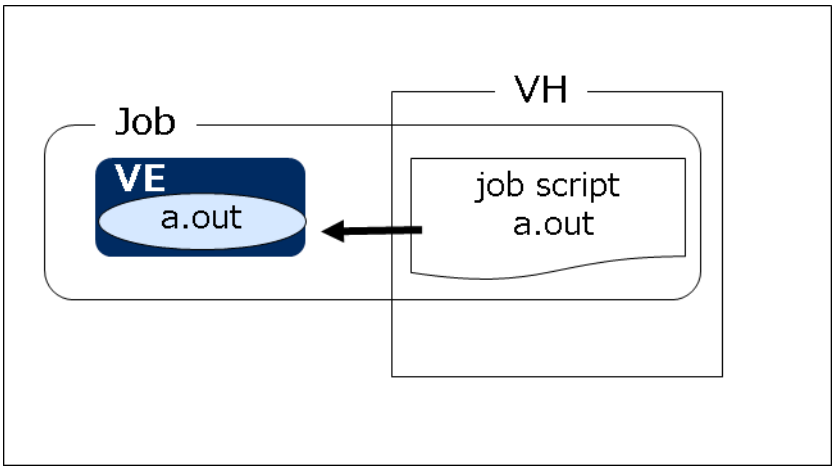


図 5-2：プログラムの実行

VH の実行ホストがバインドされているキューに、qsub --venode オプション（VE ノード総数）や --venum-lhost（論理ホスト毎の VE ノード数）を指定して投入することで、各ジョブに対して適切な VE ノードが割り当てられ、ジョブを実行することができます。

また、SX-Aurora TSUBASA のモデルによっては、VH 内で VE および HCA が PCIe スイッチを経由して CPU ソケットに接続されたトポロジー構成をとる場合があります。トポロジーとは、CPU、VE および HCA の接続形状です。以下はトポロジー構成の一例を示します。

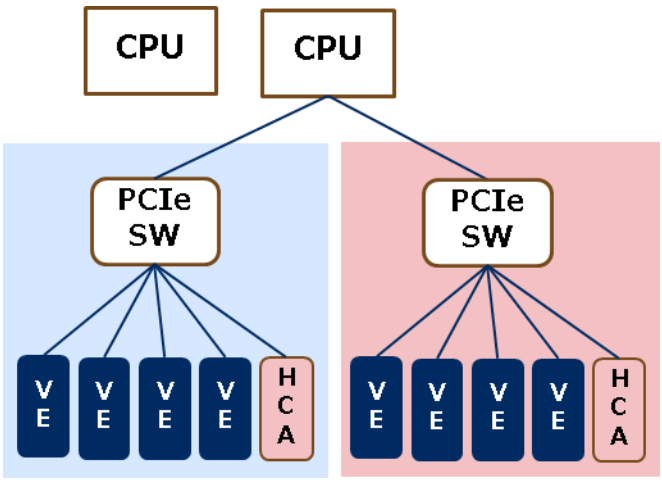


図 5-3：トポロジー構成の例

このようなトポロジー構成を管理者があらかじめ定義しておくことで、NQSV はジョブに対して VE ノードおよび HCA を割り当てて実行することが可能になります。

5.4.2　HCA およびトポロジー情報の定義

システム管理者は HCA のデバイスごとの用途と、CPU ソケット、VE ノード、および HCA のトポロジー情報を実行ホスト上のファイルに定義します。このファイルを、デバイスリソース定義ファイ

ルと呼びます。
HCA のデバイス毎の用途として、MPI 用（RDMA【Remote Direct Memory Access】）、I/O 用（Direct I/O）が定義可能で、複数定義も可能です。運用中の設定変更は不可とし、設定変更する場合は JSV の再起動を必要とします。

同一の CPU ソケットまたは同一 PCIeSW（CPU ソケットと PCIeSW を接続した場合）に接続する VE や HCA をひとまとめにしてデバイスグループと呼びます。

(1)　デバイスグループ

デバイスグループの例は下記の通りです。

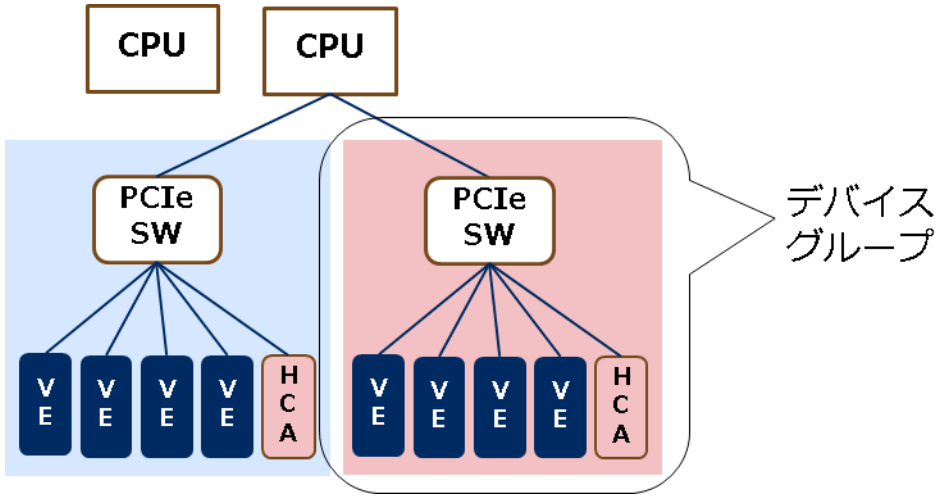


図 5-4：PCIeSW のある場合のデバイスグループの例

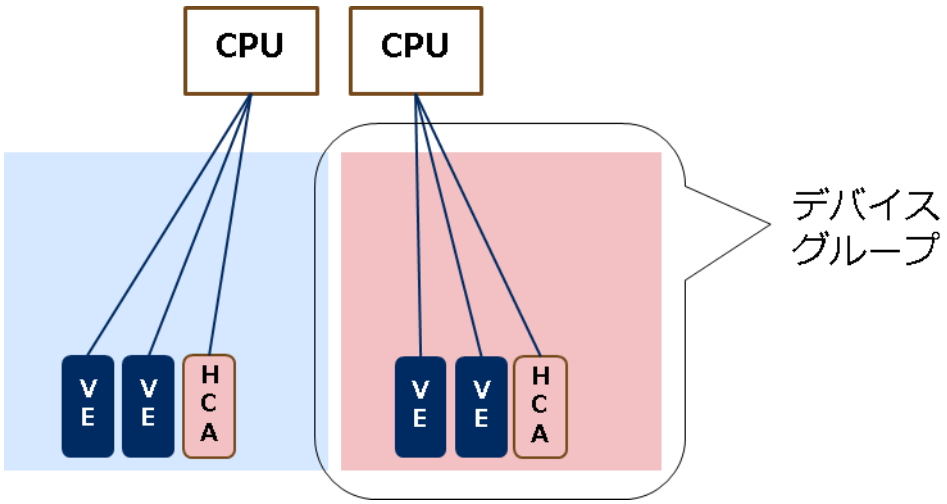


図 5-5：PCIeSW の無い場合のデバイスグループの例

(2) デバイスリソース定義ファイル

デバイスリソース定義ファイルは、VI 上の/etc/opt/nec/nqsv/resource.def です。

(3) デバイスリソース定義ファイルのフォーマット

デバイスリソース定義ファイルのフォーマットは下記の通りです。

```
形式: <Resource>
      <Resource>: リソース情報
形式 : <Type> = { <List> }

      <Type>: リソースの種別。
形式 : <Type> = Socket | PCIeSW | VE | Infiniband
  各文字列の意味は下記のとおり。
  - Socket : CPUソケット
  - PCIeSW : PCIeSW
  - VE : VEノード
  - Infiniband : HCA

      <List> : リソース詳細のリスト。リソース情報をネストして記述することで、
               トポロジー情報を表現する。
形式 : <Resource> | <Attribute>
      <Attribute>: リソース詳細情報。
形式 : <Name> : <Value>

       <Type>ごとに定義可能なリソース詳細情報は下記のとおり。いずれも設定必須。
      - Socket
            <Name>       : <Value>
        Socket Number    : ソケット番号
- PCIeSW : PCIスイッチ
         リソース詳細情報なし
      - VE
            <Name>       : <Value>
        Number           : VE物理番号(範囲指定可)
      - Infiniband
            <Name>       : <Value>
        PCI ID           : PCI識別番号
        Port Number      : ポート番号
        Mode             : HCAの利用用途 (IO とMPIが指定可。
```

```
カンマ区切りで複数の用途を指定可能)
                                IO  : I/Oのdirect通信用を示す
                                MPI : MPIのdirect通信用を示す

設定する文字列は、大文字・小文字どちらの記載も可能です。

下記条件のいずれかを満した場合、JSVの起動がエラーなります。
- PCIeSWを、Socket外のリソースとして定義した。
- VEとInfinibandを、PCIeSWまたはSocket外のリソースとして定義した。
- 存在しないPCI IDを指定した。
```

(4) デバイスリソース定義ファイルの設定例

図 5-4：PCIeSW のある場合のデバイスグループの例の構成の場合で、HCA が IO と MPI の兼用と設定する場合の設定例です。ここで HCA にはそれぞれ 2 ポート搭載されており、VH 上からはそれぞれが独立した HCA として参照できるものとします。

```
Socket = {
  Socket Number : 0
}
Socket = {
  Socket Number : 1
  PCIeSW = {
    VE = {
      Number : 0-3
    }
    Infiniband = {
      PCI ID      : 0000:05:00.0
      Port Number : 1
      Mode        : IO, MPI
    }
    Infiniband = {
      PCI ID      : 0000:07:00.0
      Port Number : 2
      Mode        : IO, MPI
    }
  }
  PCIeSW = {
    VE = {
```

```
      Number : 4-7
    }
    Infiniband = {
      PCI ID      : 0000:0b:00.1
      Port Number : 1
      Mode        : IO, MPI
    }
    Infiniband = {
      PCI ID      : 0000:0d:00.1
      Port Number : 2
      Mode        : IO, MPI
    }
  }
}
```

PCIeSW の無い構成の場合、以下のように PCIeSW の{}が無い設定となります。

```
Socket = {
  Socket Number : 0
   VE = {
    Number : 0-3
  }
  Infiniband = {
     PCI ID      : 0000:05:00.0
     Port Number : 1
     Mode        : IO, MPI
  }
}
Socket = {
  Socket Number : 1
  VE = {
    Number : 4-7
  }
  Infiniband = {
     PCI ID      : 0000:0b:00.1
     Port Number : 1
     Mode        : IO, MPI
  }
}
```

(5)　デバイスリソース定義ファイルの設定内容の参照

デバイスリソース定義ファイルの設定内容は **qstat(1)** コマンドの-E -f オプションで確認可能です。ここで PCIeSW の横の数字はデバイスグループの ID を表します。

```
qstat -E -f
  ・・・省略・・・
  Socket Resource Usage:
    NUMA Nodes = {
      Socket 0 (Cpus: 0-1) = Cpu: -/2 Memory: -/3.0GB
    }
  Device Topology:
    Socket 0 = {
      (none)
    }
    Socket 1 = {
      PCIeSW 1 = {
        VE: 0-3
        HCA: 0000:05:00.0 0 (IO,MPI)
        HCA: 0000:07:00.0 1 (IO,MPI)
      }
      PCIeSW 2 = {
        VE: 4-7
        HCA: 0000:0b:00.1 0 (IO,MPI)
        HCA: 0000:0d:00.1 1 (IO,MPI)
      }
    }
```

PCIeSW が無い場合は以下のように PCIeSW の部分が表示されません。

```
  Device Topology:
    Socket 0 = {
      (none)
    }
    Socket 1 = {
        VE: 0-3
        HCA: 0000:05:00.0 0 (IO,MPI)
        HCA: 0000:07:00.0 1 (IO,MPI)
```

```
    }
```

デバイスリソース定義ファイルが無い場合は、以下の表示となります。

```
$ qstat -E -f
 ・・・省略・・・
 Socket Resource Usage:
   NUMA Nodes = {
     Socket 0 (Cpus: 0-1) = Cpu: -/2 Memory: -/3.0GB
   }
 Device Topology: (none)
```

5.4.3　HCA の利用

(1)　リクエストの投入

qsub(1)、**qlogin(1)**、**qrsh(1)**コマンドの--use-hca オプションで Direct 通信の利用、および論理ホスト内における割り当て VE が所属するデバイスグループあたりに必要な HCA ポート数を宣言し、リクエストを投入します。

--use-hca オプションはスクリプト内の PBS 行への記載も可能です。--use-hca オプションと--venode または--venum-lhost オプションを同時に指定しない場合投入エラーとなります。

--use-hca オプションの形式は以下の通りです。

```
<hca>の形式： [<mode>:]<num>
```

<num>は論理ホストに割り当てられた VE が使用する HCA のポート数を指定します。0～32 の値が指定可能です。範囲外の数値、または数字以外の文字列を指定した場合、投入エラーとなります。

<mode>は HCA の利用用途を意味し、下記のいずれかが指定可能です。指定がない場合、all を指定したと扱います。下記以外の文字列を指定した場合、投入エラーとなります。

- io：I/O専用。デバイスリソース定義ファイルにIOを指定したHCAだけが割りあたる
- mpi：MPI専用。デバイスリソース定義ファイルにMPIを指定したHCAだけが割りあたる
- all：I/O、MPI兼用 <既定値>。デバイスリソース定義ファイルにIO,MPIの両方を指定したHCAだけが割りあたる

io、mpi、及び all は同時指定可能です。--use-hca の値は qalter で変更することはできません。

[例] 4VE ノードを要求し、VE が所属するデバイスグループあたりに HCA を 1 つ要求するリクエストを投入する場合

```
qsub --venode=4 --use-hca=1 <script>
```

[例] 4VE ノードと I/O 専用 HCA と MPI 専用 HCA をデバイスグループ内の各 VE あたり 1 つずつ要求する場合

```
qsub --venode=4 --use-hca=io:1,mpi:1 <script>
```

[例] 論理ホストあたりの VE 数が 2、論理ホストあたりの I/O, MPI 兼用の HCA 数 1、論理ホスト数が 2 であるリクエストを投入する場合

```
qsub -b 2 --venum-lhost=2 --use-hca=1 <script>
```

(2) リクエストの参照

--use-hca を指定して投入したリクエストは **qstat(1)**コマンドの**-f** オプションで参照可能です。

[例] 4VE ノードと I/O 専用 HCA と MPI 専用 HCA を 1 つずつ要求したリクエストの場合

```
$ qstat -f
～略～
VE Node Number = 2
    HCA Number = {
      For I/O = 1     <- 要求したI/O専用HCA数
      For MPI = 1     <- 要求したMPI専用HCA数
    }
```

I/O、MPI 兼用として要求された HCA 数は ForALL=<n>で表示されます。HCA の要求が無い場合、HCA Number = (none)と表示されます。

(3) HCA利用時のVEの割り当て

--use-hca を指定しリクエストを投入した場合、可能な限り同一デバイスグループに属する VE を論理ホストに割り当てます。但しリクエストの TAT や稼働率を重視し、必ずしもそうならない場合もあります。

[例]

```
(1) qsub --venode=3 --use-hca=1
(2) qsub --venode=3 --use-hca=1
```

(1)、(2)の順にリクエストを投入するとします。(1)には、同一デバイスグループの ＶＥが 3 個、(2)には、別のデバイスグループの ＶＥが 3 個割り当てられます。

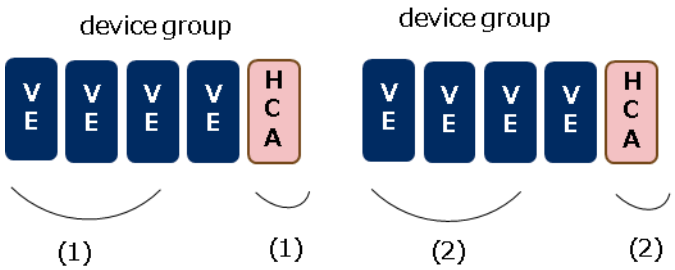


図 5-6：HCA 使用時の VE の割り当て 1

続けて、(3)のリクエストを投入するとします。

```
(3) qsub --venode=2 --use-hca=1
```

他の空き VE がまったくない場合、異なるデバイスグループに属する VE が(3)に割り当てられます。

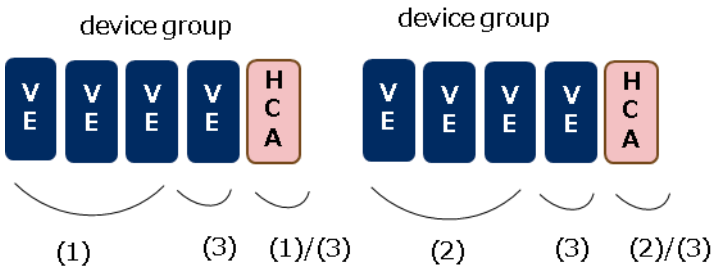


図 5-7：HCA 使用時の VE の割り当て 2

5.4.4　トポロジー情報とスケジューリング

トポロジー情報を設定していない VI は、--use-hca を指定したリクエストのスケジューリング対象にはなりません。--venode または--venum-lhost を指定し--use-hca 指定していないリクエストは、スケジューリングの対象とします。

トポロジー情報を設定した VI と設定していない VI が混在している場合は、トポロジー情報を設定していない VI は、--venode または--venum-lhost を指定したリクエストのスケジューリング対象外となります。

実行性能を最大にするために、同一キューには、CPU、VE の数やそれらの接続形状といったトポロジー構成が同じ VI をバインドしてください。

リクエストがトポロジー情報を利用した状態で、キューと JobManipulator をアンバインド、または JobManipulator や BSV を再起動の処理を行った場合、以下の制限があります。処理の時点で実行待ちであったリクエストは、処理の時点で実行中であったリクエストと同じ実行ホスト上で実行できません。実行中のリクエストの終了後に開始されます。なお、実行中リクエストにはサスペンドされたリクエストを含みます。

5.5　トポロジー性能を考慮したスケジューリング

JobManipulator は、既定値では VE ノードを VI 内の使用可能 VE ノード数制限まで詰めてジョブをアサインします。使用可能 VE ノード数制限を超えるまでは他の VI にジョブをアサインしないため、使用する VI 数を最小にすることが可能です。それにより、省電力効果が期待できます。

VE ノードと HCA のトポロジーを考慮した割り当てを行いたい場合は、トポロジー性能を考慮したスケジューリング機能を有効にすることにより実現可能です。

5.5.1　設定

トポロジー性能を考慮したスケジューリング機能は **smgr**(1M) コマンドの set device_group_topology サブコマンドにより設定します。設定には操作員以上の権限が必要です。既定値は、off であり、VE や HCA の割り当てに VE ノードと HCA の距離を重視しません。on を指定した場合、VE ノードと HCA の距離を考慮して VE や HCA の割り当てを行います。詳しくは、5.5.2 トポロジー性能を考慮した運用 の節を参照してください。

なお、off の設定時には、下位ポリシーの CPU 台数集中アサインを有効に設定することで、VI 内の使用可能 VE ノード数制限まで詰めてジョブをアサインすることができます。

本機能は R1.05 以前ではコンフィグファイルの **VE_CONCENTRATION:** パラメータにより設定していた機能です。R1.05 以前でコンフィグファイルに **VE_CONCENTRATION:** パラメータの設定がある状態で、R1.06 以降のバージョンへのアップグレードし、最初に起動した時点で設定が自動的に反映されます。それ以降は、**VE_CONCENTRATION:** パラメータは無視されます。**smgr**(1M)コマンドの set device_group_topology サブコマンドにて設定してください。

5.5.2　トポロジー性能を考慮した運用

JobManipulator による VE 割り当ては、リクエストの TAT や稼働率を重視し、同一論理ホストに対しデバイスグループを跨った割り当てとなる場合があります。

それに対し、同一論理ホストに対しデバイスグループを跨らない VE の割り当てとする運用を行うことによりトポロジー性能を重視した運用方法を実現できます。

そのためには全てのリクエストにおいて、実際の使用数に拘わらず 1 論理ホストあたりの要求 VE 数をデバイスグループに搭載されている VE 数やその倍数にします。これにより一つの論理ホストに対しデバイスグループを跨いで VE が割り当てられることが無くなり、常に性能のよい割り当て方にすることができます。

[例]

```
(1) qsub --venode=4 --use-hca=1
(2) qsub --venode=4 --use-hca=1
```

(1)、(2)の順にリクエストを投入するとします。要求 VE ノード数をいずれも 4 としているので、同一論理ホストにデバイスグループを跨ることなく 4VE を割り当て、各 VE から最も近い HCA が 1 個ずつ割り当たり、性能が良い HCA を使用できます。

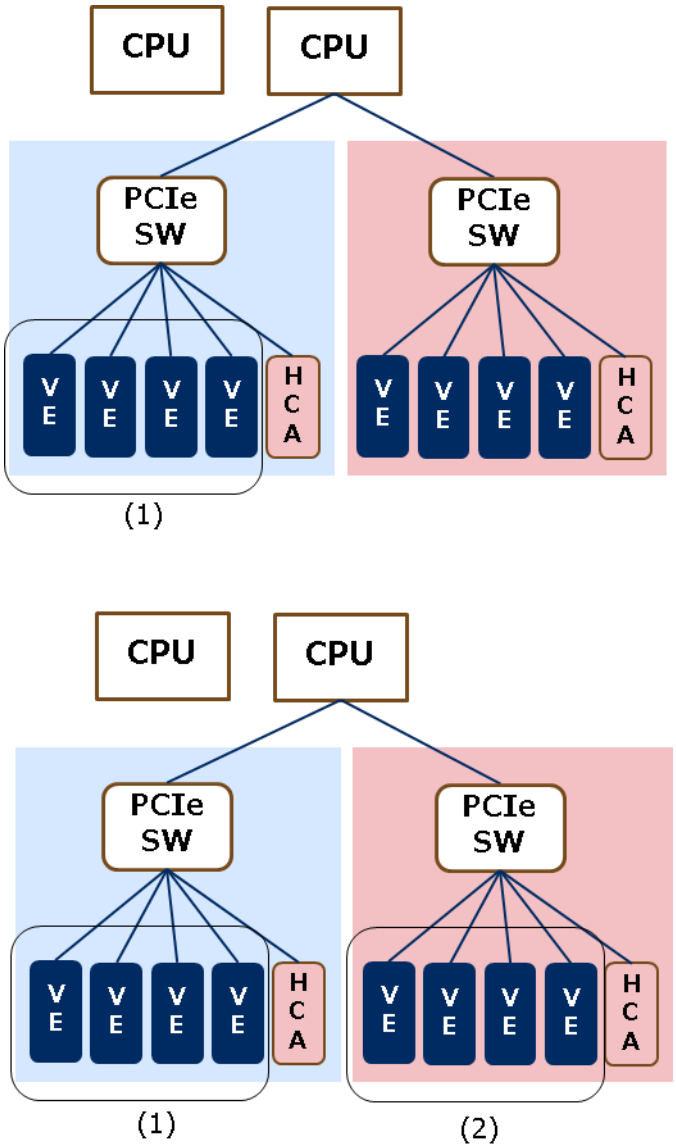


図 5-8：トポロジーを考慮した運用例 1

デバイスグループに含まれる VE 数が 4 個の場合、すべてのリクエストで 1 論理ホストあたりの要求 VE 数を 2 とすることでも同様の割り当て方が可能です。

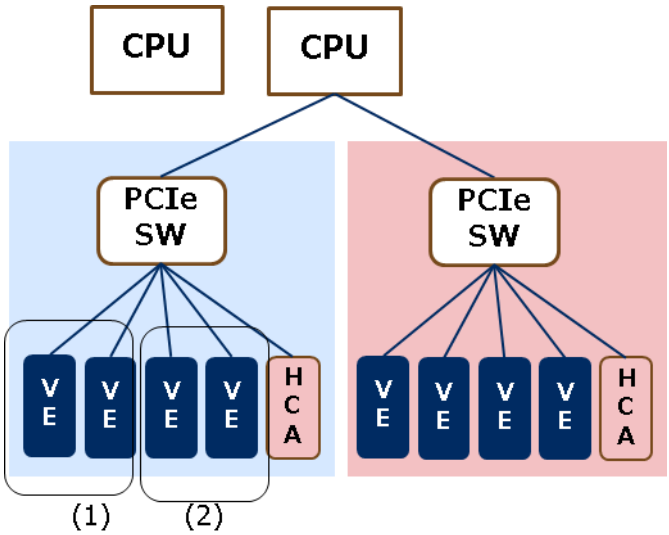


図 5-9：トポロジーを考慮した運用例 2

5.6　VEジョブのサスペンド

VEOS の Partial Process Swapping 機能（以降、PPS 機能）を利用することで、VE ジョブのメモリを VH またはストレージにスワップアウトすることや、VH またはストレージから VE ジョブのメモリを VE 上に戻すことが可能です。VE ジョブのサスペンドは、この機能を利用して実装しています。

VEOS の Partial Process Swapping 機能の設定方法については、SX-Aurora TSUBASA インストレーションガイドを参照してください。

PPS 機能を利用したサスペンドには、以下の 2 つの方法があります。

- システム管理者が**smgr**(1M)の**suspend request**サブコマンドを使用して、実行中のVEジョブをサスペンドする
- VEを使用する緊急/特別リクエストを実行する場合に、NQSVがVEを使用する通常リクエストをサスペンドして中断する

実行ホストが SX-Aurora-TSUBASA の場合、既定値では PPS 機能を利用したサスペンドが行われます。VE ジョブのサスペンド時に、PPS 機能を利用せずにプロセスの停止（SIGSTOP）のみの動作としたい場合は、システム管理者が **qmgr**(1M)の **set execution_queue partial_process_swapping** を **off** に設定してください。設定は、サスペンドされるリクエストを投入するキューに対して行います。本設定はバッチキューのみが対象で、会話キューに対して設定することはできません。

5.6.1　smgr(1M)コマンドによる VE ジョブのサスペンド

システム管理者は **smgr**(1M)の **suspend request** サブコマンドで、実行中の VE ジョブをサスペンドすることができます。この動作には、事前に PPS 機能の設定が必要です。

サスペンド中のリクエストは、**smgr**(1M)の **resume request** サブコマンドでリジュームします。その際、リクエストのサスペンド中の Elapse 時間は停止し、リジューム後に再び経過時間を計測します。

smgr(1M)コマンドでサスペンドまたはリジュームを行う際に、PPS 機能のスワップアウトやスワップインに失敗した場合は、当該リクエストはリランされます。

smgr(1M)コマンドでサスペンド中のリクエストをリジュームするときは、サスペンド中のリクエストが実行していた実行ホストに他のリクエストがない状態でリジュームを行ってください。他の実行中リクエストが存在する場合、リジュームが失敗する可能性があります。

---

1 つの実行ホスト内で複数実行している同時実行リクエスト（**qsub**(1)コマンドの**--parallel** オプションを指定して投入したリクエスト）は、**smgr**(1M)コマンドでの PPS を利用したサスペンドを利用することができません。

5.6.2　VE ジョブのサスペンドによる緊急リクエストの実行

VE を使用する通常リクエストをサスペンドして中断し、VE を使用する緊急/特別リクエストを実行する場合、PPS 機能によりスワップアウトが行われます。その際、緊急/特別リクエストはスワップアウトの完了を待ち合わせてから実行開始します。緊急/特別リクエストの終了後、サスペンドされたリクエストは自動的にスワップインし、リジュームします。

VE を使用するリクエストをサスペンドして実行した特別リクエストを、さらにサスペンドして緊急リクエストを実行することはできません。VE を使用するリクエストをサスペンドして実行する緊急リクエストの運用は、特別リクエストか緊急リクエストのいずれかで実施してください。

VE メモリ量不足時の緊急リクエストの削除

PPS機能によるスワップアウトを実施した場合でも、緊急リクエストの要求するVEメモリ量が確保できない可能性があります。このような場合、割り込まれた通常リクエストをリランし、緊急リクエストを実行することが既定の動作となります。

通常リクエストをリランさせない場合は、**qmgr**(1M) の **set execution_queue**

delete_failed_urgent_requestサブコマンドをonに設定してください。緊急リクエストの要求するVEメモリ量が確保できない場合、割り込まれた通常リクエストをリランせず、緊急リクエストを削除する動作となります。割り込まれた通常リクエストは、一旦サスペンドされたあと、緊急リクエストの削除後に再開します。本設定は緊急リクエストを投入するバッチキューに対して設定してください。

スワップする VE メモリ量の削減

緊急リクエストのVEノード毎のVEメモリサイズ制限値（--vememsz-venodeオプション）を適切に設定することにより、スワップアウトにかかる時間の短縮を期待できます。このためには下記の条件を全て満たしている必要があります。

- NQSV-ResourceManager、NQSV-JobServerのバージョンがR1.14以降。
- VEOSのバージョンが3.1.1以降。

条件を満たしている場合

緊急リクエストのVEノード毎のVEメモリサイズ制限値（--vememsz-venodeオプション)にunlimited以外の値を指定してください。これにより緊急リクエストが必要とするメモリ量だけをスワップアウトします。不要なメモリのスワップアウトを抑えることができ、スワップアウトにかかる時間の短縮を期待できます。

--vememsz-venodeオプションを指定しなかった場合は、投入したキューに設定されている既定値が適用されます。

unlimitedを指定した場合は、緊急リクエストが必要とするVEメモリ量にかかわらず、退避可能なVEメモリを全てスワップアウトします。従来と同様の時間がかかります。

条件を満たしていない場合

従来と同様に、緊急リクエストが必要とするVEメモリ量にかかわらず、退避可能なVEメモリを全てスワップアウトします。従来と同様の時間がかかります。

5.7　Dynamic JSV Priority

5.7.1　機能概要

5.6 VE ジョブのサスペンド で説明したとおり、VE を使用する緊急リクエストを実行する場合、通常リクエストを中断して、VE のメモリを一時的に開放します。緊急リクエストの要求する VE メモリ量が確保できない場合、割り込まれた通常リクエストをリラン、あるいは、緊急リクエストを削除します。

Dynamic JSV Priority を有効にすることで、VE を使用する緊急リクエストのスケジューリングにおいて、実行中 VE ジョブの状況に基づいてノード選択をすることができます。これにより下記を実現します。

- VE ジョブが実行中であるノードを避け、通常リクエストの中断回数を減らします。
- VE メモリをより多く開放可能なノードを優先します。それによりリクエストのリランや削除を抑えることができます。
- 実経過時間が短い VE ジョブの存在するノードを優先し、それによりリクエストをリランした際の影響を少なくします。

5.7.2　設定

Dynamic JSV Priority の設定はキュー単位で設定できます。本機能を有効にするには、操作員権限以上で **smgr**(1M)コマンドの下記のサブコマンドを実行してください。既定値は off です。
キュータイプが urgent 以外のキューに設定した場合、効果はありません。

Smgr: set queue dynamic_jsv_priority control = on *queue*

Dynamic JSV Priority は従来の JSV アサインプライオリティより優先のアサインポリシーです。詳細は 3.1.8 アサインポリシーを参照してください。

5.7.3　計算方法

Dynamic JSV Priority の値は、5 つの要素と最大 20 種類のカスタムリソースの要素によって算出された Priority 値の総和です。この値は緊急リクエストのスケジューリング時に JSV ごとに算出されます。計算方法の詳細は下記の通りです。

Dynamic JSV Priority = $\sum Priority$

Priority 値は次の計算式で算出します。
Priority 値は、要素の値を昇順または降順で計算することを選択できます。

昇順（**asce**）の場合 Priority = ((Current Item Value ***max_item_priority**) /**max_item_value**) * **base** 降順（**desc**）の場合 Priority = (**max_item_priority** - ((Current Item Value * **max_item_priority**) / **max_item_value**)) * **base**

base は基数です。
max_item_priority は **base** を掛ける前の Priority の最大値です。
Current Item Value はスケジューリング時点の要素の値です。
たとえば、そのノードにおいて 5 個の VE を使う通常ジョブが実行していた場合、Current Item Value は

5 です。

max_item_value は Current Item Value の最大値です。

Dynamic JSV Priority の値は、キューごとに **sstat**(1)コマンドの -Q -f -j オプションあるいは、-J --dynamic オプションで確認できます。この値は、直近に行われた緊急リクエストのスケジューリングでの値です。コマンドを実行した時点の状況とは変化している場合があります。

5.7.4　計算要素の設定

Priority の計算要素の設定はキュー単位で設定できます。操作員権限以上で **smgr**(1M)コマンドの set queue dynamic_jsv_priority item サブコマンドを実行してください。サブコマンドの詳細はリファレンス編を参照してください。

既定値の場合 Priority 値は 0 です。

付録 A　発行履歴

A.1　発行履歴一覧表

2018 年	2 月	初版
2022 年	9 月	第 15 版
2023 年	1 月	第 16 版
2023 年	3 月	第 17 版
2023 年	6 月	第 18 版
2024 年	7 月	第 19 版

A.2　追加・変更点詳細

- 第15版
  - 4.26 スケジューリング不可リクエストのキャッシュ機能を追加
  - 5.7 Dynamic JSV Priorityを拡張
- 第16版
  - 3.1.1 スケジューラマップの記述を充実
  - 4.6 事前予約機能の記述を充実
  - 4.21 最小ネットワークトポロジーノードグループ選択機能の記述を充実
  - 4.24 リクエストのアサインモードの記述を充実
- 第17版
  - 4.14.2 動的省電力運用機能に注意事項を追加
- 第18版
  - 4.21 smgr(1M)コマンドのサブコマンドに関する誤りを修正
  - 5.6.2 VEジョブのサスペンドによる緊急リクエストの実行にスワップするVEメモリ量の削減について追加

- 第19版
  - 4.23.6 クラウドインスタンス上のジョブサーバの設定 のqstatイメージを更新

索　引

へ

ゆ

よ

り

わ

NEC Network Queuing System V(NQSV)

利用の手引 [JobManipulator編]

2024年7月 第19版

日本電気株式会社

東京都港区芝五丁目７番１号

TEL(03)3454-1111（大代表）