NEC

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

[管理編]

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）によるジョブ管理システムの管理方法について説明したものです。

備考

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

(2)　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）

(3)　UNIX は The Open Group の登録商標です。

(4)　Intelは、アメリカ合衆国およびその他の国における Intel Corporation の商標です。

(5)　OpenStackは、アメリカ合衆国およびその他の国における OpenStack Foundation の商標です。

(6)　Red Hat OpenStack Platformは、アメリカ合衆国およびその他の国における Red Hat, Inc. の商標です。

(7)　Linux は Linus Torvalds氏の米国およびその他の国における登録商標あるいは商標です。

(8)　Dockerはアメリカ合衆国およびその他の国におけるDocker, Inc.の商標です。

(9)　InfiniBand は、InfiniBand Trade Associationの商標またはサービスマークです。

(10)　Zabbixはラトビア共和国にあるZabbix LLCの商標です。

(11)　その他、記載されている会社名、製品名は、各社の登録商標または商標です。

本書の対象読者

本書は、システム管理者を対象読者として書かれたものです。とくにジョブ管理システムの運用設計を行う際には、本書は有効な説明書となります。

本書は、対象読者が Linux の一般操作およびシステム管理全般を知っていて、【導入編】の内容を理解し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	ユニットの管理	ユニットの管理について説明しています	システム管理者(*)
2	バッチサーバの管理	バッチサーバの管理について説明しています	システム管理者(*)
3	実行ホストの管理	実行ホストの管理について説明しています	システム管理者(*)
4	キューの管理	キューの管理について説明しています	システム管理者(*)
5	リクエストの管理	リクエストの管理について説明しています	システム管理者(*)
6	クライアントの管理	クライアントの管理について説明しています	システム管理者(*)
7	リモート実行型会話機能	リモート実行型会話機能について説明しています	システム管理者
8	NQSVの環境保存	NQSVの環境保存について説明しています	システム管理者
9	MPIリクエスト実行環境の設定	MPIリクエスト実行環境の設定について説明しています	システム管理者
10	リクエストのグループ	リクエストのグループについて説明しています	システム管理者
11	グループ毎・ユーザ毎の制限	グループ毎・ユーザ毎の制限について説明しています	システム管理者
12	VEおよびGPU対応	VEおよびGPU対応について説明しています	システム管理者

章	タイトル	内容	対象読者
13	フックスクリプト機能	フックスクリプト機能について説明しています	システム管理者
14	ユーザプリ・ポストスクリプト機能	ユーザプリ・ポストスクリプト機能について説明しています	システム管理者
15	OpenStackと連携したプロビジョニング機能	OpenStackと連携したプロビジョニング機能について説明しています	システム管理者
16	Dockerと連携したプロビジョニング機能	Dockerと連携したプロビジョニング機能について説明しています	システム管理者
17	カスタムリソース機能	カスタムリソース機能について説明しています	システム管理者
18	ソケットスケジューリング	ソケットスケジューリングについて説明しています	システム管理者
19	障害検知・電源制御	障害検知・電源制御について説明しています	システム管理者
20	冗長化機能	冗長化機能について説明しています	システム管理者

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。複数のコアと共用メモリが実装されています。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の略語です。他ノードと通信するためのハードウェアです。

目　次

図目次

第1章　　ユニットの管理

1.1　はじめに

NQSV の各パッケージは複数のユニットから構成されています。ここでは NQSV を構成するユニットの管理について説明します。

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

NQSV の各パッケージには下記のユニットが含まれます。

パッケージ名	ユニット	
	ターゲットユニット	サービスユニット名と説明
NQSV/ResourceManager	nqs-bsv.target	nqs-bsv.service　バッチサーバ nqs-asv.service　アカウンティングサーバ nqs-acm.service　アカウンティングモニタ nqs-comd.service　コミュニケーションサーバ nqs-nag.service　ノード管理エージェント nqs-btu.service　冗長化機能の起動デーモン
NQSV/JobServer	nqs-jsv.target	nqs-jsv.service　ジョブサーバ nqs-lchd.service　ランチャーデーモン
NQSV/Client	nqs-uag.target	nqs-uag.service　ユーザエージェント
NQSV/JobManipulator	nqs-jmd.target	nqs-jmd.service　JobManipulator

拡張子が.service のユニットはサービスユニットと呼ばれ、個別の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。

各サービスユニットは同一パッケージ内のターゲットユニットに関連付けられていますが、パッケージインストール直後はこの関連付けは無効になっています。

このため、各ホストにおいて使用するサービスユニットについて関連付けを有効にします。

関連付けを有効にするには、root 権限で下記のコマンドを実行します。

```
# systemctl enable <UnitName>…
```

例えば、nqs-bsv.service と nqs-acm.service の関連付けを有効にするには、下記のコマンドを実行します。なお、拡張子が.service の場合は、拡張子を省略可能です。

```
# systemctl enable nqs-bsv.service nqs-acm.service
```

ターゲットユニットとの関連付けが有効になったユニットは、ターゲットユニットを起動することで一括して起動することができます。

ユニットを起動するには、root 権限で下記のコマンドを実行します。

```
# systemctl start <UnitName>…
```

例えば、先述の例に続いて下記のコマンドを実行すると、nqs-bsv.service と nqs-acm.service を一括して起動できます。

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

また、ターゲットユニットとの関連付けの有効・無効にかかわらず、サービスユニットを単体で起動することもできます。

例えば、下記のコマンドを実行すると、nqs-bsv.service が起動します。

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

ユニットを停止する場合は、root 権限で下記のコマンドを実行します。

```
# systemctl stop <UnitName>…
```

先述の例の場合、下記のコマンドを実行すると、nqs-bsv.target と関連付けられている nqs-bsv.service と nqs-acm.service が一括して停止します。

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

また、起動しているサービスユニットを単体で停止することもできます。

例えば、下記のコマンドを実行すると、nqs-bsv.service が停止します。

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

各パッケージに含まれるターゲットユニットは multi-user.target との関連付けが有効になっています。

これによって、OS が起動すると、関連付けを有効にしているユニットが自動起動します。

OS 起動時に自動起動させないようにするには、multi-user.target との関連付けを無効にします。

例えば、nqs-bsv.target を自動起動させないようにする場合は、下記のコマンドを実行します。

```
# systemctl disable nqs-bsv.target
```

第2章　　バッチサーバの管理

バッチサーバ（BSV）は、NQSV システムの中心コンポーネントです。リクエスト管理、キュー管理、各コンポーネント間の連携等を行います。（[導入編] バッチサーバ 参照）

2.1　バッチサーバのTCP/IPポート番号

NQSV は、バッチサーバを中心にスケジューラ、ジョブサーバ、ユーザコマンド等が TCP/IP ネットワークによってハブ状に接続されたシステムです。

このため、バッチサーバが常駐するバッチサーバホストと他のコンポーネントが実行されるホスト(クライアントホストや実行ホスト)間は TCP/IP で接続可能でなければなりません。

バッチサーバは、クライアントからの要求を受け付けるために TCP ポート 1 つをバインドします。

この TCP ポートのポート番号の既定値は、602 番です。

/etc/services にポート名"nqsv"でポート番号が登録されている場合は、そちらを優先します。

また、バッチサーバを起動する際に、nqs_bsvd コマンドに-p オプションでポート番号を指定することも可能です。

2.2　運用に必要なファイルとディレクトリ

バッチサーバの運用に必要なファイル、ディレクトリのうち主なものを以下に示します。

/etc/opt/nec/nqsv/nqs_user.map	... ユーザマップファイル
/etc/opt/nec/nqsv/nqs_passwd.def	... ローカルアカウント（ユーザ定義ファイル）
/etc/opt/nec/nqsv/nqs_group.def	... ローカルアカウント（グループ定義ファイル）
/etc/opt/nec/nqsv/nmap/	... マシン ID データベース
/etc/opt/nec/nqsv/nqs_bsv.env	... バッチサーバ起動用設定ファイル
/opt/nec/nqsv/sbin/nqs_bsvd	... バッチサーバ
/opt/nec/nqsv/sbin/nqs_logd	... ロギングサーバ
/opt/nec/nqsv/sbin/nqs_roud	... ルーティングサーバ
/opt/nec/nqsv/sbin/nqs_stgd	... ステージングサーバ
/opt/nec/nqsv/sbin/nqs_comd	... コミュニケーションサーバ
/opt/nec/nqsv/sbin/uex_prog/	... ユーザ EXIT スクリプト格納場所
/opt/nec/nqsv/sbin/stg_prog/	... ステージングスクリプト格納場所
/opt/nec/nqsv/bin/nmapmgr	... マシン ID データベース管理コマンド

/var/opt/nec/nqsv/ ... データベース
/var/opt/nec/nqsv/bsv/ ... バッチサーバデータベース
/var/opt/nec/nqsv/bsv/private/root/control/ ... コントロールファイル格納場所
/var/opt/nec/nqsv/bsv/private/root/data/ ... シェルスクリプト格納場所
/var/opt/nec/nqsv/bsv/private/root/database/queues
... キュー定義ファイル
/var/opt/nec/nqsv/bsv/private/root/database/netqueues
... ネットワークキュー定義ファイル
/var/opt/nec/nqsv/bsv/private/root/database/params
... バッチサーバ属性定義ファイル
/var/opt/nec/nqsv/bsv/private/root/database_qa/
... キューアクセスデータベース
/var/opt/nec/nqsv/bsv/private/root/input/ ... ステージインファイル格納場所
/var/opt/nec/nqsv/bsv/private/root/output/ ... ステージアウトファイル格納場所
/var/opt/nec/nqsv/bsv/private/root/failed/ ... 被障害コントロールファイル格納場所
/var/opt/nec/nqsv/bsv/private/root/outfai/ ... 未転送結果ファイル格納場所

- ユーザマップファイル
    - リモートユーザ名とローカルユーザ名のマッピングを定義するファイルです。NQSV システムへのアクセス権もこのファイルで定義します。(詳細は、2.9.1.ローカルユーザ名とリモートユーザ名、2.9.2.ユーザマッピング機能 にて説明します。)
- ローカルアカウントファイル
    - オペレーティングシステムに登録されていないアカウントを、NQSVで使用可能にするためのファイルです。(詳細は、2.9.3.ローカルアカウント機能 にて説明します。)
- マシンIDデータベース
    - バッチサーバ単位に付加されるマシンIDを格納するデータベースです。(詳細は、2.5. マシンIDの管理 にて説明します。)
- バッチサーバ
    - バッチサーバ本体です。
- ロギングサーバ
    - バッチサーバや他のNQSVサーバからのログ出力要求を処理します。
    - 既定値では /var/opt/nec/nqsv/batch_server_log へログを出力します。
    - また、致命的なエラーに関するログは、syslogへも出力されます（ファシリティ:

LOG_DAEMON, レベル: LOG_ERR)。syslogへの出力が困難な場合は、コンソールへ出力します。

- ルーティングサーバ
    - 転送キュー内にあるリクエストの転送を行うサーバです。転送処理開始時にバッチサーバから起動されます。
- ステージングサーバ
    - リクエストの結果ファイルの転送を行うサーバです。クライアントホスト上に常駐するユーザエージェントに接続し、結果ファイルの転送処理を行います。
    - 転送処理開始時にバッチサーバから起動されます。
- コミュニケーションサーバ
    - 他のバッチシステムとの連携を行うためのサーバです。バッチサーバホスト上に常駐します。(詳細は、2.10.バッチサーバ間の連携 にて説明します。)
- マシンIDデータベース管理コマンド
    - マシンIDの登録、削除や、登録済みマシンIDの表示等をおこなう管理者コマンドです。(詳細は、2.5.マシンIDの管理 にて説明します。)
- コントロールファイル格納場所
    - リクエストの実体であるコントロールファイルを格納するディレクトリです。
    - コントロールファイル内にはリクエストに関する全情報を含みます。
- シェルスクリプト格納場所
    - リクエスト投入時に指定されたシェルスクリプトを格納するディレクトリです。バッチジョブは、このシェルスクリプトの内容に従って実行されます。
- キュー定義ファイル
    - キュー(バッチキュー、会話キュー、および転送キュー)の定義を納めるファイルです。
- ネットワークキュー定義ファイル
    - ネットワークキューの定義を納めるファイルです。
- バッチサーバ属性定義ファイル
    - バッチサーバの各種属性を納めるファイルです。
- キューアクセスデータベース

    - キューに対するユーザごと、グループごとのアクセス可否情報を格納するデータベースです。

- ステージインファイル格納場所
    - ステージイン対象ファイルを一時的に格納するディレクトリです。
- ステージアウトファイル格納場所
    - ステージアウト対象ファイルを一時的に格納するディレクトリです。
- 被障害コントロールファイル格納場所
    - 致命的な障害によって処理を続けられなくなったリクエストのコントロールファイルを格納するディレクトリです。
    - バッチサーバは、このディレクトリ配下のファイルにアクセスしません。
    - したがって、不要なファイルは定期的に削除してください（バッチサーバ実行中に削除しても問題ありません）。
- 未転送結果ファイル格納場所
    - 障害などにより転送できなかった結果ファイルを格納するディレクトリです。
    - 必要であれば手動で転送してください。
    - バッチサーバは、このディレクトリ配下のファイルにアクセスしません。
    - したがって、不要なファイルは定期的に削除してください（バッチサーバ実行中に削除しても問題ありません）。

2.3　バッチサーバの設定

2.3.1　ログファイル

バッチサーバは、動作内容に関するログをファイルに出力します。ログファイルに関する設定は、qmgr(1M)の set batch_server log_file サブコマンドで行ってください。

設定できる内容は、次のとおりです。

- ログファイル名
- ログファイルの上限サイズ
- ログファイルへの出力レベル（レベルが大きいほど詳細な情報を出力します。各レベルで出力される内容は qmgr(1M)のset batch_server log_fileサブコマンド参照。）
- バックアップファイルの保存数

この属性が設定されていない場合の既定値は、 次のとおりです。

属性	既定値
ログファイル名	/var/opt/nec/nqsv/batch_server_log
ログファイルの上限サイズ	UNLIMITED（無制限）
ログファイルへの出力レベル	1
バックアップファイルの保存数	3

2.3.2　情報の採取間隔

バッチサーバは、実行ホストやバッチジョブに関する情報を採取します。

その情報の採取間隔設定は、qmgr(1M)のサブコマンドで行ってください。

各属性の設定に使用する qmgr(1M)のサブコマンドと、この属性が指定されない場合の既定値は、次のとおりです。

属性	qmgr(1M)サブコマンド	既定値
実行ホストの負荷情報やハードウェア資源搭載量情報採取間隔	set batch_server load_interval	30秒
ジョブの資源使用量採取間隔	set batch_server get_resource_interval	30秒

2.3.3　投入リクエスト数制限

バッチサーバでは、システムに投入できるリクエスト数の上限を設定できます。

投入できるリクエスト数とは、システム内に同時に存在しているリクエストの総数です。

リクエスト数の上限は、システム、グループ、ユーザごとに設定できます。

各設定は、qmgr(1M)のサブコマンドで行ってください。

各属性の設定に使用する qmgr(1M)のサブコマンドと、この属性が指定されない場合の既定値は、次のとおりです。

属性	qmgr(1M)サブコマンド	既定値
システム内に投入できるリクエスト数	set batch_server submit_limit	1000
1つのグループが投入できるリクエスト数	set batch_server group_submit_limit	0 (無制限)
1人のユーザが投入できるリクエスト数	set batch_server user_submit_limit	0 (無制限)

なお、グループ名・ユーザ名を個別指定してのリクエスト数制限も設定できます。詳細は、12.グループ毎・ユーザ毎の制限 を参照してください。

また、キュー単位の投入リクエスト数制限に関しては、4.1.2.バッチキューの設定 (5)、4.2.2.会話キューの設定 (5)、4.3.2.転送キューの設定 (2) を参照してください。

2.3.4　同時転送可能リクエスト数

バッチサーバでは、転送キュー、ネットワークキューの同時転送可能リクエスト数のシステム全体の値を 設定できます。

各設定は、qmgr(1M)のサブコマンドで行ってください。

各属性の設定に使用する qmgr(1M)のサブコマンドと、この属性が指定されない場合の既定値は、次のとおりです。

属性	qmgr(1M)サブコマンド	既定値	最大値
転送キューの同時転送可能リクエスト数	set batch_server routing_queue run_limit	100	100
ネットワークキューの同時転送可能ネットワークリクエスト数	set batch_server network_queue run_limit	1000	1000

【注意】 キューごとの同時転送可能リクエスト数に余裕があっても、システム全体が本属性で指定した値になるとそれ以上転送しません。
（転送キュー、ネットワークキューのキュー単位での設定については、4.3.2.転送キューの設定 (3)、4.4.2.ネットワークキューの設定 (2) 参照）

2.3.5　転送待ち時間

転送待ち時間とは、転送キューにおけるリクエスト転送処理や、ネットワークキューにおけるステージング処理に失敗した時刻を起点として、次に処理を開始するまでの時間間隔です。

この転送待ち時間は、転送キュー、ネットワークキュー別に設定できます。

各設定は、qmgr(1M)のサブコマンドで行ってください。

各属性の設定に使用する qmgr(1M)のサブコマンドと、この属性が指定されない場合の既定値は、次のとおりです。

属性	qmgr(1M)サブコマンド	既定値
転送キュー転送待ち時間	set batch_server routing_queue retry_interval	300秒
ネットワークキュー転送待ち時間	set batch_server network_queue retry_interval	300秒

2.3.6　転送リトライ期間

転送リトライ期間とは、転送キューにおけるルーティング処理や、ネットワークキューにおけるステージング処理を最初に開始した時刻を起点として再転送動作を繰り返す期間のことです。

この転送リトライ期間は、転送キュー、ネットワークキュー別に設定できます。

各設定は、qmgr(1M)のサブコマンドで行ってください。

各属性の設定に使用する qmgr(1M)のサブコマンドと、この属性が指定されない場合の既定値は、次のとおりです。

属性	qmgr(1M)サブコマンド	既定値
転送キュー転送リトライ期間	set batch_server routing_queue retry_span	259200 秒（3日）
ネットワークキュー転送リトライ期間	set batch_server network_queue retry_span	259200 秒（3日）

転送キューに投入されたリクエストが、リクエスト転送の失敗によるリトライ処理を繰り返した結果、retry_span で示される期間を越えた場合、該当リクエストは削除され、その旨を通知するメールがリクエスト所有者宛てに送信されます。

ネットワークキューにおいて、STAGING 状態にあるリクエストが retry_span を越えた場合は、それ以上リトライ処理は行われず、STAGING に失敗したとみなして QUEUED 状態へ戻ります。QUEUED 状態へ戻ったリクエストはスケジューラによって再び STAGING が開始されます。

EXITING 状態にあるリクエストが retry_span を越えた場合は、それ以上リトライ処理が行われず、EXITING に失敗したとみなして、結果ファイル転送を中止します。

転送が中止された結果ファイルは /var/opt/nec/nqsv/bsv/private/root/outfai/ に保存されます。

2.3.7　ハートビート送信間隔

バッチサーバとジョブサーバ間のハートビート送信間隔を設定できます。

設定は、qmgr(1M)の set batch_server heartbeat_interval サブコマンドで行ってください。

この属性が設定されていない場合の既定値は、60 秒です。

2.3.8　予算管理機能

NQSV は、予算超過のチェックを行い、予算を超過しているユーザやグループによるリクエスト投入およびリクエスト実行を拒否することができます。

本機能の詳細は［アカウンティング・予算管理編］を参照してください。

2.3.9　リクエスト・ジョブアカウント情報

バッチサーバに以下の設定を行うことにより、リクエストの実行に関するアカウンティング情報をファイル（リクエストアカウントファイル、ジョブアカウントファイル）に出力することができます。

リクエストアカウントファイル並びにジョブアカウントファイルの出力に関する設定は qmgr(1M) コマンドにより、次のサブコマンドを実行することで行います。

属性	qmgr(1M)サブコマンド	既定値
リクエストアカウント情報の出力開始	set batch_server req_account on	出力停止（off）
リクエストアカウント情報の出力停止	set batch_server req_account off	
リクエストアカウントファイルパス	set batch_server req_account_file	/var/opt/nec/nqsv/bsv/account/reqacct
ジョブアカウントファイルの出力ディレクトリ	set batch_server jacct_dir	/var/opt/nec/nqsv/acm/jacct

※アカウンティング機能の詳細は、［アカウンティング・予算管理編］を参照してください。

2.3.10 サブリクエスト生成数制限

パラメトリックリクエストの実行において、バッチサーバは、各パラメトリックリクエストに対して、同時に生成するサブリクエストの数を制限しています。

このサブリクエストの生成数の制限値の既定値は 100 となっています。

qmgr(1M)コマンドの”set batch_server subrequest_entry_limit”サブコマンドにより、サブリクエストの生成数制限値を変更することができます。

2.3.11 ライセンスの取得

NQSV のライセンスは別途構築されたライセンスサーバーによって管理します。

バッチサーバ（BSV）はライセンスサーバーに接続し、ライセンスサーバーが管理するライセンス情報をもとにライセンス利用可能数をチェックして、以下のライセンス対象プロダクトの接続可否を判断します。

・　NQSV/JobServer　　（JSVライセンスと呼びます）

・　NQSV/JobManipulator　　（JMライセンスと呼びます）

(1)　ライセンスサーバーの設定

```
/opt/nec/aur_license/aur_license.conf
```

指定方法は以下の通りです。

License_server_host = <ホスト名>

ライセンスサーバーのホスト名を指定します。

License_server_port = <ポート番号>

ライセンスサーバーの待ち受けポート番号を指定します。

(2)　ライセンスの取得

BSV は起動時にライセンスサーバーに接続し、自サーバで使用するライセンスを取得します。

なお、JSV ライセンス数が 0 の状態では BSV は起動しません。

一方、JSV ライセンスが利用可能であれば、JM ライセンス 0 の状態でも BSV は起動します。

■　同一のライセンスサーバーを複数のBSVで共有して利用する場合

各 BSV において使用するライセンスの数を、ライセンス種別ごとに定義する必要があります。

この時、各 BSV で定義した各種別のライセンス数の合計が、ライセンスサーバー上で利用可能なライセンスを超えないように定義してください。

ライセンス数の定義は BSV ホスト上の nqsd.conf ファイル（/etc/opt/nec/nqsv/nqsd.conf）に定義します。フォーマットは以下の通りです。

```
<ライセンス定義名> <ライセンス数>
```

ライセンス定義名として指定できる文字列は以下の通りです。

ライセンス定義名	内容
jsv_license_use	JSVライセンス数
jmn_license_use	JMライセンス数

例えば、JSV ライセンスを 10 個、JM ライセンスを 10 個使用する場合、以下のように定義します。

```
jsv_license_use 10
jmn_license_use 10
```

本定義ファイルで定義したライセンス数をライセンスサーバーから取得できなかった場合、BSV は起動しません。

【注意】

・上記ライセンス数の設定行がない場合や設定の記述が誤っている場合、ライセンス

サーバー上で管理されているライセンス数の最大値を取得することになります。
・同一のライセンスサーバーを複数の BSV で共有して利用する場合は、必ず本設定を行ってください。本設定を行わなかった場合の動作は不定です。

(3)　ライセンスサーバーの待ち合わせ

バッチサーバが起動するときには、ライセンスサーバーが既に起動している必要があります。
バッチサーバは、ライセンスサーバーとの通信が可能になるまで、一分間隔でリトライして待ち合わせます。
nqsd.conf ファイルにリトライする回数の上限を記述します。
書式は下記のとおりです。

```
retry_license n
```

n はリトライ回数の上限です。
0 以上 2147483647 以下の整数を指定可能で、0 を指定した場合は待ち合わせを行いません。
整数以外を指定した場合やこの行自体が存在しない場合は、5 が指定されたものとして扱います。
ライセンスサーバーを待ち合わせている間はバッチサーバを使用できません。
この状態で qstat などのコマンドを使用すると、バッチサーバに接続できないというエラーが表示されます。
ライセンスサーバーと通信が可能になり、ライセンスを取得できると、バッチサーバは通常稼働状態になります。
n で指定した回数をリトライしてもライセンスサーバーとの通信ができない場合は、バッチサーバは終了します。

2.3.12 ライセンスの更新

バッチサーバ運用中に、利用するライセンス数を以下のような操作で変更する場合、qmgr の update license サブコマンドを実行して、変更後の値を反映してください。

- ライセンスサーバー側でライセンス数の設定を変更した場合
- nqsd.confによるライセンス数の定義を変更した場合

qmgr の update license サブコマンドは以下のように管理者権限で実行します。

```
$ qmgr -Pm
Mgr: update license
Update License
```

2.3.13 利用可能なライセンスの参照

バッチサーバが保持している最大ライセンス数と現在払い出しているライセンス数は、qstat -Bf にて参照できます。

```
$ qstat -Bf
Batch Server: bsv.example.com
    NQSV Version = R1.00 (linux)
    Batch Server State = Active
     … 省略 …
  Use License :
    License (NQSV/JobServer)      = 36 (Max: 2048)
    License (NQSV/JobManipulator) = 36 (Max: 2048)
```

また、バッチサーバが保持するライセンスの詳細情報は、qstat -B -L -Pm によって参照できます。

```
$ qstat -B -L -Pm
Sysname    Product             Ver   Expiration    Num
---------- ------------------- ----- ----------- -----
VESYS      NQSV/JobServer      V1.0  31-Aug-2019  2048
VESYS      NWSV/JobManipurator V1.0  01-Sep-2019  2048
```

2.3.14 DB への遅延書き込み

バッチサーバが管理するリクエストの情報を DB 内のファイルに書き込む際、遅延書き込みによってリクエスト処理性能を大幅に向上させることができます。

本機能は、以下の形式で nqsd.conf ファイル（/etc/opt/nec/nqsv/nqsd.conf）に設定してください。その後、バッチサーバを再起動、または qmgr の load nqsd_conf サブコマンドで設定ファイルを再読み込みすることで有効になります。

```
no_sync_mode on
```

本設定を行わない場合は、同期 I/O モードで書き込みを行います。本設定を行うと、データの書き込み時に同期が行われないため、万が一障害が発生してバッチサーバが異常終了した場合、そのタイミングで I/O 行っていたリクエストのデータが破損する可能性が高くなります。また、バッチサーバの DB 領域（/var/opt/nec/nqsv/bsv/）のファイルシステムの種類やマウントのオプションによっては書き込みに遅延が生じる可能性があります。そのため、本設定を行う際はこの点に注意して実施の有無を判断してください。

2.3.15 ジョブ実行時の SIGTERM 捕捉

ジョブ実行時、デフォルトではジョブを実行するシェルを起動する際に SIGTERM を無視するように設定します。これはジョブの資源制限超過時に送信される SIGTERM によって、ジョブの起動シェルが他のプロセスよりも先に終了してしまうことで、正確なジョブの終了ステータスが得られなくなることを防止するために行っています。一方、この設定があることで、ジョブスクリプトにおいて SIGTERM の捕捉が必要である trap コマンドを利用することができません。

qsub 時に sigterm 対応オプション（--accept-sigterm）を指定することにより、ジョブ実行時の SIGTERM 捕捉を有効にしてジョブスクリプト内で trap コマンドを利用できるようになりますが、以下の設定を行うことで、投入される全てのリクエストに対して常に SIGTERM 捕捉を有効にするように設定することができます。qsub コマンドにおける sigterm 対応オプションの詳細については[操作編] を参照してください。

本機能は、nqsd.conf ファイル（/etc/opt/nec/nqsv/nqsd.conf）に以下のフォーマットで設定し、バッチサーバを再起動または qmgr の load nqsd_conf サブコマンドを実行することで有効になります。

```
accept_sigterm yes
```

本指定がない場合、および指定方法が誤っている場合はでデフォルトの動作となり、qsub の sigterm 対応オプションが有効な場合にのみ trap コマンドが利用できます。

[注意事項]

qsub 時の sigterm 対応オプションまたは本設定により trap コマンドを有効に設定した場合、ジョ

ブが資源制限超過で終了した際に、プロセスの終了状況によっては、ジョブアカウントに記録される終了ステータスがデフォルト（本オプションが無効の場合）とは異なる場合があります。

qstat コマンドでリクエストの情報を参照すると、本設定による trap コマンドを有効・無効に関わらず、リクエスト投入時に指定した sigterm 対応オプションの指定内容がそのまま表示されます。そのため、qstat での表示内容と実際の動作が異なる場合があります。

2.3.16 リクエスト ID のシーケンス番号最大値の設定

リクエスト ID のシーケンス番号の最大値を変更することができます。設定範囲は 999999～99999999 とし 100 万件から最大 1 億件のリクエストに対応可能となります。

本機能は、nqsd.conf ファイル（/etc/opt/nec/nqsv/nqsd.conf）に以下のフォーマットで設定し、バッチサーバを再起動または qmgr の“load nqsd_conf“サブコマンドを実行することで有効になります。

```
rid_seqno_max <value>
```

<value>は 999999～99999999 の範囲の整数で指定してください。本指定がない場合は、リクエスト ID のシーケンス番号の最大値が既定値 999999 となります。また、下限（999999）未満の値を指定した場合はリクエスト ID のシーケンス番号の最大値は 999999 とし、上限（99999999）を超える値を指定した場合はリクエスト ID のシーケンス番号の最大値は 99999999 となります。

2.3.17 次に投入されるリクエストのシーケンス番号の設定

次に投入されるリクエストに割り当てるシーケンス番号をカスタマイズすることができます。設定範囲は、0～シーケンス番号の最大値となります。シーケンス番号の最大値については、「2.3.16　リクエスト ID のシーケンス番号の最大値の設定」を参照してください。

本機能は、seqno.conf ファイル（/var/opt/nec/nqsv/seqno.conf）に以下のフォーマットで設定し、バッチサーバを再起動または qmgr の“load nqsd_conf“サブコマンドを実行することで有効になります。

```
<value>
```

<value>は 0～リクエスト ID のシーケンス番号の最大値の範囲の整数で指定してください。範囲外の値が指定された場合は指定値が無視されます。また、指定された値で割り当てられているリクエストがすでに存在する場合は、指定値が無視され、その値より大きい使用可能な値が割り当てられます。

seqno.conf ファイルは、NQSV/ResourceManager のパッケージではインストールされません。設定を変更したい場合は、/var/opt/nec/nqsv/配下にファイルを作成してください。障害等での BSV 再起動時に seqno.conf の以前の設定値を誤って適用することを避けるため、seqno.conf の設定値を適用後に当該ファイルは seqno.conf.accept にリネームされます。

2.3.18 ユーザ通知用スクリプトの設定

ノードヘルスチェック機能では、異常を検知したリクエストに対してメール等でリクエストのオーナーに通知することができます。通知はシェルスクリプトで行います。スクリプトの内容やスクリプトのディレクトリはユーザが自由にカスタマイズすることができます。

このスクリプトは nqsd.conf ファイル（/etc/opt/nec/nqsv/nqsd.conf）に以下のフォーマットで設定してください。その後、バッチサーバを再起動、または qmgr の“load nqsd_conf“サブコマンドを実行することで有効になります。

```
notify_script_path <script_path>
```

<script_path>はユーザ通知用スクリプトのパスを指定してください。上記が指定されていない場合は、下記のデフォルトの通知用スクリプトを使用します。

/opt/nec/nqsv/sbin/notify_prog/nqsv_notify.sh

スクリプトの作成方法やデフォルトのスクリプトの動作については、「19.5 ノードヘルスチェック機能」を参照してください。

2.3.19 memory cgroup によるメモリ管理

Linux カーネルの cgroups を使用したメモリ管理が利用できます。ジョブ毎のメモリ使用量の取得とリソース制限が正確に行えるようになります。本機能の有効、無効はバッチサーバ全体で選択できます。本機能のメモリ管理方法をキュー毎や実行ホスト毎に変更することはできません。

本機能を利用する場合は、nqsd.conf ファイル（/etc/opt/nec/nqsv/nqsd.conf）に以下のフォーマットで設定を記載してください。その後、バッチサーバの再起動、または qmgr の“load nqsd_conf“サブコマンドの実行により、設定ファイルを再読み込みすることで有効になります。

設定は 3 種類のコンフィグパラメータにより行います。

1. enable_memory_cgroup

メモリ資源の管理に memory cgroup を使用します。また、ジョブ毎のプロセス ID 管理も cgroup を使用するように変更されます。以下のフォーマットで設定を記載してください。 この指定がない場合、本機能は無効（既定値）です。

```
enable_memory_cgroup on
```

2. memory_cgroup_excluding_cache

memory cgroup によるメモリ管理が行われている場合、ファイルキャッシュもメモリ使用量に含まれます。ファイルキャッシュをメモリ使用量から除外するには、以下のフォーマットで設定を記載

してください。この指定がない場合、本機能は無効（既定値）です。

```
memory_cgroup_excluding_cache on
```

3.　enable_cgroup_process_list

ジョブ毎のプロセス ID 管理に cgroups を使用するようになります。以下のフォーマットで設定を記載してください。この指定がない場合、本機能は無効（既定値）です。ただし、この指定がない場合、かつ enable_memory_cgroup が on の場合、本機能は有効になります。

```
enable_cgroup_process_list on
```

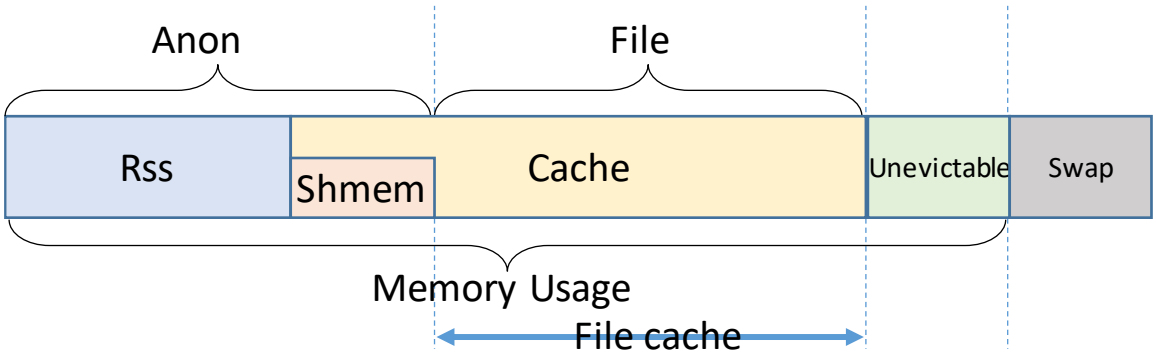
これらコンフィグパラメータの組み合わせは以下のようになります。

*1: enable_memory_cgroup
*2: memory_cgroup_excluding_cache
*3: enable_cgroup_process_list

Config parameters			Function performed		
*1: memory	*2: cache	*3: process	*1: memory	*2: cache	*3: process
on	on	-	yes	yes	-
on	off	-	yes	no	-
on	*not set*	-	yes	no	-
on	-	on	yes	-	yes
on	-	off	yes	-	no
on	-	*not set*	yes	-	yes
off	*don't care*	on	no	no	yes
off	*don't care*	off	no	no	no
off	*don't care*	*not set*	no	no	no
not set	*don't care*	on	no	no	yes
not set	*don't care*	off	no	no	no
not set	*don't care*	*not set*	no	no	no

【注意】

- memory_cgroup_excluding_cache on 指定により除外されるファイルキャッシュは Cache から Shmem を引いた領域となります。



- memory_cgroup_excluding_cache off 指定時は、下記機能のメモリ使用量にファイルキャッシュが含まれます。
  - qstat で表示されるリクエスト・ジョブの基本情報表示・詳細情報表示
  - ジョブアカウント・リクエストアカウントのメモリ使用量
  - メモリ使用量の実績値に対する課金

対象となるメモリ資源制限

以下のメモリ資源制限の最大値の制限が対象となります。

項目	意味
-l memsz_job	ジョブ(論理ホスト)ごとのメモリサイズ
--memsz-lhost	ジョブ(論理ホスト)ごとのメモリサイズ

仮想メモリサイズ制限(--vmemsz-lhost)、警告値の制限は従来通りの制御になります。

メモリ資源制限超過時の動作

NQSV は memory.oom_control のフラグを oom_kill_disable 1 として管理します。メモリ資源制限超過時には、プロセスは OOM Killer で KILL されずに、停止状態になります。NQSV は OOM により停止されたプロセスを確認し、SIGKILL 送信によりジョブを強制終了処理します。

qstat -Jf によるメモリ使用量の参照

memory cgroup によるメモリ管理が有効な場合、qstat -Jf によりメモリ使用量の詳細が表示されます。

```
Resources Information:
  Memory     = 580.878906MB
  Memory Cgroup Resources = {
    Memory Usage       = 580.906250MB
    Max Memory Usage   = 606.296875MB
    Cache              = 529.683594MB
    Rss                = 51.222656MB
    Shmem              = 529.656250MB
    Unevictable        = 0.000000B
    Swap               = 25.343750MB
  }
```

2.3.20 リクエスト結果ファイルのステージアウト処理バッファの調整

バッチサーバからユーザエージェントへ、リクエスト結果ファイルを内部ステージング方式で転送する際に使用するバッファサイズを調整する機能です。

ご使用のファイルシステムの種類やマウントのオプションに適したファイルバッファサイズを指定します。併せて書き込み保障であるシンクモード（sync の有無）を指定します。

本機能は以下の書式で nqsd.conf ファイル（/etc/opt/nec/nqsv/nqsd.conf）に設定してください。その後、バッチサーバを再起動、または qmgr の load nqsd_conf サブコマンドによる、設定ファイル再読み込みを行うことで有効になります。

項目	意味
file_transfer_buffer_size	ご使用のファイルシステムの種類や、マウントのオプションに適したファイルバッファサイズをバイト単位で指定します。 有効範囲は 1024～10485760 (1Kbyte～10Mbytes)です。
file_transfer_sync_mode	ファイルを閉じるときに、書込み処理を同期させるか否かを指定します。 同期させる場合は on、同期させない場合は off です。

設定例:ファイルバッファサイズ 8Kbyte、sync 有りの場合

```
file_transfer_buffer_size 8192
file_transfer_sync_mode on
```

バッファサイズの指定は file_transfer_buffer_size に続けてバッファサイズを byte 単位で記入ください。有効範囲は 1024～10485760 (1Kbyte～10Mbytes)です。

シンクモードの指定は file_transfer_sync_mode に続けて on または off を記入ください。

本設定を行わない場合は、リクエスト結果ファイルのステージアウト処理バッファは 1048576 (1Mbyte)で動作します。シンクモードは on（ファイルクローズ前に sync 有り）で動作します。

値が未指定、または設定範囲外の場合は、上記設定を行わない場合と同じ動作です。

ご使用の環境（同時並走するリクエストの数、ファイルシステム種類やマウントのオプション）によっては、処理遅延が生じる可能性があります。そのため本設定を行う際は、この点に注意して値を決定してください。

2.3.21 リクエスト転送時の投入リクエスト数制限超過処理変更

転送キューからバッチキューへのリクエスト転送は、キューに設定された投入リクエスト数制限の影響を受けます。この投入リクエスト数制限を超過した場合の動作を変更します。

バッチキューの投入リクエスト数制限を設定する qmgr(1M)のサブコマンドと、指定されていない場合の既定値は、次のとおりです。

属性	qmgr(1M)サブコマンド	既定値
キューに投入できるリクエスト数	set execution_queue submit_limit	0（無制限）
1つのグループがキューに投入できるリクエスト数	set execution_queue group_submit_limit	0（無制限）
1人のユーザがキューに投入できるリクエスト数	set execution_queue user_submit_limit	0（無制限）

これらの投入リクエスト数制限はそれぞれの制限値に従って動作しますが、例えばあるユーザが転送キューへのリクエストを投入し、転送先バッチキューがユーザの投入リクエスト数制限を超過した場合に、転送先バッチキューは一時的にリクエスト転送が不可となり、他のユーザのリクエスト転送が行われないことがあります。

これを変更し、あるグループやユーザの転送キューへのリクエストの投入が、転送先キューの投入リクエスト数制限超過となった時に、他のグループやユーザがそのキューへリクエスト転送できるようにします。以下の書式で nqsd.conf ファイル（/etc/opt/nec/nqsv/nqsd.conf）に設定してください。

```
change_routing_retry_behavior on
```

この指定がない場合、本機能は無効（既定値）です。その後、バッチサーバを再起動、または qmgr の load nqsd_conf サブコマンドによる、設定ファイル再読み込みを行うことで有効になります。

これにより動作が変更されるのは、転送キューからのリクエスト転送時に、バッチキューの１つのグループが投入できるリクエスト数(group_submit_limit)と、１つのユーザが投入できるリクエスト数(user_submit_limit)が制限超過した場合のみとなります。キューに投入できるリクエスト数(submit_limit)や、それ以外のキューへの制限については動作変更しません。

2.4　バッチサーバの状態確認

バッチサーバの状態を確認するときは、qstat(1) コマンドを -B オプションを付けて実行してください。

```
$ qstat -B
BatchServer      MachineID UMAX GMAX RRL  NRL  TOT ARR WAI QUE RUN EXT HLD SUS Status
---------------- --------- ---------- -------------------------------- ------
bsv.example.com        100    0    0 100 1000    0   0   0   0   0   0   0   0 Active
```

なお、バッチサーバの詳細情報を表示する場合は、qstat(1)コマンドに -B、-f オプションを付けて実行してください。

```
$ qstat -B -f
Batch Server: bsv.example.com
    NQSV Version = R1.00 (linux)
    Batch Server State = Active
    Batch Server Machine ID = 404
    Logfile Path       = /var/opt/nec/nqsv/batch_server_log
    Logfile Level      = 2
    Logfile Save Count = 3
    Logfile MAX Size   = 512.0MB
    Accounting Server Host Name   = asv.example.com
    Accounting Server Port Number = 4595
    Jacct directory = /var/opt/nec/nqsv/acm/jacct
    Budget function              = OFF
    Request Accounting           = OFF
    Request Accounting File Path = /var/opt/nec/nqsv/bsv/account/reqacct
    Reservation Accounting       = OFF
    Reservation Accounting File  = /var/opt/nec/nqsv/acm/rsvacct/rsvacct
    Specify Group for Request = OFF
    Allow Absolute Exepath = OFF
    Auto Delete Failed Request = ON
    Total Request         = 0
    Arriving Request      = 0
    Waiting Request       = 0
    Queued Request        = 0
    Pre-running Request   = 0
    Running Request       = 0
```

```
    Post-running Request  = 0
    Exiting Request       = 0
    Held Request          = 0
    Holding Request       = 0
    Restarting Request    = 0
    Suspending Request    = 0
    Suspended Request     = 0
    Resuming Request      = 0
    Migrating Request     = 0
    Transiting Request    = 0
    Staging Request       = 0
    Checkpointing Request = 0
    Forwarding Request    = 0
    Heart Beat Interval   = 60S
    Load Interval         = 30S
    Get Resource Interval = 30S
    Max Subrequest Entry Limit = 100
    Max Run Limit of Routing Queue  = 100
    Retry Interval of Routing Queue = 300S
    Retry Span of Routing Queue     = 259200S
    Max Run Limit of Network Queue  = 1000
    Retry Interval of Network Queue = 300S
    Retry Span of Network Queue     = 259200S
    Submit Number Limitation Value       = 1000
    Submit User  Number Limitation Value = UNLIMITED
    Submit Group Number Limitation Value = UNLIMITED
  Use License :
    License (NQSV/JobServer)      = 1 (Max: 10)
    License (NQSV/JobManipulator) = 0 (Max: 10)
```

qstat(1)コマンドで出力する項目（アイテム）については、-F オプションを使用することにより、必要なものだけ選択し、それらを組合せて表示することが可能です。

```
$ qstat -B -F bsvhost,mid,bstt
BatchServer      MachineID Status
---------------- ---------- ------
bsv.example.com        147 Active
```

指定可能なアイテムは以下のとおりです。

-F指定アイテム	内容	サマリ表示形式
bsvhost	バッチサーバホスト名	BatchServer
mid	マシンID	MachineID
umax	ユーザ毎のリクエスト投入数制限値	UMAX
gmax	グループ毎のリクエスト投入数制限値	GMAX
rrlm	転送キュー同時実行数制限値	RRL
nrlm	ネットワークキュー同時実行数制限値	NRL
tot	バッチサーバが管理しているリクエスト数	TOT
arr	各状態のリクエスト数	ARR
wai		WAI
gqd		GQD
que		QUE
run		RUN
ext		EXT
hld		HLD
sus		SUS
fwd		FWD
bstt	バッチサーバの状態	Status

（qstat(1)コマンド-F オプションの使用方法詳細については、[操作編]出力情報のカスタマイズ を参照してください。）

qstat(1)コマンドの-o オプション（昇順ソート）、または-O オプション（降順ソート）を使用することで、出力情報を任意の項目をキーとしてソートすることも可能です。

ソート対象項目名は上記の表にて示されるアイテムと共通であり、","で区切ることで複数指定することができます。（ソートオプションの使用方法詳細については、[操作編] 出力情報のソート を参照してください。）

qstat(1)コマンドのサマリ表示(-f オプションなし)では、出力する内容が規定の文字数で表示されるため、実行ホスト名等が途中で切られてしまうケースがあります。

-l オプションを指定することにより、途中で切ることなく全内容を表示することが可能です。

（-l オプションの使用方法詳細については、[操作編] 全内容の表示 を参照してください。）

2.5　マシンIDの管理

2.5.1　マシン ID とは

マシン ID とは、バッチサーバを特定するために使用される 32 ビット長の整数値です。

NQSV では、あらかじめ、バッチサーバに対してマシン ID を割り当てて、登録しておく必要があります（クライアントホスト、実行ホストに対しては、マシン ID を登録する必要はありません）。

2.5.2　マシン ID の登録

マシン ID の登録は、nmapmgr(1M)コマンドで、"add mid"サブコマンドを使用します。

ホスト名が host0.example.com であるホストにマシン ID 10 を割り当てる場合は、以下のように実行してください。nmapmgr コマンドは root 権限で起動してください。

```
# nmapmgr
NMAPMGR>: add mid 10 host0.example.com
```

マシン ID は、ホストに対して一つ割り当てます。 ホストが持つネットワークインタフェースごとにマシン ID を割り当てる必要はありません。

2.5.3　エイリアスの登録

バッチサーバホストが、複数のネットワークインタフェースを持つ等で、ホスト名が複数存在する場合は、上記の"add mid"サブコマンドで登録したホスト名以外の各ホスト名を同じマシン ID に対するエイリアスとして登録してください。

エイリアスの登録には、nmapmgr コマンドの"add name"サブコマンドを使用します。

```
# nmapmgr
NMAPMGR>: add name host1.example.com 10
```

エイリアス名を登録することにより、バッチサーバがどのホスト名でアクセスされても常に同一のマシン ID に変換されるため、バッチサーバを一意に特定することができます。

2.6　バッチサーバデータベースの初期化

NQSV システムにおいて、バッチサーバを運用するためには、バッチサーバの専用データベース（/var/opt/nec/nqsv/bsv/）が構築されていなければなりません。

したがって、インストール直後や何らかの理由でデータベースが破壊された場合は、バッチサーバデータベースの初期化を行ってください。

バッチサーバデータベースの初期化を行うには、root 権限で、バッチサーバ（/opt/nec/nqsv/sbin/nqs_bsvd）を -i オプション付で起動してください。

```
# /opt/nec/nqsv/sbin/nqs_bsvd -i
BatchServer database was initialized.
```

すでにバッチサーバデータベースが存在する場合は、確認を求めるメッセージが表示され、入力を求められます。

```
# /opt/nec/nqsv/sbin/nqs_bsvd -i
 BatchServer database is already exists. May I delete it(yes/no/save)?
```

この入力要求に対する入力により、バッチサーバは以下のように処理を行います。

- yesと入力した場合:
    - 既存のデータベースを削除した後、データベースを作成します。
- noと入力した場合:
    - データベースの初期化を中止します。
- saveと入力した場合:
    - 既存のデータベースを/var/opt/nec/nqsv/bsv.oldとして保存した後、データベースを作成します。

2.7　バッチサーバの起動

バッチサーバ単体を手動で起動する場合は、root 権限で下記のコマンドを実行してください。

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

バッチサーバホスト起動時に自動的にバッチサーバを起動する場合は、nqs-bsv サービスユニットを有効化してください。

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

systemctl enable を設定したサービスユニットを、手動でまとめて起動する場合は下記のコマンドを実行してください。

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

【注意】バッチサーバ起動時に所有者が不明のリクエストは削除されます。
ただし、nqs_bsvd に -u オプションを指定して起動すると、所有者不明のリクエストが存在する場合はバッチサーバの起動を中止し、リクエストを保持します。
本オプションにより、一時的にユーザ認証サービスが停止している場合等のバッチサーバ起動によるリクエスト削除を防止できますが、バッチサーバ停止中にバッチサーバが保持するリクエストの所有者のユーザアカウントが失効した場合にバッチサーバを起動することができません。
このような場合は、本オプションを指定しないでバッチサーバを起動することで、従来どおりリクエストが削除されバッチサーバが起動します。

2.8　バッチサーバの停止

バッチサーバはバッチサーバホストのシャットダウン時に自動的に停止します。

バッチサーバ単体を手動で停止する場合は、root 権限で下記のコマンドを実行するか、qmgr(1M)の shutdown サブコマンドを実行してください。

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

また、下記のコマンドを実行すると、systemctl enable を設定したサービスユニットをまとめて停止します。

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

バッチサーバが停止すると、それに接続していた各ジョブサーバは、バッチサーバのダウンを検出し、定期的に再接続を試みるモードに移行しますが、ジョブサーバ上でのジョブの実行はそのまま続行します。

また、ジョブサーバ上で実行中のジョブが存在する状態で、バッチサーバがダウンした場合も、ジョブの実行は正常に継続されます。

バッチサーバのダウン中にジョブが終了した場合、ジョブサーバはその終了状態を自身のデータベース内（ジョブサーバデータベースの詳細は、3.1.運用に必要なファイルとディレクトリ 参照）に記録しバッチサーバに再接続するまで保存します。

バッチサーバが再起動すると、各ジョブサーバはバッチサーバに再接続します。

その時点で保存されたジョブの状態をバッチサーバにレポートします。

バッチサーバは、この情報を元にジョブの管理データを再構築し、通常の運用状態に戻ります。

2.9　ユーザの管理

2.9.1　ローカルユーザ名とリモートユーザ名

ローカルユーザ名とは、バッチサーバホスト上のユーザ名です。

NQSV では、アクセス権のチェックやリクエスト所有者の特定などにローカルユーザ名を使用します。

リモートユーザ名とは、ユーザコマンドが実行されるクライアントホスト上やジョブが実行される実行ホスト上のユーザ名です。

ローカルユーザ名とリモートユーザ名が異なる場合は、両ユーザ名間の関係を定義してください。NQSV にはローカルユーザ名とリモートユーザ名の関係を定義するユーザマッピング機能と、独自のユーザアカウントを定義するローカルアカウント機能が実装されており、ローカルユーザ名とリモートユーザ名間の関連付けを柔軟に定義できます。

2.9.2　ユーザマッピング機能

ユーザマッピング機能とは、バッチサーバホスト上のユーザマップファイル（/etc/opt/nec/nqsv/nqs_user.map）の定義に従って、ローカルユーザ名とリモートユーザ名を関連付ける機能です。

/etc/opt/nec/nqsv/nqs_user.map ファイルを更新すると、更新内容はバッチサーバによるユーザマッピング処理に対して即座に反映されます。

ユーザマッピングには、リモートユーザ名からローカルユーザ名へのマッピング（RL マッピング）とその逆（LR マッピング）があります。

それぞれ以下のタイミングでマッピング処理を実施します。

- RLマッピング

  スケジューラがバッチサーバへ接続する時

  （スケジューラを起動したリモートユーザ名からローカルユーザ名へのマッピング）

  ユーザコマンド、管理者コマンドがバッチサーバへ接続する時

  （コマンドを起動したリモートユーザ名からローカルユーザ名へのマッピング）

- LRマッピング

  バッチサーバからの指示によりジョブサーバがジョブを生成する時

  （リクエスト所有者であるローカルユーザ名からジョブ実行者であるリモートユーザ名へのマッピング）

バッチサーバからの指示により他の実行ホスト上へジョブがマイグレーションされる時

（リクエスト所有者であるローカルユーザ名からジョブ実行者であるリモートユーザ名へのマッピング）

バッチサーバからクライアントホストへリクエストの結果ファイルを転送する時

（リクエスト所有者であるローカルユーザ名から結果ファイルの所有者であるリモートユーザ名へのマッピング）

ユーザマップファイルの形式

以下はユーザマップファイルのサンプルです。

```
#
# NQSV User Mapping File.
#

DefaultPrivilege PRIV_SPU

#-------------------------------------------------------------------------
# Privilege   LocalUser    RemoteUser(s)
#-------------------------------------------------------------------------

PRIV_SCH      root         root:127.0.0.1/32
PRIV_NON      root         root:0.0.0.0/0
PRIV_MGR      nqsmgr       user00:192.168.77.0/24  user01:172.16.128.0/25
PRIV_OPE      nqsope       user00:192.168.10.0/24
```

ユーザマップファイルの記述形式は以下のとおりです。行数は上記サンプルファイルに対応します。

- #記号から行末までと空行は無視します。
- デフォルトのアクセス権を、DefaultPrivilegeに指定します。（5行目）
- その他の行でローカルユーザ名とリモートユーザ名を関連付けます。（11行目から14行目）
- 各行の優先順位は上から下です（より上位の行を優先）。
- 第3カラム以降に複数のリモートユーザ指定がある場合は、より左で定義されたリモートユーザを優先します。

・

ローカルユーザ名とリモートユーザ名を関連付ける行の詳細は、以下のとおりです。

- 第１カラム

・　この行にマッチしたリモートユーザに対するアクセス権の上限を指定します。下表のキーワード内の一つを指定してください。

キーワード	種別	説明
PRIV_SCH	スケジューラ権限	最も高い権限でバッチサーバが持つ全機能にアクセスできます。主にスケジューラが使用します。
PRIV_MGR	管理者権限	管理者用の権限です。キューの作成や削除、他人のリクエストの削除、サーバの停止等ができます。PRIV_SCHの次に高い権限です。
PRIV_OPE	操作員権限	操作員用の権限です。 キューの停止や再開、他人のリクエストのホールド／リリース、各種の属性値変更等ができます。
PRIV_GMGR	グループ管理者権限	グループ管理者用の権限です。 管理するグループの範囲に限定して管理者の権限を持ちます。リクエストの投入グループが管理するグループである場合のみ、削除、ホールド／リリース、各種属性値変更等ができます。
PRIV_SPU	特別利用者権限	特別利用者用の権限です。一般利用者用の権限に加えて、他人が所有するリクエストやバッチジョブの属性値を参照できます。
PRIV_USR	一般利用者権限	一般利用者用の権限です。最も低い権限で、自身が所有するリクエストの操作や許可されているキュー、サーバ属性値の参照等しかできません。
PRIV_NON	権限なし	NQSVへのアクセス権がありません。アクセスを禁止するリモートユーザに対して指定します。

- 第 2 カラム

・　ローカルユーザ名を指定します。

RLマッピングでは、第3カラム以降のリモートユーザ指定にマッチしたユーザがこのローカルユーザ名にマップされます。

LRマッピングでは、このローカルユーザ名に対応するリモートユーザ名を第3カラム以降のリモートユーザ指定の中から検索します。

- 第 3 カラム以降

・　リモートユーザリストを指定します。複数指定の場合はスペースで区切ってください。

RLマッピングでは、このリスト内からリモートユーザ名を検索します。

LRマッピングでは、第2カラムで指定されたローカルユーザ名をこのリスト中でマッチするリモートユーザ名にマップします。

・　リモートユーザリストの構文は以下のとおりです。

```
<ユーザ名>:<IPアドレス>/<ビットマスク> [<ユーザ名>:<IPアドレス>/<ビットマスク> ...]
```

・　対象リモートホストのIPアドレスと<IPアドレス>で指定されたIPアドレスを<ビットマスク>で指定された ビット数分だけ左からチェックします。

・　一致していれば<ユーザ名>をリモートユーザ名として採用します。

・　次行：(PRIV_GMGR 指定の場合のみ)

・　グループ管理者権限 PRIV_GMGR の指定行に限り、: から始まる次行で管理対象グループを定義します。管理するグループは、スペースで区切り、複数指定できます。指定されたグループ名が不正の場合は、無視します。

・　: から始まる次行がない場合、管理対象グループがないものとします。

・　従って、グループ管理者権限のユーザマッピングの構文は２行指定で、以下となります。

```
PRIV_GMGR 〈ローカルユーザ名〉 〈リモートユーザリスト〉
: <group名> [<group名> ...]
```

ユーザマップファイル内に一致するエントリがなかった場合は、リモートユーザ名とローカルユーザ名は同一であるとみなし、デフォルトのアクセス権を設定します。

デフォルトのアクセス権は、DefaultPrivilege で指定したものです。

この指定がない場合のデフォルトのアクセス権の既定値は、PRIV_USR です。

ユーザマップファイルの例

先に示したサンプルを例に、以下では実際のユーザマッピングがどのように行われるか説明します。

最初の 2 行は、リモートユーザ名 root に関する定義です。

```
PRIV_SCH      root        root:127.0.0.1/32
PRIV_NON      root        root:0.0.0.0/0
```

・　RLマッピングの場合

・　1行目の指定により、バッチサーバホスト上からのrootでのアクセスは、ローカルユーザ名rootにマップされ、PRIV_SCH以下のアクセス権が許可されます(127.0.0.1/32

はループバックIPアドレス、127.0.0.1のみに一致します)。

- 2行目の指定により、全てのリモートホストからのrootでのアクセスは禁止されます（ビットマスクが0の場合、全てのIPアドレスと一致します)。
- ただし、優先順位の高い一行目の指定からローカルホストからのアクセスは禁止されません。
-

- LRマッピングの場合
  - 1行目の指定により、ローカルホスト上でのジョブ実行は、rootで行われます。
  - また、ローカルホストへの結果ファイル転送では、作成されるファイルの所有者がrootに設定されます。
  - ローカルホスト以外でのジョブ実行、結果ファイル転送は2行目の指定により禁止されます。

次の 2 行はリモートユーザ名 user00、user01 に関する定義です。

```
PRIV_MGR      nqsmgr      user00:192.168.77.0/24  user01:172.16.128.0/25
PRIV_OPE      nqsope      user00:192.168.10.0/24
```

- RLマッピングの場合
  - 1行目の指定から、IPアドレスが192.168.77.0/24に一致するリモートホスト上のユーザuser00と、172.16.128.0/25に一致するリモートホスト上のユーザuser01は、PRIV_MGR以下のアクセス権を持った ローカルユーザ名nqsmgrへマップされます。
  - ただし、2行目の指定により、user00がリモートホスト192.168.10.0/24からアクセスした場合には ローカルユーザ名nqsopeにマップされ、PRIV_OPE以下のアクセス権しか認められません。
  - リモートユーザリスト中に存在しないリモートホストからのuser00でのアクセスは、ローカルユーザ名user00、アクセス権PRIV_USRにマップされます。user01に関しても同様です。
- LRマッピングの場合
  - nqsmgrが所有者であるリクエストが実行される際、192.168.77.0/24にマッチする実行ホスト上に バッチジョブが生成される場合は、その実行者はuser00になりますが、172.16.128.0/25にマッチする実行ホストではuser01がバッチジョブの実行者になり

ます。

・　それ以外の実行ホストではバッチジョブの実行者はnqsmgrとなります。

2.9.3　ローカルアカウント機能

独自のユーザアカウントを定義したい場合は、ローカルアカウント機能を使用してください。

ローカルアカウント機能とは、NQSV 専用のアカウントデータベースをバッチサーバ上に構築することで、OS に登録されていないユーザに対しても NQSV の利用を可能にする機能です。

バッチサーバホスト上でのローカルユーザ名は、OS が提供するアカウントデータベース（/etc/passwd ファイルや NIS 等）に登録されている必要があります。

しかし、セキュリティ等の問題から一般ユーザ用のアカウントをバッチサーバホスト上に設定できない場合も考えられます。

このような場合には、ローカルアカウント機能を使用してください。

ローカルアカウントは、バッチサーバホスト上の/etc/opt/nec/nqsv/nqs_passwd.def および/etc/opt/nec/nqsv/nqs_group.def で定義します。

これらのファイル形式は、OS 標準の/etc/passwd、/etc/group ファイルと同じ形式のため、/etc/passwd、/etc/group や NIS のマップファイル等をそのまま流用できます。

/etc/opt/nec/nqsv/nqs_passwd.def および/etc/opt/nec/nqsv/nqs_group.def は、対象ユーザのエントリがシステム標準のアカウントデータベース内で見つからなかった場合にのみ参照します。

・　nqs_passwd.defファイル

　・　nqs_passwd.defは、ユーザ情報を定義するファイルです。各行をユーザごとのレコードと解釈します。レコードは":"で区切られた、少なくとも4つのカラムから構成されます（第5カラム以降は無視します）。 各カラムの意味は以下のとおりです。

カラム	説明
1	ユーザ名
2	(未使用)
3	ユーザID
4	グループID

・　nqs_group.defファイル

　・　nqs_group.defは、グループ情報を定義するファイルです。

　・　各行をグループごとのレコードと解釈します。レコードは":"で区切られた、少なくとも3つのカラムから構成されます（第4カラム以降は無視します）。各カラムの意味は

以下のとおりです。

カラム	説明
1	グループ名
2	(未使用)
3	グループID

・

nqs_passwd.def、nqs_group.def の変更は、自動的にバッチサーバへ反映します。

2.10 バッチサーバ間の連携

NQSV では、コミュニケーションサーバを介して、バッチサーバ間でリクエストの転送をすることができます。

2.10.1 サーバ環境設定

バッチサーバ間でリクエストの転送を行うためには、バッチサーバホスト上でのマシン ID の登録と、コミュニケーションサーバの起動、また、リクエストを投入するクライアントホスト上でのユーザエージェントの起動の際に、バッチサーバホストの指定が必要となります。

- マシンIDの登録
  - バッチサーバホスト上で、自バッチサーバのマシンIDに加えて、連携するバッチサーバのマシンIDを登録してください。マシンIDの登録方法については、2.5.マシンIDの管理 を参照してください。
  - なお、連携するバッチサーバには、バッチサーバ間で重複しない一意なマシンIDを割り当てておく必要があります。

- コミュニケーションサーバの起動
  - バッチサーバの起動時に同時にコミュニケーションサーバが起動されるよう設定します。連携する全バッチサーバホスト上で、root権限で下記のコマンドを実行します。

```
# systemctl enable nqs-comd.service
# systemctl restart nqs-bsv.target
```

・　ユーザエージェントの起動

　・　連携するバッチサーバへのリクエスト投入を行うクライアントホスト上で、ユーザエージェントの起動用設定ファイル（/etc/opt/nec/nqsv/nqs_uag.env）中のBSV_HOSTS変数に、連携する全バッチサーバホストのホスト名を指定して、ユーザエージェントを起動してください。

2.10.2 リクエスト転送設定

リクエストの転送を行うための転送キューを生成し、転送キューの転送先キューに転送先のバッチサーバホスト名とそのバッチサーバ上のキュー名を「キュー名@バッチサーバホスト名」の形式で設定してください。

転送キューの生成、および、転送先キューの設定方法については、4.3.1.転送キューの生成 および、4.3.2.転送キューの設定 を参照してください。

第3章　　実行ホストの管理

3.1　運用に必要なファイルとディレクトリ

実行ホストの運用に必要なファイル、ディレクトリを以下に示します。

/etc/opt/nec/nqsv/nqs_jsv.env	... ジョブサーバ起動用設定ファイル
/opt/nec/nqsv/sbin/nqs_shpd	... ジョブサーバ
/opt/nec/nqsv/sbin/nqs_lchd	... ランチャーデーモン
/var/opt/nec/nqsv/jsv/	... ジョブサーバデータ用ディレクトリ
/var/opt/nec/nqsv/jsv/<jsvno>/	... ジョブサーバデータベース
(<jsvno>はジョブサーバ番号)	
/var/opt/nec/nqsv/jsv/<jsvno>/nqs_shpd.pid	... ジョブサーバ PID ファイル
/var/opt/nec/nqsv/jsv/<jsvno>/<x.y.z>/	... ジョブデータベース(<x.y.z>はジョブ ID)
/var/opt/nec/nqsv/jsv/jobfile/<x.y.z>/	... ジョブファイル格納場所
/var/opt/nec/nqsv/jsv/jobfile/<x.y.z>/stdout	... 標準出力ファイル
/var/opt/nec/nqsv/jsv/jobfile/<x.y.z>/stderr	... 標準エラー出力ファイル
/var/opt/nec/nqsv/jsv/jobfile/<x.y.z>/stgfile/	... ステージングファイル格納場所

- ジョブサーバ
    - ジョブサーバ本体です。
- ランチャーデーモン
    - ジョブサーバのリモート制御を実現するデーモンです。
    - ランチャーデーモンが常駐している実行ホストでは、qmgr(1M)からジョブサーバの起動が可能です。
    - （起動方法は、3.5.1.qmgr(1M)からの起動 にて説明します。）
- ジョブサーバデータベース
    - ジョブサーバ用データベースです。 <jsvno>はジョブサーバ番号で4桁の数字（左0パディング）です。
- ジョブサーバPIDファイル
    - ジョブサーバのプロセスIDをASCIIで格納します。
- ジョブデータベース

    - ジョブサーバの制御下にあるジョブの情報を格納します。
    - 親リクエストがSTAGING状態に遷移した時点で作成され、EXITING状態で削除されます。
- ジョブファイル格納場所
    - ジョブからアクセスされるファイル群（ジョブファイル）を格納します。
    - 親リクエストがSTAGING状態に遷移した時点で作成され、EXITING状態で削除されます。
- 標準出力ファイル
    - ジョブの標準出力がリダイレクトされるファイルです。
- 標準エラー出力ファイル
    - ジョブの標準エラー出力がリダイレクトされるファイルです。
- ステージングファイル格納場所
    - ステージイン対象ファイル、ステージアウト対象ファイルを格納するディレクトリです。

3.2　実行ホストの組み込み

NQSV システムにおいてリクエストの実行に使用する実行ホストは、システムを構成する実行ホストとして、バッチサーバに登録します。

また、実行ホスト上では、投入されたリクエストに対するジョブを管理、実行するために、ジョブサーバを各実行ホストに対して１つずつ起動する必要があります。

このとき、バッチサーバがジョブサーバを一意に識別するために、各ジョブサーバに対してジョブサーバ番号を割り当てておく必要があります。

ジョブサーバ番号は、0～10239 の整数値で各ジョブサーバに一意となるように割り当てる必要があります。

そこで、バッチサーバへの実行ホストを登録するには、qmgr(1M)コマンドを管理者権限で起動して、実行ホストのホスト名と、そのホスト上で起動するジョブサーバのジョブサーバ番号を指定して、"attach execution_host"サブコマンドを実行します。

```
$ qmgr -Pm
Mgr: attach execution_host host = ehost001 job_server_id = 101
Attach Execution_Host (ehost001 : JSVID = 101).
Mgr:
```

実行ホストの登録は、NQSV システムで運用する全実行ホストに対して行う必要があります。

そこで、あらかじめ登録する実行ホストのホスト名とジョブサーバ番号の一覧ファイルを作成しておき、そのファイルを指定することで一括して実行ホストを登録することもできます。

実行ホストの一覧ファイルの書式は、以下のように、ホスト名とジョブサーバ番号を空白文字で区切って１行ずつ実行ホスト数分記述します。

```
ehost100   100
ehost101   101
ehost102   102
    :
```

そのうえで、この一覧ファイルのパス名を指定して、qmgr(1M)コマンドの"attach execution_host"サブコマンドを次のように実行してください。

```
$ qmgr -Pm
Mgr: attach execution_host file = /tmp/hlist_file
Attach Execution_Host (ehost100 : JSVID = 100).
Attach Execution_Host (ehost101 : JSVID = 101).
Attach Execution_Host (ehost102 : JSVID = 102).
     :
Attach Execution_Host (file = /tmp/hlist_file) .
```

実行ホストの登録は、qmgr(1M)コマンドの"attach execution_host"サブコマンドで行いますが、次節で説明するように、実行ホスト上で、ジョブサーバ番号を起動時の引数に指定して、ジョブサーバを直接起動すると、接続したバッチサーバ上で自動的に実行ホストが登録されます。

実行ホストの登録において、同一の実行ホストに対して異なるジョブサーバ番号で複数の登録を行ったり、同じジョブサーバ番号を複数の異なる実行ホストに対して割り当てたりすることはできません。

3.3　ノードグループ

NQSV では、複数の実行ホストをひとつのグループとして扱い、グループ単位で、次節以降で説明する、ジョブサーバの起動や、キューとのバインドを行うことができます。

この実行ホストのグループをノードグループと呼びます。

3.3.1　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行います。

以下に、"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 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 = ng1
Node_group ng1 created.
```

ノードグループは、最初は実行ホストを含まない状態で生成されます。

ノードグループには、複数の実行ホストをまとめて扱うための通常ノードグループ、ネットワークトポロジーを考慮したスケジューリングを行うためのネットワークトポロジーグループ、およびクラウドのコンピューティング資源へジョブをバースティングするためのクラウドバースティンググループの 3 つのタイプが存在します。

タイプ	説明
common	複数の実行ホストをグループとして扱うための通常のノードグループ
nw_topo	ネットワークトポロジーを考慮したスケジューリングを行うために、同一スイッチ配下の実行ホスト群を定義するためのネットワークトポロジーノードグループである。
cloud	クラウドのコンピューティング資源へジョブをバースティングするためのノードグループである。

上記のノードグループの作成例のように、タイプ指定なしでノードグループを作成した場合、common タイプのノードグループが生成されます。

以降、単に「ノードグループ」と記載した場合は、common タイプのノードグループを意味します。

また、複数の実行ホストの一括操作のためにグループ化する場合は、common タイプのノードグループを使用してください。

ネットワークトポロジーノードグループについては、ノードグループの作成時に、ノードグループのタイプとして、"nw_topo"を指定して作成します。

```
$ qmgr -Pm
Mgr: create node_group = nwtplg type = nw_topo comment = "NW topology"
Node_group nwtplg created.
```

ネットワークトポロジーの詳細は［JobManipulator 編］を参照して下さい。

クラウドバースティングノードグループについては、ノードグループの作成時に、ノードグループのタイプとして、"cloud"を指定して作成します。

```
$ qmgr -Pm
Mgr: create node_group = clng type = cloud comment = "Cloud bursting"
Node_group clng created.
```

クラウドバースティング機能の詳細は[JobManipulator 編]を参照して下さい。

3.3.2　ノードグループのコメントの設定

ノードグループの属性としてコメントを付加することができます。コメントは、作成時に設定することもできますが、qmgr の"set node_group comment"サブコマンドによって設定、変更することができます。

```
$ qmgr -Pm
Mgr: set node_group comment = "My nodegroup" ng1
Set comment to Node_group (ng1).
```

3.3.3　ノードグループへの実行ホストの追加

作成したノードグループへの実行ホストの追加には、qmgr(1M)コマンドの"edit node_group add"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。

以下に、先ほど作成したノードグループ"ng1"に、ジョブサーバ番号=100 のジョブサーバを追加する場合の例を示します。

```
$ qmgr -Pm
Mgr: edit node_group add job_server_id = 100 ng1
Add Job_Server to Node_group (ng1).
```

また、ノードグループへ実行ホストを追加する際の指定方法として、次のように複数のジョブサーバ番号を一度に指定することもできます。

```
$ qmgr -Pm
Mgr: edit node_group add job_server_id = ( 101,103,107 ) ng1
Add Job_Server to Node_group (ng1).
```

ジョブサーバ番号が連番になっている場合、次のようにジョブサーバ番号の範囲指定で実行ホストをノードグループに追加することもできます。

```
$ qmgr -Pm
Mgr: edit node_group add job_server_id = 110-120 ng1
```

```
Add Job_Server to Node_group (ng1).
```

さらに、既に実行ホストが登録されたノードグループがある場合、そのノードグループ名を指定することによって、指定したノードグループに属している実行ホストをまとめて、別のノードグループの実行ホストとして追加することができます。

```
$ qmgr -Pm
Mgr: edit node_group add node_group = ng1 ng2
Add node_group to Node_group (ng2).
```

上記の例では、既に存在するノードグループ ng1 に属している全実行ホストが、ノードグループ ng2 に追加されます。この場合、ng2 に追加される実行ホストは、サブコマンド実行時点で、指定したノードグループ ng1 に属している実行ホストで、追加操作実行後に、ng1 に属する実行ホストを変更しても、ng2 に含まれる実行ホストには影響しません。

3.3.4　ノードグループからの実行ホストの削除

ノードグループから実行ホストを削除するには、qmgr(1M)コマンドの"edit node_group delete"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。

以下に、ノードグループ"ng1"から、ジョブサーバ番号=100 のジョブサーバを削除する場合の例を示します。

```
$ qmgr -Pm
Mgr: edit node_group delete job_server_id = 100 ng1
Delete Job_Server from Node_group (ng1).
```

また、ノードグループから実行ホストを削除する際の指定方法として、追加の場合と同様に、ジョブサーバ番号の複数指定、範囲指定、および、別のノードグループを指定することにより、複数のジョブサーバ番号を一度に指定することもできます。

```
$ qmgr -Pm
Mgr: edit node_group delete job_server_id = ( 101,103,107 ) ng1
Delete Job_Server from Node_group (ng1).
$ qmgr -Pm
Mgr: edit node_group delete job_server_id = 110-120 ng1
Delete Job_Server from Node_group (ng1).
$ qmgr -Pm
Mgr: edit node_group delete node_group = ng1 ng2
```

```
Delete Job_Server from Node_group (ng1).
```

3.3.5　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 = ng1
Node_group ng1 deleted.
```

ノードグループとキューのバインドに関しては、3.6.ジョブサーバとキューのバインド を参照してください。

3.4　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。

```
$ qstat -G
NodeGroup       Type      BatchServer     Comment          JSVs BindQueue
--------------- --------- --------------- ---------------- ---- --------------------
ng1             common    bsv1.example.co (none)              1 bq1
nwtg            nw_topo   bsv1.example.co NW topology         0 (none)
cl_ng1          cloud     bsv1.example.co cloud               1 cloud_bq
```

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: ng1
    Type = common
    Comment = (none)
    Bind Queue list = {
      bq1
    }
    Job Server number list = {
        100
    }
```

```
Node Group: nwtplg_grp
    Type = nw_topo
    Comment = (none)
    Bind Queue list = {
      (none)
    }
    Job Server number list = {
      (none)
    }

Node Group: cl_ng1
    Type = cloud
    Comment = (none)
    Bind Queue list = {
      cloud_bq
    }
    Job Server number list = {
      1000
    }
    Lock State = UNLOCK
    Priority = 10
    Instances = Live: 0 / Max: 1
    Network Name = (none)
    Template = {
      (none)
    }
```

3.5　ジョブサーバの起動

3.5.1　qmgr(1M)からの起動

実行ホスト上で、NQSV のランチャーデーモンを起動しておくことにより、クライアントホストから、qmgr(1M)コマンドを使用してジョブサーバを起動することができます。

ランチャーデーモンは、下記のコマンドを root 権限で実行することで起動できます。

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

なお、ランチャーデーモンは、実行ホストに NQSV/JobServer パッケージをインストールすると、OS 起動時に自動的に起動するよう設定されています。

ランチャーデーモンを自動起動させたくない場合は、下記のコマンドを root 権限で実行して、nqs-jsv.target との関連付けを無効にしてください。

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

qmgr(1M)によるジョブサーバの起動には、"start job_server"サブコマンドを使用して、起動する実行ホストのホスト名とジョブサーバ番号を指定します。

```
$ qmgr -Pm
Mgr: start job_server execution_host=ehost100 job_server_id=100
Start Job_Server (ID = 100) on Execution_host (ehost100).
```

あらかじめ実行ホストの登録を行っている場合は、実行ホストのホスト名か、ジョブサーバ番号のみを指定してジョブサーバを起動することができます。

```
$ qmgr -Pm
Mgr: start job_server execution_host=ehost100
Start Job_Server on Execution_host (ehost100).
Mgr: start job_server job_server_id=101
Start Job_Server (ID = 101).
```

バッチサーバに実行ホストが登録されている場合、登録した実行ホストに対しては、登録したジョブサーバ番号以外の番号を指定してジョブサーバを起動することはできません。

また、登録されているジョブサーバ番号を他の実行ホストに対して指定して起動することもできません。

また、qmgr(1M)の"start job_server"サブコマンドを以下のように実行することによって、バッチサーバに登録済の全実行ホストでジョブサーバを起動することができます。

```
$ qmgr -Pm
Mgr: start job_server all
Start Job_Server all Execution_Host.
```

バッチサーバに未登録の実行ホストに対して qmgr(1M)でジョブサーバを起動する場合は、実行ホスト名と未登録のジョブサーバ番号を指定して"start job_server"サブコマンドを実行します。

この場合、ジョブサーバの起動によって、実行ホストが、指定したジョブサーバ番号で、自動的にバッチサーバに登録されます。

3.5.2　コマンドラインからの起動

各実行ホスト上でコマンドラインからジョブサーバを起動する場合は、root 権限で /opt/nec/nqsv/sbin/nqs_shpd を起動します。

このとき、引数で接続するバッチサーバホストと、ジョブサーバ番号の指定が必要です。

```
# /opt/nec/nqsv/sbin/nqs_shpd -h bsv0.example.com -n 0
```

nqs_shpd の起動時に指定可能なオプションは以下のとおりです。

オプション	省略	説明
-n *jsvno*	不可	ジョブサーバ番号として*jsvno*を割り当てます。
-h *bsv_host*	不可	*bsv_host*で指定したホストのバッチサーバへ接続します。
-p *bsv_port*	可	バッチサーバへの接続の際に、*bsv_port*に指定したポート番号を使用します。省略時は602を使用します。
-s *jsv_name*	可	ジョブサーバ名として*jsv_name*を使用します。 省略時のジョブサーバ名はJobServerXXXX(XXXXはジョブサーバ番号)です。

3.5.3　systemctl による起動

systemctl コマンドによってジョブサーバを起動するには、まず、/etc/opt/nec/nqsv/nqs_jsv.env 中の BSV_HOST_NAME 変数と JSV_NUMBER 変数にそれぞれバッチサーバホスト名とジョブサーバ番号を設定してください。下記は設定例です。

```
BSV_HOST_NAME="bsv1.example.com"
JSV_NUMBER="10"
```

修正後、下記のコマンドを root 権限で実行することでジョブサーバを起動できます。

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

実行ホスト立ち上げ時に自動的にジョブサーバを起動させるには、下記のコマンドを root 権限で実行して、nqs-jsv.target との関連付けを有効にしてください。

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

また、実行ホストをシャットダウンして再起動する際に、シャットダウン時に起動していたものと

同じジョブサーバ番号、接続先バッチサーバで自動的にジョブサーバを起動するには、/etc/opt/nec/nqsv/nqs_jsv.env を以下のように編集してください。

```
JSV_NUMBER=AUTO
```

この設定を行う場合、BSV_HOST_NAME 変数の指定は不要です。

本起動方法でジョブサーバを起動する際、ジョブサーバの追加オプション（-h および-n 以外のオプション）を指定したい場合は、/etc/opt/nec/nqsv/nqs_jsv.env の JSV_PARAM にオプションの文字列を設定してください。

以下は-s オプション(ジョブサーバ名)を指定する場合の例です。

```
JSV_PARAM="-s MY_JSV01"
```

【注意】

・省電力運用機能を利用している場合は、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場合があります。

3.6　ジョブサーバとキューのバインド

実行ホストでジョブを実行するためには、リクエストを投入するキューに対してジョブサーバを接続しておく必要があります。

ジョブサーバをキューに対して接続することを「バインド」といい、ジョブサーバとキューの接続関係を切断することを「アンバインド」といいます。

ジョブサーバをキューにバインドするには、qmgr(1M)を管理者権限で起動し、以下のサブコマンドを使用します。

バインドするキュー種別	サブコマンド
バッチキュー	bind execution_queue job_server
	bind execution_queue node_group
会話キュー	bind interactive_queue job_server
	bind interactive_queue node_group

ジョブサーバをキューにバインドする場合、直接ジョブサーバ番号を指定する方法と、ノードグループを指定する方法があります。ジョブサーバ番号を指定してキューへのバインドを行うには、"bind xxxx_queue job_server"サブコマンドを使用します。

```
$ qmgr -Pm
```

```
Mgr: bind execution_queue job_server bq1 job_server_id = 101
Bound Job_Server_ID (101) to Queue (bq1)
Mgr:
```

また、ノードグループを指定してジョブサーバをキューにバインドするには、" bind xxxx_queue node_group"サブコマンドを使用します。

```
$ qmgr -Pm
Mgr: bind execution_queue node_group bq1 node_group = ng1
Bound Node_group (ng1) to Queue (bq1).
Mgr:
```

同一ジョブサーバを複数のキューに同時にバインドすることが可能です。

また、ノードグループを指定した場合、指定したノードグループに属しているすべてのジョブサーバがキューにバインドされます。

ノードグループを指定してバインドを行った場合は、ノードグループとしてもキューとのバインド関係が生成されます。そのため、キューにバインドしたノードグループに対して実行ホストの追加を行った場合、追加された実行ホストのジョブサーバは、ノードグループがバインドされているキューに自動的にバインドされます。また、キューにバインドしたノードグループに対して実行ホストの削除を行った場合、削除された実行ホストのジョブサーバは、ノードグループがバインドされているキューから自動的にアンバインドされます。

ジョブサーバとキューとの接続（バインド）関係を切断する（アンバインド）には、qmgr(1M)を管理者権限で起動し、以下のサブコマンドを使用します。

バインドするキュー種別	サブコマンド
バッチキュー	unbind execution_queue job_server
	unbind execution_queue node_group
会話キュー	unbind interactive_queue job_server
	unbind interactive_queue node_group

キューにバインドしたジョブサーバは、ジョブサーバがダウンしても、キューとのバインド関係は保持されます。ただし、ジョブサーバがダウンした際に、バインドしているキューから自動的にアンバインドする場合は、対象のキューに対して、qmgr(1M)の"set execution_queue auto_bind_jobserver off"サブコマンドで、自動バインドを OFF に設定してください。

ノードグループを指定してアンバインドを行った場合、指定したノードグループに属する全実行ホストのジョブサーバがキューからアンバインドされます。

また、ノードグループを指定してキューにバインドしたジョブサーバを、"unbind execution_queue

job_server"サブコマンドにより、個別にキューからアンバインドすることも可能です。この場合、ノードグループとしてのキューとのバインド関係は保持されます。

3.7　ジョブサーバの状態確認

ジョブサーバの状態は、qstat(1) コマンドの -S オプションで確認してください。

```
$ qstat -S
JSVNO JobServerName   BatchServer     ExecutionHost   LINK BIND Queue    Jobs  Load   Cpu
----- --------------- --------------- --------------- ---- ---- -------- ---- ----- -----
    0 JobServer0100   bsv0.example.co ehost100        UP   Y    execque1    0   0.6   0.0
    1 JobServer0101   bsv0.example.co ehost101        UP   Y    execque1    0   0.6   0.0
    2 JobServer0102   bsv0.example.co ehost102        UP   Y    execque1    0   0.6   0.0
```

qstat -S オプションでは、リンクアップしているジョブサーバのみが表示されます。

リンクダウンしているジョブサーバを含む、バッチサーバに登録されている全実行ホストのジョブサーバを表示するには、qstat -S に-t オプションをつけて実行してください。

```
$ qstat -St
JSVNO JobServerName   BatchServer     ExecutionHost   LINK BIND Queue    Jobs  Load   Cpu
----- --------------- --------------- --------------- ---- ---- -------- ---- ----- -----
    0 JobServer0100   bsv0.example.co ehost100        UP      1 execque1    0   0.6   0.0
    1 JobServer0101   bsv0.example.co ehost101        UP      1 execque1    0   0.6   0.0
    2 JobServer0102   bsv0.example.co ehost102        UP      1 execque1    0   0.6   0.0
    3  -              bsv0.example.co ehost103        DOWN    0  -          -     -     -
```

この場合、BIND の項目は、バインドされているキューの数が表示されます。

また、qstat -S に -f オプションつけることにより、各ジョブサーバの詳細情報を表示することができます。

```
$ qstat -S -f
Job Server Name: JobServer0100
    Job Server Number  = 0
    Job Server Version = R1.00 (linux)
    Batch Server   = bsv0.example.com
    Execution Host = ehost100
    LINK Batch Server = UP
    BIND Queue = BIND
    Queue = {
```

```
        execque1
    }
    Assign JobManipulator license = YES
    Network Topology Group = {
        (none)
    }
  Resource Information:
    Memory          = Assign:    1035264 Using:     925696 Maximum:    1035264
    Swap            = Assign:    1014784 Using:       3072 Maximum:    1014784
    Number of Cpus = Assign:           8 Using:          1 Maximum:          8
  Average Information:
    LOAD (Latest  1 minute ): 0.010000
    LOAD (Latest  5 minutes): 0.040000
    LOAD (Latest 15 minutes): 0.050000
    CPU  (Latest  1 minute ): 0.016000
    CPU  (Latest  5 minutes): 0.016000
    CPU  (Latest  5 minutes): 0.016000
    CPU  (Latest 15 minutes): 0.864000
  Migration_file Transfer Parameter Information:
    Interface Hostname = ehost100.example.com
    Socketbuffer Size  = (OS default)
    I/Obuffer Size     = 512.0KB

Job Server Name: JobServer0101
    Job Server Number  = 1
    Job Server Version = R1.00 (linux)
          :
```

qstat -S オプションによるジョブサーバ情報の表示においても、-F オプションで 表示項目（アイテム）の選択表示が可能です。

```
$ qstat -S -F jsvno,ehost,quenm
JSVNO ExecutionHost   Queue
----- --------------- --------
    0 ehost100        execque1
    1 ehost101        execque1
    2 ehost102        execque1
```

qstat -S で-F オプションに指定可能なアイテムは以下のとおりです。

-F指定アイテム	内容	サマリ表示形式
jsvno	ジョブサーバ番号	JSVNO
jsvnm	ジョブサーバ名	JobServerName
bsvhost	バッチサーバホスト名	BatchServer
ehost	ジョブサーバ実行ホスト	ExecutionHost
link	リンク状態	LINK
bind	バインド状態	BIND
quenm	キュー名	Queue
jobs	バッチジョブ数	Jobs
ldavg1	過去1分間のロードアベレージ	Load
cpuavg1	過去1分間のCPUアベレージ	CPU

（qstat(1)コマンド-F オプションの使用方法詳細については、[操作編]出力情報のカスタマイズ を参照してください。）

また、qstat -S においても、-O, -o オプションにより、上記アイテムをキーとしてソートすることができます。（qstat(1)コマンド-O, -o オプションの使用方法詳細については、［操作編］出力情報のソート を参照してください。）

3.8　ジョブサーバの停止

ジョブサーバは、実行ホストのシャットダウン時に systemd によって終了させられます。

実行ホスト運用中に終了させたい場合は、ジョブサーバへ SIGTERM シグナルを送信してください。

また、ジョブサーバを systemd（systemctl コマンド）で起動した場合、下記のコマンドを root 権限で実行することで停止させることもできます。

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

ジョブサーバは qmgr(1M)の stop job_server サブコマンドを使って終了させることもできます。

例えば、ジョブサーバ ID が 100 番のジョブサーバを終了させる場合は、以下のコマンドを実行します。

```
$ qmgr -Pm
Mgr: stop job_server job_server_id = 100
```

【注意】実行中のジョブを持つジョブサーバが終了した場合、実行中のジョブは強制終了します。
リクエストのホールド等を行い実行中のジョブが存在しない状態で、ジョブサーバを終

了させてください。

3.9　ジョブサーバ単位のリクエスト一斉ホールド

ジョブサーバ内で実行しているすべてのリクエストをホールドするには、 qmgr(1M)の hold job_server サブコマンドを 実行してください。

このサブコマンドを実行すると、ジョブサーバ内で実行中のジョブの親リクエストを 一斉にホールドします。 ジョブサーバを停止するときなど、実行ホスト上に実行中のジョブが存在しない状態にする際に、本機能を利用してください。

3.10 実行ホストの情報表示

実行ホストの状態は、qstat(1) コマンドの -E オプションで確認してください。

```
$ qstat -E
ExecutionHost   BatchServer     OS         Release    Hardware     VE  Load   Cpu
--------------- --------------- ---------- ---------- ---------- ---- ----- -----
ehost100        bsv0.example.co Linux      3.10.0-327 x86_64        8   0.0   0.0
ehost101        bsv0.example.co Linux      3.10.0-327 x86_64        8   0.0   0.0
ehost102        bsv0.example.co Linux      3.10.0-327 x86_64        0   0.0   0.0
```

qstat -E オプションでは、ジョブサーバがリンクアップしている実行ホストのみが表示されます。

ジョブサーバがリンクダウンしている実行ホストを含む、バッチサーバに登録されている全実行ホストを表示するには、qstat -E に-t オプションをつけて実行してください。

また、-t オプション指定時には、実行ホストの状態（STT）も表示されます。

```
$ qstat -E -t
ExecutionHost   JSVNO JSV      S OS         Release    Hardware     VE  Load   Cpu STT
--------------- ----- -------- - ---------- ---------- ---------- ---- ----- ----- ---
ehost100          100 LINKUP   - Linux      3.10.0-514 x86_64        8   0.0   0.0 ACT
ehost101          101 LINKUP   - Linux      3.10.0-514 x86_64        8   0.0   0.0 ACT
ehost102          102 LINKUP   - Linux      3.10.0-514 x86_64        0   0.0   0.0 ACT
ehost103          103 LINKDOWN - --         --         --            -     -     - INA
```

qstat -E オプションによる実行ホスト情報の表示においても、-F オプションで 表示項目（アイテム）の選択表示が可能です。

```
$ qstat -E -F ehost,hwnm,ldavg1
ExecutionHost   Hardware    Load
--------------- ---------- -----
ehost100        x86_64       0.0
ehost101        x86_64       0.0
ehost102        x86_64       0.0
```

qstat -E で-F オプションに指定可能なアイテムは以下のとおりです。

-F指定アイテム	内容	サマリ表示形式
ehost	実行ホスト	ExecutionHost
bsvhost	バッチサーバホスト名	BatchServer
osnm	オペレーティングシステム名	OS
rel	OSリリース番号	Release
hwnm	ハードウェア名	Hardware
ldavg1	過去1分間のロードアベレージ	Load
cpuavg1	過去1分間のCPUアベレージ	CPU
ucpu	使用CPU数	Used_CPU
fcpu	未使用CPU数	Free_CPU
umem1	使用メモリサイズ	Used_MEM1
fmem1	未使用メモリサイズ	Free_MEM1
uswap1	使用スワップサイズ	Used_SWAP1
fswap1	未使用スワップサイズ	Free_SWAP1
quenm	キュー名	Queue
stt	実行ホストの状態	STT

（qstat(1)コマンド-F オプションの使用方法詳細については、[操作編]出力情報のカスタマイズ を参照してください。）

また、qstat -E においても、-O, -o オプションにより、上記アイテムをキーとしてソートすることができます。（qstat(1)コマンド-O, -o オプションの使用方法詳細については、[操作編] 出力情報のソート を参照してください。）

実行ホストの詳細情報を表示するには、qstat -E に -f オプションを指定します。

```
$ qstat -E -f
Execution Host: ehost100
    Batch Server = bsv0.example.com
    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
```

```
  Vector Engine Information:
    (none)
  Resource Information:
    Memory          = Assign:    1035264 Using:     925696 Maximum:    1035264
    Swap            = Assign:    1014784 Using:       3072 Maximum:    1014784
    Number of Cpus = Assign:           8 Using:          1 Maximum:          8
  Average Information:
    LOAD (Latest  1 minute ): 0.020000
    LOAD (Latest  5 minutes): 0.050000
    LOAD (Latest 15 minutes): 0.050000
    CPU  (Latest  1 minute ): 0.016000
    CPU  (Latest  5 minutes): 0.928000
    CPU  (Latest 15 minutes): 0.928000
  Cpuset Information:
    Resource Sharing Groups = {
      (none)
    }
  Socket Resource Usage:
    NUMA Nodes = {
      (none)
    }
  Device Topology:
    (none)

Execution Host: ehost101
    Batch Server = bsv0.example.com
       :
```

qstat -Ef オプションでは、ジョブサーバがリンクアップしている実行ホストのみが表示されます。

ジョブサーバがリンクダウンしている実行ホストを含む、バッチサーバに登録されている全実行ホストを表示するには、qstat -Ef に-t オプションをつけて実行してください。

なお､ジョブサーバがリンクダウンしている実行ホストについては一部の情報しか表示されません。

また、-Ef に重ねて-t オプションを指定することにより、リンクアップしている実行ホストについては、以下の情報も加えて表示されます。

・　実行ホストの現在の状態（Current State）

Active　　稼働状態

Inactive　　停止状態

- 現在の状態に遷移した時刻（State Transition Time）
- 現在の状態に遷移した理由（State Transition Reason）
- ジョブサーバ番号（Job Server Number）
- ジョブサーバのリンク状態（LINK Batch Server）

  UP　　ジョブサーバがバッチサーバとリンクされている状態

  DOWN　　ジョブサーバがバッチサーバとリンクされていない状態
- ノード管理エージェント情報（Node Agent）

```
$ qstat -Etf
Execution Host: ehost100
    Batch Server = bsv0.example.com
    Current State           = Active
    State Transition Time   = Mon Jul  1 16:00:26 2024
    State Transition Reason = JSV LINKUP
    Job Server Number  = 0
    LINK Batch Server = UP
    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
    Node Agent = (none)
    Substitute Status = Normal
  Vector Engine Information:
    (none)
  Resource Information:
    Memory         = Assign:    1035264 Using:     925696 Maximum:    1035264
    Swap           = Assign:    1014784 Using:       3072 Maximum:    1014784
    Number of Cpus = Assign:          8 Using:          1 Maximum:          8
  Average Information:
    LOAD (Latest  1 minute ): 0.020000
    LOAD (Latest  5 minutes): 0.050000
    LOAD (Latest 15 minutes): 0.050000
    CPU  (Latest  1 minute ): 0.016000
    CPU  (Latest  5 minutes): 0.928000
    CPU  (Latest 15 minutes): 0.928000
  Cpuset Information:
    Resource Sharing Groups = {
```

```
      (none)
    }
  Socket Resource Usage:
    NUMA Nodes = {
      (none)
    }
  Device Topology:
    (none)

Execution Host: ehost101
    Batch Server = bsv0.example.com
       :
       :

Execution Host: ehost103
    Batch Server = bsv0.example.com
    Current State           = Inactive
    State Transition Time   = Mon Mar 3 15:00:00 2017
    State Transition Reason = JSV LINKDOWN
    Job Server Number  = 103
    LINK Batch Server = DOWN
```

3.11 VEノードの情報表示

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。

VE ノードの状態は、qstat(1) コマンドの --venode オプションで確認してください。

本オプションは引数に VI のホスト名を指定することで、指定された VI（VH）に搭載されている VE の情報のみを表示することができます。

引数が指定されていない場合、JSV が LINKUP している全ての VI（VH）についての VE の情報を表示します。

```
$ qstat --venode
VectorIsland VE_No  Cores Memory Status        OS_Status
------------ ----- ------ ------ ------------ ----------
ehost100         0     10   48GB ONLINE       ONLINE
ehost100         1     10   48GB UNINITIALIZED OFFLINE
ehost100         2     10   48GB OFFLINE      OFFLINE
```

```
ehost100        3     10    48GB MAINTENANCE   OFFLINE
ehost101        0     10    48GB UNAVAILABLE   OFFLINE
ehost101        1     10    48GB ONLINE        OFFLINE
ehost101        2     10    48GB ONLINE        ONLINE
ehost101        3     10    48GB ONLINE        ONLINE
```

第4章　　キューの管理

4.1　バッチキュー

4.1.1　バッチキューの生成

バッチキューとは、バッチリクエストの実行を制御するキューです。

バッチキューは、qmgr(1M)の ″create execution_queue″サブコマンドで作成します。

```
$ qmgr -Pm
Mgr: create execution_queue = exec1 priority = 20
```

以上の手続でバッチキュー exec1 が作成されます。 このとき指定する priority はキュープライオリティです。

4.1.2　バッチキューの設定

(1)　資源制限値

バッチキューに投入されるリクエストに対して、キューごとにジョブが使用する計算資源使用量の制限値を設定することができます。

リクエスト投入時に指定した資源使用量がキューに設定した資源制限値より大きい場合は、キューへの投入を拒否します。

この機能を使って、資源を大量に使用することを許すキュー、または少ししか許さないキューというようにキューのクラス分けができます。

資源制限値は qmgr(1M)の″set execution_queue″サブコマンドで設定してください。

キューに対して資源制限値の設定が可能な計算資源と、設定のための qmgr のサブコマンドは、以下のとおりです。本設定には操作員権限が必要です。

計算資源	qmgrサブコマンド
プロセス単位	
CPU時間制限値	set execution_queue per_prc cpu_time_limit
オープンファイル数制限値（*1）	set execution_queue per_prc open_file_number_limit
データサイズ制限値	set execution_queue per_prc data_size_limit
スタックサイズ制限値	set execution_queue per_prc stack_size_limit
コアファイルサイズ制限値	set execution_queue per_prc core_size_limit
ファイルサイズ制限値	set execution_queue per_prc file_size_limit
VE CPU時間制限値	set execution_queue per_prc vecpu_time_limit
VE メモリサイズ制限値	set execution_queue per_prc vememory_size_limit
仮想メモリサイズ制限値	set execution_queue per_prc virtual_memory_size_limit

ジョブ単位	
CPU時間制限値	set execution_queue per_job cpu_time_limit
同時実行CPU台数制限値	set execution_queue per_job cpu_number_limit
メモリサイズ制限値	set execution_queue per_job memory_size_limit
同時実行GPU台数制限値	set execution_queue per_job gpu_number_limit
仮想メモリサイズ制限値	set execution_queue per_job virtual_memory_size_limit
リクエスト単位	
経過時間制限値	set execution_queue per_req elapse_time_limit

(*1) ･･･unlimited に設定した場合、Linux の仕様上の制約から実行ホスト上では 1024 が上限値として適用されます。

なお、経過時間制限値については、グループ名・ユーザ名を個別指定してのリクエスト数制限も設定できます。詳細は、12.グループ毎・ユーザ毎の制限 を参照してください。

以下に、バッチキューexec1 のプロセスごとのファイルサイズ制限を変更する場合の例を示します。この操作により、キューexec1 のプロセスごとのファイルサイズ制限が 100.5 キロバイトになります。

```
$ qmgr -Po
Mgr(bsv1.example.com): set execution_queue per_prc file_size_limit = (100.5kb) exec1
Set Permanent File Size (Per-Process) limit: exec1
```

キューに設定した資源制限値は、qstat(1)コマンドの-Q、-f オプション（Resources Limits）で確認することができます。 （4.5.1.バッチキュー 参照）

メモリサイズ資源制限は cgroup を使った制御も可能です。（2.3.19 memory cgroup によるメモリ管理参照）

VE ノード毎の資源制限値

バッチキューに投入されるリクエストに対して、VE ノード毎に資源使用量の制限値を設定することが可能です。VE ノード毎の資源制限値は、資源の上限と下限を「範囲」で制限できます。リクエスト投入時に指定した資源使用量がキューに設定した資源制限値の範囲外の場合は、キューへの投入を拒否します。

資源制限値は qmgr(1M)の"set execution_queue venode"サブコマンドで設定してください。キューに対して資源制限値の設定が可能な計算資源と、設定のための qmgr のサブコマンド以下のとおりです。本設定には操作員権限が必要です。

計算資源	**qmgrサブコマンド**
VE CPU時間制限値	set execution_queue venode vecpu_time_range = (<min>,<max> [,<warn>]) <queue-name>
VE メモリサイズ制限値	set execution_queue venode vememory_size_range = (<min>,<max> [,<warn>]) <queue-name>

<min>は最小値、<max>は最大値であり、それぞれ整数値を指定します。

<min>は<max>以下の値を指定してください、<min>と<max>が同じ値の時は、リクエスト投入時に、その値しか指定できなくなります。

キューに設定した論理ホスト毎の資源制限値は、qstat(1)コマンドの-Qf オプション（VE Node Resource Ranges）で確認することができます。（4.5.1.バッチキュー 参照）

論理ホスト毎の資源制限値

バッチキューに投入されるリクエストに対して、ジョブに割り当てる論理ホスト毎に資源使用量の制限値を設定することが可能です。

論理ホスト毎の資源制限値は、従来のジョブ毎の資源制限値と異なり、資源の上限だけでなく下限も指定でき、「範囲」で制限できます。

リクエスト投入時に指定した資源使用量がキューに設定した資源制限値の範囲外の場合は、キューへの投入を拒否します。

下限が設定できる点を除けば、論理ホスト毎の資源制限値に関する動作は従来のジョブ毎の資源制限値と同じです。

資源制限値は qmgr(1M)の"set execution_queue lhost"サブコマンドで設定してください。

キューに対して資源制限値の設定が可能な計算資源と、設定のための qmgr のサブコマンド以下のとおりです。本設定には操作員権限が必要です。

計算資源	qmgrサブコマンド
VEノード数制限値	set execution_queue lhost ve_number_range = (<min>,<max>) <queue-name>
同時実行CPU台数制限値	set execution_queue lhost cpu_number_range = (<min>,<max>) <queue-name>
同時実行GPU台数制限値	set execution_queue lhost gpu_number_range = (<min>,<max>) <queue-name>
CPU時間制限値	set execution_queue lhost cpu_time_range = (<min>,<max> [,<warn>]) <queue-name>
メモリサイズ制限値	set execution_queue lhost memory_size_range = (<min>,<max>[,<warn>]) <queue-name>
VE CPU時間制限値	set execution_queue lhost vecpu_time_range = (<min>,<max>[,<warn>]) <queue-name>
VE メモリサイズ制限値	set execution_queue lhost vememory_size_range = (<min>,<max> [,<warn>]) <queue-name>
仮想メモリサイズ制限値	set execution_queue lhost virtual_memory_size_range = (<min>,<max> [,<warn>]) <queue-name>
標準出力の結果ファイルサイズ制限値	set execution_queue lhost stdout_size_range = (<min>,<max> [,<warn>]) <queue-name>
標準エラー出力の結果ファイルサイズ制限値	set execution_queue lhost stderr_size_range = (<min>,<max> [,<warn>]) <queue-name>

<min>は最小値、<max>は最大値であり、それぞれ整数値を指定します。

<min>は<max>以下の値を指定してください、<min>と<max>が同じ値の時は、リクエスト投入時に、その値しか指定できなくなります。

キューに設定した論理ホスト毎の資源制限値は、qstat(1)コマンドの-Qf オプション（Logical Host Resource Ranges）で確認することができます。（4.5.1.バッチキュー 参照）

なお、これらの論理ホスト毎の資源制限値（VE ノード数制限値を除く）は、以下に示す、ジョブ毎の資源制限値の設定方法（per_job 指定）でも代替可能です。

```
set execution_queue per_job cpu_number_limit = <max> <queue-name>
set execution_queue per_job gpu_number_limit = <max> <queue-name>
set execution_queue per_job cpu_time_limit = (<max>,[<warn>]) <queue-name>
set execution_queue per_job memory_size_limit = (<max>,[<warn>]) <queue-name>
set execution_queue per_job virtual_memory_size_limit = (<max>,[<warn>]) <queue-name>
```

この設定方法で設定した場合、論理ホスト毎の資源制限値の<min>の値は cpu_number_limit については 1 を、それ以外は 0 を指定したものと見なし、指定された<max>および<warn>はそのまま適用されます。

ただし、lhost による指定方法でも、per_job による指定方法でも、後から設定したものが有効となります。

HCA のポート数の制限値

バッチキューに投入されるリクエストに対して、キューごとにジョブが使用する HCA のポート数の制限値を設定することができます。

リクエスト投入時に指定した HCA のポート数がキューに設定した HCA のポート数の制限値の範囲外である場合は、キューへの投入を拒否します。

バッチキューに対する HCA のポート数の制限値は qmgr(1M)の

"set execution_queue hca_number_range"サブコマンドにより設定を行います。

本設定には操作員権限が必要です。

計算資源	qmgr サブコマンド
HCA のポート数の制限値	set execution_queue hca_number_range = (<min>,<max>) mode=<hca-mode> <queue-name>

<min>は最小値、<max>は最大値であり、それぞれ整数値を指定します。

<min>は<max>以下の値を指定してください。

<min>と<max>が同じ値の場合、リクエスト投入時にその値のみ指定可能となります。それ以外の値は指定できなくなります。

(2)　資源既定値

資源既定値とは、資源制限値を指定していないでキューにリクエストを投入した場合に、リクエストの資源制限値として自動的に設定される資源制限値の既定値で、キューに対する属性値として設定しておくことができます。リクエストに資源制限値が設定されている場合は、この属性は参照されません。

バッチキューに対する資源既定値は、qmgr(1M)の以下のサブコマンドにより設定を行います。本設定には操作員権限が必要です。

計算資源	qmgrサブコマンド
プロセス単位	
CPU時間	set execution_queue standard per_prc cpu_time_limit

オープンファイル数	set execution_queue standard per_prc open_file_number_limit
データサイズ	set execution_queue standard per_prc data_size_limit
スタックサイズ	set execution_queue standard per_prc stack_size_limit
コアファイルサイズ	set execution_queue standard per_prc core_size_limit
ファイルサイズ	set execution_queue standard per_prc file_size_limit
VE CPU時間	set execution_queue standard per_prc vecpu_time_limit
VE メモリサイズ	set execution_queue standard per_prc vememory_size_limit
仮想メモリサイズ	set execution_queue standard per_prc virtual_memory_size_limit
ジョブ単位	
CPU時間	set execution_queue standard per_job cpu_time_limit
同時実行CPU台数	set execution_queue standard per_job cpu_number_limit
メモリサイズ	set execution_queue standard per_job memory_size_limit
仮想メモリサイズ	set execution_queue standard per_job virtual_memory_size_limit
同時実行GPU台数	set execution_queue standard per_job gpu_number_limit
リクエスト単位	
経過時間	set execution_queue standard per_req elapse_time_limit

以下に、プロセスごとのファイルサイズ制限の既定値を設定する場合の例を示します。

この操作により、バッチキューexec1 のプロセスごとのファイルサイズ制限の既定値が 100.5 キロバイトになります。

```
$ qmgr -Po
Mgr: set execution_queue standard per_prc file_size_limit = ( 100.5kb ) exec1
Set standard Permanent File Size (Per-Process): glbque1
```

VE ノード毎の資源既定値

VE ノード毎に資源使用量の既定値を、キューに対する属性値として設定することが可能です。

バッチキューに対する VE ノード毎の資源既定値の設定は、qmgr(1M)の以下のサブコマンドにより行います。本設定には操作員権限が必要です。

計算資源	qmgrサブコマンド
VE CPU時間	set execution_queue standard venode vecpu_time_limit = (<time>) <queue-name>
VE メモリサイズ	set execution_queue standard venode vememory_size_limit = (<size>) <queue-name>

論理ホスト毎の資源既定値

また、上記とは異なる指定方法で、ジョブに割り当てる論理ホスト毎の資源既定値を、キューに対する属性値として設定することができます。リクエスト投入時の論理ホスト毎の資源既定値に関する動作は、ジョブ毎の資源既定値と同じです。

バッチキューに対する論理ホスト毎の資源既定値の設定は、qmgr(1M)の以下のサブコマンドにより行います。本設定には操作員権限が必要です。

計算資源	qmgrサブコマンド
VEノード数	set execution_queue standard lhost ve_number_limit = <num> <queue-name>
同時実行CPU台数	set execution_queue standard lhost cpu_number_limit = <num> <queue-name>
同時実行GPU台数	set execution_queue standard lhost gpu_number_limit = <num> <queue-name>
CPU時間	set execution_queue standard lhost cpu_time_limit = (<time>) <queue-name>
メモリサイズ	set execution_queue standard lhost memory_size_limit = (<size>) <queue-name>
VE CPU時間	set execution_queue standard lhost vecpu_time_limit = (<time>) <queue-name>
VE メモリサイズ	set execution_queue standard lhost vememory_size_limit = (<size>) <queue-name>
仮想メモリサイズ	set execution_queue standard lhost virtual_memory_size_limit = (<size>) <queue-name>
標準出力の結果ファイルサイズ	set execution_queue standard lhost stdout_size_limit = (<size>) <queue-name>
標準エラー出力の結果ファイルサイズ	set execution_queue standard lhost stderr_size_limit = (<size>) <queue-name>

資源既定値の設定値は、資源制限値の下限(min)以上、上限(max)以下の値でなければいけません。

また、単位を付加する項目(cpu_time, memory_size, virtual_memory_size, vecpu_time, vememory_size, stdout_size, stderr_size)は指定値を()で囲むこと必要があります。

なお、これらの論理ホスト毎の資源既定値（VE ノード数制限値を除く）は、以下に示す、

従来のジョブ毎の資源制限値の設定方法（per_job 指定）でも代替可能です。

```
set execution_queue standard per_job cpu_number_limit = <num> <queue-name>
set execution_queue standard per_job gpu_number_limit = <num> <queue-name>
set execution_queue standard per_job cpu_time_limit = (<time>) <queue-name>
set execution_queue standard per_job memory_size_limit = (<size>) <queue-name>
set execution_queue standard per_job virtual_memory_size_limit = (<size>) <queue-name>
```

HCA のポート数の制限値

HCA のポート数の制限値を指定してしないでキューにリクエストを投入した場合に、リクエストの HCA のポート数の制限値として自動的に設定される制限値で、キューに対する属性値として設定しておくことができます。

リクエストに HCA のポート数の制限値が設定されている場合は、この属性は参照されません。

バッチキューに対する HCA のポート数の制限値は、qmgr(1M)の"set execution_queue standard hca_number"サブコマンドにより設定を行います。本設定には操作員権限が必要です。

計算資源	qmgr のサブコマンド
HCA のポート数の制限値	set execution_queue standard hca_number = <num> mode=<hca-mode> <queue-name>

既定値の設定値は、制限値の下限(min)以上、上限(max)以下の値でなければいけません。

(3)　キュープライオリティ

キュープライオリティとは、キュー間の優先度で、これにより他のキューとの間でスケジューリングの優先順位をつけることができます。

バッチキューのキュープライオリティは、バッチキューの作成時に指定しますが、qmgr の"set execution_queue priority"サブコマンドによって変更することができます。

```
$ qmgr -Po
Mgr: set execution_queue priority = 10 exec1
Set Priority: exec1
```

上記の手続きでバッチキュー exec1 のキュープライオリティを 10 に変更します。

キュープライオリティの変更には操作員権限が必要です。

(4)　カーネルパラメータ

バッチキューに設定するカーネルパラメータは、バッチジョブに受け継がれ、実行ホスト上でカーネルがジョブをスケジューリングするときのパラメータとなります。

バッチキューに対するカーネルパラメータは qmgr(1M)の"set execution_queue"サブコマンドで設定します。本設定には操作員権限が必要です。NQSV でサポートしているカーネルパラメータと、対応する qmgr のサブコマンド、キュー作成時の既定値は、以下のとおりです。

カーネルパラメータ	qmgrサブコマンド	既定値
ナイス値	set execution_queue kernel_param nice	0
リソースシェアリンググループ番号(*1)	set execution_queue kernel_param rsg_number	0

(*1) ･･･ソケットスケジューリング機能ならびに CPUSET 機能利用時のみ有効です。

以下に、バッチキューexec1 のナイス値を 5 に設定する場合の例を示します。

```
$ qmgr -Po
Mgr: set execution_queue kernel_param nice = 5 exec1
Set Nice Value (Kernel-Parameter): exec1
```

(5)　投入リクエスト数制限

バッチキューに対して、投入できるリクエスト数の上限をキューごとに設定できます。投入できるリクエスト数とは、1 つのキュー内に同時に存在しているリクエストの総数です。

バッチサーバ単位の投入数制限よりさらに詳細なリクエスト投入数制限をかけることが可能となります（バッチサーバ単位の投入数制限の詳細に関しては、2.3.3.投入リクエスト数制限 を参照してください)。

投入リクエスト数制限は、バッチキューごとに、キュー全体、グループ別、ユーザ別に設定できます。 各設定は、qmgr(1M)の"set execution_queue"サブコマンドで行ってください。

投入リクエスト数制限を設定するには、操作員以上の権限が必要です。

バッチキューの投入リクエスト数制限を設定する qmgr(1M)のサブコマンドと、指定されていない場合の既定値は、次のとおりです。

属性	qmgr(1M)サブコマンド	既定値
キューに投入できるリクエスト数	set execution_queue submit_limit	0（無制限）
1つのグループがキューに投入できるリクエスト数	set execution_queue group_submit_limit	0（無制限）
1人のユーザがキューに投入できるリクエスト数	set execution_queue user_submit_limit	0（無制限）

なお、グループ名・ユーザ名を個別指定してのリクエスト数制限も設定できます。詳細は、12.グループ毎・ユーザ毎の制限 を参照してください。

設定されている投入リクエスト数制限値は qstat コマンドの-Q、-f オプション(キューの詳細表示)で下記の内容にて確認することができます。(4.5.1.バッチキュー 参照。)

- リクエストのキュー単位の投入数制限値: Submit Number Limit
- リクエストのキュー単位のユーザ別投入数制限値: Submit User Number Limit
- リクエストのキュー単位のグループ別投入数制限値: Submit Group Number Limit
    - （各項目に対して、制限値が設定されている場合はその値が、設定されていない場合はUNLIMITEDが表示されます。）

バッチサーバ単位の投入数制限とキュー単位の投入数制限が同時に設定されている場合、リクエス

ト数がいずれかの制限値に達すると、それ以上リクエストの投入はできません。

(6)　ホールド要求権限

ホールド要求権限とは、リクエストに対してホールド要求ができる権限のことです。（リクエストのホールドについては、[操作編] バッチリクエストのホールド 参照。）

バッチキューにホールド要求権限を設定しておくことにより、そのキューに投入されたリクエストに対して、設定された権限以上の権限を持つユーザからのホールド要求のみが受け付けられるようになります。 バッチキューに対するホールド要求権限は、qmgr(1M)の"set execution_queue hold_privilege"サブコマンドで設定してください。

以下にバッチキュー exec1 のホールド要求権限に操作員権限以上を設定する場合の例を示します。

```
$ qmgr -Po
Mgr: set execution_queue hold_privilege = operator exec1
Set Hold Privilege. queue: exec1
```

ホールド要求権限を解除するには、qmgr(1M)の"delete execution_queue hold_privilege"サブコマンドを使用してください。

(7)　サスペンド要求権限

サスペンド要求権限とは、リクエストに対してサスペンド要求ができる権限のことです。（リクエストのサスペンドについては、[操作編] バッチリクエストのサスペンド 参照。）

バッチキューにサスペンド要求権限を設定しておくことにより、そのキューに投入されたリクエストに対して、設定された権限以上の権限を持つユーザからのサスペンド要求のみが受け付けられるようになります。 バッチキューに対するサスペンド要求権限は、qmgr(1M)の"set execution_queue suspend_privilege"サブコマンドで設定してください。

以下にバッチキュー exec1 のサスペンド要求権限に操作員権限以上を設定する場合の例を示します。

```
$ qmgr -Po
Mgr: set execution_queue suspend_privilege = operator exec1
Set Suspend Privilege. queue: exec1
```

サスペンド要求権限を解除するには、qmgr(1M)の"delete execution_queue suspend_privilege"サブコマンドを実行してください。

(8)　ユーザEXITスクリプトファイル

ユーザ EXIT スクリプトファイルとは、キューに投入されたリクエストが特定の状態に状態遷移した際に、指定されたシェルスクリプトを実行するユーザ EXIT 機能における、実行用のシェル・スクリプトファイルのことです（ユーザ EXIT 機能に関しては、5.1.2.ユーザ EXIT 参照）。

バッチキューに対するユーザ EXIT スクリプトファイルの設定は、qmgr コマンドの"set execution_queue userexit"サブコマンドで行います。

以下に、バッチキューexec1 に対して、リクエストの実行開始前（PRE-RUNNING 状態への状態遷移時）にスクリプトファイル script1 を実行するよう設定する場合の例を示します。

```
$ qmgr -Po
Mgr: set execution_queue userexit location = pre-running script=(script1) queue = exec1
Set Userexit Script: exec1
```

バッチキューに対するユーザ EXIT スクリプトファイルの設定を削除するには、qmgr コマンドの"delete execution_queue userexit"サブコマンドを使用します。

上記の例でバッチキューexec1 に対して設定した、リクエストの実行開始前（PRE-RUNNING 状態への状態遷移時）のユーザ EXIT を削除する場合、以下のように実行します。

```
$ qmgr -Po
Mgr: delete execution_queue userexit location = pre-running queue = exec1
Delete Userexit Script: exec1
```

(9)　ユーザEXITのタイムアウト時間設定

ユーザ EXIT のタイムアウト時間を設定し、ストールなどの実行異常が発生した場合に、ユーザ EXIT に KILL シグナルを送信し、その実行を強制終了することができます。

タイムアウト時間の設定は、qmgr(1M)のサブコマンドで行います。

以下に、バッチキューexec1 に対して、ユーザ EXIT のタイムアウト時間を 900 秒に設定する場合の例を示します。このとき、qmgr(1M)は操作員権限で起動してください。

```
$ qmgr -Po
Mgr: set execution_queue userexit_timeout = 900 exec1
Set UserExit Timeout: exec1
```

なお、ユーザ EXIT のタイムアウト時間の既定値は 0（OFF）です。0 の場合、タイムアウト監視は行いません。

(10)　生成ジョブ数制限

バッチキューに対して、生成ジョブ数制限を設定しておくことにより、そのキューに投入されるリクエストに対して、リクエストごとに生成するジョブ数の範囲を制限することができます。

バッチキューに生成ジョブ数制限を設定するには、qmgr(1M)コマンドの"set execution_queue jobs_range"サブコマンドを使用します。

以下に、バッチキューexec1 の生成ジョブ数を 100～500 の範囲に制限する場合の設定例を示します。

```
$ qmgr -Po
Mgr: set execution_queue jobs_range = (100,500) exec1
Set Jobs Range: exec1
```

なお、グループ名・ユーザ名を個別指定してのリクエスト数制限も設定できます。詳細は、12.グループ毎・ユーザ毎の制限 を参照してください。

(11)　リクエストプライオリティ範囲の制限

バッチキューに対して、リクエストプライオリティの範囲の制限値を設定しておくことにより、そのキューに投入されるリクエストに対して、投入するユーザのユーザ権限に応じてリクエストプライオリティの範囲を制限することができます。

バッチキューに対して、リクエストプライオリティの範囲の制限値を設定するには、qmgr(1M)コマンドの"set execution_queue request_priority_range"サブコマンドを使用します。

以下に、バッチキューexec1 に対して一般利用者権限でのリクエストプライオリティを-100～0 の範囲に制限する場合の設定例を示します。

```
$ qmgr -Po
Mgr: set execution_queue request_priority_range user = (-100,0) exec1
Set Request Priority Range (User): exec1
```

(12)　サブリクエスト数制限

バッチキューに対して、そのバッチキューに投入されるパラメトリックリクエストに指定可能なサブリクエスト数を制限することができます。ここで設定された制限値を超えるサブリクエスト数を指定したパラメトリックリクエストの投入はエラーとなります。

バッチキューに対して、サブリクエスト数の制限値を設定するには、qmgr(1M)コマンドの"set execution_queue subrequest_limit"サブコマンドを使用します。

以下に、バッチキューexec1 に対してパラメトリックリクエストのサブリクエスト数を 100 に制限する場合の設定例を示します。

```
$ qmgr -Po
Mgr: set execution_queue subrequest_limit = 100 exec1
Set queue subrequest limit: exec1
```

バッチキューにおけるサブリクエスト数の制限値の初期値は、1000 です。

(13)　排他実行リクエストの投入可否

排他実行リクエスト（qsub --exclusive）の投入可否を設定することができます。排他実行リクエストはキューに設定された論理ホスト毎資源制限値（CPU、MEM、VMEM、GPU、VE）によらず常に 1 ホストあたり 1 ジョブが実行されるため、本設定によりそのリクエストの投入可否を決定します。

バッチキューに対する排他実行リクエストの投入可否は管理者権限で、qmgr のサブコマンドで以下のように設定します。

```
set execution_queue exclusive_submit {on | off} <queue>
```

off に設定すると、--exclusive 指定による投入は拒否します。on に設定すると、--exclusive 指定による投入を許可します。キュー作成時のデフォルトは off です。

(14)　ステージアウトの実施有無

バッチキューに投入されたリクエストにおいてステージアウトを実施するかどうかを設定することができます。本設定を off（ステージアウトを実施しない）に設定すると、そのキューに投入されたリクエストでステージアウトを行わず、リクエストを処理するスループットが向上します。

バッチキューに対するステージアウトの実施有無は操作員権限で、qmgr のサブコマンドで以下のように設定します。

```
set execution_queue file_stageout = { on | off }  <queue>
```

off に設定すると、リクエストにステージングファイルが指定されていてもステージアウトは実施されません。on に設定すると、ステージアウトを実施します。キュー作成時のデフォルトは on です。

注意事項

本オプションを off に設定した場合、バッチサーバの処理速度を優先するために以下の動作となります。

- qsub を実行したホストへの標準出力／エラー出力ファイルのステージアウトは行われません。qsub –e オプションまたは-o オプションを使用して標準出力／エラー出力の出力先パスを指定してください。本オプションで指定した出力先パスは、実行ホスト上のパスになります。
- 標準出力／エラー出力ファイルの出力先のディレクトリが実行ホスト上に存在することを確認

してください。ディレクトリが存在しない場合はジョブの起動が失敗します。

- qsub -e オプションまたは-o オプションにおいて、ファイル名にメタ文字%[0n]j および%f は利用できません。
- リクエストログは出力されません
- マルチノード NECMPI ジョブを実行する場合は、各ランクの標準出力/エラーが正しく出力されないため、qsub 時に-f オプションを指定してランク毎に出力を行うよう投入してください。

4.2　会話キュー

4.2.1　会話キューの生成

会話キューとは、会話リクエストの実行を制御するためのキューです。

会話キューは、qmgr(1M)の ″create interactive_queue″サブコマンドで作成します。

```
$ qmgr -Pm
Mgr: create interactive_queue = inter1 priority = 20
```

以上の手続で会話キュー inter1 が作成されます。 このとき指定する priority はキュープライオリティです。

4.2.2　会話キューの設定

(1)　資源制限値

会話キューに投入される会話リクエストに対しても、キューごとに資源制限を設定することができます。

会話キューに対して資源制限値の設定が可能な計算資源と、設定のための qmgr のサブコマンドは以下の通りです。本設定には操作員権限が必要です。

計算資源	qmgrサブコマンド
プロセス単位	
CPU時間制限値	set interactive_queue per_prc cpu_time_limit
オープンファイル数制限値（*1）	set interactive_queue per_prc open_file_number_limit
データサイズ制限値	set interactive_queue per_prc data_size_limit
スタックサイズ制限値	set interactive_queue per_prc stack_size_limit
コアファイルサイズ制限値	set interactive_queue per_prc core_size_limit
ファイルサイズ制限値	set interactive_queue per_prc file_size_limit
VE CPU時間制限値	set interactive_queue per_prc vecpu_time_limit
VE メモリサイズ制限値	set interactive_queue per_prc vememory_size_limit

仮想メモリサイズ制限値	set interactive_queue per_prc virtual_memory_size_limit
ジョブ単位	
CPU時間制限値	set interactive_queue per_job cpu_time_limit
同時実行CPU台数制限値	set interactive_queue per_job cpu_number_limit
メモリサイズ制限値	set interactive_queue per_job memory_size_limit
仮想メモリサイズ制限値	set interactive_queue per_job virtual_memory_size_limit
同時実行GPU台数制限値	set interactive_queue per_job gpu_number_limit
リクエスト単位	
経過時間制限値	set interactive_queue per_req elapse_time_limit

(*1) ･･･unlimited に設定した場合、Linux の仕様上の制約から実行ホスト上では 1024 が上限値として適用されます。

なお、経過時間制限値については、グループ名・ユーザ名を個別指定してのリクエスト数制限も設定できます。詳細は、12.グループ毎・ユーザ毎の制限 を参照してください。

メモリサイズ資源制限は cgroup を使った制御も可能です。(2.3.19 memory cgroup によるメモリ管理参照)

VE ノード毎の資源制限値

会話キューに投入されるリクエストに対して、VE ノード毎に資源使用量の制限値を設定することが可能です。VE ノード毎の資源制限値は、資源の上限と下限を、「範囲」で制限できます。リクエスト投入時に指定した資源使用量がキューに設定した資源制限値の範囲外の場合は、キューへの投入を拒否します。

資源制限値は qmgr(1M)の"set interractive_queue venode"サブコマンドで設定してください。キューに対して資源制限値の設定が可能な計算資源と、設定のための qmgr のサブコマンド以下のとおりです。本設定には操作員権限が必要です。

計算資源	qmgrサブコマンド
VE CPU時間制限値	set interactive_queue venode vecpu_time_range = (<min>,<max>[,<warn>]) <queue-name>
VE メモリサイズ制限値	set interactive_queue venode vememory_size_range = (<min>,<max>[,<warn>]) <queue-name>

<min>は最小値、<max>は最大値であり、それぞれ整数値を指定します。

<min>は<max>以下の値を指定してください、<min>と<max>が同じ値の時は、リクエスト投入時に、その値しか指定できなくなります。

キューに設定した論理ホスト毎の資源制限値は、qstat(1)コマンドの-Qf オプション（VE Node Resource Ranges）で確認することができます。(4.5.2.会話キュー 参照)

論理ホスト毎の資源制限値

会話キューに投入されるリクエストに対して、ジョブに割り当てる論理ホスト毎に資源使用量の制限値を設定することが可能です。

論理ホスト毎の資源制限値は、従来のジョブ毎の資源制限値と異なり、資源の上限だけでなく下限も指定でき、「範囲」で制限できます。

リクエスト投入時に指定した資源使用量がキューに設定した資源制限値の範囲外の場合は、キューへの投入を拒否します。

下限が設定できる点を除けば、論理ホスト毎の資源制限値に関する動作は従来のジョブ毎の資源制限値と同じです。

資源制限値は qmgr(1M)の"set interactive_queue lhost"サブコマンドで設定してください。

キューに対して資源制限値の設定が可能な計算資源と、設定のための qmgr のサブコマンド以下のとおりです。本設定には操作員権限が必要です。

計算資源	qmgrサブコマンド
VEノード数制限値	set interactive_queue lhost ve_number_range = (<min>,<max>) <queue-name>
同時実行CPU台数制限値	set interactive_queue lhost cpu_number_range = (<min>,<max>) <queue-name>
同時実行GPU台数制限値	set interactive_queue lhost gpu_number_range = (<min>,<max>) <queue-name>
CPU時間制限値	set interactive_queue lhost cpu_time_range = (<min>,<max>[,<warn>]) <queue-name>
メモリサイズ制限値	set interactive_queue lhost memory_size_range = (<min>,<max>[,<warn>]) <queue-name>
VE CPU時間制限値	set interactive_queue lhost vecpu_time_range = (<min>,<max>[,<warn>]) <queue-name>
VE メモリサイズ制限値	set interactive_queue lhost vememory_size_range = (<min>,<max>[,<warn>]) <queue-name>
仮想メモリサイズ制限値	set interactive_queue lhost virtual_memory_size_range = (<min>,<max>[,<warn>]) <queue-name>

<min>は最小値、<max>は最大値であり、それぞれ整数値を指定します。

<min>は<max>以下の値を指定してください、<min>と<max>が同じ値の時は、リクエスト投入時に、その値しか指定できなくなります。

キューに設定した論理ホスト毎の資源制限値は、qstat(1)コマンドの-Qf オプション（Logical Host Resource Ranges）で確認することができます。（4.5.1.バッチキュー 参照）

なお、これらの論理ホスト毎の資源制限値（VE ノード数制限値を除く）は、以下に示す、ジョブ毎の資源制限値の設定方法（per_job 指定）でも代替可能です。

```
set interactive_queue per_job cpu_number_limit = <max> <queue-name>
set interactive_queue per_job gpu_number_limit = <max> <queue-name>
set interactive_queue per_job cpu_time_limit = (<max>,[<warn>]) <queue-name>
set interactive_queue per_job memory_size_limit = (<max>,[<warn>]) <queue-name>
set interactive_queue per_job virtual_memory_size_limit = (<max>,[<warn>]) <queue-name>
```

この設定方法で設定した場合、論理ホスト毎の資源制限値の<min>の値は cpu_number_limit については 1 を、それ以外は 0 を指定したものと見なし、指定された<max>および<warn>はそのまま適用されます。

ただし、lhost による指定方法でも、per_job による指定方法でも、後から設定したものが有効となります。

HCA のポート数の制限値

会話キューに投入されるリクエストに対して、キューごとにジョブが使用する HCA のポート数の制限値を設定することができます。

リクエスト投入時に指定した HCA のポート数がキューに設定した HCA のポート数の制限値の範囲外である場合は、キューへの投入を拒否します。

会話キューに対する HCA のポート数の制限値は qmgr(1M)の

"set interactive_queue hca_number_range"サブコマンドにより設定を行います。

本設定には操作員権限が必要です。

計算資源	qmgr サブコマンド
HCA のポート数の制限値	set interactive_queue hca_number_range = (<min>,<max>) mode=<hca-mode> <queue-name>

<min>は最小値、<max>は最大値であり、それぞれ整数値を指定します。

<min>は<max>以下の値を指定してください。

<min>と<max>が同じ値の場合、リクエスト投入時にその値のみ指定可能となります。それ以外の値は指定できなくなります。

(2)　資源既定値

会話キューの場合も、資源制限値を指定せずに投入された会話リクエストに対する資源制限値として使用する既定値を設定することができます。

会話キューに対する資源既定値は、qmgr(1M)の以下のサブコマンドにより設定を行います。本設定には操作員権限が必要です。

計算資源	qmgrサブコマンド
プロセス単位	
CPU時間	set interactive_queue standard per_prc cpu_time_limit
オープンファイル数	set interactive_queue standard per_prc open_file_number_limit
データサイズ	set interactive_queue standard per_prc data_size_limit
スタックサイズ	set interactive_queue standard per_prc stack_size_limit
コアファイルサイズ	set interactive_queue standard per_prc core_size_limit
ファイルサイズ	set interactive_queue standard per_prc file_size_limit
VE CPU時間	set interactive_queue standard per_prc vecpu_time_limit
VE メモリサイズ	set interactive_queue standard per_prc vememory_size_limit
仮想メモリサイズ	set interactive_queue standard per_prc virtual_memory_size_limit
ジョブ単位	
CPU時間	set interactive_queue standard per_job cpu_time_limit
同時実行CPU台数	set interactive_queue standard per_job cpu_number_limit
メモリサイズ	set interactive_queue standard per_job memory_size_limit
仮想メモリサイズ	set interactive_queue standard per_job virtual_memory_size_limit
同時実行GPU台数	set interactive_queue standard per_job gpu_number_limit
リクエスト単位	
経過時間	set interactive_queue standard per_req elapse_time_limit

VE ノード毎の資源既定値

ジョブに割り当てる VE ノード毎の資源既定値を、キューに対する属性値として設定することができます。

会話キューに対する VE ノード毎の資源既定値の設定は、qmgr(1M)の以下のサブコマンドにより行います。本設定には操作員権限が必要です。

計算資源	qmgrサブコマンド
VE CPU時間	set interactive_queue standard venode vecpu_time_limit = (<time>) <queue-name>
VE メモリサイズ	set interactive_queue standard venode vememory_size_limit = (<size>) <queue-name>

論理ホスト毎の資源既定値

ジョブに割り当てる論理ホスト毎の資源既定値を、キューに対する属性値として設定することができます。

リクエスト投入時の論理ホスト毎の資源既定値に関する動作は、ジョブ毎の資源既定値と同じです。

会話キューに対する論理ホスト毎の資源既定値の設定は、qmgr(1M)の以下のサブコマンドにより行います。本設定には操作員権限が必要です。

計算資源	qmgrサブコマンド
VEノード数	set interactive_queue standard lhost ve_number_limit = <num> <queue-name>
同時実行CPU台数	set interactive_queue standard lhost cpu_number_limit = <num> <queue-name>
同時実行GPU台数	set interactive_queue standard lhost gpu_number_limit = <num> <queue-name>
CPU時間	set interactive_queue standard lhost cpu_time_limit = (<time>) <queue-name>
メモリサイズ	set interactive_queue standard lhost memory_size_limit = (<size>) <queue-name>
VE CPU時間	set interactive_queue standard lhost vecpu_time_limit = (<time>) <queue-name>
VE メモリサイズ	set interactive_queue standard lhost vememory_size_limit = (<size>) <queue-name>
仮想メモリサイズ	set interactive_queue standard lhost virtual_memory_size_limit = (<size>) <queue-name>

資源既定値の設定値は、資源制限値の下限(min)以上、上限(max)以下の値でなければいけません。

また、単位を付加する項目(cpu_time, meory_size, virtual_memory_size)は指定値を()で囲む必要があります。

なお、これらの論理ホスト毎の資源既定値（VE ノード数制限値を除く）は、以下に示す、従来のジョブ毎の資源制限値の設定方法（per_job 指定）でも代替可能です。

```
set interactive_queue standard per_job cpu_number_limit = <num> <queue-name>
set interactive_queue standard per_job gpu_number_limit = <num> <queue-name>
set interactive_queue standard per_job cpu_time_limit = (<time>) <queue-name>
set interactive_queue standard per_job memory_size_limit = (<size>) <queue-name>
set interactive_queue standard per_job virtual_memory_size_limit = (<size>) <queue-name>
```

HCA のポート数の制限値

HCA のポート数の制限値を指定してしないでキューにリクエストを投入した場合に、リクエストの HCA のポート数の制限値として自動的に設定される制限値で、キューに対する属性値として設定しておくことができます。

リクエストに HCA のポート数の制限値が設定されている場合は、この属性は参照されません。

会話キューに対する HCA のポート数の制限値は、qmgr(1M)の"set interactive_queue standard hca_number"サブコマンドにより設定を行います。本設定には操作員権限が必要です。

計算資源	qmgr のサブコマンド
HCA のポート数の制限値	set interactive_queue standard hca_number = <num> mode=<hca-mode> <queue-name>

既定値の設定値は、制限値の下限(min)以上、上限(max)以下の値でなければいけません。

(3)　キュープライオリティ

会話キューのキュープライオリティも、キューの作成時に指定しますが、

qmgr の"set interactive_queue priority"サブコマンドによって変更することができます。

```
$ qmgr -Po
Mgr: set interactive_queue priority = 10 inter1
Set Priority: inter1
```

上記の手続き会話キュー inter1 のキュープライオリティを 10 に変更します。

キュープライオリティの変更には操作員権限が必要です。

(4)　カーネルパラメータ

会話キューに対するカーネルパラメータは qmgr(1M)の”set interactive_queue”サブコマンドで設定します。

本設定には操作員権限が必要です。

NQSV でサポートしているカーネルパラメータと、対応する qmgr のサブコマンド、キュー作成時の既定値は、以下のとおりです。

カーネルパラメータ	qmgrサブコマンド	既定値
ナイス値	set interactive_queue kernel_param nice	0
リソースシェアリンググループ番号(*1)	set interactive_queue kernel_param rsg_number	0

(*1) ･･･ソケットスケジューリング機能ならびに CPUSET 機能利用時のみ有効です。

以下に、会話キューinter1 のナイス値を 5 に設定する場合の例を示します。

```
$ qmgr -Po
Mgr: set interactive_queue kernel_param nice = 5 inter1
```

```
Set Nice Value (Kernel-Parameter): inter1
```

(5)　投入リクエスト数制限

会話キューに対しても、投入できるリクエスト数の上限をキューごとに設定できます。

投入できるリクエスト数とは、1 つの会話キュー内に同時に存在している会話リクエストの総数です。

バッチサーバ単位の投入数制限よりさらに詳細なリクエスト投入数制限をかけることが可能となります（バッチサーバ単位の投入数制限の詳細に関しては、2.3.3.投入リクエスト数制限 を参照してください)。

会話キューの投入リクエスト数制限は、会話キューごとに、キュー全体、グループ別、ユーザ別に設定できます。

各設定は、qmgr(1M)の"set interactive_queue"サブコマンドで行ってください。

投入リクエスト数制限を設定するには、操作員以上の権限が必要です。

会話キューの投入リクエスト数制限を設定する qmgr(1M)のサブコマンドと、指定されていない場合の既定値は、次のとおりです。

属性	qmgr(1M)サブコマンド	既定値
キューに投入できるリクエスト数	set interactive_queue submit_limit	0（無制限）
1つのグループがキューに投入できるリクエスト数	set interactive_queue group_submit_limit	0（無制限）
1人のユーザがキューに投入できるリクエスト数	set interactive_queue user_submit_limit	0（無制限）

なお、グループ名・ユーザ名を個別指定してのリクエスト数制限も設定できます。

詳細は、12.グループ毎・ユーザ毎の制限 を参照してください。

設定されている投入リクエスト数制限値は qstat コマンドの-Q、-f オプション(キューの詳細表示)で下記の内容にて確認することができます。(4.5.2.会話キュー 参照)

- リクエストのキュー単位の投入数制限値: Submit Number Limit
- リクエストのキュー単位のユーザ別投入数制限値: Submit User Number Limit
- リクエストのキュー単位のグループ別投入数制限値: Submit Group Number Limit
    - （各項目に対して、制限値が設定されている場合はその値が、設定されていない場合はUNLIMITEDが表示されます。）

バッチサーバ単位の投入数制限とキュー単位の投入数制限が同時に設定されている場合、リクエス

ト数がいずれかの制限値に達すると、それ以上リクエストの投入はできません。

(6)　サスペンド要求権限

会話キューにサスペンド要求権限を設定しておくことにより、そのキューに投入された会話リクエストに対して、設定された権限以上の権限を持つユーザからのサスペンド要求のみが受け付けられるようになります。

会話キューに対するサスペンド要求権限は、qmgr(1M)の"set interactive_queue suspend_privilege"サブコマンドで設定してください。

以下に会話キュー inter1 のサスペンド要求権限に操作員権限以上を設定する場合の例を示します。

```
$ qmgr -Po
Mgr: set interactive_queue suspend_privilege = operator inter1
Set Suspend Privilege. queue: inter1
```

会話キューのサスペンド要求権限を解除するには、qmgr(1M)の"delete interactive_queue suspend_privilege"サブコマンドを実行してください。

(7)　ユーザEXITスクリプトファイル

会話キューに対するユーザ EXIT スクリプトファイルの設定は、

qmgr コマンドの"set interactive_queue userexit"サブコマンドで行います。

以下に、会話キューinter1 に対して、リクエストの実行開始前（PRE-RUNNING 状態への状態遷移時）にスクリプトファイル script1 を実行するよう設定する場合の例を示します。

```
$ qmgr -Po
Mgr: set interactive_queue userexit location = pre-running script=(script1) queue = inter1
Set Userexit Script: inter1
```

会話キューに対するユーザ EXIT スクリプトファイルの設定を削除するには、qmgr コマンドの"delete interactive_queue userexit"サブコマンドを使用します。

上記の例で会話キューinter1 に対してセットした、リクエストの実行開始前（PRE-RUNNING 状態への状態遷移時）のユーザ EXIT を削除する場合、以下のように実行します。

```
$ qmgr -Po
Mgr: delete interactive_queue userexit location = pre-running queue = inter1
Delete Userexit Script: inter1
```

(8)　ユーザEXITのタイムアウト時間設定

ユーザ EXIT のタイムアウト時間を設定し、ストールなどの実行異常が発生した場合に、ユーザ EXIT に KILL シグナルを送信し、その実行を強制終了することができます。

タイムアウト時間の設定は、qmgr(1M)のサブコマンドで行います。

以下に、会話キューinter1 に対して、ユーザ EXIT のタイムアウト時間を 900 秒に設定する場合の例を示します。このとき、qmgr(1M)は操作員権限で起動してください。

```
$ qmgr -Po
Mgr: set interactive_queue userexit_timeout = 900 inter1
Set UserExit Timeout: inter1
```

なお、ユーザ EXIT のタイムアウト時間の既定値は 0（OFF）です。0 の場合、タイムアウト監視は行いません。

(9)　生成ジョブ数制限

会話キューに対して、生成ジョブ数制限を設定しておくことにより、そのキューに投入される会話リクエストに対して、リクエストごとに生成するジョブ数の範囲を制限することができます。

会話キューに生成ジョブ数制限を設定するには、qmgr(1M)コマンドの"set interactive_queue jobs_range"サブコマンドを使用します。

以下に、会話キューinter1 の生成ジョブ数を 10～50 の範囲に制限する場合の設定例を示します。

```
$ qmgr -Po
Mgr: set interactive_queue jobs_range = (10,50) inter1
Set Jobs Range: inter1
```

なお、グループ名・ユーザ名を個別指定してのリクエスト数制限も設定できます。

詳細は、12.グループ毎・ユーザ毎の制限 を参照してください。

(10)　待ち合わせ指定

会話リクエストの投入時に、投入したリクエストに実行ホストが割り当てられて会話セッションが開始されるまで、投入コマンド（qlogin）において待ち合わせを行うか、即時に実行が開始できる実行ホストが割り当てられなかった場合は投入をキャンセルするか、会話リクエスト投入時の動作を会話キューに設定することができます。

会話キューの待ち合わせ設定を行うには、

qmgr(1M)の"set interactive_queue real_time_scheduling"サブコマンドを使用します。

会話キューinter1 の待ち合わせ設定として、待ち合わせを行うよう設定する場合は、以下のように

実行します。

```
$ qmgr -Po
Mgr: set interactive_queue real_time_scheduling = wait inter1
Set Real Time Schaduling. queue: inter1
```

待ち合わせ設定として、"set interactive_queue real_time_scheduling ="の後ろに指定可能なモードには以下の種類があります。

設定モード	説明
wait	実行ホストの割り当てを待ち合わせる
submit_cancel	即時割り当てが行えない場合、投入をキャンセルする
manual	待ち合わせを行うか否かは、投入コマンド(qlogin)の実行時オプション(-W)の指定に従う

(11)　強制実行シェル

会話リクエストの実行時に実行ホスト上で起動する会話セッション用のシェルは、通常投入ユーザのログインシェルとして設定されているコマンドが使用されます。

ただし、会話キューに強制実行シェルが設定されている場合は、そのキューに投入される会話リクエストで起動するシェルとして、キューに設定されているシェルコマンドが使用されます。

会話キューに強制実行シェルを設定するには、qmgr(1M)の"set interactive_queue restrict_shell"サブコマンドを使用します。

会話キューinter1 の強制実行シェルとして、/bin/bash を設定する場合は、以下のように実行します。

```
$ qmgr -Po
Mgr: set interactive_queue restrict_shell = /bin/bash inter1
Set Restrict Shell. queue: inter1
```

会話キューの強制実行シェルの設定を削除する場合は、qmgr(1M)の"delete interactive_queue restrict_shell"サブコマンドを使用してください。

(12)　アイドルタイマー

qlogin を使用した会話リクエストにおいて、シェルプロンプト上で何も実行せずセッションがアイドル状態のまま一定時間経過すると、自動的にセッションを切断して会話リクエストを終了させるようにアイドルタイマーを設定することができます。

アイドルタイマーは、会話キューごとに設定することが可能です。

会話キューにアイドルタイマーを設定するには、qmgr(1M)の"set interactive_queue idle_timer"サブコマンドを使用して、タイマーの時間を分単位で指定します。

会話キューinter1 のアイドルタイマーを５分に設定する場合は、以下のように実行します。

```
$ qmgr -Po
Mgr: set interactive_queue idle_timer = 5 inter1
Set idle_timer. inter1
```

アイドルタイマーの時間を 0 に設定するとアイドルタイマー機能は OFF になります。アイドルタイマーを設定しない場合の既定値は 0 です。

> **【注意】** qlogin(1)コマンドで、-T オプションにて MPI ジョブを指定した場合には、アイドルタイマーによる自動セッション切断はできません。

(13)　排他実行リクエストの投入可否

排他実行リクエスト（qlogin --exclusive）の投入可否を設定することができます。

排他実行リクエストはキューに設定された論理ホスト毎資源制限値（CPU、MEM、VMEM、GPU、VE）によらず常に 1 ホストあたり 1 ジョブが実行されるため、本設定によりそのリクエストの投入可否を決定します。

会話キューに対する排他実行リクエストの投入可否は管理者権限で、qmgr のサブコマンドで以下のように設定します。

```
set interactive_queue exclusive_submit {on | off} <queue>
```

off に設定すると、--exclusive 指定による投入は拒否します。on に設定すると、--exclusive 指定による投入を許可します。キュー作成時のデフォルトは off です。

4.3　転送キュー

4.3.1　転送キューの生成

転送キューとは、バッチリクエストの転送を制御するキューです。

転送キューに投入されたリクエストは、投入した転送キューに設定されている転送先のキューへ自動的に転送されます。

転送キューは、qmgr(1M)の"create routing_queue"サブコマンドで作成します。

```
$ qmgr -Pm
Mgr: create routing_queue = route1 priority = 20
```

以上の手続で転送キュー route1 が作成されます。 このとき指定する priority はキュープライオリティです。

4.3.2　転送キューの設定

(1)　キュープライオリティ

キュープライオリティとは、キュー間での優先度で、これにより他の転送キューとの間でリクエスト転送の優先順位をつけることができます。

キュープライオリティはキュー作成時に必ず設定しなければなりませんが、後からでも変更できます。 キュープライオリティの変更は、qmgr(1M) の"set routing_queue priority"サブコマンドで行ってください。

```
$ qmgr -Po
Mgr: set routing_queue priority = 10 route1
```

以上の手続きで転送キュー route1 のキュープライオリティを 10 に変更します。

(2)　投入リクエスト数制限

転送キューごとに、投入できるリクエスト数の上限を設定することができます。

投入できるバッチリクエスト数とは、1 つのキュー内に同時に存在しているリクエストの総数です。

バッチサーバ単位の投入リクエスト数制限よりさらに詳細な投入リクエスト数制限をかけることが可能となります（(バッチサーバ単位の投入数制限の詳細に関しては、2.3.3.投入リクエスト数制限を参照してください)。

投入リクエスト数制限は、転送キューごとに、キュー全体、グループ別、ユーザ別に設定できます。

各設定は、qmgr(1M)のサブコマンドで行ってください。

投入リクエスト数制限を設定するには、操作員以上の権限が必要です。

転送キューキューの投入リクエスト数制限を設定するための qmgr(1M)のサブコマンドと、指定されていない場合の既定値は、次のとおりです。

属性	qmgr(1M)サブコマンド	既定値
キューに投入できるリクエスト数	set routing_queue submit_limit	0（無制限）
1つのグループがキューに投入できるリクエスト数	set routing_queue group_submit_limit	0（無制限）
1人のユーザがキューに投入できるリクエスト数	set routing_queue user_submit_limit	0（無制限）

なお、グループ名・ユーザ名を個別指定してのリクエスト数制限も設定できます。

詳細は、12.グループ毎・ユーザ毎の制限 を参照してください。

設定されている投入リクエスト数制限値は qstat コマンドの-Q、-f オプション(キューの詳細表示)で下記の内容にて確認することができます。(4.5.3.転送キュー 参照)

- リクエストのキュー単位の投入数制限値: Submit Number Limit
- リクエストのキュー単位のユーザ別投入数制限値: Submit User Number Limit
- リクエストのキュー単位のグループ別投入数制限値: Submit Group Number Limit
    - (各項目に対して、制限値が設定されている場合はその値が、設定されていない場合はUNLIMITEDが表示されます。)

バッチサーバ単位の投入数制限とキュー単位の投入数制限が同時に設定されている場合、リクエスト数がいずれかの制限値に達すると、それ以上リクエストの投入はできません。

(3)　同時転送可能リクエスト数

同時転送可能リクエスト数とは、同時に転送キューで転送を実行できるリクエストの最大数です。

同時転送可能リクエスト数はキュー作成時に設定できますが、後から変更することも可能です。

同時転送可能リクエスト数の変更は、qmgr(1M) の"set routing_queue run_limit"サブコマンドで行ってください。

```
$ qmgr -Po
Mgr: set routing_queue run_limit = 10 route1
```

以上の手続きで転送キュー route1 の同時転送可能リクエスト数を 10 に変更します。

設定可能な最大値は 100 です。

(4)　転送先キュー

転送キューに投入されたリクエストの転送先を、転送キューごとに定義することができます。

転送キューに複数の転送先キューが設定されている場合、投入されたリクエストは、設定されている転送先キューに対して順番に転送を試み、最初にみつかった転送が可能な転送先のキューに転送されます。

転送キューの転送先キューとして有効なのは、バッチキューのみで、会話キューを指定しても転送はできません。

転送キューへの転送先キューの設定は、転送キューの作成時に行うことができますが、あとから変更、追加が可能です。

転送先の設定・変更

転送キューの転送先の設定、変更は、qmgr(1M)コマンドの"set routing_queue destination"サブコマンドで行ってください。

```
$ qmgr -Po
Mgr: set routing_queue destination = (batch1@host1) route1
```

上記の手続きで転送キュー route1 の転送先キューとしてバッチサーバ host1 上の batch1 キューが設定されます。

この設定により、転送先キューが既に設定されていた場合は上書きされます。

同一のバッチサーバ上のキューへ転送する場合は、"@host1"の部分は省略することができます。

転送先の追加

転送キューの転送先キューの追加は、qmgr(1M)コマンドの"add routing_queue destination"サブコマンドで行ってください。

```
$ qmgr -Po
Mgr: add routing_queue destination = (batch2@host1) route1
```

上記の手続きで転送キュー route1 の転送先キューとしてバッチサーバ host1 上の batch2 キューが追加されます。

これにより、すでに route1 の転送先として batch1 が定義されていた場合は、route1 の転送先キューは batch1 と batch2 になります。

転送先の削除

転送キューの転送先キューの削除は、qmgr(1M)コマンドの"delete routing_queue destination"サブコマンドで行ってください。

$ qmgr -Po

Mgr: delete routing_queue destination = (batch2@host1) route1

上記の手続きにより、転送キュー route1 の転送先から バッチサーバ host1 上の batch2 キューが削除されます。

(5)　転送待ち時間

転送キューに投入されたリクエストを転送する際、転送先として設定されているキューがすべてリクエスト受付不可であった場合には、一定時間待った後で再度転送を試みます。

このときの待ち時間を転送キューに対する転送待ち時間として、バッチサーバ単位に設定可能です。（2.3.5.転送待ち時間 参照）

4.4　ネットワークキュー

4.4.1　ネットワークキューの生成

ネットワークキューとは、 クライアントホストとバッチサーバホスト（または実行ホスト）間で、リクエストのファイルステージングの際に NQSV システムが使用するファイル転送用のキューです（ファイルステージングに関しては、5.1.3.ファイルステージング を参照してください）。

転送先のクライアントホストとバッチサーバ間のネットワークによって、ファイル転送における転送パラメータの変更が必要な場合や、クライアントホストと実行ホスト間で、外部ステージング機能を使用する場合などには、転送先クライアントホストに対してネットワークキューを作成してください。

転送先クライアントホストに対応するネットワークキューが作成されていない場合は、デフォルト・ネットワークキュー(DefaultNetQue)が 使用されます。

ネットワークキューは、qmgr(1M)の create network_queue サブコマンドで作成します。

```
$ qmgr -Pm
Mgr: create network_queue = net1 staging_machine = host1 priority = 20
```

上記の手続きにより、クライアントホスト host1 に対するネットワークキュー net1 が作成されます。このとき指定する priority はキュープライオリティです。

4.4.2　ネットワークキューの設定

(1)　キュープライオリティ

キュープライオリティとは、キュー間の優先度で、これにより他のネットワークキューとの間でファイルステージングの優先順位をつけることができます。

キュープライオリティは、ネットワークキュー作成時に必ず設定しなければなりませんが、後から変更することもできます。

ネットワークキューのキュープライオリティの変更は、qmgr(1M)の"set network_queue priority"サブコマンドで行ってください。

```
$ qmgr -Po
Mgr: set network_queue priority = 10 net1
```

以上の手続きでネットワークキュー net1 のキュープライオリティを 10 に変更します。

(2)　同時転送可能ネットワークリクエスト数

同時転送可能ネットワークリクエスト数とは、ネットワークキュー上で同時に転送できるネットワークリクエストの最大数です。

同時転送可能リクエスト数は、ネットワークキュー作成時に定義できますが、後から変更することもできます。

同時転送可能ネットワークリクエスト数の変更は、qmgr(1M) の"set network_queue run_limit"サブコマンドで行ってください。

```
$ qmgr -Po
Mgr: set network_queue run_limit = 10 net1
```

以上の手続きでネットワークキュー net1 の同時転送可能リクエスト数を 10 に変更します。

設定可能な最大値は 1000 です。

(3)　リクエスト毎同時転送可能ネットワークリクエスト数

また、ネットワークキュー上で同時に転送できるネットワークリクエストの制限値は、リクエスト毎に設定することも可能です。

リクエスト毎の同時転送可能ネットワークリクエスト数の変更は、

qmgr(1M) の"set network_queue batch_request_run_limit" サブコマンドで行ってください。

```
$ qmgr -Po
Mgr: set network_queue batch_request_run_limit = 10 net1
```

以上の手続きでネットワークキュー net1 のリクエスト毎の同時転送可能ネットワークリクエスト数を 10 に設定します。

(4)　転送先ホスト

転送先ホストとは、ネットワークキューのファイルステージングの対象となるクライアントホストです。転送先ホストは､キュー作成時に指定する必要がありますが､後から変更することもできます。

ネットワークキューの転送先ホストの変更は、qmgr(1M) コマンドの"set network_queue staging_machine"サブコマンドで行ってください。

```
$ qmgr -Po
Mgr: set network_queue staging_machine = host1 net1
```

以上の手続きでネットワークキュー net1 の転送先ホストを host1 に変更します。

(5)　ステージング方式の切り替え

ネットワークキューを使用して実行されるファイルステージングの方式には、内部ステージング方式と外部ステージング方式があり、このステージング方式を、
qmgr(1M) の"set network_queue staging_method"サブコマンドにより、変更することが可能です。

```
$ qmgr -Po
Mgr: set network_queue staging_method = external net1
```

以上の手続きでネットワークキュー net1 のステージング方式を外部ステージング方式に変更します。

ネットワークキューのステージング方式の初期値は、内部ステージングです。

ステージング方式に関しては、5.1.3.ファイルステージング を参照してください。

(6)　転送バッファサイズの変更

内部ステージング方式に設定されているネットワークキューにおいて、ファイル転送に使用するバッファサイズは、キューに対して明示的に設定されていない場合、既定値として 4K バイトのバッファが使用されます。

この転送バッファサイズは、qmgr(1M)の"set network_queue staging_extended_buffer_size"サブコマンドによって設定することができます。

```
$ qmgr -Po
Mgr: set network_queue staging_extended_buffer_size = 128 net1
```

以上の手続きでネットワークキュー net1 のファイル転送バッファサイズを 128 キロバイトに変更します。

(7)　　転送待ち時間

ネットワークキューによるファイルステージングの際、転送先ホストへのファイル転送が行えなかった場合には、一定時間待った後で再度転送を試みます。

このときの待ち時間をネットワークキューにおける転送待ち時間として、バッチサーバ単位に設定可能です。(2.3.5.転送待ち時間 参照)

4.5　キューの情報表示

4.5.1　バッチキュー

キューの状態は、qstat(1) コマンドの-Q オプションで表示することができます。

バッチキューのみの状態を表示するには、-e オプションを付けて実行してください。

```
$ qstat -Q -e
[EXECUTION QUEUE] Batch Server Host: host1
====================================
QueueName       SCH JSVs ENA STS  PRI  TOT ARR WAI QUE PRR RUN POR EXT HLD HOL RST SUS MIG STG CHK
--------------- --- ---- ------- ---- ---------------------------------------------------------------
batch1            0   64 ENA ACT   10    1   0   0   0   0   1   0   0   0   0   0   0   0   0   0
batch2            0   32 ENA ACT   20    3   0   0   2   0   1   0   0   0   0   0   0   0   0   0
--------------- --- ---- ------- ---- ---------------------------------------------------------------
 <TOTAL>                             4   0   0   2   0   2   0   0   0   0   0   0   0   0   0
--------------- --- ---- ------- ---- ---------------------------------------------------------------
```

SCH にスケジューラ番号、JSVs には、バインドされているジョブサーバでリンクアップしている数、ENA,STS の欄には、キューの状態、TOT に、投入されている総リクエスト数と、その右側に各状態ごとのリクエスト数の内訳が表示されます。

qstat -Q　-e に -t オプションを付加することにより、JSVs の欄のジョブサーバ数の表示に、各キューにバインドされている総ジョブサーバ数（リンクアップ、リンクダウン両方を含む）が付加されます。

```
$ qstat -Q batch1
[EXECUTION QUEUE] Batch Server Host: host1
====================================
QueueName       SCH JSVs ENA STS  PRI  TOT ARR WAI QUE PRR RUN POR EXT HLD HOL RST SUS MIG STG CHK
--------------- --- ---- ------- ---- ---------------------------------------------------------------
```

```
batch1           0   64 ENA ACT   10    1   0   0   0   0   1   0   0   0   0   0   0   0   0   0
-------------- --- ---- ------- ---- ----------------------------------------------------------------
 <TOTAL>                            1   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0
-------------- --- ---- ------- ---- ----------------------------------------------------------------
```

また、qstat -Q に-f オプションをつけて実行すると、キューの詳細情報を表示します。

```
$ qstat -Q -f batch1
Execution Queue: batch1@host1
    Run State    = Active
    Submit State = Enable
    Scheduler ID   = 1
    Scheduler Name = Scheduler0001
    Job Server Number = {
       412*  414*
    }
    Node Group Name = {
      (UNBIND)
    }
    Use Supplementary Groups ID for Access Privileges Check = OFF
    Noaccess User list  = {
      (none)
    }
    Noaccess Group list  = {
      (none)
    }
    Refuse submission = {
      (none)
    }
    NUMA Control = OFF
    Hold Privilege    = (none)
    Suspend Privilege = (none)
    Queue Priority = 10
    Submit Number Limit       = UNLIMITED
    Submit User Number Limit  = UNLIMITED
    Submit Group Number Limit = UNLIMITED
    Subrequest Number Limit = 1000
    Checkpoint Interval = 0
    Auto Bind JobServer = ON
    Restart File Directory = /var/opt/nec/nqsv/jsv/rstfdir
```

```
    qattach command = Enable
    IntelMPI Process Manager = hydra
    Hook Function = OFF
    Exclusive Submit = ON
    UserExit Timeout = OFF
    UserPP Timeout = 300
    File Stageout = ON
    Delete Failed Urgent Request = OFF
    Partial Process Swapping = ON
    GPU-CPU Affinity = OFF
    GPU-CPU Affinity CPU per GPU = 1
    NEC MPI Process Manager = mpd
    Range of Jobs Limit per Batch Request (min,max) = 1,10240
    Range of Request Priority = {
      manager      (min,max) = -1024,1023
      operator     (min,max) = -1024,1023
      specialuser  (min,max) = -1024,1023
      user         (min,max) = -1024,1023
    }
    Defined VE Number     = 1
    Submit VE Node Range (min,max) = 0,UNLIMITED
    Total Request         = 0
    Arriving Request      = 0
    Waiting Request       = 0
    Queued Request        = 0
    Pre-running Request   = 0
    Running Request       = 0
    Post-running Request  = 0
    Exiting Request       = 0
    Held Request          = 0
    Holding Request       = 0
    Restarting Request    = 0
    Suspending Request    = 0
    Suspended Request     = 0
    Resuming Request      = 0
    Migrating Request     = 0
    Staging Request       = 0
    Checkpointing Request = 0
  UserExit Script:
    (none)
```

```
Logical Host Resource Ranges:
  VE Node Number        = Min:         0 Max: UNLIMITED Warn:       --- Std:         0
  CPU Number            = Min:         1 Max: UNLIMITED Warn:       --- Std:         1
  GPU Number            = Min:         0 Max: UNLIMITED Warn:       --- Std:         0
  CPU Time              = Min:        0S Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
  Memory Size           = Min:        0B Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
  Virtual Memory Size   = Min:        0B Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
  VE CPU Time           = Min:        0S Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
  VE Memory Size        = Min:        0B Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
  Stdout Size           = Min:        0B Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
  Stderr Size           = Min:        0B Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
Logical Host HCA Resource Ranges:
  For I/O               = Min:      0 Max: UNLIMITED Std:         0
  For MPI               = Min:      0 Max: UNLIMITED Std:         0
  For ALL               = Min:      0 Max: UNLIMITED Std:         0
VE Node Resource Ranges:
  VE CPU Time           = Min:        0S Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
  VE Memory Size        = Min:        0B Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
Resources Limits:
  (Per-Req) Elapse Time Limit       = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
  (Per-Job) CPU Time                = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
  (Per-Job) CPU Number              = Max: UNLIMITED Warn:       --- Std:         1
  (Per-Job) Memory Size             = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
  (Per-Job) Virtual Memory Size     = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
  (Per-Job) GPU Number              = Max: UNLIMITED Warn:       --- Std:         0
  (Per-Prc) CPU Time                = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
  (Per-Prc) Open File Number        = Max: UNLIMITED Warn:       --- Std: UNLIMITED
  (Per-Prc) Virtual Memory Size     = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
  (Per-Prc) Data Segment Size       = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
  (Per-Prc) Stack Segment Size      = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
  (Per-Prc) Core File Size          = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
  (Per-Prc) Permanent File Size     = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
  (Per-Prc) VE CPU Time             = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
  (Per-Prc) VE Memory Size          = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
Kernel Parameter:
  Resource Sharing Group     = 0
  Nice Value                 = 0
```

Job Server Number = {}の項目に表示される数字は、ジョブサーバ番号です。数字の右側にアスタ

リスク(*)がついているジョブサーバはリンクアップしていることを示します。

4.5.2　会話キュー

会話キューの状態も、バッチキューと同様 qstat(1) コマンドの -Q オプションで表示します。

会話キューのみの状態を表示する場合は、-i オプションを付けて実行してください。

```
$ qstat -Q -i
[INTERACTIVE QUEUE] Batch Server Host: host1
========================================
QueueName       SCH JSVs ENA STS  PRI  TOT ARR WAI QUE PRR RUN POR EXT HLD SUS
--------------- --- ---- ------- ---- ---------------------------------------
inter1            1    4 ENA ACT   10    1   0   0   0   0   1   0   0   0   0
--------------- --- ---- ------- ---- ---------------------------------------
 <TOTAL>                             1   0   0   0   0   1   0   0   0   0
--------------- --- ---- ------- ---- ---------------------------------------
```

会話キューの場合も、-f オプションで詳細情報を表示します。

また、キュー名を指定することで特定のキューの情報のみを表示することができます。

```
$ qstat -Q -f inter1
Interactive Queue: inter1@host1
    Run State    = Active
    Submit State = Enable
    Scheduler ID   = 1
    Scheduler Name = Scheduler0001
    Job Server Number = {
       414*
    }
    Node Group Name = {
      (UNBIND)
    }
    Use Supplementary Groups ID for Access Privileges Check = OFF
    Noaccess User list  = {
      (none)
    }
    Noaccess Group list  = {
      (none)
    }
    Refuse submission = {
      (none)
    }
    NUMA Control = OFF
    Suspend Privilege = (none)
    Queue Priority = 10
    Submit Number Limit       = UNLIMITED
    Submit User Number Limit  = UNLIMITED
    Submit Group Number Limit = UNLIMITED
    Auto Bind JobServer = ON
    qattach command = Enable
    IntelMPI Process Manager = hydra
    Hook Function = OFF
    UserExit Timeout = OFF
    UserPP Timeout = 300
    Exclusive Submit = ON
    Range of Jobs Limit per Interactive Request (min,max) = 1,10240
    Real Time Scheduling = wait
```

```
    Idle Timer = 0
    Restrict Shell = (none)
    Defined VE Number     = 1
    Submit VE Node Range (min,max) = 0,UNLIMITED
    Total Request         = 0
    Arriving Request      = 0
    Waiting Request       = 0
    Queued Request        = 0
    Pre-running Request   = 0
    Running Request       = 0
    Post-running Request  = 0
    Exiting Request       = 0
    Held Request          = 0
    Suspending Request    = 0
    Suspended Request     = 0
    Resuming Request      = 0
  UserExit Script:
    (none)
  Logical Host Resource Ranges:
    VE Node Number        = Min:         0 Max: UNLIMITED Warn:       --- Std:         0
    CPU Number            = Min:         1 Max: UNLIMITED Warn:       --- Std:         1
    GPU Number            = Min:         0 Max: UNLIMITED Warn:       --- Std:         0
    CPU Time              = Min:        0S Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
    Memory Size           = Min:        0B Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
    Virtual Memory Size   = Min:        0B Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
    VE CPU Time           = Min:        0S Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
    VE Memory Size        = Min:        0B Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
  Logical Host HCA Resource Ranges:
    For I/O               = Min:      0 Max: UNLIMITED Std:         0
    For MPI               = Min:      0 Max: UNLIMITED Std:         0
    For ALL               = Min:      0 Max: UNLIMITED Std:         0
  VE Node Resource Ranges:
    VE CPU Time           = Min:        0S Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
    VE Memory Size        = Min:        0B Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
  Resources Limits:
    (Per-Req) Elapse Time Limit       = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
    (Per-Job) CPU Time                = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
    (Per-Job) CPU Number              = Max: UNLIMITED Warn:       --- Std:         1
    (Per-Job) Memory Size             = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
    (Per-Job) Virtual Memory Size     = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
```

```
    (Per-Job) GPU Number              = Max: UNLIMITED Warn:       --- Std:         0
    (Per-Prc) CPU Time                = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
    (Per-Prc) Open File Number        = Max: UNLIMITED Warn:       --- Std: UNLIMITED
    (Per-Prc) Virtual Memory Size     = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
    (Per-Prc) Data Segment Size       = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
    (Per-Prc) Stack Segment Size      = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
    (Per-Prc) Core File Size          = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
    (Per-Prc) Permanent File Size     = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
    (Per-Prc) VE CPU Time             = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
    (Per-Prc) VE Memory Size          = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
  Kernel Parameter:
    Resource Sharing Group     = 0
    Nice Value                 = 0
```

4.5.3　転送キュー

転送キューの状態も、バッチキューと同様 qstat(1) コマンドの -Q オプションで表示します。

転送キューのみの状態を表示する場合は、-r オプションを付けて実行してください。

```
$ qstat -Q -r
[ROUTING QUEUE] Batch Server Host: host1
====================================
QueueName       ENA STS PRI RLM  TOT ARR WAI QUE HLD TRS
--------------- --- --- --- --- ----------------------
pipe1           ENA INA  20   1    2   2   0   0   0   0
pipe2           DIS INA  30   1    0   0   0   0   0   0
netpipe1        ENA ACT  20   1    2   1   1   0   0   0
--------------- --- --- --- --- ----------------------
 <TOTAL>                   100    4   3   1   0   0   0
--------------- --- --- --- --- ----------------------
```

転送キューの場合も、-f オプションで詳細情報を表示します。

また、キュー名を指定することで特定のキューの情報のみを表示することができます。

```
$ qstat -Q -f pipe1
Routing Queue: pipe1@host1
    Run State    = Active
    Submit State = Enable
```

```
    Queue Priority = 10
    Total Run Limit = 1
    Submit Number Limit = UNLIMITED
    Submit User Number Limit = UNLIMITED
    Submit Group Number Limit = UNLIMITED
    Hook Function = OFF
    Total Request         = 0
    Arriving Request      = 0
    Waiting Request       = 0
    Queued Request        = 0
    Held Request          = 0
    Transiting Request    = 0
    Destinations = {
      batch1@host1
      batch2@host1
    }
    Use Supplementary Groups for Access Privileges Check = OFF
    Noaccess User list  = {
      (none)
    }
    Noaccess Group list  = {
      (none)
    }
    Refuse submission = {
      (none)
    }
```

4.5.4　ネットワークキュー

ネットワークキューの状態も、バッチキューと同様 qstat(1) コマンドの -Q オプションで表示します。 ネットワークキューのみの状態を表示する場合は、-N オプションを付けて実行してください。

```
$ qstat -Q -N
[NETWORK QUEUE] Batch Server Host: host1
====================================
QueueName       StagingMachine       ENA STS PRI RLM  TOT WAI QUE RUN
--------------- -------------------- --------------- ----------------
DefaultNetQue    (any)               ENA ACT   0   1    0   0   0   0
net1            client1              ENA ACT  10   1    0   0   0   0
```

```
-------------- -------------------- -------------- ----------------
 <TOTAL>                                      1000     0    0    0    0
-------------- -------------------- -------------- ----------------
```

ネットワークキューの場合も、-f オプションで詳細情報を表示します。

また、キュー名を指定することで特定のキューの情報のみを表示することができます。

```
$ qstat -Q -f net1
Network Queue: net1@host1
    Run State    = Active
    Submit State = Enable
    Queue Priority = 10
    Total Run Limit = 1
    Run Limit per Batch Request = 1
    Staging Machine = client1
    Staging Method  = Internal
    Staging Extended Buffer = 512KB
    Total Request   = 0
    Waiting Request = 0
    Queued Request  = 0
    Running Request = 0
```

4.5.5　出力情報のカスタマイズ

キューの状態表示の際、qstat コマンドの-F オプションに、表示したい項目名（アイテム）を指定することで項目の選択的表示が可能です。

キューの状態表示において-F オプションによる表示項目選択を行う際には、-e（バッチキュー）、-i（会話キュー）、-r（転送キュー）、-N（ネットワークキュー）の指定が必要です。

また、-f オプションによる詳細表示では、表示項目選択は行えません。

```
$ qstat -Q -e -F quenm,stt1,stt2,qtot
[EXECUTION QUEUE] Batch Server Host: host1
=======================================
QueueName       ENA STS  TOT
--------------- --- --- ----
batch1          ENA ACT    1
batch2          ENA ACT    3
--------------- --- --- ----
 <TOTAL>                   4
```

```
--------------- --- --- ----
```

キューの状態表示で-F オプションに指定可能なアイテムは以下のとおりです。

-F指定アイテム	内容	サマリ表示形式
バッチキュー(-Q -e)		
quenm	キュー名	QueueName
schid	バインドされているスケジューラID	SCH
jsvs	バインドされているジョブサーバ数	JSVs
stt1	キューのリクエスト受付可否状態	ENA
stt2	キューのリクエスト実行可否状態	STS
pri	キュープライオリティ	PRI
qtot	キューに投入されているバッチリクエスト数	TOT
arr	キューに投入されている各状態のリクエスト数	ARR
wai		WAI
que		QUE
prr		PRR
run		RUN
por		POR
ext		EXT
hld		HLD
hol		HOL
rst		RST
sus		SUS
mig		MIG
stg		STG
chk		CHK
ehost*	バインドされている実行ホスト	ExecutionHost
attbl*	リクエストへのアタッチの可否	ATTBL
会話キュー(-Q -i)		
quenm	キュー名	QueueName
schid	バインドされているスケジューラID	SCH
jsvs	バインドされているジョブサーバ数	JSVs
stt1	キューのリクエスト受付可否状態	ENA
stt2	キューのリクエスト実行可否状態	STS
pri	キュープライオリティ	PRI
qtot	キューに投入されている会話リクエスト数	TOT
arr	キューに投入されている各状態のリクエスト数	ARR
wai		WAI
que		QUE
prr		PRR
run		RUN
por		POR
ext		EXT
hld		HLD
sus		SUS
ehost*	バインドされている実行ホスト	ExecutionHost

attbl*	リクエストへのアタッチの可否	ATTBL
転送キュー(-Q -r)		
quenm	キュー名	QueueName
stt1	キューのリクエスト受付可否状態	ENA
stt2	キューのリクエスト実行可否状態	STS
pri	キュープライオリティ	PRI
rqrlm	同時転送実行数制限値	RLM
qtot	キューに投入されているリクエスト数	TOT
arr	キューに投入されている各状態のリクエスト数	ARR
wai		WAI
que		QUE
hld		HLD
trs		TRS
ネットワークキュー(-Q -N)		
quenm	キュー名	QueueName
stghost	ステージングホスト	StagingMachine
stt1	キューのリクエスト受付可否状態	ENA
stt2	キューのリクエスト実行可否状態	STS
pri	キュープライオリティ	PRI
rqrlm	同時転送実行数制限値	RLM
qtot	キューに投入されているネットワークリクエスト数	TOT
wai	キューに投入されている各状態のネットワークリクエスト数	WAI
que		QUE
run		RUN

* 本アイテムは qstat コマンドの-F オプション指定時にのみ選択表示可能な情報です。

（qstat(1)コマンド-F オプションの使用方法詳細については、[操作編]出力情報のカスタマイズ を参照してください。）

また、qstat -Q においても、-O, -o オプションにより、上記アイテムをキーとしてソートすることができます。（qstat(1)コマンド-O, -o オプションの使用方法詳細については、[操作編] 出力情報のソート を参照してください。）

4.6　キューの状態

キューの状態は、リクエストの投入可否を表す状態と、投入されたリクエストの実行の可否を表す状態が以下のように定義されています。

状態種別	状態	内容
リクエスト投入可否	ENABLE	リクエスト投入可能
	DISABLE	リクエスト投入不可
リクエスト実行可否	ACTIVE	リクエスト実行可能
	INACTIVE	リクエスト実行不可

4.6.1　リクエスト投入可否状態の変更

キューを生成した際の各状態の初期値は、DISABLE 状態、INACTIVE 状態になっています。

キューをリクエスト投入可能な状態（ENABLE 状態）に変更するには、qmgr(1M)の"enable"サブコマンドを実行します。

キュー	qmgr(1M)サブコマンド
バッチキュー	enable execution_queue
会話キュー	enable interactive_queue
転送キュー	enable routing_queue
ネットワークキュー	enable network_queue

バッチキューbatch1 をリクエスト投入可能な状態（ENABLE 状態）に変更する場合は、以下のように実行します。

```
$ qmgr -Po
Mgr: enable execution_queue = batch1
Enable Queue: batch1
```

"enable"サブコマンドの実行には、操作員権限が必要です。

また、リクエスト投入可能な状態（ENABLE 状態）のキューをリクエスト投入不可状態（DISABLE 状態）に変更するときには、qmgr(1M)の"disable"サブコマンドを実行します。

キュー	qmgr(1M)サブコマンド
バッチキュー	disable execution_queue
会話キュー	disable interactive_queue
転送キュー	disable routing_queue
ネットワークキュー	disable network_queue

バッチキューbatch1 をリクエスト投入不可状態（DISABLE 状態）に変更する場合は、以下のように実行します。

```
$ qmgr -Po
Mgr: disable execution_queue = batch1
Disable Queue: batch1
```

"disable"サブコマンドの実行には、操作員権限が必要です。

4.6.2　リクエスト実行可否状態の変更

キューをリクエスト実行可能な状態（ACTIVE 状態）に変更するには、qmgr(1M)の”start”サブコマンドを実行します。

キュー	qmgr(1M)サブコマンド
バッチキュー	start execution_queue
会話キュー	start interactive_queue
転送キュー	start routing_queue
ネットワークキュー	start network_queue

バッチキューbatch1 をリクエスト実行可能な状態（ACTIVE 状態）に変更する場合は、以下のように実行します。

```
$ qmgr -Po
Mgr: start execution_queue = batch1
Start Queue: batch1
```

"start"サブコマンドの実行には、操作員権限が必要です。

また、qmgr(1M)の"start all queue"サブコマンドにより、バッチサーバ上に存在するバッチキュー、会話キュー、転送キュー、ネットワークキューの全キューを一斉に実行可能状態（ACTIVE 状態）に変更することができます。

リクエスト実行可能な状態（ACTIVE 状態）のキューをリクエスト実行不可状態（INACTIVE 状態）に変更するときには、qmgr(1M)の”stop”サブコマンドを実行します。

キュー	qmgr(1M)サブコマンド
バッチキュー	stop execution_queue
会話キュー	stop interactive_queue
転送キュー	stop routing_queue
ネットワークキュー	stop network_queue

バッチキューbatch1 をリクエスト実行不可の状態（INACTIVE 状態）に変更する場合は、以下のように実行します。

```
$ qmgr -Po
Mgr: stop execution_queue = batch1
Stop Queue: batch1
```

"stop"サブコマンドの実行には、操作員権限が必要です。

また、qmgr(1M)の"stop all queue"サブコマンドにより、バッチサーバ上に存在するバッチキュー、会話キュー、転送キュー、ネットワークキューの全キューを一斉に実行不可状態（INACTIVE 状態）

に変更することができます。

4.7　ジョブサーバ、スケジューラとのバインド

バッチキュー、および、会話キューは、リクエスト実行用のキューであるため、キューを運用するためには、ジョブを実行するためのジョブサーバと、投入されたリクエストのスケジューリングを行うスケジューラをバインドしておく必要があります。

バッチキュー、会話キューに対する、ジョブサーバ、スケジューラのバインドを行うための qmgr(1M)のサブコマンドは、以下のとおりです。

キュー種別	ジョブサーバのバインド	スケジューラのバインド
バッチキュー	bind execution_queue job_server	bind execution_queue scheduler
会話キュー	bind interactive_queue job_server	bind interactive_queue scheduler

バッチキューbatch1 にジョブサーバ番号 100 のジョブサーバをバインドする場合、以下のように実行します。

```
$ qmgr -Pm
Mgr: bind execution_queue job_server batch1 job_server_id = 100
Bound Job_Server_ID (100) to Queue (batch1).
```

会話キューinter1 にスケジューラ番号１のスケジューラをバインドする場合、以下のように実行します。

```
$ qmgr -Pm
Mgr: bind interactive_queue scheduler inter1 scheduler_id = 1
Bound Scheduler_ID (1) to Queue (inter1).
```

また、バッチキュー、会話キューにジョブサーバ、スケジューラがバインドされている状態から、qmgr(1M)によりそのバインド関係を削除する（アンバインドする）ことができます。

アンバインドを行う qmgr(1M)のサブコマンドは、以下のとおりです。

4.8　アクセス制限の設定

バッチキュー、会話キュー、転送キューに対するリクエストの投入を特定のユーザ、グループおよび投入経路に対して制限したい場合は、キューに対してアクセス制限を設定してください。

4.8.1　ユーザ、グループに対するアクセス制限

バッチキュー、会話キュー、転送キューに、ユーザ・グループに対するアクセス制限を設定することにより、キューの表示やキューへのリクエスト投入を、アクセスを許可されたユーザ（またはグループ）のみに制限することができます。

ユーザ・グループに対するキューへのアクセス制限は、キューに対して、アクセスを許可するユーザ（またはグループ）のリストを設定するか、または、アクセスを拒否するユーザ（またはグループ）のリストを設定することにより行います。

アクセス許可設定

キューに対して、アクセスを許可するユーザ（またはグループ）のリストを設定するには、まず、キューのアクセス制限モードを ACCESS モードにセットして、キューのアクセスユーザリストに、アクセスを許可するユーザ名のリスト（または、アクセスグループリストにアクセスを許可するグループ名のリスト）を設定していきます。

キューのアクセス制限モードを ACCESS モードにセットするための qmgr(1M)のサブコマンドは、以下のとおりです。

キュー種別	qmgr(1M)サブコマンド
バッチキュー	set execution_queue access
会話キュー	set interactive_queue access
転送キュー	set routing_queue access

バッチキューbatch1 のアクセス制限モードを ACCESS モードにセットするには、qmgr コマンドを管理者権限で起動し、以下のように"set execution_queue access"サブコマンドを実行します。

```
$ qmgr -Pm
Mgr: set execution_queue access batch1
Set Access_List Access.
```

キューのアクセス制限モードを ACCESS モードに変更すると、アクセスユーザリストには初期値として root ユーザのみが登録された状態に、また、アクセスグループリストは空の状態に初期化されます（この状態では、root ユーザのみがキューにアクセス可能です）。

次に、キューのアクセスユーザリストに、アクセスを許可するユーザ名のリストを設定します。この設定には qmgr コマンドで以下のサブコマンドを使用します。

設定操作	サブコマンド(バッチキュー、会話キュー、転送キュー)
アクセスユーザリストにユーザ名のリストを設定する	set execution_queue users set interactive_queue users set routing_queue users
アクセスユーザリストにユーザを追加する	add execution_queue users

	add interactive_queue users add routing_queue users
アクセスユーザリストからユーザを削除する	delete execution_queue users delete interactive_queue users delete routing_queue users

たとえば、バッチキューbatch1 のアクセスユーザリストに、ユーザ名が user1 のユーザを追加する場合は、以下のように実行します。

```
$ qmgr -Pm
Mgr: add execution_queue users = user1 batch1
Set user list. queue: batch1
```

また、キューのアクセスグループリストに、アクセスを許可するグループ名のリストを設定するには qmgr コマンドで以下のサブコマンドを使用します。

設定操作	サブコマンド(バッチキュー、会話キュー、転送キュー)
アクセスグループリストにグループ名のリストを設定する	set execution_queue groups set interactive_queue groups set routing_queue groups
アクセスグループリストにグループを追加する	add execution_queue groups add interactive_queue groups add routing_queue groups
アクセスグループリストからグループを削除する	delete execution_queue groups delete interactive_queue groups delete routing_queue groups

バッチキューbatch1 のアクセスグループリストに、グループ名が group1 のグループを追加する場合は、以下のように実行します。

```
$ qmgr -Pm
Mgr: add execution_queue groups = group1 batch1
Set group list. queue: batch1.
```

アクセス許可の設定を行った場合、アクセスを許可するユーザ、グループの情報は、qstat -Qf により Access User list、Access Group list として以下のように表示されます。

```
$ qstat -Qf
          :
    Access User list  = {
      root     user1
    }
    Access Group list  = {
      group1
    }
         :
```

アクセス拒否設定

キューにアクセスを拒否するユーザ（またはグループ）のリストを設定するには、まず、キューのアクセス制限モードを NOACCESS モードにセットして、キューのアクセスユーザリストに、アクセスを拒否するユーザ名のリスト（または、アクセスグループリストにアクセスを拒否するグループ名のリスト）を設定していきます。

キューのアクセス制限モードは、ACCESS モードか NOACCESS モードかのどちらか一方が設定されるようになっており、既定値は、NOACCESS モードです。

キューの生成直後は、アクセス制限モードが NOACCESS モードで、アクセスユーザリスト、および、アクセスグループリストが空の状態となっており、アクセスを拒否するユーザ、グループがない状態（＝全ユーザ、全グループからアクセス可能な状態）となっています。

キューのアクセス制限モードを NOACCESS モードにセットするための qmgr(1M)のサブコマンドは、以下のとおりです。

キュー種別	qmgr(1M)サブコマンド
バッチキュー	set execution_queue noaccess
会話キュー	set interactive_queue noaccess
転送キュー	set routing_queue noaccess

バッチキューbatch1 のアクセス制限モードを NOACCESS モードにセットするには、qmgr コマンドを管理者権限で起動し、以下のように"set execution_queue noaccess"サブコマンドを実行します。

```
$ qmgr -Pm
Mgr: set execution_queue noaccess batch1
Set Access_List Noaccess.
```

キューのアクセス制限モードをNOACCESSモードに変更すると、アクセスユーザリスト、および、アクセスグループリストは空の状態に初期化されます（この状態では、全ユーザがキューにアクセス可能です）。

次に、キューのアクセスユーザリスト、アクセスグループリストに、アクセスを拒否するユーザ名、グループ名のリストを設定します。

この設定方法は、アクセス許可設定の場合と同じです。

アクセス拒否の設定を行った場合、アクセスを拒否するユーザ、グループの情報は、qstat -Qf により Noaccess User list、Noaccess Group list として以下のように表示されます。

```
$ qstat -Qf
          :
    Noaccess User list  = {
      user1    user2
    }
    Noaccess Group list  = {
      group1
    }
          :
```

4.8.2　補助グループによるアクセス制限

上記のグループに対するアクセス制限の設定で、アクセスグループリストに指定したグループ名は、ユーザのプライマリグループ名と一致するかどうかによって、アクセス可否のチェックが行われます。

ただし、キューに対して補助グループチェックの設定を行うことにより、そのキューへのリクエスト投入時の投入プロセスの補助グループが、投入可否のチェックに使用されるようになります。

キューに対する補助グループチェックの設定は、qmgr(1M)の以下のサブコマンドで行います。

キュー種別	qmgr(1M)サブコマンド
バッチキュー	set execution_queue supplementary_groups_check
会話キュー	set interactive_queue supplementary_groups_check
転送キュー	set routing_queue supplementary_groups_check

以下は、バッチキューbatch1 に対して補助グループによるアクセス制限を有効にする例です。

```
$ qmgr -Pm
Mgr: set execution_queue supplementary_groups_check on batch1
Set Supplementary Groups Check ON. queue: batch1
```

4.8.3　投入経路に対する制限

バッチキュー、会話キュー、転送キューへリクエストを投入する経路には以下の 4 つがあります。

- qsub、qlogin、qrshによるリクエスト投入
- qmoveによるリクエスト投入（会話キューを除く）
- ローカル転送キューからのリクエスト転送（同一バッチサーバの転送キューからの転送。会話キューを除く）
- リモート転送キューからのリクエスト転送（別のバッチサーバの転送キューからの転送。会話キューを除く）

この４つの投入経路に対して、キューごとにリクエストの投入を受け付けるかどうかを設定することができます。

設定は、キューに対するそれぞれの経路からの投入拒否の設定、解除を qmgr(1M)コマンドのサブコマンドで行います。

投入拒否の設定、解除を行うための qmgr(1M)のサブコマンドは、以下のとおりです。

キュー種別	投入拒否設定用サブコマンド	投入拒否解除用サブコマンド
バッチキュー	set execution_queue refuse_submission	delete execution_queue refuse_submission
会話キュー	set interactive_queue refuse_submission	delete interactive_queue refuse_submission
転送キュー	set routing_queue refuse_submission	delete routing_queue refuse_submission

バッチキューbatch1 に対して、ローカル転送キューからのリクエスト投入のみを許可したい場合、以下のように設定します。

```
$ qmgr -Po
Mgr: set execution_queue refuse_submission = (qsub,qmove,remote_routing) batch1
Set Refuse submission: batch1
```

4.9　キューのアボート

バッチキュー、会話キュー内で実行中のリクエストをすべて強制終了させることをキューのアボートといいます。強制終了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

キューのアボートは、qmgr(1M)コマンドの以下のサブコマンドで行います。

キュー種別 qmgr(1M)サブコマンド

キュー種別	qmgr(1M)サブコマンド
バッチキュー	abort execution_queue
会話キュー	abort interactive_queue

```
$ qmgr -Pm
Mgr: abort execution_queue = batch1
Delete 1.host1
```

上記の手続きで バッチキューbatch1 内で実行中の全リクエストを削除します。

アボート操作によりキュー内で実行中の全ユーザのリクエストを削除するためには、qmgr(1M)を管理者権限で起動する必要があります。

操作員権限以下で起動した場合は、起動したユーザが所有するリクエストのみを削除します。

4.10 キューのパージ

バッチキュー、会話キュー、転送キュー内でキューイング状態のリクエストをすべて削除することをキューのパージと呼びます。

削除の対象となるリクエストの状態は、以下のとおりです。

QUEUED
WAITING
HELD
STAGING

キューのパージは、qmgr(1M)コマンドの以下のサブコマンドで行います。

キュー種別	qmgr(1M)サブコマンド
バッチキュー	purge execution_queue
会話キュー	purge interactive_queue
転送キュー	purge routing_queue

```
$ qmgr -Pm
Mgr: purge execution_queue = batch1
```

```
Delete 2.host1
```

上記の手続きで バッチキューbatch1 内でキューイング状態の全リクエストを削除します。

パージ操作によりキュー内でキューイング状態の全ユーザのリクエストを削除するためには、qmgr(1M)を管理者権限で起動する必要があります。操作員権限以下で起動した場合は、起動したユーザが所有するリクエストのみを削除します。

4.11 キューの削除

キューの削除は、qmgr(1M)の "delete"サブコマンドで行います。キューの削除を行う際は、キュー内にリクエストが存在しない状態で、かつ、キューがリクエスト投入不可の状態（DISABLE 状態）になっている必要があります。

キュー種別ごとの削除用サブコマンドは以下のとおりです。

キュー種別	qmgr(1M)サブコマンド
バッチキュー	delete execution_queue
会話キュー	delete interactive_queue
転送キュー	delete routing_queue
ネットワークキュー	delete network_queue

```
$ qmgr -Pm
Mgr: delete execution_queue queue = batch1
Queue batch1 deleted.
```

上記の手続きで バッチキューbatch1 を削除します。

第5章　　リクエストの管理

5.1　バッチリクエスト

5.1.1　リクエスト属性変更

qalter(1)コマンドにより、リクエストに設定されている属性を変更することができます。

変更が可能な属性に関しては、qalter(1)コマンドを参照してください。

カーネルパラメータ属性の変更は、リクエストの実行中でも変更可能ですが、その他のリクエスト属性の変更は、リクエストの実行が開始する前に行ってください。

以下に、リクエスト 72.host1 のカーネルパラメータ属性のナイス値を 20 に変更する場合の例を示します。

```
$ qalter -P o -K nice=20 72.host1
Attribute of Request is altered.
```

5.1.2　ユーザ EXIT

ユーザ EXIT とは、リクエストが特定の状態に遷移した際に、 各ジョブが実行される実行ホスト上においてユーザ定義の任意のシェルスクリプト (ユーザ EXIT スクリプト)を実行する機能です。

ユーザ EXIT スクリプトは、以下に示すリクエストの状態に遷移したタイミング（ロケーション）で実行することができ、各タイミングで実行するスクリプトをそれぞれ最大４つまで設定しておくことができます。

- PRE-RUNNING（実行開始前）
- POST-RUNNING（実行終了直後）

(1) ユーザEXITの設定

ユーザ EXIT の設定は、バッチキュー、または、会話キューに対して、以下の qmgr(1M)のサブコマンドで実行します。

キュー種別	qmgr(1M)サブコマンド
バッチキュー	set execution_queue userexit
会話キュー	set interactive_queue userexit

上記のキュー種別に対応したサブコマンドに、"location="に対して、スクリプトを実行するタイミングを指定し、"script="で実行するスクリプトを指定し、さらに、"queue="に設定先のキュー名を指定します。

"location="に指定するスクリプトの実行タイミングは、以下のとおりです。

タイミング	指定キーワード
PRE-RUNNING	pre-running
POST-RUNNING	post-running

また、"script="でのスクリプトの指定は、スクリプトファイル名を最大４つまで、カンマ(,)または、コロン(:)で区切ったうえで、() で囲んで指定します。

このとき、カンマで区切って指定したスクリプトは、左に指定したスクリプトから順番に実行され、コロンで区切って指定したスクリプトは同時に実行を行います。

ここで指定したスクリプトファイル名のスクリプトは、バッチサーバホスト上の/opt/nec/nqsv/sbin/uex_prog/ディレクトリの下に配置しておく必要があります。

以下はバッチキューbatch1 に、リクエスト実行開始前にユーザ EXIT スクリプト script1 と script2 を順番に実行するよう、ユーザ EXIT の設定する場合の設定例です。

```
$ qmgr -Po
Mgr: set execution_queue userexit location = pre-running script = (script1,script2) queue = batch1
Set Userexit Script:batch1
```

ユーザ EXIT の設定結果は、qstat(1)コマンドの-Q -f オプションで以下のように表示されます。

```
Execution Queue: batch1@host1
    :
  UserExit Script:
    Pre-running  = {
      1st = script1
      2nd = script2
    }
    :
```

(2)　ユーザEXITの実行

キューにユーザ EXIT の設定がされている場合、ユーザ EXIT スクリプトの実行直前に、バッチサーバからリクエストの各ジョブを実行するジョブサーバへ/opt/nec/nqsv/sbin/uex_prog/ディレクトリの下に配置されたユーザ EXIT スクリプトファイルをコピーして実行します。例えば、"location="に pre-running が指定されている場合は、リクエストの PRE-RUNNING 時にユーザ EXIT スクリプトが当該ジョブサーバにコピーされて実行されます。

このとき、複数のユーザ EXIT スクリプトを順番に実行するように設定されている場合は、各実行

順のスクリプトがジョブ間で同期をとって実行されます。

つまり、1st.に設定されているスクリプトの実行が全ジョブにおいて終了した後、2nd.に設定されているスクリプトの実行が開始されます。

また、各状態遷移時に、ユーザ EXIT スクリプトは、環境変数 UEX_PROCTYPE に" EXECUTION"をセットして実行されます。このときの終了コードが 0 の場合、ユーザ EXIT スクリプト実行成功として、リクエストの状態遷移を次に進めますが、終了コードが 0 以外の場合は、ユーザ EXIT スクリプト実行失敗として、そのユーザ EXIT 実行時の状態遷移をエラーとしてリクエストの処理を行います。

また、その場合、POST-RUNNING 状態以外のユーザ EXIT の実行においては、実行エラーとなったユーザ EXIT スクリプトから、キューに設定された順番とは逆順に、環境変数 UEX_PROCTYPE に" ROLLBACK"をセットしてユーザ EXIT スクリプトが再度実行されます。

したがって、ユーザ EXIT スクリプトは、状態遷移失敗となった場合にそれまでの実行結果をキャンセルするロールバック処理が必要であれば、環境変数 UEX_PROCTYPE によって、順方向の処理とロールバック処理を切り分けて作成しておく必要があります。

ユーザ EXIT スクリプト実行時には、以下の環境変数がセットされます。

環境変数名	説明	環境変数の値
UEX_LOCATION	ユーザEXITスクリプトが起動されたロケーション	"PRERUNNING", "POSTRUNNING"
UEX_PROCTYPE	処理のタイプ	"EXECUTION"(順方向処理時) "ROLLBACK"(ロールバック時)
UEX_ORDERNO	ユーザEXITスクリプトの実行順序番号	0～3の整数値
UEX_JOBID	ユーザEXITを実行しているジョブのジョブID	ジョブ番号.シーケンス番号.ホスト名
UEX_NJOBS	リクエスト内のジョブの総数	1以上の整数
UEX_USER	ジョブ所有者のユーザ名	ユーザ名
UEX_HOME	ジョブ所有者のホームディレクトリ	ホームディレクトリの絶対パス名
UEX_STGDIR	ステージングファイル格納場所のトップ・ディレクトリ	ステージングファイル格納場所の絶対パス名
UEX_JOBDB	ジョブデータベースのトップ・ディレクトリ	ジョブデータベースの絶対パス名
UEX_RSTPREV	一つ前のリクエスト状態	"QUEUED" (QUEUED状態) "RUNNING"(RUNNING状態) "SUSPENDING"(SUSPENDING状態) "SUSPENDED"(SUSPENDED状態) "RESUMING"(RESUMING状態)
UEX_RSTRSON	状態遷移が発生した理由	"RUN"(実行開始要求) "RERUN"(リラン要求)

		"DELETE"(デリート要求) "EXIT"(実行終了) "SYSTEM_FAILURE"(システム障害)
UEX_SID	ジョブのセッションID	1以上の整数
UEX_START_TIME	ジョブの開始時刻	時刻形式：YYYY/MM/DD hh:mm:ss
UEX_END_TIME	ジョブの終了時刻	時刻形式：YYYY/MM/DD hh:mm:ss
その他ジョブに設定される環境変数		

【スクリプト例】

```
#! /bin/sh


LINK_FILE=$UEX_HOME/jobfiles


case $UEX_LOCATION in
PRERUNNING|RESTARTING)
    case $UEX_PROCTYPE in
    EXECUTION)
        rm $LINK_FILE > /dev/null 2>&1
        ln -s $UEX_STGDIR $LINK_FILE || exit 1
        ;;
    ROLLBACK)
        rm $LINK_FILE > /dev/null 2>&1
        ;;
    esac
    ;;
POSTRUNNING|HOLDING)
    case $UEX_PROCTYPE in
    EXECUTION)
        rm $LINK_FILE > /dev/null 2>&1
        ;;
    ROLLBACK)
        ln -s $UEX_STGDIR $LINK_FILE  > /dev/null 2>&1
        ;;
    esac
    ;;
esac


exit 0
```

(3)　ユーザEXITの入出力

ユーザ EXIT スクリプト実行による標準出力、標準エラー出力は、ジョブサーバ経由でバッチサーバへ転送されて、バッチサーバのログファイルに出力します。

ログファイルへの出力に関しては、以下に注意してください。

- ユーザEXITの出力をバッチサーバのログファイルへ出力させたい場合は、ログレベルを2以上に設定してください。 ログレベルの変更は、qmgr(1M)の"set batch_server log_file"サブコマンドを使用します。
- メッセージ一行の最大長は、1023バイトです。それを越えた場合は、複数行に分割されてログに出力されます。

ユーザ EXIT スクリプトの標準入力は、/dev/null へリダイレクトされ、3 番以降ファイルディスクリプタはクローズされます。

(4)　ユーザEXITのタイムアウト監視

ユーザ EXIT スクリプトのストールなど、ユーザ EXIT の実行に異常が発生した場合に備え、ユーザ EXIT の実行に対してタイムアウトによる監視ができます。

キューにユーザ EXIT のタイムアウト時間を設定します。ユーザ EXIT が実行を開始してから、キューに設定したタイムアウト時間だけ経過した場合、実行を中断します。

キューの設定は以下の qmgr(1M)のサブコマンドを使用します。

設定	qmgr(1M)のサブコマンド	既定値
ユーザExitのタイムアウト時間（秒）	set execution_queue userexit_timeout set interactive_queue userexit_timeout	0　(OFF) ※タイムアウト監視しません

5.1.3　ファイルステージング

ファイルステージング機能とは、ジョブの実行に関係するファイル群をクライアントホストと 実行ホスト間で転送する機能です。

ファイルステージング機能には、転送する方向によって、次の二種類の機能があります。

- ステージイン
    - クライアントホストから実行ホストへのファイル転送で、ジョブの実行に必要なファイルをあらかじめ実行ホスト上に展開しておく場合等に使用します。
    - ステージインを行うファイルは、リクエスト投入時に、qsub(1)コマンドの-Iオプションによって指定します。

・　ステージアウト

　・　実行ホストからクライアントホストへの転送で、主にバッチジョブの出力ファイルをユーザの元へ転送する目的で使用します。

　・　ステージアウトを行うファイルは、リクエスト投入時に、qsub(1)コマンドの-Oオプションによって指定したファイル、および、ジョブの標準出力、標準エラー出力のファイルです。

ネットワークキュー

リクエストのファイルステージングの際の、クライアントホストへのファイル転送は、NQSV システム内でネットワークリクエストとして実行され、そのためのキューとしてネットワークキューが存在します。

ネットワークキューは、ファイル転送を行うクライアントホストごとに作成し、以下に説明するステージング方式、ファイル転送に使用するソケットバッファやファイル転送の同時実行数などを設定することができます。

ネットワークキューには、システムの既定値として作成されている DefaultNetQueue があり、明示的にステージング用のネットワークキューが作成されていないクライアントホストに対しては、この DefaultNetQueue によってステージングが行われます。

ステージング方式

ファイルステージングの方式として、内部ステージング方式 と 外部ステージング方式 があります。ステージング方式は、ネットワークキューに対して設定します。ネットワークキューのステージング方式の初期値は、内部ステージングです。

ネットワークキューのステージング方式の変更には、

qmgr(1M)の"set network_queue staging_method" サブコマンドを実行してください。

```
$ qmgr -Po
Mgr: set network_queue staging_method = external net1
Set Staging Method. Network_queue: net1
```

ステージングファイルの絶対パス指定機能

外部ステージングに限り、実行ホスト上にあるステージングファイルのパスを絶対パスで指定することができる機能です。

本機能を有効化するには、qmgr(1M)の"set batch_server allow_absolute_exepath" サブコマンドを実行してください。設定には管理者権限が必要です。

```
$ qmgr -Pm
```

```
Mgr: set batch_server allow_absolute_exepath on
```

【注意】本機能は外部ステージングにおいてのみ有効であり、内部ステージングでは絶対パス指定はできません。

(1)　内部ステージング方式（システム既定のファイル転送機能）

内部ステージング方式とは、NQSV に組み込まれている標準のステージング方式です。

ファイル転送処理は、NQSV が全て行いますので特別な設定をしなくても運用を開始できます。

バッチサーバホストを中継して転送されるため転送速度は低く、大規模ファイルのステージングには向きません。

内部ステージングのファイル転送は、以下の 2 つのシーケンスに分割して実行します。

1. クライアントホストとバッチサーバホスト間の転送

バッチサーバホスト上で起動された、ステージングサーバとクライアントホスト上に常駐しているユーザエージェント間でファイルを転送します。 バッチサーバホスト上では、対象ファイルを一時的にバッチサーバデータベース内に保存します。

2. 実行ホストとバッチサーバホスト間の転送

バッチサーバとジョブサーバ間を常時接続しているTCPコネクションを使って、ファイルを転送します。 この転送で使用されるバッファのサイズは

qmgr(1M)の"set network_queue staging_extended_buffer_size"サブコマンドで変更してください。

一般的に、バッファサイズを大きくするほど転送速度は向上します。

(2)　外部ステージング

外部ステージング方式とは、サイト独自のファイル転送機能(ステージングスクリプト)を NQSV から呼び出すことでステージングを行う方式です。

①　設定方法

外部ステージングを行うには、ファイル転送を行うクライアントホストに対して、ネットワークキューを生成し、ステージング方式を外部転送（external）に設定したうえで、転送処理を行うためのステージングスクリプトをバッチサーバホスト上の/opt/nec/nqsv/sbin/stg_prog配下に、ネットワークキューのキュー名と同じファイル名で配置しておきます。

② ステージングスクリプト

ステージングスクリプトは、設定されたネットワークキューに対するファイルステージングを行う際に、バッチサーバ上でステージングを行うファイルごとにroot権限で実行されます。

このとき、ステージングスクリプトに対して、以下の環境変数が設定されます。

環境変数	説明	変数値
STG_DIRECTION	ステージング方向 (インorアウト)	"STAGE_IN"(ステージイン) "STAGE_OUT"(ステージアウト)
STG_ORDERFILE	ステージング指示ファイル	絶対パス名
STG_STGNO	ステージング番号	0 以上の整数（4 桁左 0 パディング)
STG_REQID	リクエストID	<シーケンス番号>.<ホスト名>
STG_USER	リクエスト所有者のバッチサーバホスト上でのユーザ名	文字列
STG_GROUP	リクエスト所有者のバッチサーバホスト上でのグループ名、または、リクエストのグループ指定実行機能ONの場合のリクエスト投入時に指定したグループ名（詳細は 第10章.リクエストのグループ 参照)	文字列
STG_UID	リクエスト所有者のバッチサーバホスト上でのユーザID	0 以上の整数値
STG_GID	リクエスト所有者のバッチサーバホスト上でのグループID、または、リクエストのグループ指定実行機能ONの場合のリクエスト投入時に指定したグループのグループID（詳細は 第10章.リクエストのグループ参照)	0 以上の整数値

③ ステージング指示ファイル

ステージングスクリプトに環境変数STG_ORDERFILEで渡される、ステージングを行うファイルの情報が記載されたファイルです。

ステージングスクリプトでは、このファイルの内容を参照してファイル転送処理を行うよう記述してください。

ステージング指示ファイルの内容は、1行 1 転送で、フォーマットは以下のとおりです。

```
stgno jobno cli_user cli_host cli_path exe_user exe_host exe_path
```

stgno　　ステージング番号
jobno　　ジョブ番号
cli_user クライアントホスト上でのユーザ名
cli_host クライアントホストのホスト名
cli_path クライアントホスト上での対象ファイルの絶対パス名
exe_user　　実行ホスト上でのユーザ名
exe_host実行ホストのホスト名
exe_path　　実行ホスト上での対象ファイルの絶対パス名

ステージング指示ファイルには、ステージングを行う全ファイル転送についての情報が記載されているため、ステージングスクリプト内で、ステージング番号によって自身が処理すべき転送の切り分けを行う必要があります。

【ステージング指示情報例】

```
0000 0000 cuser0 chost0 /home/user0/dir0/ euser0 ehost0 /var/opt/nec/nqsv/jsv/jobfile/0.1508.105/stgfile/dir0/
0000 0001 cuser0 chost1 /home/user1/file1 euser0 ehost1 /var/opt/nec/nqsv/jsv/jobfile/0.1508.105/stgfile/file1
```

cli_pathまたはexe_path中に空白文字または逆スラッシュが含まれる場合、ステージング指示ファイル中では下表のとおり通常文字列にマップされます。
ステージングスクリプト作成時には注意してください。

マップ対象文字	マップ後の文字列
スペース（' '）	"¥s"
水平タブ（'¥t'）	"¥t"
改行（'¥n'）	"¥n"
復帰（'¥r'）	"¥r"
逆スラッシュ（'¥¥'）	"¥¥"

【ステージングスクリプト例】

```
#! /bin/sh
#
# User-defined Staging Script. (scp version)
#

while read stgno jobno cli_user cli_host cli_path exe_user exe_host exe_path
do
    [ ${stgno} != ${STG_STGNO} ] && continue

    src=""
```

```
    dst=""
    case ${STG_DIRECTION} in
    STAGE_IN)
        src=${cli_user}@${cli_host}:${cli_path}
        dst=${exe_user}@${exe_host}:${exe_path}
    ;;
    STAGE_OUT)
        src=${exe_user}@${exe_host}:${exe_path}
        dst=${cli_user}@${cli_host}:${cli_path}
    ;;
    esac
    su ${STG_USER} -c "scp -rp ${src} ${dst}" 2>&3 || exit 1

done < ${STG_ORDERFILE}

exit 0
```

④　ステージングスクリプトの出力

ステージングスクリプトは標準出力、標準エラー出力、およびファイルディスクリプタの3番がオープンされた状態で起動されます（他のファイルディスクリプタはクローズ）。

各ファイルディスクリプタの出力先は、以下のとおりです。

- 標準出力、標準エラー出力

  バッチサーバのログファイルへリダイレクトされ、ログレベルが2以上の場合にログへ出力します。

  メッセージ一行の最大長は、1023バイトです。

  それを越えた場合は複数行に分割されてログに出力します。

- ファイルディスクリプタの3番

  最後に出力された一行を転送失敗時のエラーメッセージとみなし、バッチサーバへ転送します。

  このメッセージは、qstat(1)コマンドの-T、-fオプションで参照できます。

  メッセージの最大長は127バイトです。それを越えた部分は切り捨てられます。

  以下は、バッチサーバログの出力例です。

```
06/29 12:01:00 NQSV(STAGE): nqs_stgd(netque0): host00: No address associated with hostname
```

⑤　ステージングスクリプトの終了コード

ステージングサーバは、ステージングスクリプトの終了コードから転送結果を判断し、その結果以下のように動作します。

終了コード	意味	動作
0	ステージング成功	[STAGE_IN] リクエストはステージング成功でQUEUED状態に遷移する。 [STAGE_OUT] リクエストはステージアウト成功でEXITED状態に遷移する。
1	リトライ可能エラー	[STAGE_IN] リクエストはSTAGING状態のまま、一定時間(*1)後に再度ステージインを試みる。 [STAGE_OUT] リクエストはEXITING状態のまま、一定時間(*1)後に再度ステージアウトを試みる。
2以上	致命的エラー	[STAGE_IN] ステージイン失敗でQUEUED状態へ戻り、スケジューラによりステージングを再実行。 [STAGE_OUT] リクエストはステージアウト失敗でEXITED状態へ遷移する。この時ステージアウト対象ファイルは転送されない。

*1 ネットワークキューに設定された転送待ち時間（2.3.5.転送待ち時間 参照）

【注意】外部ステージングではリトライ可能エラーが発生した場合、そのエラーが発生したネットワークキューがリトライを行うまで停止します。その間、同じキューにあるほかのネットワークリクエストも停止します。

5.1.4　ジョブマイグレーション

ジョブマイグレーションとは、あるジョブサーバが管理しているジョブを、別のジョブサーバの管理下に移動させる機能です。

ジョブの移動と共に、ユーザが指定した任意のファイル（マイグレーションファイル）を移動先ホストへ転送する機能もサポートしています。

(1) 実行手順

ジョブマイグレーションは、以下の手順で実行します。

1)　対象ジョブのリクエストをホールド

ジョブの状態を確認します。

```
$ qstat -J 123.bsv0.example.com
  JNO RequestID         EJID   Memory      CPU JSVNO ExecutionHost   UserName Exit
 ---- --------------- ----- -------- -------- ----- --------------- -------- ----
    0 123.bsv0.exampl   136   10.15G   184.24     6 ehost1          user        -
    1 123.bsv0.exampl    12    0.00B     0.00     7 ehost2          user        -
```

リクエスト全体をホールドします。

```
$ qhold 123.bsv0.example.com
Hold request 123.bsv0.example.com is accepted.
```

リクエストがHELD状態になったことを確認します。

```
$ qstat 123.bsv0.example.com
 RequestID       ReqName  UserName Queue     Pri STT S   Memory      CPU   Elapse R H M Jobs
 --------------- -------- -------- -------- ---- --- - -------- -------- -------- - - - ----
 123.bsv.example test1    user     execque1    0 HLD -    0.00B     0.00       68 Y Y Y    2
```

2)　qmgr(1M)の"migrate job"サブコマンドにより、ジョブデータベース、マイグレーションファイル等を移動元ジョブサーバから移動先ジョブサーバに転送

```
$ qmgr -Po
 Mgr: migrate job = 0:123.bsv0.example.com job_server_id=8
 Migrated Job
```

上記は、ジョブIDが 0:123.bsv0.example.comのジョブを、ジョブサーバ番号8のジョブサーバへマイグレーションする場合の実行例です。

```
$ qstat -J 123.bsv0.example.com
JNO  RequestID         EJID   Memory      CPU JSVNO ExecutionHost   UserName Exit
---- --------------- ----- -------- -------- ----- --------------- -------- ----
   0 123.bsv0.exampl     -    0.00B     0.00     8 ehost3          user        -
   1 123.bsv0.exampl     -    0.00B     0.00     7 ehost2          user        -
```

ジョブ番号0のジョブがジョブサーバ番号8のジョブサーバ上に存在することを確認します。

3)　対象のリクエストをリリース

```
$ qrls 123.bsv0.example.com
Request 123.bsv0.example.com was release.
```

実行が再開します。

```
$ qstat 123.bsv0.example.com
 RequestID       ReqName  UserName Queue     Pri STT S   Memory      CPU   Elapse R H M Jobs
 --------------- -------- -------- -------- ---- --- - -------- -------- -------- - - - ----
 123.bsv.example test1    user     execque1    0 RUN -   20.31G  1311.20      161 Y Y Y    2
```

(2)　ジョブマイグレーションの対象ファイル

ジョブマイグレーション処理では、ジョブの実行制御に必要なファイル群を移動先ジョブサーバへ転送します。このファイル転送の対象となるファイルは以下の通りです。

ファイル種別	パス	備考
ジョブデータベース	/var/opt/nec/nqsv/jsv/<jsvno>/<jobid>/	ジョブ管理情報
ジョブファイル	/var/opt/nec/nqsv/jsv/jobfile/<jobid>/	ステージングファイル等
マイグレーションファイル	任意のパス	ユーザ指定のファイル

ジョブデータベースとジョブファイルは、ジョブマイグレーション時に NQSV により自動的に転送されますが、プログラムが使用しているファイル等、ジョブマイグレーション時に同時に転送が必要なファイルは、あらかじめリクエスト投入時に qsub(1)の-G オプションによって指定しておくことにより、マイグレーションファイルとして転送することができます。

転送可能なファイルの種別は、通常ファイル、ディレクトリ、および FIFO(名前付きパイプ)で、それ以外のファイルは無視されます。 ただし、シンボリックリンク・ファイルは通常ファイルとして転送されます。

移動元に存在しない対象ファイルは無視されます。

また、移動先に同名のファイルが既に存在している場合、対象ファイルは転送されません。

ジョブマイグレーションに伴って転送されたファイルは、対象ジョブが消滅する時点で NQSV によって自動的に削除されます。

(3)　転送パラメータの設定

ジョブマイグレーションを行う際にファイル転送を最適化するため、ジョブサーバごとに、

- 接続用ネットワークインタフェースのホスト名
- ソケットバッファサイズ
- I/Oバッファサイズ

を設定することができます。

設定には、qmgr(1M)の"set job_server migration_file_transfer_parameter"サブコマンドを実行して

ください。

以下に、ジョブサーバ番号 1 のジョブサーバに、接続用ホスト名＝host1、ソケットバッファサイズ＝OS 既定値、ファイル I/O バッファサイズ＝1024KB とする場合の例を示します。

```
$ qmgr -Po
Mgr: set job_server migration_file_transfer_parameter interface_hostname = host1 socketbuffer_size = (0)
iobuffer_size = (1048576) job_server_id = 1
Set Migration_file_transfer_parameter: Job_Server_ID (1)
```

(4)　ジョブサーバデータベースの共有化

ジョブマイグレーションの際、ジョブデータベース、ジョブファイル等のファイルは通常、ジョブサーバホスト間でファイル転送が行われます。

しかし、ジョブサーバのデータベース領域（/var/opt/nec/nqsv/jsv）をジョブサーバホスト間でファイル共有するように設定しておくと、ファイル転送を行わず、共有ファイルシステム内でのファイルコピーで処理を行うようにすることができます。

ただし、共有ファイルシステムへのファイル I/O はローカルファイルシステムへの I/O に比べて一般に低速ですので、 デメリットがある点も考慮してください。

> **【注意】**共有ファイルシステムには、NFS、ScaTeFS を使用してください。他の共有ファイルシステムを使った場合の動作は保証できません。

ジョブサーバデータ領域の共有化は、以下の手順で作業を行ってください。

1. 実行ホスト上で実行中のジョブサーバを終了させる。
2. NFSサーバ上のファイルシステムを各実行サーバがNFSマウントできるよう設定する。ジョブサーバがデータベースにroot権限でアクセスするため、rootによるアクセスを許可するよう設定してください。
3. 各実行ホストに上記ファイルシステムをNFSマウントする。
   マウントポイントは /var/opt/nec/nqsv/jsv としてください。また、ファイルシステムは読み書き可能でNFSマウントしてください。
4. ジョブサーバを起動する。

5.1.5　連携リクエスト投入失敗時の残存リクエストの削除機能

連携リクエスト投入時、連携リクエストの投入が途中で失敗した場合に、それまでに投入した連携対象リクエストは自動的に削除されます。本機能は無効化することができます。無効にした場合は、連携対象リクエストの投入が途中で失敗すると、それまでに投入した連携対象リクエストを HELD 状

態で保持します。

連携リクエスト投入失敗時の残存リクエストの削除機能は、以下の qmgr のサブコマンドで変更できます。

set batch_server auto_delete_failed_request { on | off }

on　投入に失敗した連携リクエストを自動的に削除します。(既定値)

off　投入に失敗した連携リクエストを自動的に削除しません。

HELD状態で保持します。

【アクセス権】管理者権限が必要。

設定内容は qstat -Bf（バッチサーバ情報）にて参照できます。

```
$qstat -Bf
Batch Server: bsv.example.com
     :
   Allow Absolute Exepath = OFF
   Auto Delete Failed Request = ON
     :
```

5.1.6　リラン回数制限機能

バッチリクエストを実行しているノードで障害に遭遇し、リクエストのジョブが継続実行できないことがあります。その際に NQSV はリクエストをリランして、再実行する場合があります。しかし、ノード障害の起因となるジョブの場合は、リクエストを無制限にリランすると、全ノードが障害状態になる可能性があります。バッチリクエストのリラン回数制限を設定することで、この状況を避けることが可能です。バッチリクエストのリラン制限は qmgr(1M)コマンドのサブコマンドでキュー毎に行います。

キュー種別	qmgr(1M)サブコマンド
バッチキュー	set execution_queue rerun_default = {yes | no} [limit = <limit>] <queue-name>

```
$ qmgr -Pm
Mgr: set execution_queue rerun_default = yes limit = 1 batch1
Set rerun_default(limit:1). queue: batch1
```

yes を指定した場合は、バッチリクエストがリラン可能です。<limit>にリラン可能な回数を設定します。指定されていない場合、または 0 が指定された場合は、リラン回数が無制限となります。指定可

能な値範囲は 0～2147483647 の整数です。no を指定した場合は、バッチリクエストがリラン不可となります。

バッチリクエストのリラン回数が制限値に達した場合は、当該リクエストがリラン不可状態（qsub -r n と同様）となります。再度リランが起こる状況になった場合は、当該リクエストが削除されます。

リラン回数制限の対象は、NQSV でリランされるすべてのケースです。
・ジョブサーバの LINKDOWN によりジョブがストールしてリランされる場合
・ノードヘルスチェック機能によりリランされる場合
・qrerun(1)コマンドでリランされる場合
・JobManipulator の実行中ジョブ強制リラン機能によりリランされる場合
・緊急リクエストのアサインにより下位の緊急度のリクエストがリランされる場合
・qmove(1)コマンドで-f オプションによりジョブがあるリクエストをキュー移動する場合

5.1.7　HW 障害時の自動リラン・課金除外機能

バッチリクエストを実行しているノードで障害に遭遇し、ジョブサーバが LINKDOWN した場合、リクエストはストール状態となり、ストール状態のまま elaps 超過でリクエストが終了すると、リクエストを削除し、正常に終了したジョブに対して課金します。

しかし、障害遭遇ジョブの異常により、正常終了したジョブがあったとしても実行結果に異状が出る場合があります。HW 障害時の自動リラン・課金除外機能を有効にすることで、ストール状態で elaps 超過したリクエストをリランし、課金対象外とすることができます。

本機能は、nqsd.conf ファイル（/etc/opt/nec/nqsv/nqsd.conf）及び asvd.conf ファイル（/etc/opt/nec/nqsv/asvd.conf）に以下のフォーマットで設定し、バッチサーバを再起動または qmgr の load nqsd_conf サブコマンドを実行、およびアカウンティングサーバの再起動することで有効になります。

nqsd.conf ファイル

```
enable_jsv_failure_handling  on
```

asvd.conf ファイル

```
NO_CHANGE_ON_HW_FAILURE=on
```

既定値は off です。

課金機能を ON にしていて、LINKDOWN による HW 障害遭遇リクエストに対して課金対象外とし

たい場合は、BSV 側、ASV 側両方の設定を有効にしてください。

5.2　会話リクエスト

5.2.1　リクエスト属性変更

会話リクエストに対しても、バッチリクエストと同様に、qalter(1)コマンドによりリクエストに設定されている属性を変更することができます。
変更が可能な属性に関しては、qalter(1)コマンドを参照してください。

5.2.2　ユーザ EXIT

会話リクエストに対しても､バッチリクエストと同様に､ユーザ EXIT の設定をすることが可能です。
設定方法に関しては、5.1.2.ユーザ EXIT を参照してください。

5.2.3　待ち合わせ指定

qlogin(1)による会話リクエストの投入時に、即時に実行が行える実行ホストの割り当てができない場合の動作として、投入をキャンセルさせるか、割り当ての待ち合わせを行うかを会話キューの属性として設定することができます。
設定の方法に関しては、4.2.2.会話キューの設定（10）を参照してください。

5.2.4　強制実行シェル

会話リクエストの実行時には、実行ホスト上で起動するシェルとして、通常は投入ユーザのログインシェルが使用されますが、会話キューの属性として強制実行シェルを設定しておくことにより、会話リクエストの実行時に使用するシェルをキューに設定されたパスのプログラムに限定することができます。強制実行シェルの設定方法に関しては、4.2.2.会話キューの設定（11）を参照してください。

5.2.5　アイドルタイマー

会話キューに対してアイドルタイマーを設定しておくことにより、会話リクエスト実行時に、会話セッション上で一定時間の間、プログラムの実行も入力もないアイドル状態を検知した際に、自動的に会話リクエストを終了させることができます。
アイドルタイマーの設定に関しては、4.2.2.会話キューの設定（12）を参照してください。

5.2.6　注意事項

1. 会話リクエストに対しては、ファイルステージング、およびジョブマイグレーションを行うことはできません。
2. 会話リクエストは、転送キューに投入できません。
3. 会話リクエストでは、ジョブを起動した実行ホストから qlogin/qrsh コマンドを起動したクラ

イアントホストに対して接続を行うため、クライアントホストのファイアウォールの設定が必要です。詳細は 6.3 会話リクエスト実行およびリクエストへのアタッチのための設定を参照してください。

第6章　　クライアントの管理

NQSV クライアントは、リクエストを投入・操作、NQSV の設定や操作、情報表示のための各種コマンド操作などを行います。

6.1　api_client.confの設定

NQSV クライアントの各種コマンドは、バッチサーバに接続することにより処理を行います。

このとき、クライアントコマンドが接続するバッチサーバは、各コマンドのオプション等で指定することができますが、コマンドオプションで指定されない場合には、接続先のバッチサーバの情報を /etc/opt/nec/nqsv/api_client.conf ファイルから取得します。

api_client.conf の構文は、キーワードとその値のペアで構成されます。

#文字以降は行末までコメントとみなします。

キーワード	説明
batch_server_host	バッチサーバホストのホスト名
batch_server_port	バッチサーバに接続するためのTCPポート番号

指定可能なキーワードは下表のとおりで、大文字／小文字の区別はありません。

【api_client.conf のサンプル】

```
#
# NQSV Client Configuration file.
#


batch_server_host    bsv0.example.com
batch_server_port    602
```

また、" batch_server_host"の行で指定したバッチサーバへの接続に失敗した場合に、自動的に代替のバッチサーバとして接続を行うバッチサーバホストを登録しておくことも可能です。

代替バッチサーバは、" batch_server_host"の行の下に、"alt_server_host"のキーワードを使用して記述します。

```
#
# NQSV Client Configuration file.
#
```

```
batch_server_host    bsv0.example.com
alt_server_host      bsv1.example.com
batch_server_port    602
```

この設定をすることにより、クライアントコマンドは、最初に" batch_server_host"の行に指定されたバッチサーバホストへの接続を行い、接続に失敗した場合に、" alt_server_host"で指定されたホストへ接続を行います。

"alt_server_host" の行で指定されたホストへの接続にも失敗した場合は、クライアントコマンドはバッチサーバへの接続失敗でエラーとなります。

クライアントコマンドが、" batch_server_host"の行に指定されたバッチサーバホストへの接接続に失敗して、" alt_server_host"で指定されたホストへ接続を行った場合は、以下のように、接続したバッチサーバホストを標準エラー出力に表示します。

```
$ qsub job_script
Connected to bsv1.example.com.

Request 204.bsv1.example.com submitted to queue: batch.
```

ただし、"alt_server_host"行の設定を行っても、qmgr コマンド、および、クライアントコマンドのコマンドラインで接続先のバッチサーバを明示的に指定した場合は、自動的にバッチサーバの接続の切り替えを行いません。

> **【注意】**"alt_server_host"の設定により、既定のバッチサーバへの接続に失敗した場合、自動的に別のバッチサーバへ接続されるため、リクエストの操作を行う際に操作対象として指定するリクエストが、目的のリクエストであるかどうかは十分確認するようにしてください。

6.2　ユーザエージェント

クライアントホスト上で、リクエストの投入、参照等ができるようにするために、NQSV コマンドが接続するバッチサーバの設定、および、実行結果ファイルの転送のためのデーモン（＝ユーザエージェント）を起動しておく必要があります。

ユーザエージェントの起動／終了には、通常、systemctl を使用します。

ユーザエージェントは、起動時に、接続を許可するバッチサーバを引数に指定する必要があり、クライアントホスト上の/etc/opt/nec/nqsv/nqs_uag.env の BSV_HOSTS＝にクライアントホストから

接続するバッチサーバのホスト名を設定してください（複数ある場合はスペースで区切って指定してください)。

```
BSV_HOSTS="bsv1.example.com bsv2.example.com bsv3.example.com"
```

クライアントホスト起動時にユーザエージェントを自動起動する場合は、以下のように systemctl コマンドで nqs-uag のサービスを有効化してください。

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

【起動方法】

nqs-uag のサービスが有効化されている場合、systemctl を使用して、root 権限で以下のように起動します。

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

nqs-uag のサービスが有効化されていない場合、systemctl を使用して、root 権限で以下のように nqs-uag.service を起動します。

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

systemctl を使用せず、コマンドラインから起動する場合は、以下のように引数に接続を許可するバッチサーバのホスト名を指定してください。

```
# /opt/nec/nqsv/sbin/nqs_uagd bsv0.example.com bsv1.example.com
```

ユーザエージェントは、起動するとデフォルトではポート番号 603 の TCP ポートをバインドします。/etc/services にポート名"nqsv-uag"または"nqsII-uag"でポート番号が登録されている場合は、そちらのポート番号を優先して使用します。"nqsv-uag"と"nqsII-uag"の両方が登録されている場合は"nqsv-uag"の設定を採用します。そして、バッチサーバ(正確にはステージングサーバ)からの接続要求を待ちます。

【終了方法】

nqs-uag のサービスが有効化されている場合、systemctl を使用して、root 権限で以下のように実行します。

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

nqs-uag のサービスが有効化されていない場合、systemctl を使用して、root 権限で以下のように nqs-uag.service を停止します。

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

手動で終了させる場合には、nqs_uagd プロセスに対して SIGTERM を送信してください。

6.3　会話リクエスト実行およびリクエストへのアタッチのための設定

会話リクエストでは、ジョブを起動した実行ホストから qlogin/qrsh コマンドを起動したクライアントホストに対して TCP 接続を行います。この接続にはクライアントホストのエフェメラルポートを利用するため、会話リクエストを実行するクライアントホストのファイアウォールにて、エフェメラルポートへの接続を許可するよう設定してください。

また、リクエストへのアタッチ（qattach）を利用する際も上記の会話リクエストと同様の設定が必要です。

第7章　　リモート実行型会話機能

リモート実行型会話機能とは、ユーザがクライアントホスト上で、プログラム実行要求を行うことで、NQSV へリモート実行を行う会話リクエストを自動的に投入するもので、あたかもユーザのクライアントマシン上でプログラムを実行しているようなインタフェースで、実行ホスト上でプログラムを実行するものです。

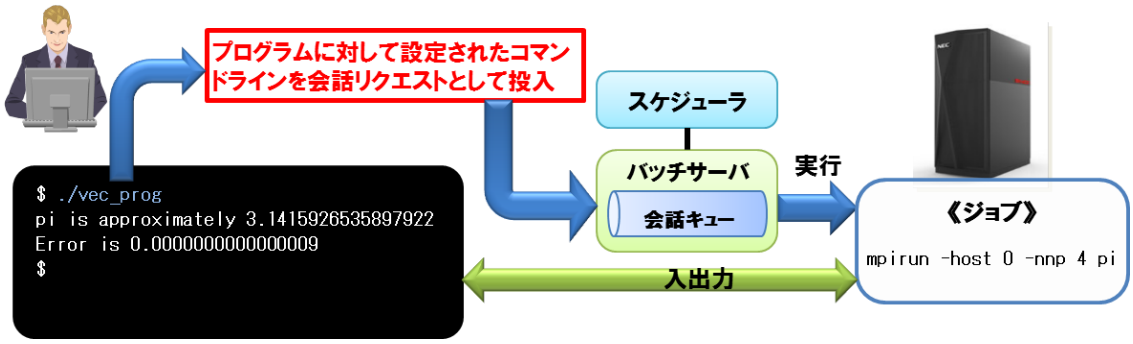


図 7-1：リモート実行型会話リクエスト

NQSV へリモート実行タイプの会話リクエストを投入するのは、qrsh コマンドが行います。

ユーザがリモート実行型会話機能を利用するには、クライアントホスト上で、事前に、リモート実行したいプログラムのコマンドラインと、それを会話リクエストとして投入するための qrsh 用のオプションを登録し、さらに、登録内容どおりに qrsh コマンドで会話リクエストを投入するリモート実行型プログラムを作成する必要があります。これには、qcmdconf(1)コマンドを使用します。

7.1　リモート実行型プログラムの登録・作成

リモート実行したいプログラムの登録とリモート実行型プログラムの作成は、qcmdconf(1)の -a オプションで行います。

```
qcmdconf  -a  --name=entry-name  --cmd=remote_cmd-line
--queue = queue-name  [--opt=qrsh_options]  [-f file-name]  [-S]
```

--name=*entry-name*

リモート実行したいコマンドラインの登録名を指定します。下記の-fオプションの指定をしない場合、この登録名がリモート実行型プログラムのファイル名となります。

--cmd=*remote_cmd-line*

リモートで実行したいコマンドラインを指定します。指定可能なコマンドラインの文字数

は最大1023文字です。

コマンドライン内に、オプションや引数の複数指定等で、スペースが入る場合は、シングルコート（'）またはダブルコート（"）で囲んでください。また、リモートへ制御文字等を引き渡したい場合は¥でエスケープしてください。

--queue = *queue-name*

qrshコマンドに指定する投入先の会話キュー名を指定します。

--opt=*qrsh_options*

qrshコマンドに指定するオプション（資源制限等）を指定します。

オプションや引数の複数指定等で間にスペースが入る場合は、シングルコート（'）またはダブルコート（"）で囲んでください。

-f *file-name*

リモート実行型プログラムのファイル名を、*file-name*にします。

-S

管理者権限で登録を実行します。本オプション指定可能であるのは、rootユーザのみです。

管理者権限で登録を行うことにより、全ユーザから利用可能なシステム共通のリモート実行型プログラムを登録します。

-Sオプションなしで登録を行った場合、qcmdconfコマンドを実行したユーザの個人用のリモート実行型プログラムを登録します（ユーザのホームディレクトリ配下に登録データファイルを作成します）。

登録したデータは以下の場所に配置されます。

- 一般ユーザの場合：

  $HOME/.nqsv/qcmd_list.....ユーザの個人用登録データ

- 管理者権限指定の場合（rootユーザによる -Sオプション指定）：

  /etc/opt/nec/nqsv/qcmd_list.....システム共通の登録データ

以下は実行例です。

```
$ qcmdconf -a --name vecprog --cmd='mpirun -host 0 -nnp 4 -ve 0 ./pi' --queue iq1 --opt='-T necmpi --cpunum-lhost=4
--venum-lhost=1'
$ ls -l vecprog
-rwxr-xr-x 1 user1  grp1 37  2月 14 14:54 2017 vecprog
```

指定した登録名 vecprog と同じ名前で、リモート実行型プログラム vecprog がカレントディレクトリに作成されました。

この vecprog を起動すると、qrsh コマンドで、「-T necmpi --cpunum-lhost=4」というオプション

指定で、会話キューiq1 にリモートタイプの会話リクエストが投入されます。

この場合、以下のコマンドラインが、実行ホスト上で実行されることになります。

```
mpirun -host 0 -nnp 4 -ve 0 ./pi
```

※mpirun へのホスト指定

会話リクエストでの -host オプションの引数には、バッチリクエストと同様に、ホスト名でなくジョブ番号を指定します。

7.2　リモート実行型プログラムの登録情報の参照

リモート実行型プログラムの登録情報は、qcmdconf (1)の -l オプションで参照できます。

qcmdconf　-l　[-S]

利用できるリモート実行型プログラムの登録情報の一覧を標準出力に表示します。

システム共通の登録情報と、起動ユーザの個人用登録情報を参照して、リモート実行型プログラム登録情報の一覧を表示します。

-S

管理者権限で登録情報の表示を行います。管理者権限で実行する場合、システム共通の登録情報のみを参照して、リモート実行型プログラム登録情報の一覧を表示します。

【一般ユーザでの使用例】

```
$ qcmdconf -l
------------------------------
Common configuration
------------------------------
name         remote-cmd                 queue      qrsh-option
-------------------------------------------------------------------------------
AP1          "AP1 -x"                   iqs        "--cpunum-lhost=2"
------------------------------
User configuration
-------------------------------------------------------------------------------
name         remote-cmd                 queue      qrsh-option
-------------------------------------------------------------------------------
vecprog      " mpirun -host 0 -nnp 4 /opt/nec/ve/bin/ve_exec ./pi" iq1 "-T necmpi --cpunum-lhost=4 --venum-lhost=1"
```

【管理者権限での使用例】

```
# qcmdconf -l -S
```

```
-----------------------------
Common configuration
-----------------------------
name          remote-cmd                  queue       qrsh-option
-------------------------------------------------------------------------------
AP1           "AP1 -x"                    iqs         "--cpunum-lhost=4 "
```

7.3　リモート実行型プログラムの実行

ユーザがクライアントホスト上で、上記で作成したリモート実行型プログラムを実行すると、対応する登録情報に従い※、自動的に qrsh(1)でリモート実行タイプの会話リクエストが投入され、即時に実行ホストでプログラムがリモード実行されます。

※一般ユーザが作成したリモート実行型プログラムは、作成したユーザの個人用登録データ $HOME/.nqsv/qcmd_list を参照します。

この時、クライアントホストと、リモート実行されている実行ホスト上のプログラムの標準入出力・エラー出力が接続されます。

【リモート実行型プログラムの実行例】

```
$ ./vecprog
pi is approximately 3.1415926535897922, Error is 0.0000000000000009
```

上記例では、実行ホスト上で以下のようなコマンドラインが実行され、実行ホスト上の標準出力がクライアントホストの標準出力に出力されます。

```
mpirun -host 0 -nnp 4 /opt/nec/ve/bin/ve_exec ./pi
```

なお、実行端末上のコマンドラインで、アプリケーションプログラムの標準入出力・エラー出力をリダイレクションやパイプを使って、切り替えることも可能です。

例えば、以下のような指定も可能です。

```
$ ./vecprog > outfile
```

outfile は、クライアントホスト（ユーザの実行端末）上に作成されます。

また、リモート実行型プログラムには、引数を渡すことも可能です。リモート実行型プログラム起

動時に引数を指定すると、そのまま実行ホスト上での実行コマンドラインに引き渡されます。

例えば、クライアントホストでのコマンドラインで引数を指定すると、

```
$ ./vecprog -opt1
```

実行ホスト上では以下のようなコマンドラインが実行されます。

```
mpirun -host 0 -nnp 4 /opt/nec/ve/bin/ve_exec ./pi -opt1
```

リモート実行型プログラムは、リモートタイプの会話リクエストの自動投入であるので、実行中に他の端末で qstat(1)コマンドで、以下のように会話リクエストとして情報が参照できます。

```
$ qstat
RequestID       ReqName  UserName Queue     Pri STT S   Memory      CPU   Elapse R H M Jobs
--------------- -------- -------- -------- ---- --- - -------- -------- -------- - - - ----
17608.bsv1      QRSH     user1    iq1         0 RUN -    0.00B     0.00        2 N N N    1
```

7.4　リモート実行型プログラムの登録削除

リモート実行型プログラムの登録を削除するには、qcmdconf(1)の -d オプションで行います。

```
qcmdconf  -d entry-name  [-S]
```

登録名を指定してそれに対応するリモート実行型プログラムの登録情報を削除します。

削除するのは、登録情報のみです。リモート実行型プログラムのファイルは、rm コマンドで削除してください。

entry-name

削除したいリモート実行型プログラムの登録名を指定します。

-S

管理者権限で登録の削除を行います。管理者権限で登録したシステム共通の登録情報から、リモート実行型プログラムの登録削除を行います。

本オプションを指定しない場合は、起動ユーザの個人用登録データから、リモート実行型プログラムの登録削除を行います。

7.5　システム共通のリモート実行型プログラムの運用

qcmdconf -a によるリモート実行型プログラムの登録・作成を、管理者権限（root ユーザで-S オプ

ション指定）で行った場合、登録情報は /etc/opt/nec/nqsv/qcmd_list に登録され、作成されたリモート実行型プログラムは、全ユーザから利用可能なシステム共通のものとなります。

システム共通の登録データの場所は /etc/opt/nec/nqsv/qcmd_list 固定ですが、その情報で作成されるリモート実行型プログラムの配置場所は問いません。

本機能を利用して、リモート実行型プログラムをツールとしてユーザで共通利用するといった運用を行う場合には、作成したリモート実行型プログラムを、ユーザが参照できる任意の場所に配置し、各ユーザでパスを通す等してください。

なお、root ユーザが、-S オプションを指定せずに、qcmdconf -a で登録・作成したリモート実行型プログラムは、root ユーザ専用（個人用）となります。

この場合は、他ユーザによる登録情報の参照・リモート実行型プログラムの実行はできません。

第8章　　NQSV 環境の保存

8.1　バッチサーバとキューの環境の保存

現在運用されているバッチサーバ、実行ホスト（ジョブサーバ）、ノードグループ、キューに関する情報を保存するには、qmgr(1M)の list all サブコマンドを実行してください。

```
# qmgr
Mgr: list all file=savefile.bsvque
```

以上で、バッチサーバ、実行ホスト（ジョブサーバ）、ノードグループ、キューに関する情報を qmgr のサブコマンドの形に変換してファイル（savefile.bsvque）に保存します。

また、テンプレート情報、並びに、カスタムリソースの情報が定義されている場合は、その情報も合わせて保存します。

上記で出力されたファイルを qmgr コマンドに読み込ませることで、NQSV 環境を容易に復元することができます。

```
# qmgr -Pm < savefile.bsvque
Set Logfile Path.
Set Log Level.
Set Save Logfile Number.
Set Logfile Size.
Set Heartbeat Interval.
Set Load Interval.
     :
```

実行キューのバインド関係の情報については、ジョブサーバ、スケジューラの構成が変更される可能性があるため、本サブコマンドでは出力しません。バインド関連情報を出力する場合は、list bind サブコマンドを使用してください。

また、テンプレート情報のみを保存するには list template サブコマンドを、カスタムリソース情報のみを保存するには list custom_resource サブコマンドを利用してください。

8.2　バッチサーバの環境の保存

現在運用されているバッチサーバに関する情報を保存するには、qmgr(1M)の list batch_server サブコマンドを実行してください。

```
# qmgr
Mgr: list batch_server file=savefile.bsv
```

以上で、バッチサーバに関する情報を qmgr のサブコマンドの形に変換してファイルに保存します。そのファイルを qmgr コマンドに読み込ませることで、NQSV 環境を容易に復元することができます。

```
# qmgr -Pm < savefile.bsv
Set Logfile Path.
Set Log Level.
Set Save Logfile Number.
Set Logfile Size.
Set Hartbeat Interval.
Set Load Interval.
    :
```

実行ホスト（ジョブサーバ）とノードグループに関する情報を保存するには、qmgr(1M)の list node サブコマンドを利用してください。

8.3　キューの環境の保存

現在運用されているキューに関する情報を保存するには、 qmgr(1M)の list queue サブコマンドを実行してください。

```
# qmgr
Mgr: list queue file=savefile.queue
```

以上で、すべてのキューに関する情報を qmgr のサブコマンドの形に変換してファイルに保存します。

そのファイルを qmgr コマンドに読み込ませることで、NQSV 環境を容易に復元することができます。

```
# qmgr -Pm < savefile.queue
Queue exec1 created.
Enable Queue: exec1
Start Queue: exec1
Set Checkpoint: exec1
Set Access_List Noaccess.
Set Reserve ID: exec1
Set Restart option: exec1
Set Elapse Time (Per-Request) limit: exec1
Set standard Elapse Time (Per-Request): exec1
Set CPU Time (Per-Job) limit: exec1
Set standard CPU Time (Per-Job): exec1
    :
```

8.4　バインド状態の保存

現在運用されているキューとジョブサーバのバインド情報を保存するには、 qmgr(1M)の list bind サブコマンドを実行してください。

```
# qmgr
Mgr: list bind file=savefile.bind
```

以上で、現在運用されているキューとジョブサーバのバインド情報を qmgr のサブコマンドの形に変換してファイルに保存します。

そのファイルを qmgr コマンドに読み込ませることで、NQSV 環境を容易に復元することができます。

```
# qmgr -Pm < savefile.bind
Bound Job_Server_ID (0) to Queue (exec1).
Bound Job_Server_ID (1) to Queue (exec1).
Bound Job_Server_ID (2) to Queue (exec1).
    :
```

第9章　　MPI リクエスト実行環境の設定

NQSV にて、以下の MPI を使用する場合には、実行環境の設定が必要です。

- OpenMPI　(OpenMPI Version 4.1.x, Version 5.0.x)
- IntelMPI　(Intel(R) MPI Library for Linux OS Version 4.1 Update 3, Version 5.1 Update 3, 2017 Update 1, 2018 Update 3, 2019 Update 7, 2021 Update 5)
- Intel OneAPI　(Intel(R) OneAPI Toolkits Release 2021.4)
- MVAPICH2　(MVAPICH2 Version 2.0, Version 2.3.4, Version 2.3.6, Version 2.3.7)
- Platform MPI (Platform MPI Version 09.01.04.03)

9.1　OpenMPI環境の設定

(1)　実行ホストの設定

OpenMPI ジョブを実行する場合は、各実行ホスト上で OpenMPI ジョブ起動用のスクリプトファイル（/opt/nec/nqsv/sbin/startopenmpi.sh）に、OpenMPI のインストール場所の情報を設定してください。

```
startopenmpi.sh
:
# Installation path of your OpenMPI
OMPIHOME=/opt/openmpi-5.0.5
:
```

startopenmpi.sh スクリプトでは、PATH および LD_LIBRARY_PATH を設定してスレーブジョブに渡しています。ご利用になる環境でパスの設定順番に依存するようなジョブを実行する場合は、環境に合わせて PATH および LD_LIBRARY_PATH を適切に設定してください。

(2)　ジョブとMPIプロセスの配置

NQSV は、OpenMPI のリモートプロセス生成機能と連携して、各ジョブ内で OpenMPI プロセスを起動します。

1. NQSVは、リクエスト投入コマンドで指定されたジョブ数分ジョブを作成し、ジョブに割り当てられたホスト名と利用できるCPU数の情報を、MPIに引き渡します。
2. MPIは、NQSVから渡された情報を元に、MPIプロセスを配置します。
   MPIは、通常1ノードにあたり1ジョブであることを前提にしています。

【注意】１つの実行ホスト上に、複数のジョブが配置された場合、OpenMPI プログラムの実行がエラーとなります。
各実行ホストに対してジョブが一つのみ配置されるよう調整してください。

9.2　IntelMPI環境の設定

NQSV は、IntelMPI のリモートプロセス生成機能と連携して、各ジョブ内で IntelMPI プロセスを起動します。

1. NQSVは、リクエスト投入コマンドで指定されたジョブ数分ジョブを作成し、ジョブに割り当てられたホスト名と利用できるCPU数の情報を、MPIに引き渡します。
2. MPIは、NQSVから渡された情報を元に、MPIプロセスを配置します。
   MPIは、通常1ノードにあたり、1ジョブであることを前提にしています。

【注意】１つの実行ホスト上に、複数のジョブが配置された場合、IntelMPI のプロセスはそれらのジョブの内の１つに偏って配置され、ジョブが正常に実行されません。
各実行ホストに対してジョブが一つのみ配置されるよう調整してください。

9.3　MVAPICH2環境の設定

NQSV は、MVAPICH2 のリモートプロセス生成機能と連携して、各ジョブ内で MVAPICH プロセスを起動します。

1. NQSV は、リクエスト投入コマンドで指定されたジョブ数分ジョブを作成し、ジョブに割り当てられたホスト名と利用できるCPU 数の情報を、MPI に引き渡します。
2. MPI は、NQSV から渡された情報を元に、MPI プロセスを配置します。MPI は、通常1ノードあたり1ジョブであることを前提にしています。

【注意】１つの実行ホスト上に、複数のジョブが配置された場合、MVAPICH プログラムの実行がエラーとなります。
各実行ホストに対してジョブが一つのみ配置されるよう調整してください。

9.4　Platform MPI環境の設定

NQSV は、Platform MPI のリモートプロセス生成機能と連携して、各ジョブ内で Platform MPI プロセスを起動します。

1. NQSVは、リクエスト投入コマンドで指定されたジョブ数分ジョブを作成し、ジョブに割り当てられたホスト名と利用できるCPU数の情報を、MPIに引き渡します。
2. MPIは、NQSVから渡された情報を元にMPIプロセスを配置します。MPIは、通常1ノードあたり1ジョブであることを前提にしています。

【注意】１つの実行ホスト上に、複数のジョブが配置された場合、Platform MPI のプロセスはそれらのジョブの内の１つに偏って配置され、ジョブが正常に実行されません。
各実行ホストに対してジョブが一つのみ配置されるよう調整してください。

9.5　NEC MPI環境の設定

プロセスマネージャの設定

NQSV において、NEC MPI ジョブを実行する際のプロセスマネージャとして Hydra を使用するか、MPD を使用するかを投入するキューに対して設定することができます。

キューに対して、NEC MPI ジョブのプロセスマネージャを設定するには、qmgr(1M)を操作員権限で起動し、以下のサブコマンドを使用します。

```
set execution_queue necmpi_process_manager = { hydra | mpd } queue
```

queue で指定したキューに投入する NEC MPI のリクエストが使用するプロセスマネージャを hydra または mpd に設定します。キューを作成した際の既定値のプロセスマネージャは、mpd になっています。

この設定ができるのはバッチキューに対してだけです。会話キューは常にプロセスマネージャとして hydra を使用します。

キューに設定されている NEC MPI のプロセスマネージャは、以下のように、qstat -Qf により確認することができます。

```
$ qstat -Qf bq
Execution Queue: bq@bsvhost
    Run State    = Active
    Submit State = Enable
    Scheduler ID   = 1
        :
    GPU-CPU Affinity = OFF
    GPU-CPU Affinity CPU per GPU = 1
    NEC MPI Process Manager = mpd
    Range of Jobs Limit per Batch Request (min,max) = 1,10240
    Range of Request Priority = {
        :
```

第10章　リクエストのグループ

リクエスト情報には、ユーザと共にグループの情報を保持します。このグループは、通常、リクエストを投入したユーザのプライマリグループとなり、ジョブ実行時は実行ホスト上のプライマリグループで実行します。

一方、リクエストのグループ指定実行機能 を利用すると、リクエストの投入時に指定したグループがリクエストのグループとなり、ジョブ実行時にはそのグループで実行します。本節では、このリクエストのグループ指定実行機能について説明します。

10.1 リクエストのグループ指定実行機能

リクエスト投入時に指定したグループ（補助グループ）で、ジョブ実行、およびアカウンティングを行う機能です。

本機能使用時、リクエストのグループを、リクエスト投入コマンド（qsub, qlogin, qrsh）で、--group=<group_name> オプションにてグループ（補助グループ）名を指定することができます。--group オプションを指定しない場合は、リクエスト投入コマンド実行時のグループ名がリクエストのグループとなります。また、newgrp により実行グループを補助グループに切り替えて、投入コマンドを実行することにより、補助グループを指定することも可能です。

本機能を使用して、リクエストのグループが指定されると、そのリクエストに対して以下の動作を行います。

- リクエストに指定されたグループを使用して、キューのアクセス制限、投入数制限、キューの資源制限のチェックを行います。
- リクエストに指定されたグループを実行GIDとして、ジョブを実行します。
- 実行ホスト上に該当グループが存在しない場合、ジョブの実行が失敗します。
- リクエストアカウントおよびジョブアカウントのグループ情報として、リクエストに指定されたグループを記録します。
- そのリクエストのファイルステージングにおいて、内部ステージングの場合、指定されたグループでファイルが生成されます。
- 外部ステージングを使用する場合、ステージングスクリプトに対して設定される環境変数 STG_GROUP（グループ名）、STG_GID（グループのGID）の値が、ユーザが指定したグループのものになります。

10.2 機能の有効化と設定情報の参照

リクエストのグループ指定実行機能を利用するには、以下の qmgr のサブコマンドで機能を有効化します。

形式

```
set batch_server specify_group_request { on | off }
```

on　リクエストのグループ指定実行機能を使用します。
　　リクエスト投入時に補助グループが指定でき、そのグループでジョブを実行します。
off　リクエストのグループ指定実行機能を使用しません。

【アクセス権】操作員権限が必要。

上記で行った設定内容は qstat -Bf（バッチサーバ情報）にて参照できます。

```
$qstat -Bf
Batch Server: bsv.example.com
     :
  Specify Group for Request = ON
  Allow Absolute Exepath = OFF
     :
```

10.3 機能利用時の注意事項

本機能を利用する場合、以下の点にご注意ください。

- NQSVシステム内（クライアントホスト、BSVホスト、ジョブサーバホスト）でユーザ（UID）／グループ（GID）の設定を共通にしてください。
    - 設定が異なると、投入エラー、実行エラー、意図しないUID/GIDでのジョブ実行などが発生することがあります。
- ユーザマッピングは行わないでください。
    - ユーザマッピングがされている（クライアント－BSV－実行ホスト間のいずれかでユーザ名が別のものにマッピングされている）場合は、リクエスト投入時のグループの指定は無効となります。マッピングされたユーザのプライマリグループで動作します。

・ 転送キューによる他BSVへのリクエスト転送を行う場合、以下の設定をBSVホスト間で共通にしてください。

本機能のON/OFF

ユーザ（UID）／グループ（GID）の設定

ユーザマッピングの設定

・ キューのアクセス制限のチェック時、キューの補助グループチェック設定の状態（ON/OFF）にかかわらず、リクエスト投入時に指定されたグループでのみアクセス制限のチェックを行います。

・ ワークフロー（wstart）に対してグループを指定することはできません。

　・ なお、ワークフロー内のリクエストに対して個別にグループを指定することは可能です。

第11章　グループ毎・ユーザ毎の制限

バッチサーバならびにキューでの制限において、グループ名またはユーザ名を個別に指定して、制限の上限を設定できるものがあります。

対象となる制限は、バッチサーバ、ならびに、バッチキュー、会話キュー、転送キューの投入数制限と、バッチキュー、会話キューの生成ジョブ数制限と経過時間制限です。

11.1 バッチサーバ単位の投入数制限

バッチサーバで、グループ名またはユーザ名を個別に指定して、システムに投入できるリクエスト数の上限を設定できます。

なお、グループ名・ユーザ名個別指定での投入数制限を設定したユーザおよびグループでは、1 グループ当たり・１ユーザ当たりの投入数制限値は無効となります。

(1)　制限の設定

グループ名またはユーザ名を個別指定して投入数制限を設定する、qmgr(1M)の"set batch_server"サブコマンドは、次のとおりです。本設定は、操作員以上の権限が必要です。

属性	qmgr(1M)サブコマンド
指定グループが投入できるリクエスト数	set batch_server group_submit_limit=<limit> groups=<group_name>
指定ユーザが投入できるリクエスト数	set batch_server user_submit_limit=<limit> users=<user_name>

(2)　制限の解除

グループ名またはユーザ名を個別指定して投入数制限を解除する qmgr(1M)の"delete batch_server"サブコマンドは次のとおりです。設定の解除は、操作員以上の権限が必要です。

属性	qmgr(1M)サブコマンド
指定グループが投入できるリクエスト数	delete batch_server group_submit_limit groups=<group_name>
指定ユーザが投入できるリクエスト数	delete batch_server user_submit_limit users=<user_name>

11.2 キュー単位の制限

11.2.1 投入数制限

キュー単位で、グループ名またはユーザ名を個別に指定して、キューに投入できるリクエスト数の上限を設定できます。本設定が可能なキュー種別は、バッチキュー、会話キュー、転送キューです。

なお、グループ名・ユーザ名個別指定での投入数制限を設定したユーザおよびグループでは、1 グループ当たり・１ユーザ当たりの投入数制限値は無効となります。

ただし、キュー全体に対するリクエスト投入数制限を超えることはできません。

(1) 制限の設定

グループ名またはユーザ名を個別指定して投入数制限を設定する、qmgr（1M)の″set exection_queue″, ″set interactive_queue″, ″set routing_queue″サブコマンドは、次のとおりです。本設定は、操作員以上の権限が必要です。

属性	キュー種別	qmgr(1M)サブコマンド
指定グループが投入できるリクエスト数	バッチキュー	set execution_queue group_submit_limit=<limit> groups=<group_name> <queue>
	会話キュー	set interactive_queue group_submit_limit=<limit> groups=<group_name> <queue>
	転送キュー	set routing_queue group_submit_limit=<limit> groups=<group_name> <queue>
指定ユーザが投入できるリクエスト数	バッチキュー	set execution_queue user_submit_limit=<limit> users=<user_name> <queue>
	会話キュー	set interactive_queue user_submit_limit=<limit> users=<user_name> <queue>
	転送キュー	set routing_queue user_submit_limit=<limit> users=<user_name> <queue>

(2) 制限の解除

グループ名またはユーザ名を個別指定して投入数制限を解除する qmgr(1M)の”delete exection_queue”, ”delete interactive_queue”, ”delete routing_queue”サブコマンドは次のとおりです。設定の解除は、操作員以上の権限が必要です。

属性	キュー種別	qmgr(1M)サブコマンド
指定グループが投入できるリクエスト数	バッチキュー	delete execution_queue group_submit_limit groups=<group_name> <queue>
	会話キュー	delete interactive_queue group_submit_limit groups=<group_name> <queue>
	転送キュー	delete routing_queue group_submit_limit groups=<group_name> <queue>
指定ユーザが投入できるリクエスト数	バッチキュー	delete execution_queue user_submit_limit users=<user_name> <queue>
	会話キュー	delete interactive_queue user_submit_limit users=<user_name> <queue>

	転送キュー	delete routing_queue user_submit_limit users=<user_name> <queue>

11.2.2 生成ジョブ数制限

グループ名またはユーザ名を個別に指定して、キューに投入されるリクエストに対して、リクエストごとに生成するジョブ数の範囲を制限できます。

リクエスト投入時に指定したジョブ数が、グループまたはユーザに設定したジョブ数の範囲外である場合は、キューへの投入を拒否します。本設定が可能なキュー種別は、バッチキューと会話キューです。

なお、本設定をしていても、キューに設定された生成ジョブ数制限は超えることはできません。

(1)　制限の設定

グループ名またはユーザ名を個別指定してキューの生成ジョブ数制限を設定する、qmgr (1M)の"set exection_queue", "set interactive_queue"サブコマンドは、次のとおりです。

本設定は、操作員以上の権限が必要です。

属性	キュー種別	qmgr(1M)サブコマンド
指定グループが投入できるリクエストの生成ジョブ数範囲	バッチキュー	set execution_queue jobs_range = <limit> groups=<group_name> <queue>
	会話キュー	set interactive_queue jobs_range = <limit> groups=<group_name> <queue>
指定ユーザが投入できるリクエストの生成ジョブ数範囲	バッチキュー	set execution_queue jobs_range = <limit> users=<user_name> <queue>
	会話キュー	set interactive_queue jobs_range = <limit> users=<user_name> <queue>

(2)　制限の解除

グループ名またはユーザ名を個別指定してキューの生成ジョブ数制限を解除する qmgr(1M)の"delete exection_queue", "delete interactive_queue"サブコマンドは次のとおりです。設定の解除は、操作員以上の権限が必要です。

属性	キュー種別	qmgr(1M)サブコマンド
指定グループが投入できるリクエストの生成ジョブ数範囲	バッチキュー	delete execution_queue jobs_range groups=<group_name> <queue>
	会話キュー	delete interactive_queue jobs_range groups=<group_name> <queue>
指定ユーザが投入できるリクエストの生成ジョブ数範囲	バッチキュー	delete execution_queue jobs_range users=<user_name> <queue>
	会話キュー	delete interactive_queue jobs_range users=<user_name> <queue>

11.2.3 経過時間制限

グループ名またはユーザ名を個別に指定して、キューに投入されるリクエストに対して、リクエストごとに経過時間の制限値を設定できます。リクエスト投入時に指定した経過時間がグループまたはユーザに設定した経過時間の制限値より大きい場合は、キューへの投入を拒否します。

本設定が可能なキュー種別は、バッチキューと会話キューです。

なお、本設定をしていても、キューに設定された経過時間の制限値を超えてリクエストを投入することはできません。

また、リクエスト投入時に経過時間を指定しなかった場合、リクエストの経過時間は、下記の 3 つの値の内、最小値となります。

- キューの経過時間既定値
- グループに設定した経過時間制限の最大値
- ユーザに設定した経過時間制限の最大値

(1)　制限の設定

グループ名またはユーザ名を個別指定してキューの経過時間制限値を設定する、qmgr (1M)の"set exection_queue", "set interactive_queue"サブコマンドは、次のとおりです。

本設定は、操作員以上の権限が必要です。

属性	キュー種別	qmgr(1M)サブコマンド
指定グループが投入できるリクエストの経過時間制限値	バッチキュー	set execution_queue per_req elapse_time_limit = <limit> groups=<group_name> <queue>
	会話キュー	set interactive_queue per_req elapse_time_limit = <limit> groups=<group_name> <queue>
指定ユーザが投入できるリクエストの経過時間制限値	バッチキュー	set execution_queue per_req elapse_time_limit = <limit> users=<user_name> <queue>
	会話キュー	set interactive_queue per_req elapse_time_limit = <limit> users=<user_name> <queue>

(2)　制限の解除

グループ名またはユーザ名を個別指定してキューの経過時間の制限値を解除する qmgr(1M)の"delete exection_queue", "delete interactive_queue"サブコマンドは次のとおりです。設定の解除は、操作員以上の権限が必要です。

属性	キュー種別	qmgr(1M)サブコマンド
指定グループが投入できるリクエストの経過時間制限値	バッチキュー	delete execution_queue per_req elapse_time_limit groups=<group_name> <queue>
	会話キュー	delete interactive_queue per_req elapse_time_limit groups=<group_name> <queue>
指定ユーザが投入で	バッチキュー	delete execution_queue per_req elapse_time_limit users=<user_name> <queue>

きるリクエストの経過時間制限値	会話キュー	delete interactive_queue per_req elapse_time_limit users=<user_name> <queue>

11.3 グループ毎・ユーザ毎の制限情報の参照

グループ名またはユーザ名を個別指定して設定した各種制限値については、qstat(1)コマンドの--limit オプションにて確認してください。

また、バッチサーバの各種設定は -Bf オプション、キューの各種設定は -Qf オプションにて参照できますが、これらのオプションではグループ名またはユーザ名を個別指定しての制限値の設定の有無のみを確認できます。

(1) qstat --limit

qstat --limit では、バッチサーバならびにキュー（バッチキュー・会話キュー・転送キュー）での制限において、グループ名またはユーザ名を個別に指定して設定できる制限に関連する情報を表示します。

ただし、qstat(1)コマンド実行時のアクセス権限により、下記のとおり表示内容が異なります。

- 管理者権限、操作員権限
    - 全てのグループ・ユーザに関する制限値を表示します。

    バッチサーバの投入数制限値

    キュー情報

        アクセス制限（ユーザリスト、グループリスト）

        投入数制限

        生成ジョブ数制限

        経過時間制限

- グループ管理者権限
    - 管理するグループに関する制限値のみ表示します。

    バッチサーバの投入数制限値

    キュー情報（管理するグループにアクセス権のあるキューのみ）

        アクセス制限（グループリスト）

        投入数制限

        生成ジョブ数制限

        経過時間制限

・　特別利用者権限、一般利用者権限

　　・　qstat(1)コマンド実行時のユーザ・グループに関する制限値のみ表示します。

　　・　キュー情報はアクセス権のあるキューのみで、アクセス制限のリストは表示しません。

表示例）管理者権限で qstat(1)コマンドの--limit オプションを実行

```
$qstat -Pm --limit
Batch Server: bsvhost
  Submit Number Limitation Value       = 1000
  Submit User  Number Limitation Value = UNLIMITED
        user1       : Limit = 200
        user2       : Limit = 100
  Submit Group Number Limitation Value = 300
        grpA        : Limit = 100
        grpB        : Limit = 400


Execution Queue: exec1
  Access User list = {
      user2
  }
  Access Group list = {
      grpA     grpB     grpZ
  }
  Submit Number Limit       = 100
  Submit User  Number Limit = 10
        user1       : Limit = 30
        user2       : Limit = 5
  Submit Group Number Limit = 50
        grpA        : Limit = 40
        grpZ        : Limit = UNLIMITED
  Range of Jobs Limit per Batch Request (min,max) = 1,10240
      User  Limit:
            user1       : Limit = 1,512
            user2       : Limit = 1,256
      Group Limit:
            grpA        : Limit = 512,1024
            grpZ        : Limit = 512,2048
  Resources Limits:
    (Per-Req) Elapse Time Limit = Max: UNLIMITED Warn: UNLIMITED
```

```
      User  Limit:
            user1       : limit = Max:     3000S Warn:     2940S
            user2       : limit = Max:     1500S Warn:     1450S
      Group Limit:
            grpA        : limit = Max:      600S Warn:      540S
            grpZ        : limit = Max:     5000S Warn:     4900S

Execution Queue: bq1
:
```

(2)　qstat -Bf

バッチサーバにおいて、グループ名またはユーザ名を個別指定しての投入数制限が設定されている場合には、「The other Limitation Values are setting.」のメッセージが表示されます。

表示例）qstat -Bf

```
$ qstat -Bf
Batch Server: bsvhost
:
    Submit Number Limitation Value       = 1000
    Submit User  Number Limitation Value = UNLIMITED
    Submit Group Number Limitation Value = UNLIMITED
    The other Limitation Values are setting.
  Use License :
:
```

(3)　qstat -Qf

キューにおいて、グループ名またはユーザ名を個別指定しての投入数制限、生成ジョブ数制限、経過時間制限のいずれかが設定されている場合には、各キュー情報の最後に「The other Limitation Values are setting.」のメッセージが表示されます。

表示例）qstat -Qf

```
$ qstat -Qf exec1
Execution Queue: exec1@ bsvhost
:
  Kernel Parameter:
    Resource Sharing Group     = 0
:
```

```
    Aging Range                = 160
    Slave Priority             = 0
  The other Limitation Values are setting.


Execution Queue: exec2@ bsvhost
:
```

第12章　VE および GPU 対応

NQSV では実行ホストに搭載された VE および GPU を管理し、それぞれのハードウェアを最適に利用できるジョブの実行制御機能を搭載しています。

12.1 VE対応機能の設定

実行ホストが SX-Aurora TSUBASA システムの場合は、本設定を必ず実施してください。

VE を使用するジョブを実行するためには、VEOS 用として、VH 上の CPU コアを確保する必要があります。VH に搭載されている VE 数と同じ数の CPU コアを確保する必要があるため、NQSV の CPUSET 機能でジョブが利用する CPU コアを適切な数に制限してください。

VH 上に 40CPU コア、8VE が搭載されている環境を例に、具体的な設定手順を以下に説明します。CPUSET 機能の詳細については、18.2CPUSET 機能も合わせて参照してください。

1. cpuset.conf の作成

   実行ホスト上の/etc/opt/nec/nqsv 配下に cpuset.conf を作成します。

2. ホスト全体の資源量を表す CPUSET を定義

   cpuset.conf に以下の設定を記載してください。

   ```
   cpuset   0-39    0-1    0
   ```

   この設定は、ホストにトータル 40 コア（0～39 番のコア）が搭載され、２つのメモリノード（ソケット）があることを示しています。

3. ジョブ用 CPU コアの CPUSET を定義

   cpuset.conf に以下の設定を記載してください。

   ```
   forjob    8-39    0-1    1
   ```

   この設定は、ジョブ実行用に 8～39 番の計 32 コアを確保し、それに対応するメモリノード（ソケット）番号 0～1 を割り当てています。この設定は仮想的な RSG 番号の 1 番にマッピングします。残りの 8 コア(0~7 番)が VEOS 用に確保されることになります。

   このとき、ジョブ用 CPU コアには、VE が接続されているソケット内の CPU コアがなるべく多くなるよう設定してください。

4. JSV の起動

   上記設定後、JSV プロセスを起動してください。

5. キューのソケットスケジューリング機能を有効にする

VE ジョブを投入するキューで CPUSET が利用できるよう、ソケットスケジューリング機能を有効にします。

qmgr で以下のサブコマンドを使用してください。（exec1 キューに設定する場合）

```
set execution_queue numa_control = on exec1
```

※ソケットスケジューリング機能の使用には注意事項があります。第 18 章ソケットスケジューリング(18.1.1 ソケットスケジューリング機能の有効化、および 18.1.3 メモリアロケーションポリシー)を参照してください。

6. キューに CPUSET（仮想 RSG）を割り当て

VE ジョブを投入するキューに上記で設定した CPUSET(仮想 RSG)番号 1 を割り当てます。

qmgr で以下のサブコマンドを使用してください。（exec1 キューに設定する場合）

```
set execution_queue kernel_param rsg_number =1 exec1
```

以上の設定を行うことで、ジョブは常に forjob で設定した 32 コア（8～39 番コア）を使用し、VEOS 用に 8 コアを確保しておくことが可能になります。この例を参考に、実行ホストに搭載されている実際の VE 数に合わせて CPU コア番号を適切に設定してください。

また、Hyper-Threading 機能が有効な場合は、論理コア数で計算してください。上記の例では、8 論理コアを VEOS 用に確保してください。

12.2 VEの総数指定投入とキューの既定VE搭載数の設定

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。

NQSV ではリクエストで使用する VE の総数を指定してリクエストを投入することができます。この投入方法ではジョブ数が自動的に決定されるため、ユーザはジョブ数を気にすることなく、VE の数だけで簡単にリクエスト投入ができます。

VE 総数指定によるリクエスト投入を行う場合、キューに対して既定搭載 VE ノード数を設定します。既定搭載 VE ノード数は、キューにバインドされている各実行ホストに搭載された既定の VE ノード数を表します。リクエストが VE 総数指定で投入された場合、既定搭載 VE ノード数をもとにジョブ数を自動計算してリクエストに適用します。

既定搭載 VE ノード数は qmgr にて以下のサブコマンドで実行キュー毎に設定します。設定には管理者権限が必要です。

```
set execution_queue defined_ve_number = <venum> <queue_name>
set interactive_queue defined_ve_number = <venum> <queue_name>
```

<venum>には 1 以上の整数を指定し、デフォルトは 1 です。

設定した既定搭載 VE ノード数は、qstat -Qf で以下のように参照できます。

```
$ qstat -Qf bq
Execution Queue: bq@bsv1
    Run State    = Active
    Submit State = Enable
    . . .
Defined VE Number     = 1
Submit VE Node Range (min,max)  = 0,UNLIMITED
Total Request         = 0
Arriving Request      = 0
    . . .
```

VE 総数は qsub --venode オプションで指定して投入します。
VE 総数指定が行われた場合のジョブ数の算出式は以下の通りです。小数点以下は切り捨てます。

<ジョブ数> = (<venum> + (<既定搭載 VE ノード数> - 1)) / <既定搭載 VE ノード数>

例えば、キューの既定搭載 VE ノード数が 8 の場合、--venode オプションの指定値によって算出されるジョブ数は以下のようになります。

② qsub --venode=16の場合、ジョブ数：2
② qsub --venode=9の場合、 ジョブ数：2
③ qsub --venode=8の場合、 ジョブ数：1
④ qsub --venode=4の場合、ジ ョブ数：1

ジョブ毎に既定搭載 VE ノード数の倍数に切り上げられた VE 資源が確保されるため、上記例②④のように、既定搭載 VE ノード数で割り切れない総 VE 数を指定して投入した場合、実行ホスト上の VE 資源が余ることになります。

12.3 キューのVE総数範囲制限の設定

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。
VE ノード総数指定（qsub --venode）で投入できる VE ノード総数の範囲（上限値と下限値）をキュー毎に設定できます。qsub 時に指定された VE 総数の値がこの範囲外であった場合、リクエストの投入はエラーとなります。

この範囲制限は、VE 総数指定（qsub --venode）で投入したリクエストのみ有効です。既定値では、論理ホスト毎の VE ノード数指定（qsub –venum-lhost）で投入したリクエストに対して制限されません。併せて制限するには、**投入可能な VE ノード数制限の拡張** を参照してください。

VE ノード総数の範囲は qmgr の以下のサブコマンドで設定し、設定には管理者権限が必要です。

```
set execution_queue submit_venode_range = (<min>,<max>) <queue-name>
set interactive_queue submit_venode_range = (<min>,<max>) <queue-name>
```

<min>には 0 以上<max>以下の整数値を指定しなければなりません。
<min>と<max>が同じ値であった場合、qsub --venode にはその値しか指定できなくなります。
<min>と<max>がともに 0 の場合、VE ノードを使用しないリクエストのみ投入できます。
キュー作成直後は、<min>が 0、<max>が UNLIMITED になります。

設定した範囲制限の値は qstat -Qf コマンドで以下のように参照します。

```
$ qstat -Qf bq
Execution Queue: bq@bsv1
    Run State    = Active
    Submit State = Enable
    ・・・
Defined VE Number     = 1
Submit VE Node Range (min,max)  = 0,UNLIMITED
Total Request         = 0
Arriving Request      = 0
    ・・・
```

投入可能な VE ノード数制限の拡張

論理ホスト毎の VE ノード数指定（qsub --venum-lhost）と実行されるジョブ数（qsub -b）を乗算した値に対してキューの VE 総数範囲制限を適用するには、次の設定を行ってください。

以下の書式で nqsd.conf ファイル（/etc/opt/nec/nqsv/nqsd.conf）に設定してください。その後、バッチサーバを再起動、または qmgr の load nqsd_conf サブコマンドによる、設定ファイル再読み込みを行うことで有効になります。

```
queues_for_extended_venode_range [queue ...]
```

queue は空白区切りで複数指定可能です。*queue* に指定されたキューに対して本機能を有効にします。*queue* を指定しない場合、BSV に存在するすべてのキューに対して有効にします。
queues_for_extended_venode_range の行がない場合、論理ホスト毎の VE ノード数指定（qsub --venum-lhost）で投入したリクエストに対して制限されません。

12.4 HCAの割り当て

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。

実行ホストに VE と HCA が搭載されている場合、ジョブに対して HCA を割り当てることができます。本機能の詳細については［JobManipulator 編］HCA の割り当てを参照してください。

12.5 HCA障害の検知

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。

実行ホストに VE と HCA が搭載されている場合、HCA の障害を検知した際にその JSV を自動的に運用から除外することができます。

障害を検知するタイミングは、リクエストが PRE-RUNNING 状態または POST-RUNNING 状態の時です。

本機能では、実行ホストに搭載された HCA が全て利用可能な状態が健全な状態であると考え、一台でも利用不可な HCA が検知されると障害発生と見なします。

また、本機能では実行ホストにインストールされた ibstat コマンドを実行して HCA の障害発生を検知するため、ibstat がインストールされていない環境では動作しません。

HCA 障害が発生した際に、その JSV をどのように扱うかを JSV 毎に設定することができます。設定可能な対処は以下の通りです。

off	障害を検知しても何もしない。
unbind	障害を検知すると、キューから JSV をアンバインドする。実行中のジョブが存在する場合、そのジョブは実行を継続する。
down	障害を検知すると、JSV を終了させる。実行中のジョブが存在する場合、そのジョブはリランされる。

HCA 障害時の対処は qmgr の以下のサブコマンドで設定します。

```
set job_server hca_failure_check = {off | down | unbind} job_server_id = jsvid
```

実行には操作員権限が必要です。デフォルトは off です。

上記の設定内容は qstat -Sf コマンドにて確認ができます。

```
$ qstat -Sf
Job Server Name: JobServer0001
     ・・・〈省略〉・・・
    HCA Failure Check = UNBIND
     ・・・〈省略〉・・・
```

【注意】
本節の機能は下位互換性維持のために残されている機能ですので、ご使用は推奨しません。HCA の障害検知には 19.5 ノードヘルスチェック機能をご使用ください。

12.6 キューの同時実行GPU台数制限の設定

バッチキュー、会話キューに投入されるリクエストに対して、キューごとにジョブが使用する同時実行 GPU 台数の制限値を設定することができます。

リクエスト投入時に指定した資源使用量がキューに設定した資源制限値より大きい場合は、キューへの投入を拒否します。

キューに対して同時実行 GPU 台数の制限を設定するには、qmgr の以下のサブコマンドを使用します。

```
set execution_queue per_job gpu_number_limit = limit  queue
set interactive_queue per_job gpu_number_limit = limit  queue
```

実行には、操作員権限が必要です。

同時実行 GPU 台数制限値として 0～256 の値または UNLIMITED を指定できます。

また、キューに対して同時実行 GPU 台数の資源既定値を設定するには、qmgr の以下のサブコマンドを使用します。

```
set execution_queue standard per_job gpu_number_limit = limit  queue
```

```
set interactive_queue standard per_job gpu_number_limit = limit  queue
```

なお、同時実行 GPU 台数の資源既定値のデフォルトは 0 です。

他の資源既定値のデフォルト（UNLIMITED）とは異なりますので、ご注意ください。

キューに設定した資源制限値、資源既定値は、qstat(1)コマンドの-Q -f オプションの表示（Resources Limits）で確認することができます。

12.7 VE NUMAモードの自動切り替え

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。

VE を使用するジョブの実行時に、アプリケーションによっては VE のパーティショニングモードを変更して実行することで、アプリケーションの実行性能が向上する場合があります。パーティショニングモード／NUMA モードの詳細は、「SX-Aurora TSUBASA パーティショニングモードのための VEOS NUMA モードガイド」を参照してください。

本機能を使用するには、実行ホスト上のパーティショニングモードを切り替えるためのシェルスクリプト（/opt/nec/nqsv/sbin/venuma_chg.sh）の設定が必要です。このスクリプトは、NQSV のインストール時には/opt/nec/nqsv/sbin/venuma_chg.sh.sample というファイル名になっています。これを /opt/nec/nqsv/sbin/venuma_chg.sh に変更して使用してください。

スクリプト中に、当該運用環境のパーティショニングモードの既定値を設定する DEFAULT（ON か OFF）という変数を設けています。当該運用環境の状況に合わせて DEFAULT に ON または OFF を設定してください。初期値は OFF です。

リクエスト投入時に qsub の--venuma オプションで VE NUMA モードの ON/OFF を指定することで、ジョブ実行時の VE のパーティショニングモードを自動的に切り替えます。パーティショニングモードの切り替えは、ジョブにアサインされた VE に対して切り替えを行います。リクエスト投入時に VE NUMA モードが指定されていない場合は、当該運用環境の既定の設定でジョブを実行します。

qsub の指定イメージは下記の通りです。

```
qsub --venuma={on|off}
```

上記の設定内容は qstat -f コマンドの「VE NUMA Mode」欄で確認できます。なお、投入後に qalter による設定値の変更はできません。また、会話リクエストには対応しておりません。

パーティショニングモードの切り替えに失敗した場合や、切り替えが 90 秒以内に終わらない場合は、リクエスト実行が失敗し、QUEUED 状態に戻ります。パーティショニングモードの切り替えを待機する時間を 90 秒以外に変更したい場合は、JSV 起動時に環境変数 NQSV_VENUMA_TIMEOUT を設定してください。

【注意】

- パーティショニングモードの切り替えには時間がかかります。実際の環境でかかる時間にあわせてチューニングし、その時間を smgr(1M)で投入キューの Elapse マージン時間として設定してく

ださい。Elapse マージンの詳細は、［JobManipulator 編］を参照してください。

- NQSV ではリクエスト開始時のみパーティショニングモードを変更します。リクエスト終了後、パーティショニングモードの設定が既定値と異なる可能性があります。
- 本機能と VE ジョブのサスペンドによる緊急ジョブの実行は同時に利用できません。緊急ジョブの詳細は、「JobManipulator 編」を参照してください。

12.8 マルチインスタンスGPU（MIG）の設定

12.8.1 マルチインスタンス GPU(MIG)について

マルチインスタンス GPU(以降 MIG とする)は、NVIDIA 社製の一部の GPU（A30、A100、H100 等）に搭載された、GPU 内で物理的なリソース分割を行える機能です。

MIG を使用する場合、GPU リソースへのアクセスが従来の GPU と異なるため、qsub オプションの「--gpunum-lhost」や各種 GPU 関連の機能は利用できません。そのため、qsub オプション「--mig」を使用してリソースの割り当てを行います。

12.8.2 MIG の使用方法

例えば NVIDIA 社製 H100 の場合、1 ホストに最大 8 台の GPU が搭載されます。そして GPU 毎に MIG 機能の有効と無効を設定することが可能となっています。

本機能は、下記の条件を全て満たした場合にのみ自動的に有効になります。

- 実行ホスト上に MIG 対応 GPU カードが搭載されている。
- nvidia-smi コマンドがインストールされている。
- MIG の GPU インスタンス(GI)が設定済みである。

各実行ホストの MIG 機能が有効であるかどうかは、qstat -Ef コマンド（実行ホストの情報）の出力に「Multi Instances GPU」が表示されているかどうかで確認できます。

【注意】MIG 機能が有効な場合、当該実行ホストの GI プロファイル名の先頭に"M"を付加したカスタムリソースが作られます。このカスタムリソースは NQSV システムで使用しますので、変更しないでください。

第13章　フックスクリプト機能

フックスクリプト機能とは、リクエストが特定の状態に遷移した際に、バッチサーバホスト上で管理者が定義した任意のスクリプト（フックスクリプトと呼ぶ）を実行する機能です。フックスクリプトは、リクエストが遷移しうるあらゆる状態において定義することができます。

13.1 フックスクリプトの配置

フックスクリプトは、バッチサーバホストの以下のディレクトリにキュー名のディレクトリを作成した上で、その配下に「リクエストの状態」を示す特定の名前で配置します。

```
/opt/nec/nqsv/sbin/hook_prog
```

フックスクリプトの命名規則は以下の通りです。

スクリプトを起動するリクエスト状態	フックスクリプト名
QUEUED	request_queued.sh
WAITING	request_waiting.sh
HELD	request_held.sh
HOLDING	request_holding.sh
SUSPENDING	request_suspending.sh
SUSPENDED	request_suspended.sh
ARRIVING	request_arriving.sh
TRANSITING	request_transiting.sh
STAGING	request_staging.sh
PRE-RUNNING	request_prerunning.sh
RUNNING	request_running.sh
POST-RUNNING	request_postrunning.sh
EXITING	request_exiting.sh
MIGRATING	request_migrating.sh
MOVED	request_moved.sh
EXITED	request_exited.sh
RESUMING	request_resuming.sh

以下は、batch1 キューに投入されたリクエストが RUNNING 状態に遷移した時に実行するフックスクリプトのパス名の例です。

```
/opt/nec/nqsv/sbin/hook_prog/batch1/request_running.sh
```

13.2 フックスクリプトの有効化・無効化の設定

フックスクリプトは、リクエストが投入されたキュー（バッチキュー・会話キュー・転送キュー）において、フックスクリプト機能が有効化されている場合においてのみ実行されます。前述のディレクトリ内にスクリプトを配置しただけでは実行されません。

フックスクリプト機能の有効化は、qmgr(1M)のサブサブコマンドで設定します。本設定には操作員権限が必要です。

キュー	qmgr(1M)サブコマンド
バッチキュー	set execution_queue hook_function
会話キュー	set interactive_queue hook_function
転送キュー	set routing_queue hook_function

以下は、バッチキューbatch1 のフックスクリプト機能を有効化する場合の設定例です。

```
$ qmgr -Po
Mgr: set execution_queue hook_function on batch1
```

フックスクリプト機能の有効化設定結果は、qstat(1)コマンドの-Q -f オプションで以下のように表示されます。

```
$ qstat -Qf
Execution Queue: batch1@bsv1
    Run State    = Inactive
      :
    IntelMPI Process Manager = hydra
    Hook function = ON
:
```

13.3 フックスクリプトの実行

キューのフックスクリプト機能が有効化され、所定ディレクトリにフックスクリプトが配置されている場合、そのキューに投入されたリクエストの状態遷移に応じて、バッチサーバはフックスクリプトを実行します。

フックスクリプト機能はユーザ EXIT 機能とは異なり、リクエストの状態遷移の中でフックスクリプトの実行完了の待ち合わせは行いません。

また、スクリプトの実行結果（終了ステータス）によるリクエストの状態遷移動作の違いもありません。

フックスクリプト実行時には、以下の環境変数がセットされます。

環境変数名	説明	環境変数の値
HOOK_REASON	リクエストの状態遷移理由	RUN DELETE PRERUN_FAIL　等
HOOK_RSTPREV	一つ前のリクエスト状態	QUEUED PRERUNNING RUNNING 等
HOOK_RID	リクエストID	通常リクエストの場合： <シーケンス番号>.<ホスト名> パラメトリックリクエストの場合： <シーケンス番号>[].<ホスト名>
HOOK_USER	リクエストのオーナー名	ユーザ名
HOOK_GROUP	リクエストのオーナーのグループ名	グループ名
HOOK_HOME	リクエストのオーナーのホームディレクトリ	ホームディレクトリの絶対パス名
HOOK_QUEUE	リクエストが投入されたキュー名	キュー名
HOOK_NJOBS	リクエストのジョブ数	リクエストのジョブ数
その他、リクエストに指定された環境変数に従う		

第14章　ユーザプリ・ポストスクリプト機能

ユーザプリ・ポストスクリプト機能とは、ジョブの実行前（PRE-RUNNING）または実行後（POST-RUNNING）に、リクエスト投入時に指定した任意のスクリプト（ユーザ PP スクリプトと呼ぶ）を、実行する機能です。

リクエスト所有者自身がユーザ PP スクリプトを作成、指定することが可能です。本機能を利用して、例えば、ジョブ実行開始前のノード健全性の確認や、ジョブ実行後の実行ホスト上の一時ファイルのクリアなどが可能です。なお、ユーザ PP スクリプトは、資源制限や課金の対象外です。

また、ユーザ PP スクリプトのストールや不正なスクリプトの実行による資源占有を防ぐため、タイムアウト監視が可能です。キューにユーザ PP スクリプトのタイムアウト時間を設定します。ユーザ PP スクリプトが実行を開始してからキューに設定したタイムアウト時間だけ経過した場合、実行を中断します。

14.1 ユーザPPスクリプトのタイムアウト時間設定

ユーザ PP スクリプトのタイムアウト時間を設定し、ストールなどの実行異常が発生した場合に、ユーザ PP スクリプトに KILL シグナルを送信し、その実行を強制終了することができます。

タイムアウト時間の設定は、qmgr(1M)のサブコマンドで行います。

以下に、バッチキューexec1 に対して、ユーザ PP スクリプトのタイムアウト時間を 900 秒に設定する場合の例を示します。このとき、qmgr(1M)の操作員権限で起動してください。

```
$ qmgr -Po
Mgr: set execution_queue userpp_timeout = 900 exec1
Set UserPP Timeout: exec1
```

各設定に使用する qmgr(1M)のサブコマンドと、その既定値は、次のとおりです。

設定	キュー種別	qmgr(1M)のサブコマンド	既定値
ユーザPPスクリプトのタイムアウト時間（秒）	バッチキュー	set execution_queue userpp_timeout	300
	会話キュー	set interactive_queue userpp_timeout	

なお、0 を設定した場合、タイムアウト監視は行いません。

第15章　OpenStack と連携したプロビジョニング環境

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。

NQSV は OpenStack と連携し、実行ホスト内のジョブ実行環境を動的に構成することができます。本機能により、同一ホストの環境（OS のバージョンや導入されているライブラリ、ISV 等）をジョブ毎に必要とするものに動的に切りえて、ジョブを実行することができます。

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、ベアメタルによるプロビジョニングをサポートします。

また、このプロビジョニング環境の OS イメージや資源の情報を、テンプレートとして定義し管理する機能も提供しています。

15.1 仮想マシンによるプロビジョニング環境の設定

NQSV は OpenStack の仮想マシンインスタンス生成機能と連携し、実行ホスト内のジョブ実行環境を動的に構成することができます。

本節では、以下のバージョンの OpenStack を例にとり、仮想マシンインスタンス生成機能との連携について説明します。

・ OpenStack Liberty（2015/10）　　コミュニティ版

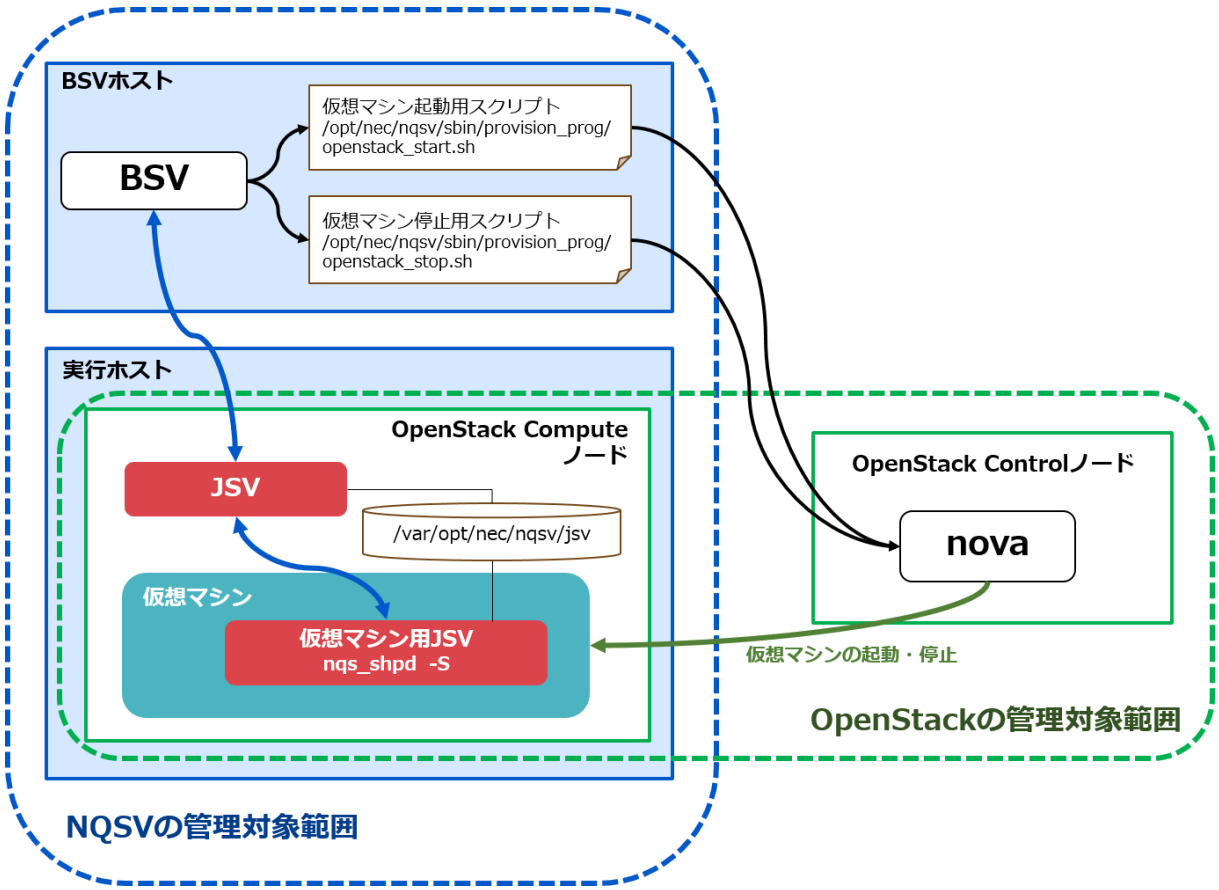


図 15-1：NQSV と OpenStack（仮想マシン）の関係を表す概念図

NQSV の実行ホストを OpenStack の Compute ノードとして構成します。

ジョブの開始時には、BSV から仮想マシン起動用スクリプト実行し、OpenStack の Control ノードに仮想マシンの起動を指示します。

nova コマンドによって仮想マシンが起動し、仮想マシン内の JSV と実行ホスト上の JSV が連係動作することにより、仮想マシン内でジョブを実行します。

ジョブの終了時には、BSV から仮想マシン停止用スクリプト実行し Control ノードに仮想マシンの停止を指示、OpenStack は仮想マシンを停止します。

15.1.1 OpenStack の環境設定

NQSV と連携を行うには、事前に OpenStack を以下に示すように環境構築してください。

(1)　BSVホストからのnovaコマンド利用

BSV ホストから、root ユーザによって OpenStack の Control ノードに対して nova コマンドが利用できるよう構築してください。

(2)　実行ホストをOpenStackのComputeノードに設定

NQSV の実行ホストを、OpenStack の Compute ノードとなるよう設定してください。

(3)　仮想マシンの起動イメージの設定

仮想マシンの起動イメージは以下の条件を満たしてください。

- 仮想マシンの起動イメージに、NQSV/JobServerパッケージをインストールし、OS起動時にsystemdスクリプトによってジョブサーバを nqs_shpd -Sの形式で自動起動するように、設定してください。なお、nqs_shpdのその他のオプション（-hや-n）の指定は不要です。
- 仮想マシン上のOSと実行ホストのOSは、相互に通信が可能な状態としてください。
- 仮想マシン上のOSと実行ホストのOSとで、 JSVDB（/var/opt/nec/nqsv/jsv）を共有してください。
- 各実行ホスト上で起動した仮想マシン間で、相互にホスト名を使用した通信ができる状態としてください。
- 仮想マシン上のOSで、ジョブ投入ユーザのアカウントが利用できるようにしてください。

15.1.2 実行ホスト上のジョブサーバの設定

仮想マシン上で起動したジョブサーバ（JSV）は、実ホスト上の JSV に接続します。この時、仮想マシン上の JSV は、実ホスト上の JSV が自身のホスト名として取得したホスト名を使って接続を行います。

ただし、実ホスト上と仮想マシン上で参照できるホスト名が異なっている場合には、仮想マシン側から見たホスト名を、実行ホスト上の JSV 起動時に-H オプションにて明示的に指定してください。

実行ホスト上の JSV 起動オプション：

```
nqs_shpd -H <vm_host> -h <bsv_host> -n <jsvno>
```

これにより、仮想マシンで参照できるホスト名<vm_host>で、仮想マシン上の JSV から実行ホスト

上の JSV に接続可能となります。

15.1.3 仮想マシン起動用スクリプト

仮想マシンの起動処理は、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。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/provision_prog/openstack_start.sh

NQSV 並びに OpenStack の実行環境に合わせて、以下に記載する処理を行うように、起動用スクリプトを作成してください。

(1)　仮想マシン起動用スクリプトで実施が必要な処理

仮想マシン起動用スクリプトでは以下の 3 つの処理を行ってください。

1)　仮想マシン起動

スクリプトに指定された環境変数に従って、novaコマンドを実行し、仮想マシンを起動します。

2)　仮想マシン起動待ち受け

全ての仮想マシンが完全に起動することを待ち受けます。

3)　仮想マシン情報ファイル（VMIP）の作成

NQS_OPENSTACK_IPINFOPATH環境変数で指定されたファイルに、以下の内容をスペース区切りで、1ジョブに対して1行ずつ記入します。（これをVMIPファイルと呼びます。）

起動できたジョブ番号

仮想マシンに割り当てたIPアドレス

起動した仮想マシンのID

【記入例】

```
0 192.168.1.101 8a850692-0362-425d-b9af-f1cb20a450c0
1 192.168.1.102 9d52de61-a439-46cc-89ee-dd4a66eaafb6
```

以上 3)までのステップが全て完了したら exit 0 で終了します。

また、1)～3)の処理の中で、nova コマンドの実行に失敗するなど、仮想マシンの起動失敗した場合は、NQS_OPENSTACK_FAILINFOPATH 環境変数で指定されたファイルに、該当するジョブ番号を 1 ジョブに対して 1 行ずつ追記します。（これを VMFAIL ファイルと呼びます。）

その後、exit 0 以外で終了します。

(2)　仮想マシン起動用スクリプト実行時の環境変数

仮想マシン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
NQS_OPENSTACK_TEMPLATE_NAME	テンプレート名
NQS_OPENSTACK_IMAGE_NAME	OSイメージ名
NQS_OPENSTACK_CPU	CPU数
NQS_OPENSTACK_GPU	GPU数
NQS_OPENSTACK_MEMORY	メモリサイズ（サイズ+単位　例：100MB）
NQS_OPENSTACK_FLAVOR	フレーバ名
NQS_OPENSTACK_CUSTOM	独自定義
NQS_OPENSTACK_QUE	リクエストが投入されたキュー名
NQS_OPENSTACK_RID	リクエストID 形式： <シーケンス番号>.<mid>の形式 <mid>は数値
NQS_OPENSTACK_IPINFOPATH	VMIPファイルを保存するパス
NQS_OPENSTACK_FAILINFOPATH	VMFAILファイルを保存するパス
NQS_OPENSTACK_PROCTYPE	"EXECUTION"（順方向処理時） "ROLLBACK"（ロールバック時）
NQS_OPENSTACK_HOSTS	起動対象のホスト名とジョブ番号 形式： <jobno>:<hostname>[<jobno>:<hostname> …] ※スペースが区切り文字

(3)　ロールバック処理

仮想マシン起動用スクリプトの実行に失敗した（exit 0 以外で終了）場合、その後にロールバック処理が行えるよう、同スクリプトを再度実行します。順方向処理なのか、ロールバック処理なのかは、環境変数 NQS_ OPENSTACK_PROCTYPE にて切り分けます。

ロールバック処理では、VMIP ファイルに記載した ID の仮想マシンを停止し、全ての仮想マシンが完全に停止することを待ち受けてください。NQSV はロールバック完了後には仮想マシンは全て停止できているものとして後続の処理を行います。また、ロールバック処理において、停止に失敗したホストについては、該当するジョブ番号を VMFAIL ファイルに記載してください。

(4)　タイムアウト

仮想マシン起動用スクリプトの実行時に、タイムアウト監視を行います。タイムアウト時間は、テンプレートに設定した起動見込み時間（Boot Timeout）の値です。

仮想マシン起動用スクリプトの順方向処理において、Boot Timeout の設定時間が経過しても終了しなかった場合、起動用スクリプトの実行に失敗したとみなして、仮想マシン起動用スクリプトの実行を中断（KILL シグナルを送信）し、ロールバック処理を行います。

また、ロールバック処理においても、同じくタイムアウト監視を行います。Boot Timeout の設定時

間が経過しても処理が完了しなかった場合は、仮想マシン起動用スクリプトの実行（ロールバック処理）を中断します。ただし、ロールバック処理をタイムアウトで中断した場合には、仮想マシンが停止できないまま実行ホスト上に存続している可能性があります。

(5)　仮想マシン起動用スクリプト作成時の注意事項

仮想マシン起動用スクリプトにおいて、複数の仮想マシンを立ち上げる処理を行う場合、その立ち上げ失敗を検知した場合の動作として、以下の 2 パターンが考えられます。

どちらのパターンを採用するかによって、その後のスケジューリング動作が異なります。それぞれにメリットとデメリットがありますので、運用に合わせて処理を行うようにしてください。

1)　全ての仮想マシンについて立ち上げ処理を待ち受け、立ち上げ失敗した仮想マシンを全て把握する：

立ち上げ対象となった仮想マシンのうち、立ち上げることができない全ての仮想マシンを把握することができます。そのため、以後のスケジューリングにおいて、該当ホストを避けてアサインするなどのスケジューリング上の対処を行うことができるというメリットがあります。

一方、全ての仮想マシンが完全に立ち上がるまで待ち受ける必要があるため、最初に立ち上げ不可を検知してからそのリクエストの再アサイン対処を行うまでに時間を要するというデメリットがあります。

2)　一つでも立ち上げ失敗の仮想マシンを検知した時点で、立ち上げ処理全体を失敗として処理を中断する：

仮想マシンの立ち上げ不可をすぐに検知できるため、そのリクエストを再アサインする等の対処をすぐにとることができるというメリットがあります。

一方、最初に立ち上げ失敗を検知したもの以外にも立ち上げできない仮想マシンが存在したとしても検知できないため、次回、同ホストにアサインされてそこで再度立ち上げ失敗となる可能性があるというデメリットがあります。

15.1.4 仮想マシン停止用スクリプト

仮想マシンの停止処理は、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。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/provision_prog/openstack_stop.sh

NQSV 並びに OpenStack の実行環境に合わせて、以下に記載する処理を行うように、停止用スクリプトを作成してください。

(1)　仮想マシン停止用スクリプトで実施が必要な処理

仮想マシン停止用スクリプトでは以下の 2 つの処理を行ってください。

1)　仮想マシンの停止

スクリプトに指定された環境変数に従って、nova コマンドを実行し、仮想マシンを停止します。

2)　仮想マシンの停止待ち受け

全ての仮想マシンが完全に停止する（もしくは停止できないことが明確になる）まで待ち受けます。

※停止用スクリプトは実行に失敗しても、ロールバック処理は行いません。

仮想マシンが確実に停止したことを見届けるようにしてください。

NQSV は仮想マシン停止用スクリプトの実行が完了することによって、仮想マシンは完全停止できているものとして後続の処理を行います。

以上 2)までのステップが全て完了したら exit 0 にて終了します。

また、仮想マシンの停止待ち受け中に異常が発生し停止できない仮想マシンがある場合は、NQS_OPENSTACK_FAILINFOPATH 環境変数で指定された VMFAIL ファイルに、該当するジョブ番号を 1 ジョブに対して 1 行ずつ追記し、exit 0 以外で終了します。

(2)　仮想マシン停止用スクリプト実行時の環境変数

仮想マシン停止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
NQS_OPENSTACK_TEMPLATE_NAME	テンプレート名
NQS_OPENSTACK_IMAGE_NAME	OSイメージ名
NQS_OPENSTACK_CPU	CPU数
NQS_OPENSTACK_GPU	GPU数
NQS_OPENSTACK_MEMORY	メモリサイズ（サイズ+単位。　例：100MB）
NQS_OPENSTACK_FLAVOR	フレーバ名
NQS_OPENSTACK_CUSTOM	独自定義
NQS_OPENSTACK_QUE	リクエストが投入されたキュー名
NQS_OPENSTACK_RID	リクエストID 形式：　<シーケンス番号>.<mid>の形式 <mid>は数値
NQS_OPENSTACK_IPINFOPATH	VMIPファイルを保存するパス
NQS_OPENSTACK_FAILINFOPATH	VMFAILファイルを保存するパス
NQS_OPENSTACK_IPADDRS	ジョブ番号と仮想マシンに割り当てたIPアドレスの一覧。形式は下記のとおり。 形式： <jobno>:<ip addr> [<jobno>:<ip addr> …] ※スペースが区切り文字

	<ip addr>はXXX.YYY.ZZZ.NNNの形式 （例：172.16.1.1）
NQS_OPENSTACK_PROCTYPE	"EXECUTION"
NQS_OPENSTACK_HOSTS	停止対象のホスト名とジョブ番号。 形式： <jobno>:<hostname>[<jobno>:<hostname> …] ※スペースが区切り文字

(3)　タイムアウト

仮想マシン停止用スクリプトの実行時にも、タイムアウト監視を行います。タイムアウト時間は、テンプレートに設定した停止見込み時間（Stop Timeout）の値です。
仮想マシン停止用スクリプトの実行が、Stop Timeoutの設定時間が経過しても終了しなかった場合、実行失敗とみなして、仮想マシン停止用スクリプト実行を中断（KILLシグナルを送信）します。
仮想マシン停止用スクリプトをタイムアウトで中断した場合は、仮想マシンが停止できないまま存続している可能性があります。

15.1.5 仮想マシン起動・停止用サンプルスクリプト

仮想マシン起動用スクリプトおよび停止用スクリプトのサンプルがBSV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仮想マシン起動用スクリプトおよび仮想マシン停止用スクリプトスクリプトを作成し、適切なパスに配置してください。

【注意】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OpenStack 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)　仮想マシン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/provision_prog/openstack_start.sh.sample

処理概要：

・　環境変数NQS_OPENSTACK_PROCTYPEがEXECUTIONの場合：

nova bootコマンドにより環境変数NQS_OPENSTACK_HOSTSで指定されたホスト上で仮想マシンを起動し、あらかじめnova floating-ip-listにて取得しておいた空きfloating-ipを割り当てます。nova bootコマンドは、--availability-zoneオプションを使用して仮想マシンを起動するホストを明示的に指定します。

その後、nova boot実行時の起動メッセージより取得したインスタンスIDをnova showコマンドで定期的に監視して、そのステータスがBUILDでなくなるまで待ち受けます。

・　環境変数NQS_OPENSTACK_PROCTYPEがROLLBACKの場合：

環境変数NQS_OPENSTACK_IPINFOPATHで指定されたVMIPファイルより取得したインスタンスIDに対してnova deleteコマンドにて仮想マシンを停止します。

その後、nova showコマンドで該当インスタンスIDの情報が参照できなくなるまで待ち受けます。

(2)　仮想マシン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/provision_prog/openstack_stop.sh.sample

処理概要：

環境変数NQS_OPENSTACK_IPADDRSで指定されたファイルより取得したIPアドレスをもとに、nova listコマンドでインスタンスIDを取得します。そのインスタンスIDに対してnova deleteコマンドにて仮想マシンを停止します。

その後、nova showコマンドで該当インスタンスの情報が参照できなくなるまで待ち受けます。

15.1.6 仮想マシンの NQSV への組み込み

OpenStack の Compute ノードとなるよう設定した仮想マシンを起動する実行ホストのジョブサーバを、キューにバインドすることで、ジョブ運用に組み込みます。

また、当該ジョブサーバをキューからアンバインドすることでジョブ運用から外すことができます。

【注意】仮想マシンを起動する実行ホストをキューにバインドする際、ベアメタルサーバ並びに通常の実行ホストとは混在できません。仮想マシンを起動する実行ホストをバインドするキューは、仮想マシン専用のキューとしてください。。

15.2 ベアメタルによるプロビジョニング環境の設定

NQSV は OpenStack のベアメタルプロビジョニング機能と連携し、ベアメタルサーバを実行ホストとしたジョブ実行環境を動的に構成することができます。

本節では、以下のバージョンの OpenStack を例にとり、ベアメタルプロビジョニング機能との連携について説明します。

・ Red Hat OpenStack Platform (RHOSP) 8

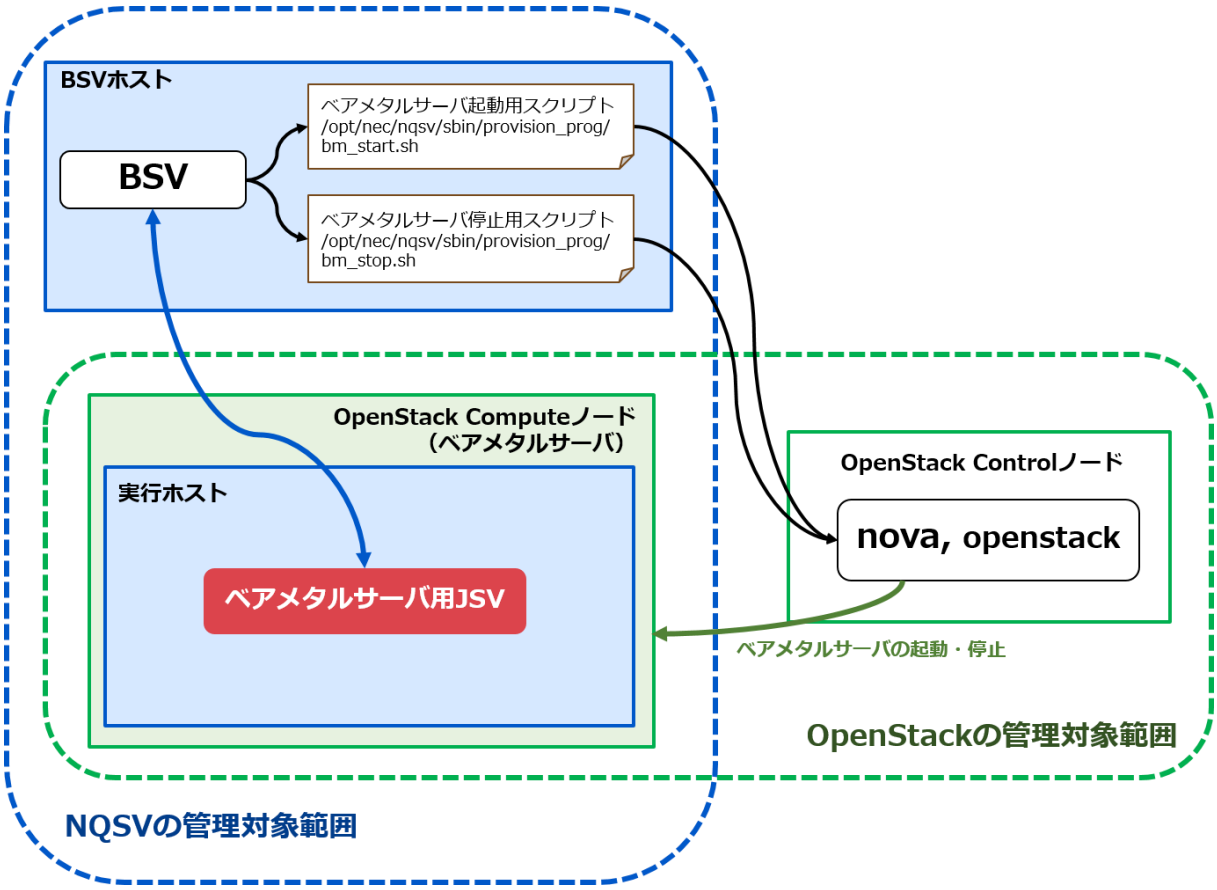


図 15-2：NQSV と OpenStack（ベアメタル）の関係を表す概念図

OpenStack の Compute ノード（ベアメタルサーバ）を、NQSV の実行ホストとして登録します。

ジョブの開始時には、BSV からベアメタルサーバ起動用スクリプト実行し、OpenStack の Control ノードにベアメタルサーバの起動を指示します。nova コマンドによってベアメタルサーバが起動し、NQSV の実行ホストとしてジョブを実行します。

ジョブの終了時には、BSV からベアメタルサーバ停止用スクリプト実行し Control ノードにベアメタルサーバの停止を指示、OpenStack はベアメタルサーバを停止します。

15.2.1 OpenStack の環境設定

NQSV と連携を行うには、事前に OpenStack を以下に示すように環境構築してください。

(1)　BSVホストからのnovaコマンド利用

BSV ホストから、root ユーザによって OpenStack の Control ノードに対して nova, OpenStack コマンドが利用できるよう構築してください。

(2)　ベアメタルサーバの起動イメージの設定

ベアメタルサーバの起動イメージは以下の条件を満たしてください。

ベアメタルサーバの起動イメージに、NQSV/JobServerパッケージをインストールし、OS起動時にsystemdによってジョブサーバを nqs_shpd -B <BSV_host> の形式で自動起動するように、設定してください。なお、nqs_shpdのその他のオプション（-hや-n）の指定は不要です。

各ベアメタルサーバ間で相互にホスト名を使用した通信ができる状態としてください。

各ベアメタルサーバから、BSVホストに対してホスト名を使用した通信ができる状態としてください。

ベアメタルサーバ上のOSで、ジョブ投入ユーザのアカウントが利用できるようにしてください。

15.2.2 ベアメタルサーバ起動用スクリプト

ベアメタルサーバの起動処理は、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。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/provision_prog/bm_start.sh

NQSV 並びに OpenStack の実行環境に合わせて、以下に記載する処理を行うように、起動用スクリプトを作成してください。

(1)　ベアメタルサーバ起動用スクリプトで実施が必要な処理

仮想マシン起動用スクリプトでは以下の 2 つの処理を行ってください。

1)　ベアメタルサーバ起動

スクリプトに指定された環境変数に従って、nova コマンドを実行し、ベアメタルサーバを起動します。

2)　ベアメタルサーバ起動待ち受け

ベアメタルサーバが完全に起動することを待ち受けます。

以上 2)までのステップが全て完了したら exit 0 で終了します。

また、1)～2)の処理の中で、nova コマンドの実行に失敗するなど、ベアメタルサーバの起動失敗した場合は、exit 0 以外で終了します。

(2)　ベアメタルサーバ起動用スクリプト実行時の環境変数

ベアメタルサーバ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
NQS_BM_TEMPLATE_NAME	テンプレート名
NQS_BM_IMAGE_NAME	OSイメージ名
NQS_BM_CPU	CPU数
NQS_BM_GPU	GPU数
NQS_BM_MEMORY	メモリサイズ（サイズ+単位　例：100MB）
NQS_BM_FLAVOR	フレーバ名
NQS_BM_CUSTOM	独自定義
NQS_BM_PROCTYPE	"EXECUTION"（順方向処理時） "ROLLBACK"（ロールバック時）
NQS_ BM _HOSTS	起動対象のホスト名 形式：　<hostname>[<hostname> …] ※スペースが区切り文字

(3)　ロールバック処理

ベアメタルサーバ起動用スクリプトの実行に失敗した（exit 0 以外で終了）場合、その後にロールバック処理が行えるよう、同スクリプトを再度実行します。順方向処理なのか、ロールバック処理なのかは、環境変数 NQS_ OPENSTACK_PROCTYPE にて切り分けます。

ロールバック処理では、NQS_BM_HOSTS 環境変数で指定されたホストを停止し、全てのベアメタルサーバが完全に停止することを待ち受けてください。NQSV はロールバック完了後にはベアメタルサーバは全て停止できているものとして後続の処理を行います。

(4)　タイムアウト

ベアメタルサーバ起動用スクリプトの実行時に、タイムアウト監視を行います。タイムアウト時間は、テンプレートに設定した起動見込み時間（Boot Timeout）の値です。

ベアメタルサーバ起動用スクリプトの順方向処理において、Boot Timeout の設定時間が経過しても終了しなかった場合、起動用スクリプトの実行に失敗したとみなして、ベアメタルサーバ起動用スクリプトの実行を中断（KILL シグナルを送信）し、ロールバック処理を行います。

また、ロールバック処理においても、同じくタイムアウト監視を行います。Boot Timeout の設定時間が経過しても処理が完了しなかった場合は、ベアメタルサーバ起動用スクリプトの実行（ロールバック処理）を中断します。

ただし、ロールバック処理をタイムアウトで中断した場合には、ベアメタルサーバが停止できない

まま存続している可能性があります。

15.2.3 ベアメタルサーバ停止用スクリプト

ベアメタルサーバの停止処理は、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。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/provision_prog/bm_stop.sh

NQSV 並びに OpenStack の実行環境に合わせて、以下に記載する処理を行うように、停止用スクリプトを作成してください。

(1)　ベアメタルサーバ停止用スクリプトで実施が必要な処理

ベアメタルサーバ停止用スクリプトでは以下の 2 つの処理を行ってください。

1)　ベアメタルサーバの停止

スクリプトに指定された環境変数に従って、nova コマンドを実行し、ベアメタルサーバを停止します。

2)　ベアメタルサーバの停止待ち受け

全てのベアメタルサーバが完全に停止する（もしくは停止できないことが明確になる）まで待ち受けます。

※停止用スクリプトは実行に失敗しても、ロールバック処理は行いません。ベアメタルサーバが確実に停止したことを見届けるようにしてください。

NQSV はベアメタルサーバ停止用スクリプトの実行が完了することによって、ベアメタルサーバは完全停止できているものとして後続の処理を行います。

以上 2)までのステップが全て完了したら exit 0 にて終了します。

また、ベアメタルサーバの停止待ち受け中に異常が発生し停止できないベアメタルサーバがある場合は、exit 0 以外で終了します。

(2)　ベアメタルサーバ停止用スクリプト実行時の環境変数

ベアメタルサーバ停止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
NQS_BM_TEMPLATE_NAME	テンプレート名
NQS_BM_IMAGE_NAME	OSイメージ名
NQS_BM_CPU	CPU数
NQS_BM_GPU	GPU数
NQS_BM_MEMORY	メモリサイズ（サイズ+単位　例：100MB）
NQS_BM_FLAVOR	フレーバ名
NQS_BM_CUSTOM	独自定義

NQS_BM_PROCTYPE	"EXECUTION"
NQS_ BM _HOSTS	停止対象のホスト名 形式：　<hostname>[<hostname> …] ※スペースが区切り文字

(3) タイムアウト

ベアメタルサーバ停止用スクリプトの実行時にも、タイムアウト監視を行います。タイムアウト時間は、テンプレートに設定した停止見込み時間（Stop Timeout）の値です。

ベアメタルサーバ停止用スクリプトの実行が、Stop Timeout の設定時間が経過しても終了しなかった場合、実行失敗とみなして、ベアメタルサーバ停止用スクリプト実行を中断（KILL シグナルを送信）します。

ベアメタルサーバ停止用スクリプトをタイムアウトで中断した場合は、ベアメタルサーバが停止できないまま存続している可能性があります。

15.2.4 ベアメタルサーバ起動・停止用サンプルスクリプト

ベアメタルサーバ起動用スクリプトおよび停止用スクリプトのサンプルが BSV 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ベアメタルサーバ起動用スクリプトおよびベアメタルサーバ停止用スクリプトスクリプトを作成し、適切なパスに配置してください。

> **【注意】**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 OpenStack 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) ベアメタルサーバ起動用スクリプトのサンプル

インスールパス：/opt/nec/nqsv/sbin/provision_prog/bm_start.sh.sample

処理概要：

・ 環境変数NQS_BM_PROCTYPEがEXECUTIONの場合：

nova bootコマンドにより環境変数NQS_BM_HOSTSで指定されたベアメタルサーバを起動します。

その後、nova bootコマンド実行時の起動メッセージより取得したインスタンスIDをnova showコマンドで定期的に監視して、そのステータスがBUILDでなくなるまで待ち受け

ます。

・

・　環境変数NQS_BM_PROCTYPEがROLLBACKの場合：

環境変数NQS_BM_HOSTSで指定されたホスト名をもとに、nova listコマンドでインスタンスIDを取得します。そのインスタンスIDに対してnova deleteコマンドにてベアメタルサーバを停止します。

その後、nova showコマンドで該当インスタンスの情報が参照できなくなるまで待ち受けます。

(2)　ベアメタルサーバ停止用スクリプトのサンプル

インスールパス：/opt/nec/nqsv/sbin/provision_prog/bm_stop.sh.sample

処理概要：

環境変数NQS_BM_HOSTSで指定されたホスト名をもとに、nova listコマンドでインスタンスIDを取得します。そのインスタンスIDに対してnova deleteコマンドにてベアメタルサーバを停止します。

その後、nova showコマンドで該当インスタンスの情報が参照できなくなるまで待ち受けます。

15.2.5 ベアメタルサーバの NQSV への組み込み

ベアメタル環境を使用する場合、事前に、ベアメタルサーバを実行ホストとして NQSV に登録する必要があります。

実行ホストとして登録した後は、通常の実行ホストと同様の操作で、キューにバインドしてジョブ運用に組み込むことができます。

(1)　ベアメタルサーバの登録

qmgr(1M)の"attach baremetal_host"サブコマンドを使用してベアメタルサーバを登録してください。登録する際、ホスト名とジョブサーバ番号に加えて、CPU 数、メモリサイズ、GPU 数の情報も合わせて登録する必要があります。

attach baremetal_host サブコマンドの指定（操作には管理者権限が必要です）

```
attach baremetal_host host = <host_name> job_server_id=<jsv_id> cpu = <cpunum> memory=<memsz> gpu=<gpunum>
```

"attach execution_host"サブコマンドと同様に、あらかじめ登録するベアメタルサーバのホスト名、

ジョブサーバ番号、CPU 数、メモリサイズ、GPU 数の情報の一覧ファイルを作成しておき、そのファイルを指定することで、一括してベアメタルサーバを登録することも可能です。

一覧ファイルは、以下のように、ホスト名、ジョブサーバ番号、CPU 数、メモリサイズ、GPU 数を空白文字で区切り、1 行ずつベアメタルサーバの数分記述します。

```
Host1 0 4 10GB 0
Host2 1 4 20GB 1
```

以下のサブコマンドで、上記一覧ファイルを指定してベアメタルサーバを一括登録します。

```
attach baremetal_host file=<file_path>
```

【注意】運用前に、ベアメタルホストで使用するライセンス情報を BSV に登録するため、ベアメタルホスト上で一度 JSV を手動で起動し、LINKUP させてください。
この時、qmgr サブコマンド attach baremetal_host で登録した JSVID を指定して、起動してください。

(2)　ベアメタルサーバの削除

登録済みのベアメタルサーバの削除は、qmgr(1M)の"detach baremetal_host"サブコマンドで行います。ベアメタルサーバの削除は、そのベアメタルサーバのジョブサーバを使用するジョブが存在しない状態で、且つ、ジョブサーバが LINKDOWN している場合のみ可能です。

detach baremetal_host サブコマンドの指定（操作には管理者権限が必要です）

```
detach baremetal_host host = <host_name>
detach baremetal_host host = (<host_name>, <host_name>, …)
detach baremetal_host job_server_id = <jsv_id>
detach baremetal_host job_server_id = (<jsv_id1>,<jsv_id2>, …)
detach baremetal_host job_server_id = < jsv_id1>-< jsv_id2>
detach baremetal_host all
```

(3)　ベアメタルサーバ登録情報の参照

登録したベアメタルサーバの情報は、通常の実行ホストと同様に、qstat -E, -Et で参照できます。ExecutionHost の項目の先頭に "[B]"が表示されているものがベアメタルサーバです。

【表示例】

```
$ qstat -Et
```

```
ExecutionHost   JSVNO JSV      S OS         Release    Hardware    Load   Cpu STT
--------------- ----- -------- - ---------- ---------- ---------- ----- ----- ---
[B]bhost         1000 LINKUP   - Linux      2.6.32-431 x86_64       0.2   0.2 ACT
host1              11 LINKDOWN - --         --         --             -     - INA
```

詳細表示の qstat -Ef, -Etf では、ExecutionHost の項目のホスト名の後ろに"[Baremetal]"が表示されているものがベアメタルサーバです。詳細表示では、ベアメタルサーバ登録時に"attach baremetal_host"サブコマンドで定義したベアメタルサーバの資源量が参照できます（Defined Baremetal Resources)。また、ベアメタルサーバの起動時に使用されたテンプレートも参照できます（OpenStack Template)。

【表示例】

```
$ qstat -Ef
Execution Host: bhost [Baremetal]
        :
    Substitute Status = Normal
    OpenStack Template = Cent7_M
  Defined Baremetal Resources:
    Memory         = 8GB
    Number of Cpus = 2
    Number of GPUs = 1
  Resource Information:
    Memory         = Assign:     254976 Using:          - Maximum:     254976
    Swap           = Assign:     525312 Using:          0 Maximum:     525312
    Number of Cpus = Assign:          8 Using:          1 Maximum:          8
  GPU Information:
    Device[0]: GeForce 9800 GT
    TotalGlobalMem = 511 MB
        :
```

ベアメタルサーバのジョブサーバ情報も、qstat -S, -St, -Sf, -Stf で参照できます。

【表示例】

```
$ qstat -S
JSVNO JobServerName   BatchServer     ExecutionHost   LINK BIND Queue    Jobs  Load   Cpu
----- --------------- --------------- --------------- ---- ---- -------- ---- ----- -----
 1000 JobServer1000   bsvhost         [B]bhost        UP      3 bq,bq2,*    0   0.0   0.0
```

```
$ qstat -Sf
Job Server Name: JobServer1000
    Job Server Number  = 1000
      :
    Execution Host = bhost [Baremetal]
      :
    Assign JobManipulator license = YES
    OpenStack Template = Cent7_M
  Defined Baremetal Resources:
    Memory          = 8GB
    Number of Cpus = 2
    Number of GPUs = 1
  Resource Information:
    Memory          = Assign:     254976 Using:          - Maximum:     254976
      :
```

(4)　ベアメタルサーバのバインド・アンバインド

"attach baremetal_host"サブコマンドで登録したベアメタルサーバのジョブサーバを、キューにバインドすることで、ジョブ運用に組み込みます。なお、スケジューリングに関しては、[JobManipulator 編］を参照してください。

また、当該ジョブサーバをキューからアンバインドすることでジョブ運用から外すことができます。

ベアメタルサーバとキューのバインド・アンバインドの操作は、通常の実行ホストの場合と同様です。

> **【注意】**ベアメタルサーバをキューにバインドする際、仮想マシンを起動する環境の実行ホスト並びに通常の実行ホストとは混在できません。
> ベアメタルサーバをバインドするキューは、ベアメタルサーバ専用のキューとしてください。

(5)　ベアメタルサーバのリセット

一度起動したベアメタルサーバが障害等によって LINKDOWN した場合、当該ベアメタルサーバの復旧が完了した後に、NQSV 上でベアメタルサーバと紐付けられたテンプレートの情報をクリアして運用（スケジューリング）復帰させる必要があります。（本操作を行わない場合、当該ベアメタルサーバが正しくスケジューリングされません。）

ベアメタルサーバと紐付けられたテンプレートの情報をクリアするには、 qmgr(1M)の"reset baremetal_host"サブコマンドを使用します。操作には管理者権限が必要です。

```
reset baremetal_host host = <host_name>
```

本サブコマンドは、対象のベアメタルサーバの JSV が LINKDOWN していて、且つ、同ベアメタルサーバが何れかのテンプレートにて起動されている状態（実行ホスト情報としてテンプレート名が参照できる状態）のままである場合のみ実行することができます。

また、本サブコマンドは対象ホスト上にストール中のジョブがない状態で実行してください。ストール中のジョブがある場合は、まず該当のリクエストをリランまたは削除してください。

15.3 OpenStack用テンプレートの設定

OpenStack と連携したプロビジョニング環境の OS イメージや資源の情報を、OpenStack 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。

ユーザは、リクエスト投入時、投入コマンド（qsub(1), qlogin(1), qrsh(1)）に--template オプションでテンプレートを指定することにより、そのテンプレートが示す環境下でジョブを実行します。

なお、テンプレートの内容は仮想マシン環境およびベアメタル環境で共通です。

15.3.1 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、OpenStack と連携したプロビジョニング環境の情報として、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。 ユーザはリクエスト投入時に、このテンプレート名を指定します。
OSイメージ名 (image)	OpenStackが起動するOSディスクイメージ名。 実行ホストで利用可能なイメージ名を指定します。 最大47文字まで定義可能です。
フレーバ名 (flavor)	OpenStackのフレーバの名前。 最大47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。
起動見込み時間 (boot_timeout)	仮想マシン・ベアメタルサーバ起動用スクリプト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900秒です。 この値は、JobManipulatorがベアメタルサーバ上にジョブをアサインする際にも使用します。
停止見込み時間 (stop_timeout)	仮想マシン・ベアメタルサーバ停止用スクリプト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900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。

【注意】転送キューを利用し他バッチサーバへリクエストを転送する場合、すべてのバッチサーバでテンプレート構成（テンプレート名とその設定値）を同じとしてください。

15.3.2 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 openstack_template"サブコマンドを使用します。

```
create openstack_template=<template_name> image=<OS_image> flavor=<flavor_name>  cpu=<cpunum> memsz=<memory_size>
[gpu=<gpunum>]        [boot_timeout=<timeout>]        [stop_timeout=<timeout>]        [custom=" <custom_define>" ]
[comment=" <comment>" ]
```

※テンプレートに指定する資源量の cpu、memsz、gpu の値は、flavor に指定する OpenStack のフレーバに設定した値と、同じにしてください。

OpenStack の設定のイメージおよびフレーバを以下のように生成した場合を例にとり、テンプレートの作成例を挙げます。

OpenStack の設定：

イメージ名	内容

rhel70	OS に RHEL7 がインストールされている
rhel71	OS に RHEL7.1 がインストールされている

フレーバ名	内容
small	小規模な環境。CPU=1, メモリ=1GB
medium	中規模な環境。CPU=2, メモリ=2GB
large	大規模な環境。CPU=4, メモリ=4GB

NQSV で定義するテンプレート：

上記 OpenStack の構成において、RHEL7（rhel70）では小規模な環境（small）・中規模な環境（medium）を使用し、RHEL7.1（rhel71）では中規模な環境（medium）・大規模な環境（large）を使用する、４種類のテンプレートを定義する場合、操作は以下となります。

```
$ /opt/nec/nqsv/bin/qmgr -Pm
Mgr: create openstack_template=os_70_small image=rhel70 cpu=1 memsz=1gb flavor=small
OpenStack Template os_70_small created.
Mgr: create openstack_template=os_70_medium image=rhel70 cpu=2 memsz=2gb flavor=medium
OpenStack Template os_70_medium created.
Mgr: create openstack_template=os_71_medium image=rhel71 cpu=2 memsz=2gb flavor=medium
OpenStack Template os_71_medium created.
Mgr: create openstack_template=os_71_large image=rhel71 cpu=4 memsz=4gb flavor=large
OpenStack Template os_71_large 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 openstack_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 openstack_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するには、qmgr(1M) を管理者権限で起動し、以下の"set openstack_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編集することができません。

```
set openstack_template image=<OS_image>  <template_name>
set openstack_template cpu=<cpunum>  <template_name>
```

```
set openstack_template memsz=<memory_size> <template_name>
set openstack_template gpu=<gpunum> <template_name>
set openstack_template custom="<custom_define>" <template_name>
set openstack_template comment="<comment>" <template_name>
set openstack_template flavor=<flavor_name> <template_name>
set openstack_template boot_timeout=<timeout> <template_name>
set openstack_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 openstack_template"および"set openstack_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をリクエスト投入時に指定した場合、投入エラーとなります。既に投入済みの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、新たなリクエスト投入でのテンプレートの使用を禁止します。

また、既に対象のテンプレートを使用するリクエストが投入されている場合は、そのリクエストが終了するまで待ちます。

使用中のリクエストが存在しなくなったら、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、以下のサブコマンドを使用します。

```
lock openstack_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、以下のサブコマンドを使用します

```
unlock openstack_template =<template_name>
```

15.3.3 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
[OpenStack Template]
================================================================================
Template   L Image      Flavor CPU  Memory GPU  Custom               Comment
---------- - ---------- ------ ---- ------ ---- -------------------- --------------------
os_70_smal - rhel70     small     1   1.0G    0 (none)               RHEL7 Small.
os_70_medi - rhel70     medium    2   2.0G    0 (none)               (none)
   :
```

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
OpenStack Template: os_70_small
  Lock State   = UNLOCK
  OS Image     = rhel70
  Flavor       = small
  CPU Number   = 1
  Memory Size  = 1GB
  GPU Number   = 0
  Boot Timeout = 900
  Stop Timeout = 900
  Custom       = (none)
  Comment      = RHEL7 Small.
  Requests     = 0
OpenStack Template: os_70_medium
  Lock State   = UNLOCK
    :
```

15.3.4 テンプレート指定によるリクエスト投入とジョブの配置

テンプレートを指定してリクエストを投入するには、qsub(1)、qlogin(1)および qrsh(1)の--template オプションにてテンプレートを指定します。

【例】

```
$qsub --template= os_70_small -q bq -l elapstim_req=1000
```

--template オプションを付けてリクエスト投入した場合、そのリクエストのジョブ毎の資源制限値（CPU 数、メモリサイズ、GPU 数）は、テンプレートで定義された CPU 数、メモリサイズ、GPU 数になります。

これらの値は、投入時にキューの資源制限値（ジョブ毎 CPU 数制限、ジョブ毎メモリサイズ制限、ジョブ毎 GPU 数制限）のチェック対象になりますが、キューのスタンダード値は適用されません。

テンプレート指定されたリクエストが仮想マシンを使用する場合、1 仮想マシンにつき 1 ジョブで実行します。

テンプレート指定されたリクエストがベアメタルサーバを使用する場合、同一リクエストの複数ジョブを 1 つのベアメタルサーバ上で実行することは可能ですが、このような構成は推奨しません。

第16章　Docker と連携したプロビジョニング環境

NQSV はコンテナ型仮想化が可能なソフトウェア Docker と連携し、ジョブを実行ホスト内の隔離したシステム（コンテナ）上で実行できます。

また、このプロビジョニング環境のコンテナイメージや資源情報を、テンプレートとして定義し管理する機能も提供しています。

16.1 Dockerによるプロビジョニング環境の設定

NQSV は Docker のコンテナ生成機能と連携し、実行ホスト内のジョブ実行環境を動的に構成することができます。

本節では、以下のバージョンの Docker を例にとり、コンテナ生成機能との連携について説明します。

- Docker CE Version 24.0.5
- Rocky Linux 8.8

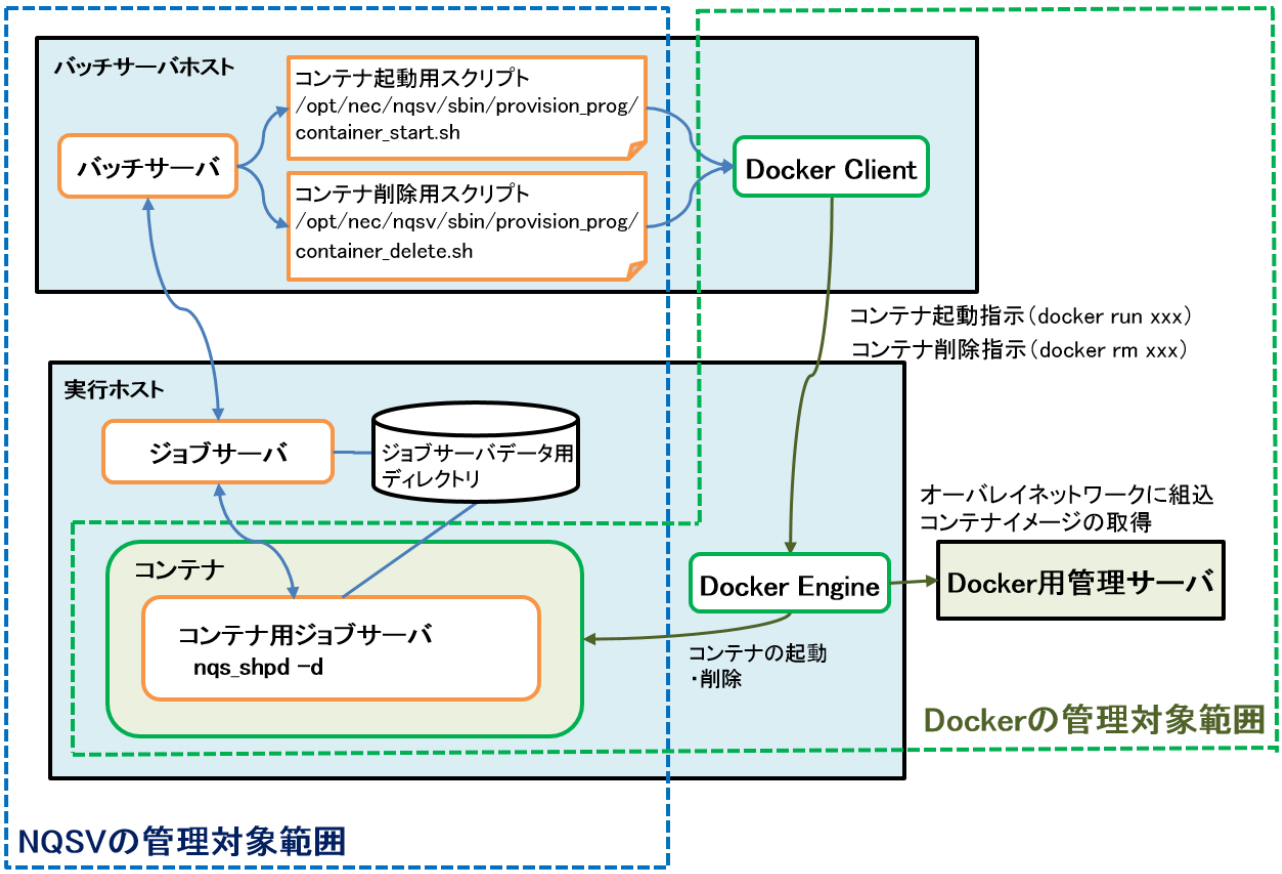


図 16-1：NQSV と Docker の関係を表す概念図

Docker 用管理サーバとは、コンテナのイメージのリポジトリを管理するための Docker Registry および Docker がオーバレイネットワークを構築するための分散 KVS（Key-Value Store）が利用できるマシンです。

NQSV の実行ホストを Docker のコンテナを作成するためのノードとして構成します。

ジョブを開始するとき、バッチサーバはコンテナ起動用スクリプトで Docker Client を実行し、実行ホスト上の Docker Engine に、コンテナ用ジョブサーバを含むテンプレート（コンテナを起動するためのイメージと資源量の情報）でコンテナ起動を指示します。

Docker Engine は指示されたイメージを Docker 用管理サーバから取得し、さらに、Docker 用管理サーバと連携しオーバレイネットワークに組み込みます。

そして、テンプレートで指定された資源量情報で、コンテナを起動し、コンテナ内でジョブを実行します。

ジョブ終了時、バッチサーバはコンテナ用削除スクリプトを実行し、Docker Engine にコンテナ削除を指示し、コンテナを削除します。

16.1.1 Docker の環境設定

NQSV と連携を行うには、事前に Docker を以下に示すように環境構築してください。

(1) バッチサーバホストのDocker環境構築

バッチサーバホストから、Docker Engine に対して、Docker Client が利用できるよう構築してください。

(2) 実行ホストのDocker環境構築

実行ホスト上に、Docker Engine を利用できるよう構築してください。

(3) Docker用管理サーバの構築

- 新たにDocker用管理サーバを用意します。
    - （なお、Docker用管理サーバはバッチサーバホストと共用する構成も可能です。）
- Docker用管理サーバに、コンテナ間ネットワークとしてオーバレイネットワークを使用するために必要な分散KVSを利用できるよう構築してください。
- ジョブ用のオーバレイネットワークをあらかじめ作成してください。
    - （なお、コンテナ起動用スクリプト内で動的に作成しても構いません。）
- Docker用管理サーバに、コンテナを起動するために必要なイメージのリポジトリである

Docker Registryを利用できるよう構築してください。

・ コンテナのイメージをあらかじめ作成し、Docker Registryに登録してください。

(4) コンテナの起動イメージの設定

コンテナの起動イメージは以下の条件を満たしてください。

・ コンテナの起動イメージに、NQSV/JobServerパッケージをインストールし、コンテナ起動時に nqs_shpd -d の形式で自動起動するように設定してください。なお、nqs_shpdのその他のオプション（-hや-n）の指定は不要です。

・ コンテナ内では、実行ホストの JSVDB（/var/opt/nec/nqsv/jsv）を共有してください。

・ Dockerがオーバレイネットワークを利用可能な状態としてください。

・ コンテナのホスト名によるコンテナ間通信（ssh等）が利用できるようにしてください。

・ コンテナ内で、ジョブ実行ユーザのアカウントが利用できるようにしてください。

・ コンテナ内から、NQSVのクライアントホスト、および実行ホストに通信ができるようにしてください。

なお、SX-Aurora TSUBASA システムのコンテナ実行で必要なソフトウェアをインストールし、コンテナ起動時に NQSV/JobServer を起動するイメージを作成する Dockerfile のサンプルを下記に示します。

（※）本サンプルは執筆時点の情報です。本サンプルを参考に Dockerfile を作成する際、パッケージのバージョンについては、最新のバージョンに合わせて編集ください。

Rocky Linux のイメージに、Dockerfile と同じディレクトリに配置している TSUBASA-soft-release-ve1-3.0-1.noarch.rpm、および IB ドライバ（MLNX_OFED_LINUX-23.04-1.1.3.0）をインストールし、インターネット上の yum レポジトリから SX-Aurora TSUBASA システムのソフトウェアをインストールします。

インターネット上からインストールするための設定ファイル（yum.conf, Rocky-BaseOS.repo, TSUBASA-repo.repo, TSUBASA-restricted.repo ） お よ び TSUBASA-soft-release-ve1-3.0-1.noarch.rpm と IB ドライバ（MLNX_OFED_LINUX-23.04-1.1.3.0-rhel8.8-x86_64.tar）は、事前に Dockefile と同じディレクトリに配置してください。

【Dockerfile のサンプル】

```
FROM            docker.io/rockylinux:8.8
MAINTAINER      NEC
```

```
ADD             yum.conf /etc
ADD             Rocky-BaseOS.repo /etc/yum.repos.d
ADD             TSUBASA-soft-release-ve1-3.0-1.noarch.rpm /tmp
ADD             TSUBASA-repo.repo /tmp
ADD             TSUBASA-restricted.repo /tmp
ARG             RELEASE_RPM=/tmp/TSUBASA-soft-release-ve1-3.0-1.noarch.rpm
RUN             yum -y install $RELEASE_RPM && ¥
                cp /tmp/*.repo /etc/yum.repos.d && ¥
                rm /tmp/*.repo && ¥
                yum clean all && ¥
                yum -y group install ve-container && ¥
                yum -y group install nec-sdk-runtime

RUN             yum -y group install nec-mpi-runtime nqsv-execution

RUN             yum -y install perl pciutils gtk2 atk cairo gcc-gfortran tcsh lsof libnl3 libmnl ethtool tcl tk
ADD             MLNX_OFED_LINUX-23.04-1.1.3.0-rhel8.8-x86_64.tar /tmp
RUN             cd /tmp/MLNX_OFED_LINUX-23.04-1.1.3.0-rhel8.8-x86_64 && ¥
                ./mlnxofedinstall --user-space-only --without-fw-update -q --all
RUN             rm -rf /tmp/MLNX*

RUN             yum -y group install ve-container-infiniband

# Enable the next line if you use ScaTeFS
#RUN             yum -y group install scatefs-client-tsubasa-container

COPY            jsvstart.sh /tmp/jsvstart.sh
CMD             ["/tmp/jsvstart.sh"]
```

下記は、ジョブ実行に必要なユーザとグループを動的に生成し、NQSV/JobServer を起動するスクリプトのサンプルです。前述の Dockerfile と同じディレクトリに配置してください。

【jsvstart.sh のサンプル】

```
#!/bin/bash
/usr/sbin/groupadd -g ${NQS_CONTAINER_GID} ${NQS_CONTAINER_GNAME}
/usr/sbin/useradd -g ${NQS_CONTAINER_GID} -d ${NQS_CONTAINER_HOME} -u ${NQS_CONTAINER_UID} ${NQS_CONTAINER_UNAME}
/opt/nec/nqsv/sbin/nqs_shpd -d
```

16.1.2 実行ホスト上のジョブサーバの設定

コンテナ内で起動したジョブサーバは、実行ホスト上のジョブサーバに接続します。この時、コンテナ内のジョブサーバは、実行ホスト上のジョブサーバが自身のホスト名として取得したホスト名を使って接続を行います。

ただし、実行ホスト上とコンテナ内で参照できるホスト名が異なっている場合には、コンテナ内から見たホスト名を、実行ホスト上のジョブサーバ起動時に-H オプションにて明示的に指定してください。

実行ホスト上のジョブサーバ起動オプション：

```
nqs_shpd -H <jhost> -h <bsv_host> -n <jsvno>
```

これにより、コンテナ内で参照できるホスト名<jhost>で、コンテナ内のジョブサーバから実行ホス

ト上のジョブサーバに接続可能となります。

16.1.3 コンテナ起動用スクリプト

コンテナの起動処理は、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/provision_prog/container_start.sh

NQSV 並びに Docker の実行環境に合わせて、以下に記載する処理を行うように、コンテナ起動用スクリプトを作成してください。

(1)　コンテナ起動用スクリプトで実施が必要な処理

コンテナ起動用スクリプトでは以下の 3 つの処理を行ってください。

1)　コンテナ起動

スクリプトに指定された環境変数に従って、docker run コマンドを実行し、コンテナを起動します。

2)　コンテナ起動待ち受け

全てのコンテナが完全に起動することを待ち受けます。

3)　コンテナ情報ファイルの作成

NQS_CONTAINER_INFOPATH 環境変数で指定されたファイルに、以下の内容をスペース区切りで、1 ジョブに対して 1 行ずつ記入します。（これをコンテナ ID ファイルと呼びます。）

起動できたジョブ番号

コンテナのホスト名（※ジョブ間で重複しないようにしてください）

コンテナID

【記入例】

```
0 NQS-0-496-10 56fc8a257a34b92eaeae379bfdf34444693966e99f5f4e451eb11637a8b2a31e
1 NQS-1-496-10 47dd3f398c44a13cfddf451cb3345233724677a56f7f5b362ce32784b9f6b72a
```

以上 3)までのステップが全て完了したら exit 0 で終了します。

また、1)～3)の処理の中で、docker run コマンドの実行に失敗するなど、コンテナの起動失敗した場合は、NQS_CONTAINER_FAILINFOPATH 環境変数で指定されたファイルに、該当するジョブ番号を 1 ジョブに対して 1 行ずつ追記します。（これを FAIL ファイルと呼びます。）

その後、exit 0 以外で終了します。

(2)　コンテナ起動用スクリプト実行時の環境変数

コンテナ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
NQS_CONTAINER_TEMPLATE_NAME	テンプレート名
NQS_CONTAINER_IMAGE_NAME	イメージ名
NQS_CONTAINER_CPU	CPU数
NQS_CONTAINER_GPU	GPU数
NQS_CONTAINER_MEMORY	メモリサイズ（サイズ+単位　例：100MB）
NQS_CONTAINER_CUSTOM	独自定義
NQS_CONTAINER_QUE	リクエストが投入されたキュー名
NQS_CONTAINER_RID	リクエストID 形式：　<シーケンス番号>.<mid>の形式 <mid>は数値
NQS_CONTAINER_INFOPATH	コンテナIDファイルを保存するパス
NQS_CONTAINER_FAILINFOPATH	FAILファイルを保存するパス
NQS_CONTAINER_PROCTYPE	"EXECUTION"（順方向処理時） "ROLLBACK"（ロールバック時）
NQS_CONTAINER_HOSTS	コンテナを起動する実行ホストのホスト名とジョブ番号 形式： <jobno>:<hostname>[<jobno>:<hostname> …] ※スペースが区切り文字
NQS_CONTAINER_CPUSET_FUNC	NQSVのCPUSET機能が有効か無効かを示す "Disable"（無効） "Enable"（有効）
NQS_CONTAINER_CPUSET_CPUS	ジョブに割り当てられたCPUコア番号 （CPUSET機能が有効の場合のみ） 形式： <jobno>:<cpus>[<jobno>:<cpus> …] ※スペースが区切り文字
NQS_CONTAINER_CPUSET_MEMS	ジョブに割り当てられたメモリノード番号 （CPUSET機能が有効の場合のみ） 形式： <jobno>:< mems>[<jobno>:< mems> …] ※スペースが区切り文字
NQS_CONTAINER_ASSIGNED_GPUS	ジョブに割り当てられたGPU番号 （GPU数が1以上の場合のみ） 形式: <jobno>:<gpus>[<jobno>:<gpus> …] ※スペースが区切り文字
NQS_CONTAINER_VE	VE数
NQS_CONTAINER_ASSIGNED_VE_DEVS	ジョブに割り当てられたVE番号 （VE数が1以上の場合のみ） 形式: <jobno>:<ves>[<jobno>:<ves> …] ※スペースが区切り文字
NQS_CONTAINER_ASSIGNED_HCA_DEVS	実行ホスト上のHCAのパス

	形式: <jobno>:<path>[<jobno>:<path> …] ※スペースが区切り文字
NQS_CONTAINER_UNAME	ジョブ実行ユーザ名
NQS_CONTAINER_UID	ジョブ実行ユーザID
NQS_CONTAINER_GNAME	ジョブ実行グループ名
NQS_CONTAINER_GID	ジョブ実行グループID
NQS_CONTAINER_WORKDIR	ジョブを投入したディレクトリ
NQS_CONTAINER_HOME	ジョブ実行ユーザのホームディレクトリ

(3)　ロールバック処理

コンテナ起動用スクリプトの実行に失敗した（exit 0 以外で終了）場合、その後にロールバック処理が行えるよう、同スクリプトを再度実行します。

順方向処理なのか、ロールバック処理なのかは、環境変数 NQS_CONTAINER_PROCTYPE にて切り分けます。

ロールバック処理では、コンテナ ID ファイルに記載したコンテナ ID と一致するコンテナを削除し、全てのコンテナが完全に削除されることを待ち受けてください。

NQSV はロールバック完了後にはコンテナは全て削除されているものとして後続の処理を行います。

また、ロールバック処理において、削除に失敗したコンテナについては、該当するジョブ番号を FAIL ファイルに記載してください。

(4)　タイムアウト

コンテナ起動用スクリプトの実行時に、タイムアウト監視を行います。タイムアウト時間は、テンプレートに設定した起動見込み時間（Boot Timeout）の値です。

コンテナ起動用スクリプトの順方向処理において、Boot Timeout の設定時間が経過しても処理が終了しなかった場合、起動用スクリプトの実行に失敗したとみなして、コンテナ起動用スクリプトの実行を中断（KILL シグナルを送信）し、ロールバック処理を行います。

また、ロールバック処理においても、同じくタイムアウト監視を行います。

Boot Timeout の設定時間が経過しても処理が完了しなかった場合は、コンテナ起動用スクリプトの実行（ロールバック処理）を中断します。

ただし、ロールバック処理をタイムアウトで中断した場合には、コンテナが削除できないまま実行ホスト上に存続している可能性があります。

(5)　コンテナ起動用スクリプト作成時の注意事項

コンテナ起動用スクリプトにおいて、複数のコンテナを立ち上げる処理を行う場合、その立ち上げ失敗を検知した場合の動作として、以下の 2 パターンが考えられます。

どちらのパターンを採用するかによって、その後のスケジューリング動作が異なります。それぞれにメリットとデメリットがありますので、運用に合わせて処理を行うようにしてください。

1)　全てのコンテナが立ち上がるまで待ち受け、立ち上げ失敗したコンテナを全て把握する：
立ち上げ対象となったコンテナのうち、立ち上げることができない全てのコンテナを把握することができます。そのため、以後のスケジューリングにおいて、該当ホストを避けてアサインするなどのスケジューリング上の対処を行うことができるというメリットがあります。
一方、全てのコンテナが完全に立ち上がるまで待ち受ける必要があるため、最初に立ち上げ不可を検知してからそのリクエストの再アサイン対処を行うまでに時間を要するというデメリットがあります。

2)　一つでも立ち上げ失敗のコンテナを検知した時点で、立ち上げ処理全体を失敗として処理を中断する：
コンテナの立ち上げ不可をすぐに検知できるため、そのリクエストを再アサインする等の対処をすぐにとることができるというメリットがあります。
一方、最初に立ち上げ失敗を検知したもの以外にも立ち上げできないコンテナが存在したとしても検知できないため、次回、同ホストにアサインされてそこで再度立ち上げ失敗となる可能性があるというデメリットがあります。

16.1.4 コンテナ削除用スクリプト

コンテナの削除処理は、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。

削除用スクリプト：　/opt/nec/nqsv/sbin/provision_prog/container_delete.sh

NQSV 並びに Docker の実行環境に合わせて、以下に記載する処理を行うように、コンテナ削除用スクリプトを作成してください。

(1)　コンテナ削除用スクリプトで実施が必要な処理

コンテナ削除用スクリプトでは以下の 2 つの処理を行ってください。

1)　コンテナの削除
スクリプトに指定された環境変数に従って、docker rm コマンドを実行し、コンテナを削除します。

2)　コンテナの削除待ち受け
全てのコンテナが完全に削除される（もしくは削除できないことが明確になる）まで待ち受けます。

※削除用スクリプトは実行に失敗しても、ロールバック処理を行いません。コンテナが確実に削除されたことを見届けるようにしてください。NQSV はコンテナ削除用スクリプトの実行が完了することによって、コンテナが完全に削除されているものとして後続の処理を行います。

以上 2)までのステップが全て完了したら exit 0 にて終了します。

また、コンテナが削除できない場合は、NQS_CONTAINER_FAILINFOPATH 環境変数で指定された FAIL ファイルに、該当するジョブ番号を 1 ジョブに対して 1 行ずつ追記し、exit 0 以外で終了します。

(2)　コンテナ削除用スクリプト実行時の環境変数

コンテナ削除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
NQS_CONTAINER_TEMPLATE_NAME	テンプレート名
NQS_CONTAINER_IMAGE_NAME	イメージ名
NQS_CONTAINER_CPU	CPU数
NQS_CONTAINER_GPU	GPU数
NQS_CONTAINER_MEMORY	メモリサイズ（サイズ+単位。　例：100MB）
NQS_CONTAINER_CUSTOM	独自定義
NQS_CONTAINER_QUE	リクエストが投入されたキュー名
NQS_CONTAINER_RID	リクエストID 形式：　<シーケンス番号>.<mid>の形式 <mid>は数値
NQS_CONTAINER_INFOPATH	コンテナIDファイルを保存するパス
NQS_CONTAINER_FAILINFOPATH	FAILファイルを保存するパス
NQS_CONTAINER_IDS	ジョブ番号とコンテナのホスト名の一覧 形式： <jobno>:<container_hostname>[　<jobno>:<container_hostname> …] ※スペースが区切り文字
NQS_CONTAINER_PROCTYPE	"EXECUTION"
NQS_CONTAINER_HOSTS	削除対象のコンテナが存在する実行ホスト名とジョブ番号 形式： <jobno>:<hostname>[<jobno>:<hostname> …] ※スペースが区切り文字

(3)　タイムアウト

コンテナ削除用スクリプトの実行時にも、タイムアウト監視を行います。タイムアウト時間は、テンプレートに設定した停止見込み時間（Stop Timeout）の値です。

コンテナ削除用スクリプトの実行が、Stop Timeout の設定時間が経過しても終了しなかった場合、実行失敗とみなして、コンテナ削除用スクリプトの実行を中断（KILL シグナルを送信）します。

コンテナ削除用スクリプトをタイムアウトで中断した場合は、コンテナが削除できないまま存続している可能性があります。

【注意】コンテナ削除スクリプトの実行に失敗し、ジョブの実行で利用したコンテナが実行ホスト上に残存した場合には、実行ホストにログインし、直接コンテナを削除してください。

16.1.5 コンテナ起動・削除用サンプルスクリプト

コンテナ起動用スクリプトおよびコンテナ削除用スクリプトのサンプルが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コンテナ起動用スクリプトおよびコンテナ削除用スクリプトを作成し、適切なパスに配置してください。

【注意】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 Docker 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) コンテナ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/provision_prog/container_start.sh.sample
処理概要：

・ 環境変数NQS_CONTAINER_PROCTYPEがEXECUTIONの場合：

docker networkコマンドにより、オーバレイネットワークを動的に生成します。

docker runコマンドにより環境変数NQS_CONTAINER_HOSTSで指定された実行ホスト上でコンテナを起動します。docker runコマンドの成否により、コンテナの起動成否を判断します。

・ 環境変数NQS_CONTAINER_PROCTYPEがROLLBACKの場合：

環境変数NQS_CONTAINER_INFOPATHで指定されたコンテナIDファイルより取得したコンテナIDに対してdocker rmコマンドにてコンテナを削除します。

その後、docker networkコマンドにより、動的に生成したオーバレイネットワークを削除します。

(2) コンテナ削除用スクリプトのサンプル

インストールパス：/opt/nec/nqsv/sbin/provision_prog/container_delete.sh.sample
処理概要：

環境変数NQS_CONTAINER_INFOPATHで指定されたファイルより取得したコンテナIDをもとに、docker rmコマンドにてコンテナを削除します。

その後、docker networkコマンドにより、動的に生成したオーバレイネットワークを削除します。

16.1.6 コンテナを起動する実行ホストとキューについての注意事項

コンテナを起動してその中でジョブを実行するリクエストを投入するキューは、コンテナ起動専用のキューとしてください。

そのキューには、Docker Engine によるコンテナが起動できる実行ホストのみをバインドしてください。仮想マシンを起動する環境の実行ホスト、ベアメタルサーバ、並びに通常の実行ホストとは混在できません。

16.2 コンテナ用テンプレートの設定

Docker と連携したプロビジョニング環境のイメージや資源の情報を、コンテナ用テンプレート（以降テンプレートと呼ぶ）として定義します。

ユーザは、リクエスト投入時、投入コマンド（qsub(1), qlogin(1), qrsh(1)）に--template オプションでテンプレートを指定することにより、そのテンプレートが示す環境下でジョブを実行します。

16.2.1 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、Docker と連携したプロビジョニング環境の情報として、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。 ユーザはリクエスト投入時に、このテンプレート名を指定します。
イメージ名 (image)	起動するコンテナのイメージ名。 実行ホストで利用可能なイメージ名を指定します。 最大47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。
HCA数 （hca）	ジョブが利用可能なHCA数（I/O用、MPI用、共通用）。 形式：(<for_io>, <for_mpi>, <for_all>) それぞれ0以上の整数を指定します。HCAの利用が不可である場合は0を指定します。既定値は0です。
起動見込み時間 (boot_timeout)	コンテナ起動用スクリプト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900秒です。
停止見込み時間 (stop_timeout)	コンテナ削除用スクリプト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900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。

【注意】転送キューを利用し他バッチサーバへリクエストを転送する場合、すべてのバッチサーバでテンプレート構成（テンプレート名とその設定値）を同じとしてください。

16.2.2 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 container_template"サブコマンドを使用します。

```
create container_template=<template_name> image=<image> cpu=<cpunum> memsz=<memory_size> [gpu=<gpunum>]
[boot_timeout=<timeout>] [stop_timeout=<timeout>] [custom="<custom_define>"] [comment="<comment>"]
```

App_A 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。

・ 起動するコンテナイメージ： App_A_Img

・　割り当てCPU数：　2

・　割り当てメモリ　：　1GB

・　コメント：　For App_A

```
$ /opt/nec/nqsv/bin/qmgr -Pm
Mgr: create container_template=App_A image=App_A_Img cpu=2 memsz=1gb comment="For App_A"
Container Template App_A 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 container_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 container_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 container_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編集することができません。

```
set container_template image=<image>  <template_name>
set container_template cpu=<cpunum>  <template_name>
set container_template memsz=<memory_size>  <template_name>
set container_template gpu=<gpunum>  <template_name>
set container_template custom="<custom_define>"  <template_name>
set container_template comment="<comment>"  <template_name>
set container_template boot_timeout=<timeout>  <template_name>
set container_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 container_template"および"set container_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をリクエスト投入時に指定した場合、投入エラーとなります。既に投入済みの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、新たなリクエスト投入でのテンプレートの使用を禁止します。

また、既に対象のテンプレートを使用するリクエストが投入されている場合は、そのリクエストが終了するまで待ちます。使用中のリクエストが存在しなくなったら、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 container_template"サブコマンドを使用します。

```
lock container_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 container_template"サブコマンドを使用します

```
unlock container_template =<template_name>
```

16.2.3 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
[Container Template]
================================================================================
Template   L Image      CPU  Memory GPU  Custom               Comment
---------- - ---------- ---- ------ ---- -------------------- --------------------
App_A      - App_A_Img     2   1.0G    0 (none)               For App_A
```

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
Container Template: App_A
  Lock State   = UNLOCK
  Image        = App_A_Img
  CPU Number   = 2
  Memory Size  = 1GB
  GPU Number   = 0
```

```
Boot Timeout = 900
Stop Timeout = 900
Custom       = (none)
Comment      = For App_A
Requests     = 5
```

16.2.4 テンプレート指定によるリクエスト投入とジョブの配置

テンプレートを指定してリクエストを投入するには、qsub(1)、qlogin(1)および qrsh(1)の--template オプションにてテンプレートを指定します。

【例】

```
$qsub --template=App_A -q bq -l elapstim_req=1000
```

--template オプションを付けてリクエスト投入した場合、そのリクエストのジョブ毎の資源制限値（CPU 数、メモリサイズ、GPU 数）は、テンプレートで定義された CPU 数、メモリサイズ、GPU 数になります。これらの値は、投入時にキューの資源制限値（ジョブ毎 CPU 数制限、ジョブ毎メモリサイズ制限、ジョブ毎 GPU 数制限）のチェック対象になりますが、キューのスタンダード値は適用されません。

コンテナ用テンプレートを指定されたリクエストは、1 コンテナにつき 1 ジョブで実行します。

第17章　カスタムリソース機能

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、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。
また、カスタムリソースの実績値を採取するかどうか、採取する場合には採取種別（瞬間値か積算値）と、実績値が制限値を超過した場合のジョブの打ち切りの有無を定義することができます。

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。
スケジューラは、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ようにスケジューリングします。
スケジューラの詳細については、［JobManipulator 編］を参照してください。
カスタムリソースのアカウンティング・予算管理については、［アカウンティング・予算管理編］を参照してください。

また、キュー毎にリクエストに指定する利用量についての既定値や制限値等の設定を保持します。これをキューのカスタムリソース利用量制限情報と呼びます。
バッチサーバは、リクエスト投入受付時、このカスタムリソース利用量制限情報を参照して、投入可否を判定し、必要に応じて既定値を適用します。

> **注意】**カスタムリソース機能が適用されるリクエストは、バッチリクエスト（ローカルリクエスト）と会話リクエストのみです。また、キューのカスタムリソース利用量制限情報が設定されるのは、バッチキューと会話キューのみとなります。

カスタムリソースの実績値を採取するように設定した場合、リクエスト実行中に定期的にそのリソースの使用量を採取し、設定に応じて実績値が制限値を超過した場合に実行を打ち切ることができます。

17.1 カスタムリソース情報

17.1.1 カスタムリソース情報

カスタムリソース情報は、任意の仮想的なリソースで、最大で 20 件まで定義できます。
カスタムリソース情報には、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

の範囲や上限値を設定します。詳細は下記の通りです。

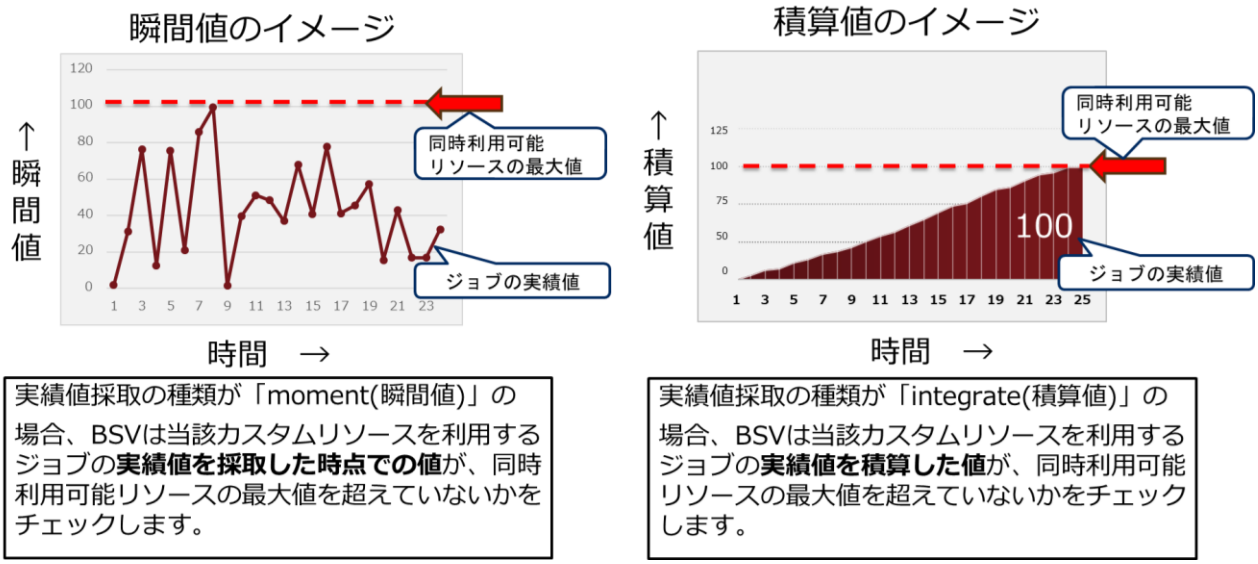
要素名		説明
カスタムリソース名		カスタムリソースの名前。 　　最大15文字までの任意の名前を指定できます。 　　名前に、空白、並びに、「"(ダブルクォーテーション)」、「@」を含むことはできません。
消費単位		リクエスト投入時に指定したカスタムリソースを消費する単位。 以下のいずれかを選択します。 **job**　　：ジョブ単位 **request**　：リクエスト単位
利用量制御情報	利用量制御の対象種別	カスタムリソースの利用量制御の対象種別。 **bsv** 　　利用量制御の対象をBSVとします。 　　BSV全体で同時に利用可能なリソースの最大値を設定し、それに収まるようにスケジューリングします。 **host** 　　利用量制御の対象を実行ホストとします。 　　１実行ホスト当たりで同時に 利用可能なリソースの最大値を設定し、それに収まるようにスケジューリングします。 　　ただし、hostの指定ができるのは、カスタムリソースの消費単位がjobの場合のみです。requestの場合は指定できません。
	利用量制御の対象	利用量制御の対象。 同時利用可能リソースの最大値の適用範囲を示します。 利用量制御の対象種別がbsvの場合： 　　BSV全体となります。 利用量制御の対象種別がhostの場合： 　　１実行ホスト当たり（既定値）、または、実行ホスト名による個別指定となります。
	同時利用可能リソースの最大値	上記、対象種別および対象で示される範囲でのカスタムリソースの同時に利用可能なリソースの最大値。 0～2147483647の整数で指定します。
実績値採取の種類		カスタムリソースの実績値採取の有無および種類（瞬間値か積算値）。瞬間値と積算値の違いは課金処理において考慮されます。 **off** 　　カスタムリソースの実績値を採取しません **moment** 　　カスタムリソースの実績値を採取し、採取した値を瞬間値として扱います。 **integrate** 　　カスタムリソースの実績値を採取し、採取した値を積算値として扱います。
制限超過時の強制停止		採取した実績値が制限値を超過していた場合にジョブの実行を強制停止する機能のon / offを設定。 　　実績値採取がoffの場合は使用することができません。
単位		カスタムリソースの単位。 　　任意の5文字以内の単位が指定可能です。指定しない場合は無名数

	となります。設定された単位の整合についてNQSVでは関知しません。（実績値が瞬間値で単位が積算値のような組み合わせでもエラーになりません）

消費単位は、job または request のいずれか一方を選択しなければなりません。

利用量制御情報については、利用量制御の対象種別、利用量制御の対象、同時利用可能リソースの最大値の３つで構成され、１つのカスタムリソース定義で複数定義することできます。

実績値採取の種類に瞬間値、または積算値と指定した場合に、BSV が採取した実績値をどのように扱うかのイメージは下記の様になります。



17.1.2 カスタムリソース情報の定義・削除

カスタムリソース情報の定義・削除の操作は、qmgr(1M)のサブコマンドで行います。カスタムリソース情報の各操作には、管理者権限が必要です。

また､バッチサーバ上にリクエストが存在しない時のみ実行可能です｡リクエストが存在する場合、カスタムリソース情報を操作するサブコマンドは実行できません。

以下、qmgr のサブコマンドによるカスタムリソース情報の各操作の詳細です。

(1)　　カスタムリソース情報の作成

カスタムリソース情報の作成は、"create custom_resource"サブコマンドで行います。

以下は、"Power"という名前で、ジョブ毎にリソースを消費するカスタムリソース情報を作成し、同時に利用量制御情報として BSV 全体での最大値を 2000 と設定する場合の例です。

```
$ qmgr -Pm
Mgr: create custom_resource=Power consumer=job type=bsv available=2000
```

```
Custom_resoure Power created.
```

カスタムリソース作成時、リクエストがカスタムリソースを消費する単位として、job または request のいずれかを選択しなければなりません。

利用量制御情報の type および available の指定を省略して、カスタムリソースを作成することも可能です。

```
$ qmgr -Pm
Mgr: create custom_resource=License consumer=job
Custom_resoure License created.
```

実績値採取の種類 check_mode、制限超過時の強制停止 terminate_job、単位 unit をカスタムリソース作成時に同時に指定することが可能です。指定しなかった場合はデフォルトの値（check_mode=off, terminate_job=off, unit=指定無し）が設定されます。

```
$ qmgr -Pm
Mgr: create custom_resource=Power consumer=job type=bsv available=2000 check_mode=moment terminate_job=on
unit=volt
Custom_resoure Power created.
```

利用量制御情報は、"edit custom_resource add"サブコマンドにて追加することができます。(詳細は、(6)カスタムリソースの利用量制御情報の追加 参照)

【注意】利用量制御情報の type および available の指定を省略してカスタムリソースを作成し、利用量制御情報が設定されていない場合、このカスタムリソースに関してはスケジューラによる利用量制御は行われません。

(2)　カスタムリソースを消費する単位の変更

カスタムリソース情報作成時に指定したカスタムリソースの消費単位を変更するには、"set custom_resource consumer"サブコマンドを使用します。

【例】

```
$ qmgr -Pm
Mgr: set custom_resource consumer=request License
Set Consumer to Custom_resoure (License).
```

ただし、job から request に変更する場合、既に host タイプの利用量制御情報が登録されている時は変更できません。その場合は、先に host タイプの利用量制御情報を削除してください。

(3)　実績値採取設定の変更

カスタムリソースの実績値採取の設定を変更するには、"set custom_resource check_mode"サブコマンドを使用します。

【例】

```
$ qmgr -Pm
Mgr: set custom_resource check_mode=moment Power
Set check_mode to Custom_resoure (Power).
```

ただし、check_mode を off に変更する場合、terminate_job が on に設定されているときは変更できません。その場合は、先に terminate_job を off に設定してください。

(4)　制限超過時の強制停止の変更

カスタムリソースの制限超過時の強制停止に関する設定を変更するには、"set custom_resource terminate_job"サブコマンドを使用します。

【例】

```
$ qmgr -Pm
Mgr: set custom_resource terminate_job=on Power
Set terminate_job to Custom_resoure (Power).
```

ただし、terminate_job を on に変更する場合、check_mode が off に設定されているときは変更できません。その場合は、先に terminate_job を moment または integrate に設定してください。

(5)　単位の変更

カスタムリソースの単位の設定を変更するには、"set custom_resource unit"サブコマンドを使用します。

【例】

```
$ qmgr -Pm
Mgr: set custom_resource unit=volt Power
Set unit to Custom_resoure (Power).
```

unit には 5 文字以下の任意の文字列が指定できます。

(6)　カスタムリソースの利用量制御情報の追加

カスタムリソースの利用量制御情報の追加は、"edit custom_resource add"サブコマンドで行います。利用量制御情報の対象種別（type）および対象（target）を指定して、同時に利用可能なリソースの最大値（Available）を設定します。なお、既に設定されているものについては上書きとなります。

以下は、"Power"という名前のカスタムリソース情報に、

- BSV全体での最大値を5000
- １実行ホスト当たりの最大値を100
- 実行ホストhost_aに限り最大値120

と設定する例です。

```
$ qmgr -Pm
Mgr: edit custom_resource add type=bsv available=5000 Power
Add Available_info from Custom_resource (Power).
Mgr: edit custom_resource add type=host available=100 Power
Add Available_info from Custom_resource (Power).
Mgr: edit custom_resource add type=host target=host_a available=120 Power
Add Available_info from Custom_resource (Power).
```

対象種別 type=bsv と指定した場合、BSV 全体の最大値の設定となります。

対象種別 type=host と指定した場合、1 実行ホスト当たりの最大値の設定となります。

なお、対象種別が host の時は、target で個別に実行ホスト名を指定し、最大値をすることが可能です。

(7)　カスタムリソースの利用量制御情報の削除

カスタムリソースの利用量制御情報の削除は、"edit custom_resource delete"サブコマンドで行います。

利用量制御情報の対象種別（type）および対象（target）を指定して削除します。

以下は、"Power"という名前のカスタムリソース情報の利用量制御情報の内、実行ホスト host_a についての情報を削除する例です。

```
$ qmgr -Pm
Mgr: edit custom_resource delete type=host target=host_a Power
Deleted Available_info from Custom_resource (Power).
```

対象種別 type=host の場合、対象（target）の指定を省略すると、対象種別 host の情報を全て一括削除します。（つまり、1 実行ホスト当たりの最大値と、実行ホスト名指定の最大値の全てを削除します。）

(8)　カスタムリソース情報の削除

カスタムリソース情報の削除は、"delete custom_resource"サブコマンドで行います。

```
$ qmgr -Pm
Mgr: delete custom_resource=Power
Custom_resource Power deleted.
```

17.1.3 カスタムリソース情報の表示

カスタムリソース情報は、qstat(1)コマンドの--custom オプションで表示します。

【例】

```
$ qstat --custom
Custom Resource : Power
  Consumer = job
  Check Mode = Integrate
  Terminate Job = On
  Unit = MW
  Type = bsv :                          Available Resource Limit = 5000
  Type = host: Target = (default)       Available Resource Limit = 100
               Target = host_a          Available Resource Limit = 120

Custom Resource : License
  Consumer  = request
  Check Mode = Integrate
  Terminate Job = Off
  Unit = (none)
  Type = bsv :                          Available Resource Limit = 50

Custom Resource : Virtual
  Consumer = job
  Check Mode = moment
  Terminate Job = Off
  Unit = (none)
  Type = bsv :                          Available Resource Limit = 100
```

```
Type = host: Target = (default)        Available Resource Limit = 1
```

17.2 キューのカスタムリソース利用量制限情報

17.2.1 キュー毎のカスタムリソース利用量制限情報

カスタムリソース情報を定義した場合、キュー毎にカスタムリソース利用量制限情報として、リクエストに指定する利用量についての既定値、指定範囲、利用量制御対象外指定の可否の設定を保持します。

バッチサーバは、リクエスト投入受付時、このカスタムリソース利用量制限情報を参照して、投入可否を判定し、必要に応じて既定値を適用します。

キューのカスタムリソース利用量制限情報の詳細は、以下です。

要素名	説明
カスタムリソース名	カスタムリソース情報で定義されたカスタムリソースの名前。
リクエストでの利用量の既定値	リクエストに指定する利用量の既定値。 リクエスト投入時に指定されなかった場合、この値を適用します。 以下の指定が可能です。 1～2147483647の整数 unused （または0）　　<既定値> ※指定のカスタムリソースを使用しない（利用量0）の意で、そのカスタムリソースについては利用量制御の対象外となります。
リクエストでの利用量の指定範囲	リクエストに指定するカスタムリソースの利用量の下限値と上限値。 1～2147483647の整数が指定できます。
利用量制御対象外指定の可否	リクエスト投入時、利用量制御対象外指定を許可するか否かの設定。 yes　利用量制御対象外指定（unusedまたは0）を許可します （既定値） no　利用量制御対象外指定は不可です

17.2.2 キューのカスタムリソース利用量制限情報の設定

キューのカスタムリソース利用量制限情報の設定は、qmgr(1M)のサブコマンドで行います。キューのカスタムリソース利用量制限情報の設定には、操作員権限が必要です。

キューのカスタムリソース利用量制限情報の各設定に使用する qmgr(1M)のサブコマンドと、その既定値は、次のとおりです。

設定	qmgr(1M)のサブコマンド	既定値
リクエストでの利用量の既定値	set execution_queue custom_resource =*cr_name* standard=*std queue* set interactive_queue custom_resource =*cr_name* standard=*std queue*	unused (0)
リクエストでの	set execution_queue custom_resource =*cr_name*	(1, 2147483847)

利用量の指定範囲	range=(*min,max*)　*queue* set　interactive_queue　custom_resource　=*cr_name* range=(*min,max*)　*queue*	
利用量制御対象指定の可否	set　execution_queue　custom_resource　=*cr_name* permit_unused={ yes | no }　*queue* set　interactive_queue　custom_resource　=*cr_name* permit_unused={ yes | no }　*queue*	yes ※利用量制御対象外指定可

なお、カスタムリソース利用量制限情報の 3 種の設定を 1 行で指定することも可能です。

以下は、"Power"という名前のカスタムリソース情報について、バッチキューbq での利用量制限情報として、既定値 30、範囲指定（10,50）、利用量制御対象外指定は不可を設定する例です。

```
$ qmgr -Po
Mgr: set execution_queue custom_resource=Power standard=30 range=(10,50) permit_unused=no bq
Set Custom_resource_info (Power). queue: bq
```

17.2.3 キューのカスタムリソース利用量制限情報の表示

キューのカスタムリソース利用量制限情報は、qstat(1)コマンドの-Qfオプションの表示（Custom Resources）で出力されます。

【例】

```
$ qstat -Qf
Execution Queue: bq@bsv
    Run State = Active
    Submit State = Enable
      :
  UserExit Script:
    (none)
  Custom Resources:
    Power          : Range (min,max) = 10,50       Std = 30         : Permit Unused = No
    License        : Range (min,max) = 1,20        Std = unused     : Permit Unused = Yes
    Virtual        : Range (min,max) = 1,1         Std = 1          : Permit Unused = No
  Resources Limits:
    (Per-Req) Elapse Time Limit       = Max: UNLIMITED Warn: UNLIMITED Std:     3600S
    (Per-Job) CPU Time                = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
      :
  Kernel Parameter:
    Resource Sharing Group     = 0
    Nice Value                 = 0
      :
```

17.3 カスタムリソース機能使用時のリクエスト

カスタムリソース機能を使用している環境では、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 オプションを使用して各カスタムリソース名と利用量を指定します。また、投入コマンドでの指定がない場合は、キューに設定している各カスタムリソースの利用量の既定値を適用します。

スケジューラは、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ようにスケジューリングします。

リクエストのカスタムリソースの利用量の情報については、ジョブの環境変数に、以下のように設定します。2 種類の命名規則がありますが、どちらも同じ内容です。

```
NQSV_CR_＜カスタムリソース名＞=＜利用量＞
```

および

```
NQSII_CR_＜カスタムリソース名＞=＜利用量＞
```

【環境変数の設定例】

```
NQSV_CR_Power = 20
NQSV_CR_License = unused
NQSV_CR_Virtual = 1
```

また、カスタムリソースに設定されている単位は以下の環境変数に設定します。2 種類の命名規則がありますが、どちらも同じ内容です。単位が設定されていない場合、<単位>には空文字が入ります。

```
NQSV_CR_UNIT_＜カスタムリソース名＞=＜単位＞
```

および

```
NQSII_CR_UNIT_＜カスタムリソース名＞=＜単位＞
```

なお、この環境変数は、以下のユーザ定義処理スクリプトにおいても、参照できます。

- ユーザEXITスクリプト （詳細は 5.1.2.ユーザEXIT 参照）
- フックスクリプト （詳細は 14.フックスクリプト機能 参照）
- ユーザPPスクリプト（詳細は 15.ユーザプリ・ポストスクリプト機能 参照）

17.4 資源監視スクリプト

資源監視及び使用量出力モードに moment または integrate を設定した場合、資源監視スクリプトによって監視を行います。資源監視スクリプトはカスタムリソースごとに作成し、/opt/nec/nqsv/sbin/custom_prog配下に格納し、スクリプト名はカスタムリソース名と同じにします。例えば、Power というカスタムリソースで資源監視を行う場合、/opt/nec/nqsv/sbin/custom_prog/Power という資源監視スクリプトを格納します。
資源監視スクリプトには下記の環境変数が渡されます。

環境変数名	値	説明
CR_ASSIGNED_VE	0～2^{31}-1	割り当て VE 番号
CR_ASSIGNED_CORE	0～2^{31}-1	割り当てコア番号（※）
CR_NUMA_NODE	0～2^{31}-1	割り当て NUMA ノード番号（※）
CR_<カスタムリソース名>	0～2^{31}-1	カスタムリソース制限値
CR_UNIT_<カスタムリソース名>	文字列	カスタムリソースの単位
CR_CPUNUM	0～2^{31}-1	ジョブ毎の CPU 台数制限値
CR_GPUNUM	0～256	ジョブ毎の GPU 台数制限値
CR_VENUM	0～256	ジョブ毎の VE 台数制限値

環境変数名	値	説明
CR_JOBID	<jobno>:<seqno>.<hosts>	監視対象のジョブ ID
CR_TMPDIR	ディレクトリパス	情報の一時保存用ディレクトリ
CR_EJID	1～プロセス ID 最大値	ジョブのセッション ID

※ソケットスケジューリング値が有効の場合のみ。

資源監視スクリプトは実行ホストに転送され、定期的に root ユーザーで実行されます。取得した実績値は標準出力に書き込みます。この時、終了ステータスが 0 以外の場合は、監視失敗として扱い、ジョブの実行はそこで打ち切られます。

資源管理スクリプトは管理者が用意します。

資源管理スクリプトの例として、VH の I/O 量を採取する例と VI の消費電力を採取する例を示します。

・VH の I/O 量の採取例

VH 上で実行中ジョブ（プロセス）の I/O 統計情報（読み出し文字数、書き込み文字数）は、/proc/<PIDs>/io に記録されます。下記スクリプトは、ジョブ内のプロセスの I/O 統計情報を参照し、読み出しバイトと書き込みバイトの合計値（KiB）を採取します。なお、本スクリプトは、VE からの Direct 通信による I/O 量は採取できません。

```
#!/bin/bash
SAVED_DIR=${CR_TMPDIR}/ioacct
SID=${CR_EJID}
declare -A PID_HASH


if [ ! -e ${SAVED_DIR} ];then
    mkdir -p ${SAVED_DIR}
else
    for fl in `ls ${SAVED_DIR}`;
    do
        PID_HASH[${fl}]=`cat ${SAVED_DIR}/${fl}`
        rm ${SAVED_DIR}/${fl}
    done
fi


PIDS=`ps -s ${SID} -o pid`
IO=0
for pid in ${PIDS};
do
    IOFL="/proc/${pid}/io";
    if [ -e ${IOFL} ];then
        RCHAR=`cat ${IOFL} | grep rchar | awk -F ':' '{print $2}'`;
        WCHAR=`cat ${IOFL} | grep wchar | awk -F ':' '{print $2}'`;
```

```
        # B to KiB
        CHAR=$((${RCHAR}/1024+${WCHAR}/1024))
        IO=$((${IO}+${CHAR}))

        if [ -n "PID_HASH[${pid}]" ];then
            PID_HASH[${pid}]=0
            echo ${CHAR} > ${SAVED_DIR}/${pid}
        fi
    fi
done
ACC=${PID_HASH["ACC"]}
PID_HASH["ACC"]=0
for val in ${PID_HASH[@]};
do
    ACC=$((${ACC}+${val}))
done

echo ${ACC} > ${SAVED_DIR}/ACC
IO=$((${IO}+${ACC}))
echo ${IO}
exit 0
```

・消費電力の採取例

インテリジェントな PDU を搭載したラックでは、SNMP を介することで、リモートで消費電力を採取することができます。下記は、実行ホストからインテリジェントな PDU に snmpwalk コマンドでアクセスし、実行ホスト全体の消費電力を取得するスクリプト例です。

本スクリプトを利用する際は、下記の変数を設定ください。

変数名　　：設定内容

・PDU_IP　：PDU の IP アドレスを指定してください。

・OUTLETS ：電源が接続されているコンセントの番号を指定してください。

・PWR_MIB：ご利用の PDU の MIB（Management Information Base）ファイルに記載された、消費電力が採取可能な Object ID を指定してください。

なお、本スクリプトは実行ホスト共通のスクリプトであるため、PDU_IP や OUTLETS は実行ホスト上のファイルを参照するなど、実行ホストごとに可変となるようにしてください。また、本スクリプトを実行する際は、1 ジョブが実行ホストを占有するようにしてください。

```
#!/bin/bash

SNMPBIN=/usr/bin/snmpwalk

PDU_IP=<IP_Address of PDU>

OUTLETS=<Outlet_numbers>

SUM=0

for outlet in ${OUTLETS};

do

    PWR_MIB=".1.3.6.1.4.1.13742.6.5.4.3.1.6.1.${outlet}.6"

    POWER=`${SNMPBIN} -v 2c -c public ${PDU_IP} ${PWR_MIB} | awk -F ':' '{print
$4}'`

    if [ $? -ne 0 ]; then

        exit 1

    fi

    SUM=$((${SUM} + ${POWER}))

done

echo $SUM

exit 0
```

第18章　ソケットスケジューリング

NUMA アーキテクチャのスカラーマシン（Linux）を実行ホストとして使用する場合、ジョブに対して、最適なリソース（CPU 数、メモリ）の割り当てを行うこと（ソケットスケジューリング）ができます。 また、Linux の CPUSET 機能と連携して、リソース分割することも可能です。

18.1 ソケットスケジューリング機能

ソケットスケジューリング機能とは、ジョブに対して、最適なリソース（CPU 数、メモリ）の割り当てを行う機能です。CPU はコア単位で、メモリはソケット単位での割り当てを行います。

ソケットスケジューリング機能の利用は、キュー単位で有効化することができます。

有効化されたキューにおいて、ソケットの割り当て方法を制御するコアバインドポリシーと、メモリの使用方法を制御するメモリアロケーションポリシーがそれぞれ指定できます。

また、ソケットスケジューリングが有効化されたキューにおいては、リクエスト投入時、以下の２つの機能が利用できるようになります。

- ジョブ毎CPU台数をソケット単位で指定する機能
- ジョブ毎CPU台数とジョブ毎メモリ量の比率チェック機能

18.1.1 ソケットスケジューリング機能の有効化

ソケットスケジューリング機能を利用する場合は、qmgr(1M) を操作員権限で起動し、以下のサブコマンドを使用して、 キューのソケットスケジューリング機能を ON に設定し、有効化してください。

キュー	qmgr(1M)サブコマンド
バッチキュー	set execution_queue numa_control = { on | off } <queue>
会話キュー	set interactive_queue numa_control = { on | off } <queue>

on　：ソケットスケジューリング機能を利用する

off　：ソケットスケジューリング機能を利用しない（既定値）

【例】

```
$ qmgr -Po
Mgr: set execution_queue numa_control = on que1
Set Numa Control ON. queue: que1
```

【注意】

・ソケットスケジューリング機能を ON に設定したキューに対しては、NUMA アーキテクチャのスカラーマシン（Linux）である実行ホストの JSV をバインドする必要があります。
・複数の実行ホストをバインドする場合は、すべて同じソケット構成の実行ホストとしてください。
・また、複数のキューに同一の実行ホストをバインドする場合、それらのキュー間で、ソケットスケジューリング機能の ON／OFF を混在させる構成はできません。
・ソケットスケジューリング機能を有効に設定した場合、論理ホスト毎の CPU 台数制限値が OMP_NUM_THREADS 環境変数に自動的に設定されます。VE を使用するジョブでは、使用する VE 数に応じてジョブスクリプト内で OMP_NUM_THREADS 環境変数を適切に設定してください。
・GPU-CPU Affinity 機能が ON の場合は、ソケットスケジューリング機能を無効にできません。GPU-CPU Affinity 機能を OFF にしてから無効にしてください。GPU-CPU Affinity 機能の詳細は 18.3 GPU-CPU Affinity 機能を参照してください。

18.1.2 コアバインドポリシー

ジョブに資源を割り当てる際の CPU コア選択のポリシーとして、以下の 2 種類があります。

- ソケット集中ポリシー
    - コアがソケットに集中するように割り当てます。空きコアがより少ないソケットが優先となります。
- ソケット分散ポリシー
    - コアがソケット間で分散するように割り当てます。空きコアがより多いソケットが優先となります。

コアバインドポリシーは、ソケットスケジューリング機能が有効化されたキューに対して、キュー単位で設定できます。設定には qmgr(1M) を操作員権限で起動し、以下のサブコマンドを使用します。

キュー	qmgr(1M)サブコマンド
バッチキュー	set execution_queue numa_option core_bind_policy = <policy> <queue>
会話キュー	set interactive_queue numa_option core_bind_policy = <policy> <queue>

<policy>には以下を指定することができます。

concentration ：ソケット集中ポリシー（既定値）
balance　　：ソケット分散ポリシー

18.1.3 メモリアロケーションポリシー

ジョブ実行時のソケット間のメモリ割り当てのポリシーとして、以下の 3 種類があります。

- membindポリシー
    - ジョブの実行コアが属するソケットのメモリのみを使用します。
    - メモリが不足した場合はスワップを使用します。
- localallocポリシー
    - ジョブの実行コアが属するソケットのメモリを優先的に使用します。
    - メモリが不足した場合は他ジョブのソケット上のメモリを使用します。
- interleaveポリシー
    - ジョブに割り当てたソケットのメモリを交互に使用します。
    - メモリが不足した場合は他ジョブのソケット上のメモリを使用します。

メモリアロケーションポリシーは、ソケットスケジューリング機能が有効化されたキューに対して、キュー単位で設定できます。設定には qmgr(1M) を操作員権限で起動し、以下のサブコマンドを使用します。

キュー	qmgr(1M)サブコマンド
バッチキュー	set execution_queue numa_option memory_allocation_policy = <policy> <queue>
会話キュー	set interactive_queue numa_option memory_allocation_policy = <policy> <queue>

<policy> には以下を指定することができます。

membind：membindポリシー

localalloc：localallocポリシー（既定値）

interleave：interleaveポリシー

【注意】

システムに HugePage を設定し、ソケットスケジューリングで membind ポリシーを指定した場合、要求する cpu 数によっては十分な HugePage が利用できないことがあります。

SX-Aurora TSUBASA では、VE 上でプログラムを高速実行するために HugePage を必要とします。そのため、メモリバインドポリシーは localalloc または interleap を指定することを推奨します。

もし、メモリバインドポリシーを membind にする場合は、SX-Aurora TSUBSA インストレーションガイド「4.11 HugePages の設定」を参照し、HugePages 設定コマンドの MEMBIND オプションを有効にしてください。

18.1.4 ジョブ毎 CPU 台数をソケット単位で指定する機能

ソケットスケジューリング機能が ON のキューの場合、リクエストを投入する際、qsub -l オプションにおいて、ジョブ毎 CPU 台数の指定（cpunum_job）をする代わりに、ジョブ毎ソケット数の指定（socknum_job）をすることができます。

(1) キューにおけるリクエスト投入時のCPU数指定方法の設定

リクエスト投入時、ジョブ毎ソケット数指定によるリクエスト投入を許可するかどうかは、ソケットスケジューリング機能が ON のキューに対して、キュー単位で設定します。

設定には qmgr(1M) を操作員権限で起動し、以下のサブコマンドを使用します。

キュー	qmgr(1M)サブコマンド
バッチキュー	set execution_queue submit_cpu_unit = { cpu | socket | any } <queue>
会話キュー	set interactive_queue submit_cpu_unit = { cpu | socket | any } <queue>

submit_cpu_unit に指定する値の意味は以下の通りです。

cpu　：cpunum_job のみ指定可能

socket　　：socknum_job のみ指定可能

any　：cpunum_job、socknum_job のどちらでも指定可能（既定値）

(2) ジョブ毎ソケット数を指定してのリクエスト投入

ソケットスケジューリング機能が ON で、且つ、リクエスト投入時の CPU 数指定方法で socknum_job 指定可と設定されているキューに対しては、qsub -l オプションにおいて、ジョブ毎ソケット数の指定（socknum_job）をすることができます。

ジョブ毎ソケット数を指定してリクエストを投入するには、qsub コマンドで -l オプションに、socknum_job=<limit> を指定します。なお、socknum_job は、-l オプションの cpunum_job と同時には指定できません。

【例】

```
$ qsub -q bq -l socknum_job=4 job_script
Request 226.bsv.example.com submitted to queue: bq.
```

socknum_job で指定したソケット数は、キューにバインドされている実行ホストに搭載されているソケットの情報をもとに、ジョブ毎 CPU 台数に自動換算されます。

自動換算では、以下のように計算します。

（指定したソケット数）×（キューにバインドされている実行ホストのソケットあたりの CPU（コア）数）

【注意】

・本機能はジョブ毎の CPU 台数制限値の指定を代替するためのものであり、ジョブに割り当てられる CPU コアの配置をソケット単位となるよう指定（制限）するものではありません。
・本機能では実行ホストのソケットに関する情報を使用するため、リクエスト投入時に、キューに JSV がバインドされていない場合、または一度も LINKUP したことがない JSV のみがバインドされている場合、投入エラーとなります。
キューにバインドされ LINKUP していたジョブサーバが全てダウンした場合は、ダウン前の情報を使用して、CPU 台数の換算を行います。

18.1.5 ジョブ毎の CPU 台数とメモリ量の比率チェック機能

リクエスト投入時に指定されたジョブ毎の CPU 台数とメモリ使用量の値が、実行ホストのソケット上の CPU(コア)数とメモリ容量の比率に一致しているかどうかをチェックし、ソケットの比率に一致しない指定をした場合は投入エラーとすることができます。

本機能により、実行ホスト上でジョブが実行される際に、なるべくソケット内のローカルメモリが使用されるように、CPU を割り当てることができます。

ジョブ毎の CPU 台数とメモリ量の比率チェック機能は、ソケットスケジューリング機能が ON のキューに対して、キュー単位で設定できます。設定には、qmgr(1M) を操作員権限で起動し、以下のサブコマンドを使用して有効化します。

キュー	qmgr(1M)サブコマンド
バッチキュー	set execution_queue numa_unit_check = { on | off } <queue>
会話キュー	set interactive_queue numa_unit_check = { on | off } <queue>

on　：ジョブ毎のCPU台数とメモリ量の比率チェック機能を利用する

off　：ジョブ毎のCPU台数とメモリ量の比率チェック機能を利用しない（既定値）

【注意】本機能では実行ホストのソケットに関する情報を使用するため、リクエスト投入時に、キューに JSV がバインドされていない場合、または一度も LINKUP したことがない JSV のみがバインドされている場合、投入エラーとなります。
キューにバインドされ LINKUP していたジョブサーバが全てダウンした場合は、ダウン前の情報を使用して、ジョブ毎 CPU 台数とジョブ毎メモリ量の比率チェックを行います。

18.1.6 ソケットスケジューリング情報の表示

(1) キュー情報

qstat -Qf でのキュー情報表示で、ソケットスケジューリング機能の各種設定（NUMA Control, NUMA option, Submit CPU Unit, NUMA Unit Check）が参照できます。

【例】

```
$ qstat -Qf
Execution Queue: que1@bsv1
    Run State    = Active
    Submit State = Enable
      :
    Restart option = {
      Ignore modify
    }
    NUMA Control = ON
    NUMA option = {
      Core Bind Policy = concentration
      Memory Allocation Policy = localalloc
    }
    Submit CPU Unit = any
    NUMA Unit Check = OFF
    Hold Privilege    = (none)
    Suspend Privilege = (none)
      :
```

(2) 実行ホスト情報

ソケットスケジューリング機能利用時、qstat -Ef での Linux 実行ホストの情報表示で、ソケット情報とソケットの利用状況（Socket Resource Usage）が参照できます。

【例】

```
$ qstat -Ef
Execution Host: host1
  Batch Server = bsv1
      :
  Average Information:
    LOAD (Latest  1 minute ): 0.230000
```

```
    LOAD (Latest  5 minutes): 0.270000
      :
  Cpuset Information:
    Resource Sharing Groups = {
      RSG Number 0  = Name: cpuset  Cpus: 0-31  Mems: 0-7
    }
  Socket Resource Usage:
    NUMA Nodes = {
      Node 0 (Cpus: 0-3)   = Cpu: 4/4 Memory: 0.5GB/4.0GB
      Node 1 (Cpus: 4-7)   = Cpu: 4/4 Memory: 0.5GB/4.0GB
      Node 2 (Cpus: 8-11)  = Cpu: 0/4 Memory: 0B/4.0GB
      Node 3 (Cpus: 12-15) = Cpu: 0/4 Memory: 0B/4.0GB
      Node 4 (Cpus: 16-19) = Cpu: 0/4 Memory: 0B/4.0GB
      Node 5 (Cpus: 20-23) = Cpu: 0/4 Memory: 0B/4.0GB
      Node 6 (Cpus: 24-27) = Cpu: 0/4 Memory: 0B/4.0GB
      Node 7 (Cpus: 28-31) = Cpu: 0/4 Memory: 0B/4.0GB
    }
```

(3) ジョブ情報

ソケットスケジューリング機能利用時、qstat -Jf で、実行中のジョブが使用しているソケット番号（Assigned Sockets）が参照できます。

【例】

```
$ qstat -Jf
Request ID: 166.bsv1
    Batch Job Number = 0
    Execution Job ID = 3662
         :
    Remaining CPU Time = UNLIMITED
    Virtual Memory = 0.000000B
    Assigned Sockets = 0-1
      :
```

18.2 CPUSET機能

NUMA アーキテクチャのスカラーマシン（Linux）を実行ホストとして使用する場合、Linux の CPUSET 機能を使用することにより、仮想的に SUPER-UX における Resource Sharing Group (RSG) に相当するリソース分割機能を提供します。これを CPUSET 機能と呼びます。

本機能では、Linux の CPUSET 機能と連携し、一つの実行ホストの資源（CPU、メモリ）を複数に分割し、キューごとに割り当てることで搭載資源量の大きい実行ホストを効率よくスケジューリングする機能を提供します。

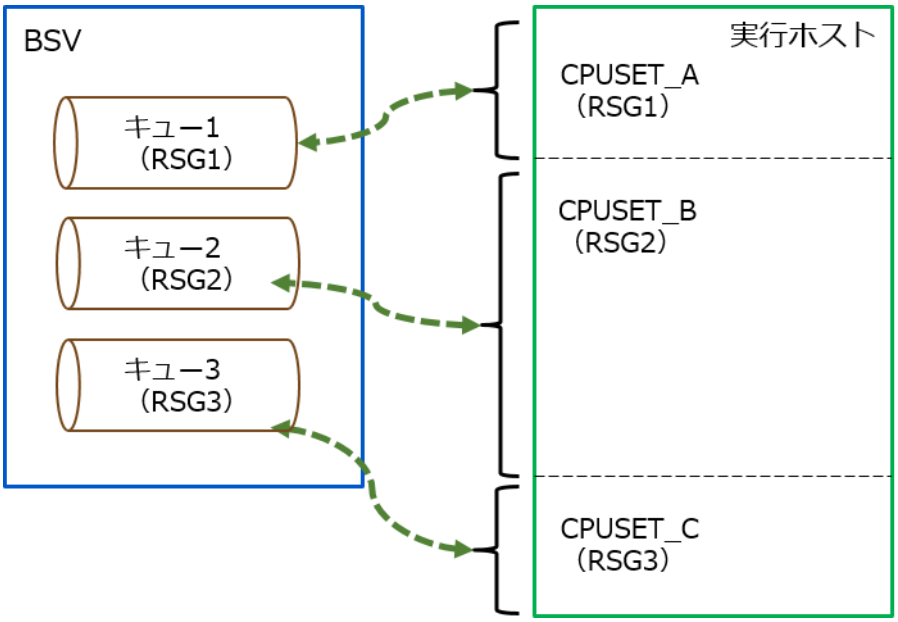


図 18-1 ：CPUSET 機能の概念図

また、CPUSET 機能では、実行ホストの各ソケット上の物理的な CPU コアとメモリノードを Linux の CPUSET 機能によりグループ化して各ジョブに割り当てます。ジョブ実行時に、ソケットスケジューリング機能によって割り当てた CPU コアとメモリノードを CPUSET として、キューに対応する CPUSET (RSG) 内に作成し、その CPUSET 内でジョブのプロセスを実行します。これにより、ジョブが実行ホストの資源を排他的に使用できるようになります。

以下は実行ホスト上の CPUSET_B（RSG2）を使用する que2 に投入されたリクエストのジョブにおいて、そのジョブ用の CPUSET を作成する概念図です。

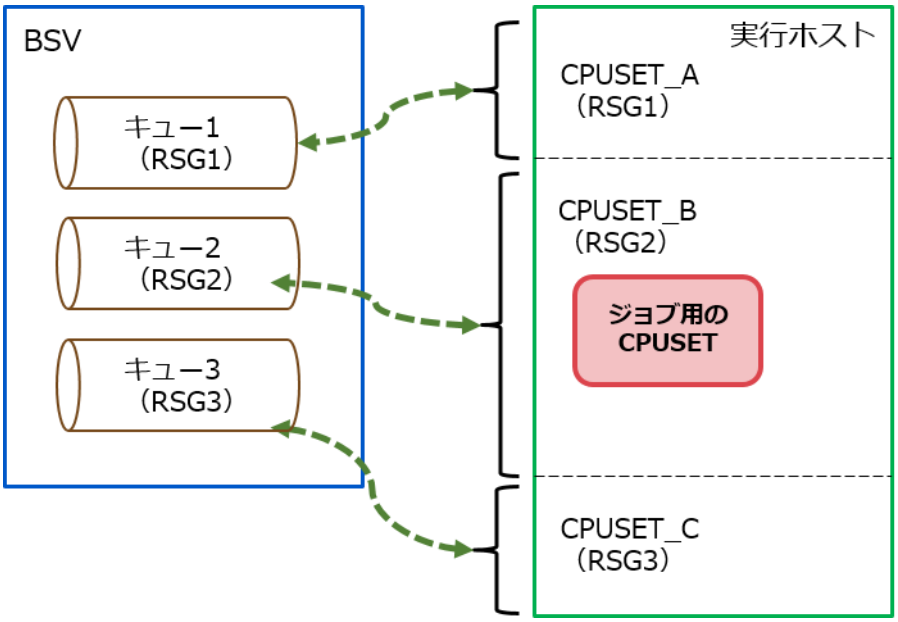


図 18-2：ジョブ用の CPUSET を作成する概念図

ジョブ用の CPUSET は、リクエストが投入されたキューに対応した CPUSET（RSG）内に実行開始のタイミングで作成され、この中の CPU コアおよびメモリノードのみを使用してジョブが動作します。

18.2.1 CPUSET 機能の設定

CPUSET 機能はソケットスケジューリング機能と連動しています。そのため、CPUSET 機能を使用するためには、まず投入先のキューのソケットスケジューリング機能を有効化してください。

（詳細は 18.1.1.ソケットスケジューリング機能の有効化 参照）

上記設定を行った上で、CPUSET 機能を使用する実行ホスト上に CPUSET の定義を行う設定ファイル（cpuset.conf）を作成してください。cpuset.conf は、各実行ホストの/etc/opt/nec/nqsv 配下に作成し、以下の形式で記述します。

・　ソケットに搭載されているメモリ量（省略可）

```
Sizeof_MemoryNode　<メモリ量>
```

本指定は省略可で、省略した場合はメモリ容量をホストから自動的に取得します。

・　CPUSETの定義（複数定義可、省略不可）

<CPUSET 名>　<CPU コア番号の範囲>　<メモリノード番号の範囲>　<対応 RSG 番号>

1 行に 1 CPUSET で、空白文字で区切って、以下の項目を設定してください。

CPUSET 名：Linux 上で生成する CPUSET の名前

CPUSET で使用するコア番号の範囲（N-M）※

CPUSET で使用するメモリノード番号の範囲（N-M）※

CPUSET と対応させる RSG 番号（0～31）

※CPU コア番号とメモリノード番号について、連続していない値を記述したい場合、
カンマ区切りで空白文字を入れずに値を並べて記述してください。（N1,N2-M2,…）

【注意】

- 最初の CPUSET は、以下の条件を満たすように定義してください。
  - CPUSET 名は "cpuset"とする
  - 実行ホスト全体のリソース量となるように CPU コア番号とメモリノード番号を指定
  - 対応 RSG 番号は "0" とする
  - CPUSET で使用するコア番号の範囲は、ソケット単位となるように指定
- 以降の CPUSET 定義行では、CPUSET 名と対応 RSG 番号にはユニークな値を設定してください。
- cpuset.conf の設定変更を行った場合は、JSV を再起動してください。
- cpuset.conf において、もともと存在していた CPUSET 定義行を削除した場合には、実行ホスト上にあるその CPUSET は手動で削除してください。
- NQSV の CPUSET 機能を使用する場合、実行ホスト上で CPUSET を手動で作成・変更することはしないでください。（上記項目の場合を除く）
- 各 CPUSET 間で CPU コア番号とメモリノード番号が重ならないように定義してください。（"cpuset" は除く）
- "cpuset"（RSG 番号 0）を使うキューと、その他の CPUSET（RSG 番号 1 以上）を使うキューに、同一の JSV をバインドしないでください。

以下は cpuset.conf の定義例です

```
Sizeof_MemoryNode 4.0GB
#####################################################
# A first CPUSET line is resources of the host total.
# The CPUSET sets 'cpuset'. The RSGNO must set '0'.
#####################################################
#CPUSET     CPUS        MEMS        RSGNO
cpuset      0-31        0-7         0
#####################################################
# Following CPUSET lines is divided resources.
# CPUSET and RSGNO should set unique values.
#####################################################
cpusetA     0-11        0-2         1
```

```
cpusetB     12-15       3           2
cpusetC     16-31       4-7         3
```

CPUSETを作成後、各キューで使用するCPUSETのRSG番号を設定します。qmgr(1M) を操作員権限で起動し、以下のサブコマンドを使用して、設定してください。

キュー	qmgr(1M)サブコマンド
バッチキュー	set execution_queue kernel_param rsg_number = <value> <queue>
会話キュー	set interactive_queue kernel_param rsg_number = <value> <queue>

以下は que1 で RSG 番号 1 の CPUSET を使用する場合の設定例です。

```
$ qmgr -Po
Mgr: set execution_queue kernel_param rsg_number = 1 que1
Set RSG Number (Kernel-Parameter): que1
```

18.2.2 CPUSET 情報の表示

(1)　実行ホスト情報

CPUSET 機能利用時、qstat -Ef での Linux 実行ホストの情報表示で、CPUSET 情報（Cpuset Information）が参照できます。

【例】

```
$ qstat -Ef
Execution Host: host1
  Batch Server = bsv1
      :
      :
  Average Information:
    LOAD (Latest  1 minute ): 0.230000
    LOAD (Latest  5 minutes): 0.270000
      :
    CPU  (Latest 15 minutes): 0.008000
  Cpuset Information:
    Resource Sharing Groups = {
      RSG Number 0  = Name: cpuset  Cpus: 0-31  Mems: 0-7
      RSG Number 1  = Name: cpusetA Cpus: 0-11  Mems: 0-2
      RSG Number 2  = Name: cpusetB Cpus: 12-15 Mems: 3
      RSG Number 3  = Name: cpusetC Cpus: 16-31 Mems: 4-7
```

```
    }
  Socket Resource Usage:
    NUMA Nodes = {
      Node 0 (Cpus: 0-3)   = Cpu: 4/4 Memory: 852.1MB/4.0GB
      Node 1 (Cpus: 4-7)   = Cpu: 4/4 Memory: 10.0MB/4.0GB
      Node 2 (Cpus: 8-11)  = Cpu: 0/4 Memory: 10.0MB/4.0GB
      Node 3 (Cpus: 12-15) = Cpu: 0/4 Memory: 10.0MB/4.0GB
      Node 4 (Cpus: 16-19) = Cpu: 0/4 Memory: 10.0MB/4.0GB
      Node 5 (Cpus: 20-23) = Cpu: 0/4 Memory: 10.0MB/4.0GB
      Node 6 (Cpus: 24-27) = Cpu: 0/4 Memory: 10.0MB/4.0GB
      Node 7 (Cpus: 28-31) = Cpu: 0/4 Memory: 10.0MB/4.0GB
    }
```

(2)　キュー情報

qstat -Qf でのキュー情報表示で、キューで使用する CPUSET の RSG 番号（Kernel Parameter: Resource Sharing Group）が参照できます。

【例】

```
$ qstat -Qf
Execution Queue: que1@bsv1
    Run State    = Active
    Submit State = Enable
      :
  Resources Limits:
    (Per-Req) Elapse Time Limit       = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
    (Per-Job) CPU Time                = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
      :
    (Per-Prc) Permanent File Capacity = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
  Kernel Parameter:
    Resource Sharing Group     = 1
    Nice Value                 = 0
      :
```

18.3 GPU-CPU Affinity機能

GPU-CPU Affinity 機能は、GPU ジョブの計算速度の向上を目的とする機能です。

NQSV は、実行ホストに搭載されている GPU と CPU ソケットの物理的距離を認識し、近い位置にある CPU と GPU の組み合わせでジョブに割り当てを行います。

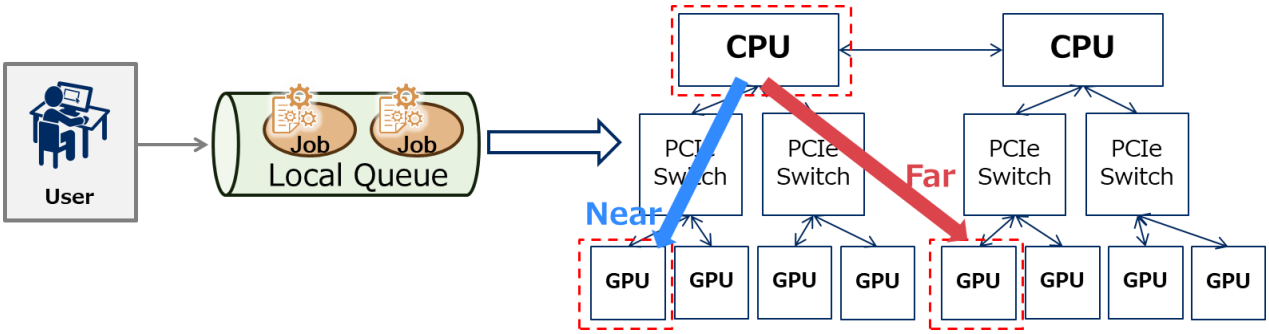


図 18-3 GPU と CPU の距離

GPU-CPU Affinity 機能は、ソケットスケジューリング機能が有効なキューにおいてキュー単位に利用できます。GPU-CPU Affinity 機能が有効化されたキューに GPU を使用するリクエストを投入すると、近い位置にある CPU と GPU の組み合わせでリクエストに割り当てることが可能です。

18.3.1 GPU-CPU Affinity 機能の有効化

GPU-CPU Affinity 機能を利用する場合は、**qmgr(1M)**を操作員権限以上で起動し、以下のサブコマンドを利用して、キューの GPU-CPU Affinity 機能を ON に設定し、有効にしてください。

キュー	qmgr(1M)サブコマンド
バッチキュー	set execution_queue gpu_affinity = { on | off } <queue>
会話キュー	set interactive_queue gpu_affinity = { on | off } <queue>

on　：GPU-CPU Affinity機能を利用する

off　：GPU-CPU Affinityを利用しない（既定値）

【例】

```
$ qmgr -Po
Mgr: set execution_queue gpu_affinity = on que1
Set GPU-CPU Affinity ON. queue: que1
```

【注意】

- ソケットスケジューリング機能におけるジョブ毎の CPU 台数とメモリ量の比率チェック機能は off としてください。当該機能が on の場合は、GPU-CPU Affinity 機能を ON にできません。また、GPU-CPU Affinity 機能が ON の場合は、当該キューのジョブ毎の CPU 台数とメモリ量の比率チェック機能の設定を cpu または socket に変更できません。

・複数のキューに同一の実行ホストをバインドする場合、それらのキュー間で、GPU-CPU Affinity 機能の ON／OFF を混在させる構成はできません。
・同一の実行ホストを、GPU あたりの CPU 台数設定が異なる複数キューにバインドしないでください。バインドした場合は、適切な GPU-CPU の割り当てができなくなります。

18.3.2 GPU あたりの CPU コア数の設定

NQSV の GPU-CPU Affinity 機能は、近い距離にある CPU と GPU の組み合わせ、これを単位として割り当てを行います。CPU ソケットの搭載コア数を、その CPU ソケットと近い GPU の台数で割ったコア数を GPU あたりの CPU コア数として設定します。これが 1GPU につき必ず設定した CPU コア数となり、近い距離にある CPU と GPU の組み合わせで選択することが可能です。

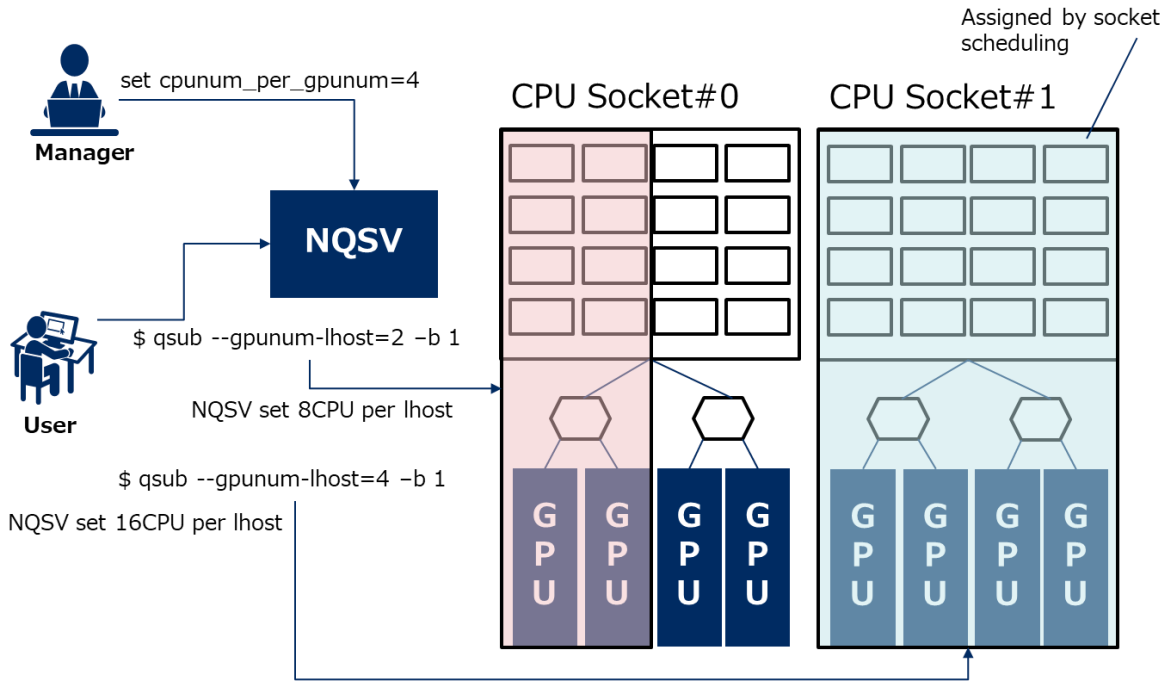


図 18-4 GPU あたりの CPU コア数によるジョブの CPU 台数の計算

GPU あたりの CPU コア数の値は、以下の式を元に算出し設定してください。

1 ソケットに搭載されている CPU コア数 ÷ 当該ソケットと物理的に近い GPU 台数

上記の図の例ですと、1 ソケットに搭載されている CPU コア数は 16、ソケットと物理的に近い GPU の数は 4 ですので、GPU あたりの CPU コア数の値は 16÷4 = 4 となります。

GPU あたりの CPU コア数の値の設定は、GPU-CPU Affinity が ON になっているキューに対して、キュー単位で設定できます。設定には **qmgr(1M)**を操作員権限で起動し、以下のサブコマンドを使用します。

キュー	qmgr(1M)サブコマンド
バッチキュー	set execution_queue cpunum_per_gpunum = <cpunum> <queue>
会話キュー	set interactive_queue cpunum_per_gpunum =　<cpunum> <queue>

<cpunum>には 0～2147483647 を指定可能です。既定値は 1 です。

【注意】

- GPU あたりの CPU コア数の値を計算する式で算出された値よりも大きい値を設定した場合、割り当てるコア数が不足してリソースの割り当てができず、ジョブが PRE-RUNNING で失敗する可能性があるため、必ず実行ホストの GPU 数および CPU 数を確認し、適切な CPU コア数を設定してください。
  もし誤って設定してジョブが PRE-RUNNING で失敗した場合は、設定を見直したうえ、ジョブが実行したジョブサーバを再起動してください。
- キューにリクエストが存在する場合、本値の設定を変更することはできません。
- GPU-CPU Affinity 機能が ON となっているキューには、リクエスト投入時に以下の制約が加わります。
  - **qsub(1)/qlogin(1)/qrsh(1)**コマンドで、**--cpunum-lhost** オプション、または **-l cpunum_job** オプションを指定してリクエストを投入することはできません。
    また、ハイブリッドリクエストを投入する場合、すべてのジョブグループにおいて **--cpunum-lhost** オプション、または**-l cpunum_job** オプションを指定することはできません。
  - **qsub(1)/qlogin(1)/qrsh(1)**コマンドで、**--cpunum-lhost** オプション、または **-l cpunum_job** オプションを指定せずにリクエストを投入した場合、GPU あたりの CPU コア数にて論理ホスト毎の CPU 台数を自動計算するため、キューの論理ホスト毎/ジョブ単位の CPU 台数制限の既定値は適用されず無視されます。
  - バッチキュー・会話キューの論理ホスト毎/ジョブ単位の GPU 台数制限の既定値が 0 と設定されている場合、**qsub(1)/qlogin(1)/qrsh(1)**コマンドで、**--gpunum-lhost** オプション、または**-l gpunum_job** オプションで 1 以上の値を必ず指定してください。
  - バッチキュー・会話キューの論理ホスト毎/ジョブ単位の GPU 台数制限の既定値が 1 以上と設定されている場合、**qsub(1)/qlogin(1)/qrsh(1)**コマンドで、**--gpunum-lhost** オプション、または**-l gpunum_job** オプションで 0 の値を指定することはできません。
  - ハイブリッドリクエストを投入する場合、すべてのジョブグループにおいて **--gpunum-lhost** オプション、または**-l gpunum_job** オプションで 1 以上の値を必ず指定してください。
  - **qalter(1)**コマンドで、投入済みのリクエストの論理ホスト毎/ジョブ単位の CPU 台数制限値および GPU 台数制限値を変更することはできません。
- 転送キューの転送先キューの GPU-CPU Affinity 機能が ON となっている場合、論理ホスト毎/ジョブ単位の GPU 台数制限値が 0、または**--cpunum-lhost(-l cpunum_job)**オプションの指定があるリクエストは転送することができません。

18.3.3 トポロジーの設定

システム管理者は、CPU ソケット、ソケット番号、および PCIeSW の GPU 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。本ファイルは、**[JobManipulator 編] 5.4.2 HCA およびトポロジー情報の定義** に記載されているファイルと同じですが、ファイルフォーマットが異なります。

運用中の設定変更は不可です。設定変更する場合は JSV の再起動を行ってください。

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 GPU 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

GPU-CPU Affinity が適用可能なデバイスグループの例は下記の通りです。

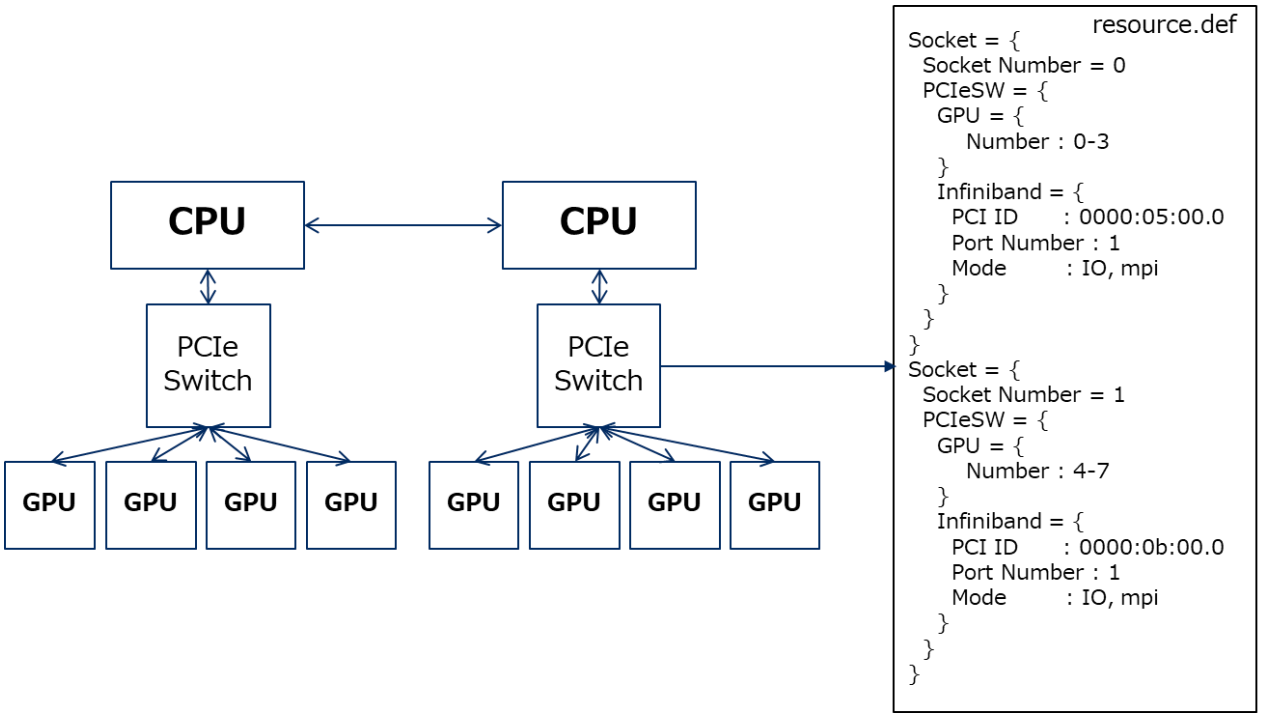


図 18-5 GPU トポロジーの設定例

(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、実行ホスト上の/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

GPU-CPU Affinity 機能における、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 | GPU
各文字列の意味は下記のとおり。

- Socket ： CPU ソケット
- PCIeSW:PCIeSW
- GPU: GPU
- 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
- GPU
  <Name>　　：<Value>
  Number　　：GPU 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・小文字のどちらの記載も可能です。

下記条件のいずれかを満たした場合、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。
- GPU 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。
- **[JobManipulator 編] 5.4.2 HCA およびトポロジー情報の定義** に記載のある、「VE」と「GPU」を同時に指定した。

(4) デバイスリソース定義ファイルの設定例

下記は、図 22-3 GPU トポロジーの設定例 の構成の場合の設定例です。CPU ソケットは 2 つ、当該ソケットに物理的に近い PCIeSW は 2 つで、1 つの PCIeSW に搭載されている GPU の数は 2 台とします。

```
Socket = {
  Socket Number = 0
  PCIeSW = {
    GPU = {
      Number : 0-1
    }
    Infiniband = {
      PCI ID      : 0000:05:00.0
      Port Number : 1
      Mode        : IO, mpi
    }
  }
  PCIeSW = {
    GPU = {
      Number : 2-3
    }
    Infiniband = {
      PCI ID      : 0000:0b:00.0
      Port Number : 1
      Mode        : IO, mpi
    }
  }
}
Socket = {
  Socket Number = 1
  PCIeSW = {
    GPU = {
      Number : 4-5
    }
    Infiniband = {
      PCI ID      : 0000:13:00.0
      Port Number : 1
      Mode        : IO, mpi
    }
  }
```

```
  PCIeSW = {
    GPU = {
      Number : 6-7
    }
    Infiniband = {
      PCI ID      : 0000:19:00.0
      Port Number : 1
      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
  GPU = {
    Number : 0-1
  }
}
Socket = {
  Socket Number = 1
  GPU = {
    Number : 2-3
  }
}
```

(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 = {
      Node 0 (Cpus: 0-3) = Cpu: -/4 Memory: -/7.9GB
      Node 1 (Cpus: 4-7) = Cpu: -/4 Memory: -/7.9GB
    }
  Device Topology:
```

```
Device Topology:
    Socket 0 = {
      PCIeSW 0 = {
        GPU: 0-1
        HCA: 0000:05:00.0 1 (IO,MPI)
      }
      PCIeSW 1 = {
        GPU: 2-3
        HCA: 0000:0b:00.0 1 (IO,MPI)
      }
    }
    Socket 1 = {
      PCIeSW 2 = {
        GPU: 4-5
        HCA: 0000:13:00.0 1 (IO,MPI)
      }
      PCIeSW 3 = {
        GPU: 6-7
        HCA: 0000:19:00.0 1 (IO,MPI)
      }
    }
```

PCIeSW が無い場合は、以下の様に PCIeSW の部分が表示されません。

```
Device Topology:
    Socket 0 = {
        GPU: 0-1
    }
    Socket 1 = {
        GPU: 2-3
    }
```

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。

```
Device Topology:(none)
```

18.3.4 cgroups の利用

GPU-CPU Affinity 機能において割り当てられた CPU および GPU に対して cgroups で制限を行うことができます。制限を行う処理はシェルスクリプトで行います。

cgroups で制限を行うには、実行ホストの下記のパスにシェルスクリプトを配置してください。このスクリプトが存在しない場合、cgroups での制限はされませんが、エラーにはなりません。このスクリプトは実行ホスト上で root 権限によって実行されます。

```
/opt/nec/nqsv/sbin/system_startup_prog/cgroups.sh
```

シェルスクリプトの実行時には下記の環境変数が渡されます。

環境変数名	説明
NQSV_CG_JOBID	ジョブ ID
NQSV_CG_GPUNUM	GPU 台数の制限値
NQSV_CG_GPULIST	割り当てられた GPU 番号 例：2 番と 3 番が割り当てられた場合、以下の値になる。 NQSV_CG_GPULIST=2,3
NQSV_CG_CPUNUM	CPU 台数の制限値 これは（--gpunum-lhost の値）×（キューに設定された GPU あたりの CPU 数）である。
NQSV_CG_CPULIST	割り当てられた CPU 番号 例：2 番と 3 番が割り当てられた場合、以下の値になる。 NQSV_CG_CPULIST=2,3
NQSV_CG_MEMMAX	メモリサイズ制限の最大値 単位は数字の最後に 1 バイトの文字で表す。具体的な値は、B、K、M、G、T、P、E である。 B：バイト、K：K バイト：M：M バイト、G：G バイト、T：T バイト、P：P バイト、E：E バイト unlimited の場合は UNLIMITED が格納される。
NQSV_CG_MEMWARN	メモリサイズ制限の警告値 単位は数字の最後に 1 バイトの文字で表す。具体的な値は、B、K、M、G、T、P、E である。 B：バイト、K：K バイト：M：M バイト、G：G バイト、T：

環境変数名	説明
	T バイト、P：P バイト、E：E バイト unlimited の場合は UNLIMITED が格納される。

NQSV/JobServer のインストール時に、サンプルのシェルスクリプトを以下の場所に配置していますので、これを参考にして cgroup.sh を作成してください。

```
/opt/nec/nqsv/sbin/system_startup_prog/cgroups.sh.sample
```

なお、cgroup の設定はジョブ終了時に自動的に削除されます。

【注意】

・GPU-CPU Affinity 機能で割り当てられた CPU および GPU に対して cgroups で制限を行う場合は、ジョブから見える GPU は全てそのジョブに割り当てられたものとなるため、そのジョブには環境変数 CUDA_VISIBLE_DEVICES は渡されません。

第19章　障害検知・電源制御

NQSV では、実行ホスト外から実行ホストの障害を検知する機能と、実行ホストの電源制御による省電力機能を搭載しています。本機能を利用する場合は、後述のノード管理エージェントを運用する必要があります。

19.1 障害検知

Linux では、Zabbix 等の運用管理のための OSS（Open Source Software）を利用することにより、CPU 高温度などのノード異常、CPU 故障などの H/W 障害、OS ストール等（以降、これらを一括して障害と呼ぶ）を検知できます。

運用管理ホスト上にノード管理エージェントを起動しておき、運用管理の OSS で障害を検知した際に、障害通知コマンドを実行することにより、運用管理の OSS からノード管理エージェントを通じてバッチサーバに障害を通知できます。

また、障害検知を行う OSS を導入しない場合には、簡易障害検知スクリプトを使用することにより、バッチサーバに障害を通知できます。

- 短時間でノードのHW障害検知が可能になり、障害発生ノードの除外を即座に行う
- 障害種別の判別が可能

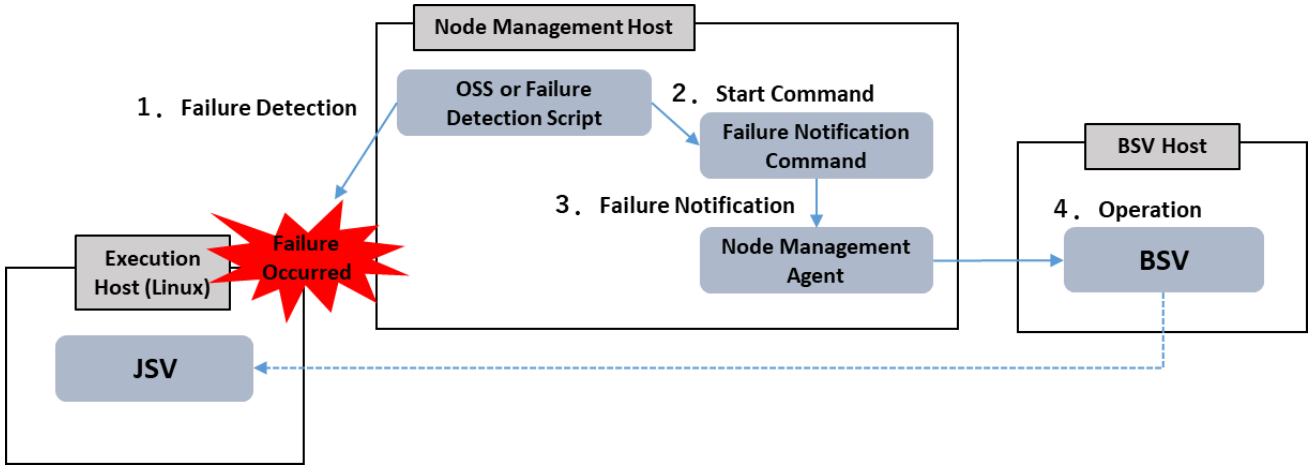


図 19-1：障害検知

19.1.1 障害検知機能の設定

(1)　簡易障害検知スクリプトによる障害検知

運用管理ホストにて、簡易障害検知スクリプト/opt/nec/nqsv/sbin/ping_check.sh を編集して、NODELIST=の行に、監視対象の実行ホストを設定してください。

```
#!/bin/bash
NODELIST="host1 host2 host3"

for node in $NODELIST;
do
    ping -c 5 $node >/dev/null 2>&1
    if [ $? -ne 0 ]; then
        /opt/nec/nqsv/sbin/nqs_ntfr -m "Down" $node >/dev/null 2>&1
    fi
done
```

同じく運用管理ホストにて、cron でこの簡易障害検知スクリプトを定期的に実行するよう設定してください。これにより、簡易障害検知スクリプトは、定期的に実行ホストの生存を確認し、接続できない場合は、BSV に障害を通知します。

簡易障害検知スクリプトから障害通知を受けた BSV は、障害が発生した実行ホストの JSV を LINKDOWN させます。

実行ホストの状態は、qstat -Etf コマンドで確認できます。

以下は、簡易障害検知スクリプトで障害通知された実行ホストの qstat -Etf コマンドでの表示イメージです。

```
$ qstat -Etf host1
Execution Host: host1
    Batch Server = host1.example.com
    Current State           = Inactive
    State Transition Time   = Tue Sep 29 13:41:03 2017
    State Transition Reason = ABNORMAL STOP
    Message = Down
    Job Server Number  = 11
    LINK Batch Server = DOWN
```

(2)　OSSによる障害検知

OSS を利用して障害検知を行うには、障害通知コマンド nqs_ntfr を使用します。障害通知コマンドは、ノード管理エージェントを介して、BSV に障害を通知します。

障害通知コマンド nqs_ntfr は、/opt/nec/nqsv/sbin にインストールされます。
形式は以下のとおりです。

```
nqs_ntfr [-o operation]  [-m message]  hostname | IP_address
```

nqs_ntfr コマンドは、hostname もしくは IP_address で指定した実行ホストで障害が発生したことを、BSV に通知します。

障害通知を受けて BSV が JSV に対してどのように操作（操作しない／JSV 停止／アンバインド）を行うかを、-o operation で指定します。また、障害の内容を -m message で指定して通知できます。message の内容は qstat -Etf の表示（Message）で確認できます。

各オプションの詳細は以下の通りです。

- -o operation
  - 障害通知時のJSVの操作を指定します。
  - operationには、下記のいずれかが指定可能です。
  - JSVがLINKUPの状態の時のみJSVを操作します。
  - ① nothing　JSVは操作しません。
  - ② down　　JSVをLINKDOWN します。
  - ③ unbind　JSVをキューからUNBIND します。
  - -oオプションの指定がない場合、downの動作になります。
  - -o operationの指定と障害通知時のJSVのLINK状態およびキューとのBIND状態による、BSVの動作は次のとおりです。

operation	障害通知時のJSVの状態		障害通知時のBSVの動作
	LINK	BIND	
nothing	UP/DOWN	BIND /UNBIND	JSVを操作しません。 JSVのBIND状態は変化しません。
down	UP	BIND /UNBIND	JSVをLINKDOWNします。 実行ホストの状態がINACTIVEになります。 JSVのBIND状態は変化しません。
	DOWN		JSVを操作しません。 JSVのBIND状態は変化しません。
unbind	UP	BIND	JSVをキューからUNBINDします。

		UNBIND	JSVを操作しません。 JSVのBIND状態は変化しません。
	DOWN	BIND /UNBIND	JSVを操作しません。 JSVのBIND状態は変化しません。

※ -o down 指定の時以外、BSV は JSV を LINKDOWN する操作は行いません。

障害によって JSV との接続が切れると、LINKDOWN となります。

- -m message
    - 障害内容をmessageで指定します。
    - messageの最大文字列長は255バイトです。
    - 255バイトを超えた場合、超過した文字列は切り捨てます。
    - JSVの状態にかかわらず、qstat -Etfで表示します。

障害検知時に障害通知コマンドを実行するための OSS での設定方法については、Zabbix での設定例を以下に示します。

NQSV では、Zabbix Version2.4.6 を使用し、Zabbix から nqs_ntfr を使用してノード管理エージェントに障害を通知できることを確認しました。

Zabbix を使用して障害を検知する場合の設定手順は以下です。なお、詳細な設定操作については、Zabbix のマニュアルを参照してください。

- 実行ホストの状態を監視できるようにZabbixを構成してください。
- Actionを作成の上、検知する障害の条件を設定してください。
    - 例えば、ホスト上のメモリ枯渇を障害として検知する場合はConditionsとして以下のように設定します。
    - Trigger = host1: Lack of available memory on server host1
- 作成したActionのOperationsとして、以下の項目を設定してください。
    - Target: Current host
    - Type: Custom script
    - Execute on: Zabbix server
    - Commands: /opt/nec/nqsv/sbin/nqs_ntfr -m ”{TRIGGER.NAME}” {HOST.NAME1}

19.2　電源制御

JobManipulator は、リクエストのスケジューリングと実行ホストの起動・停止を連携させる省電力運用機能を実現しています。

本機能では、ノード管理エージェント経由で IPMI を利用して実行ホストの起動を行います。

ノード管理エージェントは、運用管理ホスト上にインストールされた ipmitool（/usr/bin/ipmitool）を利用して実行ホストに搭載された BMC（ベースボード管理コントローラ：Baseboard Management Controller）にアクセスし、ホストの起動を行います。

電源制御機能を利用するためには、事前に以下を設定してください。

- 各実行ホストで、搭載されているBMCが利用できるように設定
- 運用管理ホストにipmitoolをインストール

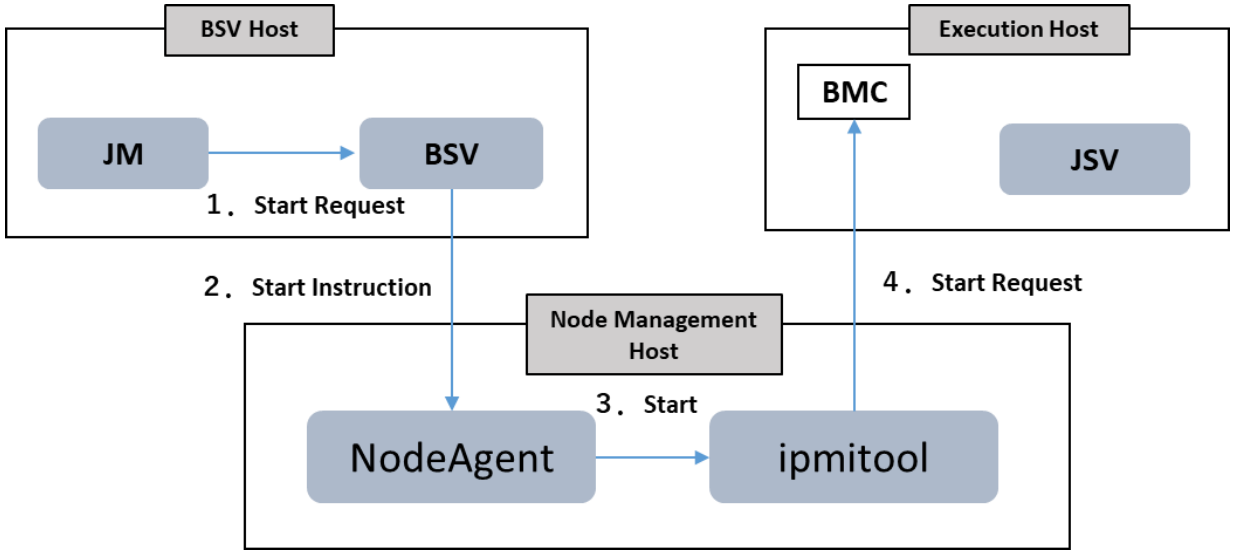


図 19-2：電源制御

19.3 ノード管理エージェントの設定

ノード管理エージェントは、Linux マシンの実行ホストを管理し、ホストで発生した障害のバッチサーバへの通知や、NQSV から実行ホストの起動を行うためのエージェントの役割を果たします。
ノード管理エージェントを運用するために下記を設定する必要があります。

(1) 各実行ホストで、搭載されているBMCが利用できるように設定

※BMCにアクセスするためのパスワードを設定した場合には、パスワードを記載したファイルを作成してください。

(3) (4) の設定でこのパスワードファイルのパスを指定します。

(2) 運用管理ホストにipmitoolをインストール

(3) ノード管理エージェントに関する設定（/etc/opt/nec/nqsv/nag.conf）

(4) 管理対象ホストおよびBMCへのアクセスに関する設定（/etc/opt/nec/nqsv/nag_nodelist）

本節では (3) と(4) について説明します。

・ /etc/opt/nec/nqsv/nag.conf（設定ファイル）

ノード管理エージェントの設定ファイル（/etc/opt/nec/nqsv/nag.conf）には、各設定項目を
パラメータ：設定値
の形式で指定します。
＃で始まる行、および空白行はコメント行として無視します。
同じパラメータの設定が、複数行書かれていた場合、後に書かれたものが有効になります。
以下の項目が設定できます。

パラメータ	値 (設定可能範囲または 設定可能最大文字数)	説明
BSVHOST	文字列 (最大255文字)	BSVホスト名。IPアドレスの指定も可能。 BSVHOSTの指定がない時は、localhostとなります。
PORT	数値 (1~65535)	BSVのポート番号。 PORTの指定がない時は、602となります。
LOGFILE	文字列 (最大1023文字)	ノード管理エージェントログファイルのパス。 LOGFILEの指定がない時、 /var/opt/nec/nqsv/node_agent_logとなります。
LOGLEVEL	数値 (0~4)	ログレベル。 値が大きいほど詳細な情報を出力します。 LOGLEVELの指定がない時は、1となります。 それぞれのレベルの意味は以下のとおりです。

		0：デバッグ用メッセージを出力せず、警告、およびエラー情報のみを出力。 1：警告、エラー情報に加えて、ノード管理エージェントの起動、終了処理、バッチサーバ接続情報、障害の検知情報を出力。 2：1に加えて、デバッグ用メッセージを出力。 3：2に加えて、各種パケットの処理状況を出力。 4：3に加えて、各種パケットの内容を出力。
LOGSIZE	数値 (1~2147483647) またはUNLIMITED	ログファイルのサイズ（単位：バイト） 設定可能な最大値は2147483647 (2GB-1) です。 LOGSIZEの指定がない時、もしくはUNLIMITEDと指定した時は最大値(2147483647)となります。
LOGSAVE	数値 (0～2147483647)	保存ファイル数。 ログファイルがLOGSIZEで指定したサイズを超えた場合、それまで出力していたファイルを別名で保存する際の保存ファイル数の上限です。 保存ファイル数がLOGSAVEで指定した数を超えた場合、古いファイルから順番に削除します。 0を指定した場合、ログファイルは保存しません。 LOGSAVEの指定がない時は3となります。
IPMIUSER	文字列 (最大47文字)	BMCにアクセスするためのユーザ名。 ipmitool実行時、-Uオプションで指定されたユーザ名を指定します。 IPMIUSERの指定がない時は、-Uオプションを指定しません。
IPMIPASSWDFL	文字列 (最大1023文字)	BMCにアクセスするためのパスワードが記載されたファイルのパス。 ipmitool実行時、-fオプションで指定されたパスワードファイルを指定します。 IPMIPASSWDFL指定がない時は、-fオプションを指定しません。

ノード管理エージェント運用中に上記設定変更を行った場合、以下のように systemctl reload にてノード管理エージェントの設定変更を反映します。

```
root# systemctl reload nqs-nag.service
```

・　/etc/opt/nec/nqsv/nag_nodelist（ノードリストファイル）

nag_nodelist には、ノード管理エージェントの管理対象の実行ホスト、BMC の IP アドレス、BMC にアクセスするためのユーザ名およびパスワードが記載されたファイルのパスを設定します。

各実行ホストで nag.conf の IPMIUSER および IPMIPASSWDFL の設定と異なる、ユーザ名およびパスワードを使用したい場合にのみ、ユーザ名およびパスワードファイルパスを設定してください。

nag_nodelist と nag.conf の両方に設定が存在する場合、nag_nodelist の設定が優先されます。

以下の形式で、スペース区切りでそれぞれの設定値を指定します。

```
<ホスト名>|<IPアドレス> <BMCのIPアドレス> [<ユーザ名> <パスワードファイルパス>]
```

＃で始まる行、および空白行はコメント行として無視します。
同じホストの設定が、複数行書かれていた場合、後に書かれたものが有効になります。

複数のノード管理エージェントを運用することができます。それぞれのエージェントのノードリストファイルに重複した実行ホストを定義しないでください。

ノード管理エージェント運用中にノードリストファイルを更新した場合、ノード管理エージェントを再起動してください。

```
root# systemctl restart nqs-nag.service
```

19.4 ノード管理エージェントの起動・停止

ノード管理エージェントは、運用管理ホストで以下のように起動します。

```
root# systemctl start nqs-nag.service
```

また、ノード管理エージェントを停止する場合には、運用管理ホストで以下のように停止します。

```
root# systemctl stop nqs-nag.service
```

19.5 ノードヘルスチェック機能

NQSV では、ジョブの開始時(PRE-RUNNING)およびジョブの終了時(POST-RUNNING)にジョブが実行するノードが健全かどうかを、シェルスクリプトを使ってチェックすることができます。チェック用のシェルスクリプトを各サイトの運用ポリシーに合わせて作成することで、CPU や GPU、VE、HCA 等の各種の HW 障害を検出できます。NQSV ではノードのヘルスチェックで障害を検知した場合、チェックのきっかけとなったリクエストを再実行します。問題が起こったノードは、UNBIND または LINKDOWN して運用から除外することができます。また、予め用意しているユーザ通知用スクリプトによってリクエストのオーナーへメール通知することができます。

19.5.1 ノードヘルスチェック設定の概要

ノードヘルスチェック機能を利用するためには、以下の順番で設定してください。

1. ヘルスチェックスクリプトの作成と配置
2. 障害検知ノードへの処置の設定
3. ユーザ通知用スクリプトの設定
4. Elapse マージンによるヘルスチェック時間の調整

以降の節ではノードヘルスチェック機能を利用するための詳細設定方法や障害検知時の動作等を説明します。

19.5.2 ヘルスチェックスクリプト

ヘルスチェックスクリプトは、各サイトの運用ポリシーに合わせてシステム管理者が作成するものです。スクリプト名と配置場所は JSV が動作する実行ホスト上の以下のパスです。

/opt/nec/nqsv/sbin/healthchk.sh

上記のスクリプトが存在しない場合や実行権限が適切に設定されない場合は、ノードヘルスチェックは行いません。実行ホストのノードヘルスチェックを行う場合は、事前に本スクリプトを作成して実行ホスト上に配置してください。

ヘルスチェックスクリプト実行時には NQSV から以下の環境変数がセットされます。スクリプトの中ではその情報をもとに各種 HW の状態をチェックします。

環境変数名	説明	環境変数の値
NQSV_HEALTHCHK_TIMING	ヘルスチェックのタイミング	PRE-RUNNING POST-RUNNING
NQSV_HEALTHCHK_RID	リクエストID	通常リクエストの場合： <シーケンス番号>.<ホスト名> パラメトリックリクエストの場合： <シーケンス番号>[].<ホスト名>
NQSV_HEALTHCHK_JID	ジョブID	<ジョブ番号>:<リクエストID>
NQSV_HEALTHCHK_SID	ジョブのセッションID	1～プロセスIDの最大値
NQSV_HEALTHCHK_ REQOWNER	リクエストのオーナー名	ユーザ名

NQSV_HEALTHCHK_ REQOWNERGRP	リクエストのオーナーのグループ名	グループ名
NQSV_HEALTHCHK_CPULIST	ジョブに割り当てられたCPU番号のリスト。ソケットスケジューリング機能が有効な場合のみ値が設定されます。	カンマ区切りのCPU番号の数字。 例：0,1,2,3,4
NQSV_HEALTHCHK_VELIST	ジョブに割り当てられたVEノード番号のリスト。 VEを使用するジョブの場合のみ値が設定されます。	カンマ区切りのVEノード番号の数字。 例：0,1,2,3,4
NQSV_HEALTHCHK_GPULIST	ジョブに割り当てられたGPU番号のリスト。GPUを使用するジョブの場合のみ値が設定されます。	カンマ区切りのGPU番号の数字。 例：0,1,2,3,4
NQSV_HEALTHCHK_HCALIST	ジョブに割り当てられたHCAデバイス名のリスト。HCAを使用するジョブの場合のみ値が設定されます。	カンマ区切りのHCAデバイス名の文字列。 例：0000:3e:00.0, 0000:4d:00.0

過剰な障害検知でリクエストやノードに対して誤った処置をしないために、スクリプトの中でジョブに割り当てられた資源の情報をもとに必要な障害検知を行ってください。例えば、VE の障害検知を行う場合は環境変数 NQSV_HEALTHCHK_VELIST の値をチェックして、VE ノード番号があればそれに応じた処理を行います。なお、CPU 資源はジョブが必ず使用するため、常に障害検知を行うことが可能です。

ヘルスチェックの結果は、シェルスクリプトの終了コードで返却してください。NQSV はその終了コードを取得して、ヘルスチェックで障害があったかどうかをチェックします。シェルスクリプトの終了コードが 0 の場合は障害がないと判断し、リクエストの実行を継続します。それ以外の値が返された場合は、障害があると判断し、リクエストの再実行や当該ノードへの処置等を行います。

VE と HCA ヘルスチェックのサンプルスクリプト

VE および HCA の障害検知用のヘルスチェックサンプルスクリプトは、実行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必要な障害検知処理を実装したスクリプトを作成して、適切なパスに配置してください。

> **【注意】**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。

サンプルスクリプトの概要は以下の通りです。

インストールパス：

/opt/nec/nqsv/sbin/healthchk.sh.sample

処理概要：

本サンプルでは、VE→HCA の順で VE と HCA の障害検知を行います。VE については、VEOS が提供している VE ノード異常チェック用コマンド（/opt/nec/ve/veos/libexec/ve_check_job：以降 ve_check_job という）を利用して、ジョブの開始時と終了時に VE および VEOS に異常がないかをチェックします。一方、HCA については、NQSV が HCA 障害検知を行うために実行ホスト上にインストールしている HCA 障害検知用のシェルスクリプト（/opt/nec/nqsv/sbin/hcachk.sh：以降、hcachk.sh という）を呼び出して、HCA デバイスに異常がないかをチェックします。

VE の障害検知で異常が発生した場合は、HCA の障害検知を実施しません。障害検知した場合は、ジョブの情報や失敗のコード、検知タイミング等の情報を Linux OS の/var/log/messages に出力します。なお、ジョブに VE または HCA が割り当てられていない場合は、当該の検知処理は行いません。

VE 障害検知は以下の流れで行います。

1. ジョブに VE が割り当てられているかをチェックします。割り当てられていない場合は、チェック処理をスキップします。
2. ve_check_job の存在チェックを行います。存在しない場合や実行権限がない場合は、ログを出力して障害なしとして VE 障害検知をスキップします。
3. ヘルスチェックのタイミング（PRE-RUNNING または POST-RUNNING）に合わせて、チェックタイミングを表すオプション（-s：ジョブ開始時、-e：ジョブ終了時）、VE ノード番号、ジョブのセッション ID を指定して ve_check_job を呼び出します。
4. ve_check_job の戻り値が 0 の場合は、障害がないと判断します。それ以外の場合は障害があると判断します。障害検知した場合は、関連情報を Linux OS の/var/log/messages にワーニング情報として出力します。なお、複数の VE がジョブに割り当てられた場合は、途中で 1 つの VE で障害検知して場合でも、残りの VE についても継続して障害検知処理を実施します。
5. ジョブに割り当てられた VE のうち 1 つ以上の VE で障害検知した場合は、ヘルスチェックの結果として終了コード 1 を返します。

HCA 障害検知は以下の流れで行います。

1. ジョブに HCA が割り当てられているかをチェックします。割り当てられていない場合は、チェック処理をスキップします。
2. hcachk.sh の存在チェックを行います。存在しない場合や実行権限がない場合は、ログを出力して障害なしとして HCA 障害検知をスキップします。
3. HCA デバイス名を指定して hcachk.sh を呼び出します。ヘルスチェックのタイミングは限定しません。
4. hcachk.sh の戻り値が 0 の場合は、障害がないと判断します。それ以外の場合は障害があると

判断します。障害検知した場合は、関連情報を Linux OS の/var/log/messages にワーニング情報として出力します。なお、複数の HCA がジョブに割り当てられた場合は、途中で 1 つの HCA の障害を検知した場合には残りの HCA の障害検知処理を継続せずに処理を中断します。

5. ジョブに割り当てられた HCA のうち 1 つの HCA で障害検知した場合は、ヘルスチェックの結果として終了コード 2 を返します。

19.5.3 障害検知ノードの処置設定

ノードヘルスチェックで障害を検知した場合は、障害が起こったノードに対して、unbind/down/nothing のいずれかの処置を実行ホスト単位で選択することができます。

unbind を選択した場合は、障害検知したノードをバインドしている全てのキューからアンバインドされます。障害検知のきっかけとなったジョブを除いて、当該ノード上で実行しているジョブはそのまま継続実行します。後方にアサインされているジョブは、アサインを取り消して他のノードに再アサインされます。VE や GPU 等で障害が発生した場合に、障害が当該ジョブにのみ影響し、同一ノード上で実行している他のジョブに影響がない場合は、unbind を選択します。

down を選択した場合は、障害検知したノードを LINKDOWN します。障害検知のきっかけとなったジョブを除いて、当該ノード上で実行しているジョブはストールします。JobManipulator の実行中ジョブの強制リラン機能が ON の場合は、当該ジョブを強制リランします。後方にアサインされているジョブはアサインを取り消して他のノードに再アサインされます。ノード上で実行している全てのジョブに影響がある障害の場合は、down を選択します。

nothing を選択した場合は、障害検知したノードに対して何もしません。

qmgr(1M)の下記のサブコマンドで設定します。設定には操作員権限が必要です。

qmgr(1M)サブコマンド
set job_server health_check_action = <action> job_server_id=<jsvid>
set job_server health_check_action = <action> job_server_id=(<jsvids>)

以下は JSV の 1 番に対して UNBIND を設定する場合の設定例です。

```
$ qmgr -Po
Mgr: set job_server health_check_action = unbind job_server_id=1
Set node health check action.
```

設定結果は、qstat(1)コマンドの-S -f オプションで以下のように表示されます。

```
$ qstat -Qf
```

```
Job Server Name: JobServer0035
    Job Server Number  = 1
      :
    HCA Failure Check = OFF
    Health Check Action = UNBIND
:
```

19.5.4 ユーザ通知用スクリプトの設定

リクエスト投入時に-m a オプションを指定した場合は、ジョブ終了時のノードヘルスチェックで障害を検知すると、障害検知したリクエストのオーナーにメールで通知します。なお、ジョブ開始時に障害を検知した場合は、ジョブの実行をまだ開始していない、かつ、QUEUED 状態にロールバックして再実行するのでユーザ通知は行いません。

以下は通知メールのフォーマットです。

```
NQSV request: <rid> aborted.
Request name:  <ジョブスクリプト名>
Request owner: <オーナー>
Mail sent at:  <送信日時>
[jobno <ジョブ番号>]: Job has aborted by node health check error.
```

ジョブ投入時に-M オプションで通知用のメールアドレスも指定した場合は、指定されたメールアドレスにメール通知を行います。ジョブ投入時に-m a オプションを指定しなかった場合は、メール通知を行いません。既定値ではバッチホストにある下記のシェルスクリプトでメール通知を行います。

/opt/nec/nqsv/sbin/nofity_prog/nqsv_notify.sh

上記のシェルスクリプトを変更することで、通知手段や通知内容、通知先等をカスタマイズすることができます。また、サイトに特化した通知用スクリプトがある場合は、バッチサーバのコンフィグファイルに当該スクリプトのパスを指定することができます。詳細は、2.3.18 ユーザ通知用スクリプトの設定 を参照してください。

ユーザ通知用スクリプト

ユーザ通知用スクリプトは、各サイトの運用ポリシーに合わせてシステム管理者が作成できます。スクリプト名と配置場所も決めることができます。作成したスクリプトはバッチサーバのコンフィグファイルにパスを指定して適用してください。

ユーザ通知スクリプト実行時には、NQSV から以下の環境変数がセットされます。

環境変数名	説明	環境変数の値

NQSNOTIFY_RID	リクエストID	通常リクエスト： <シーケンス番号>.<ホスト名> パラメトリックリクエスト： <シーケンス番号>[].<ホスト名>
NQSNOTIFY_TITLE	メールのタイトル	NQSV request:<リクエストID> aborted.
NQSNOTIFY_CONTENTS	メールの本文	Request name:　<ジョブスクリプト名> Request owner:　<オーナー> Mail sent at:　<送信日時> [jobno <ジョブ番号>]: Job has aborted by node health check error.
NQSNOFITY_REQOWNER_EMAIL	リクエストのオーナーのメールアドレス	メールアドレス
NQSNOTIFY_REQOWNERGRP	リクエストのオーナーのグループ名	グループ名
NQSNOTIFY_QUEUE	リクエストの投入キュー名	キュー名

19.5.5 Elapse マージンによるヘルスチェック時間の調整

ヘルスチェック処理はリクエストの PRE-RUNNING と POST-RUNNING で実行されます。ノードヘルスチェックに時間がかかる場合、PRE-RUNNING と POST-RUNNING の時間が長くなります。JobManipulator の Elapse マージンを適切に設定することでこの時間をチューニングしてください。それにより、後方にアサインされているリクエストとリソース占有時間が重複せずに運用することができます。Elapse マージンの詳細については、JobManipulator 編の 3.1.7 Elapse マージンを参照してください。

19.5.6 障害検知リクエストの再実行

ノードヘルスチェックで障害を検知した場合は、障害検知するきっかけとなるリクエストを再実行します。

ジョブ開始時(PRE-RUNNING)に障害を検知した場合は、当該リクエストを PRE-RUNING 状態から QUEUED 状態に戻して再実行します。1 リクエストに複数ジョブがあった場合は、一部のジョブのみヘルスチェックが失敗した場合は、ヘルスチェックが成功したジョブに対してジョブ終了時のヘルスチェックを経て QUEUED 状態に戻します。

ジョブ終了時(POST-RUNNING)に障害を検知した場合は、当該リクエストをリランして QUEUED 状態に戻して再実行します。リクエストの状態遷移理由は"SYSTEM_FAILURE"となります。

19.5.7 障害検知リクエストのアカウンティングと課金

ノードヘルスチェックで障害を検知した場合は、障害検知するきっかけとなるリクエストのアカウントにノードヘルスチェック失敗のフラグを出力します。アカウントの参照コマンド scacctjob(1)/scactreq(1)を使って--hw-failure オプションを指定してジョブ単位とリクエスト単位で障害情報を確認することができます。障害検知した場合は、下記のように HW FAILURE カラムに 20 と表示されます。複数ジョブ指定して投入したリクエストは、1 以上のジョブで障害を検知した場合、リクエストの障害検知フラグを設定します。アカウントの参照コマンドの表示イメージは下記のようになります。

```
$ scacctjob --hw-failure -I 1
=============================================================================================
JOB                              REQUEST  … QUEUED   START    END                  CPU    REAL      HW
ID     REQUEST_ID                NAME     … NAME     TIME     TIME     TIME       (SECS) (SECS) FAILURE
=============================================================================================
32816 1.bsv1(0000)               job1     …  bq       20:29:48 20:29:48 20:32:48    0.02  179.75     20
34769 1.bsv1(0001)               job1     …  bq       20:29:48 20:29:48 20:32:49    0.08  180.12      0
```

```
$ scacctreq --hw-failure -I 1
=============================================================================================================
REQUEST                   REQUEST  USER       QUEUE    QUEUED   START    END          CPU    REAL             HW
ID                        NAME     NAME       NAME     TIME     TIME     TIME       (SECS) (SECS) STATUS  FAILURE
=============================================================================================================
1.bsv1                    job1     user       bq       20:29:48 20:29:48 20:32:49    0.02     180 DELETED      20
```

また、障害検知するきっかけとなるリクエストに対しても課金を行いません。なお、パラメトリックリクエストの場合は、障害を検知したサブリクエストに対して課金を行いませんが、正常に実行できたサブリクエストに対して課金を行います。

第20章　冗長化機能

バッチサーバ、アカウンティングサーバ、JobManipulator の各コンポーネントを二重化（冗長化）し、NQSV システムをダウンさせることなく、継続稼働させることができます。

NQSV では、NQSV 単独による冗長化および CLUSTERPRO による冗長化に対応しています。

20.1 NQSV単独による冗長化

NQSV では CLUSTERPRO を使用せず、簡易的に、バッチサーバ、アカウンティングサーバ、JobManipulator を二重化することができます。この NQSV 単独での二重化を、冗長化機能と呼びます。

冗長化機能では、冗長化機能用簡易障害検知スクリプトや Zabbix などの OSS の運用監視ソフトウェアを利用して、運用系ホストの障害監視を行い、障害発生時にはあらかじめ用意していた待機系ホスト（代替ホスト）で障害発生ホストの機能を代行させます。

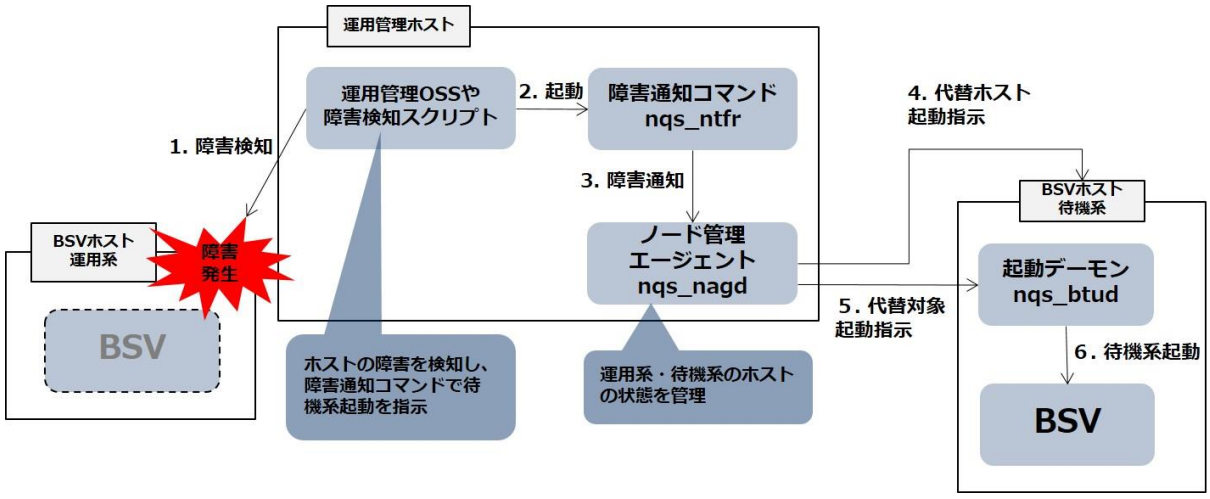


図 20-1：冗長化機能の全体像

障害検知や代替ホストへの切り替え制御については、ノード管理エージェントを利用します。（20. 障害検知・電源制御 も合わせて参照してください。）

冗長化機能の起動デーモン（nqs_btud）は、代替ホストに常駐し、ノード管理エージェントからの指示を受けて、冗長化対象のコンポーネントを起動します。

なお、本章では主にバッチサーバを二重化することを例にとり説明しますが、同様の方法でアカウンティングサーバおよび JobManipulator を二重化することができます。

20.1.1 起動デーモンのインストール

冗長化機能を使用する場合は、待機系の代替ホストに、NQSV/ResourceManager パッケージをインストールしてください。

インストール後、起動デーモン（nqs_btud）の起動・停止は、root 権限にて systemctl を使用します。

【起動方法】

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

【停止方法】

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

起動デーモンは状況に応じて下記のシェルスクリプトを実行します。これらのシェルスクリプトはお使いの環境に合わせてあらかじめ作成する必要があります。各シェルスクリプトの作成方法等については、21.1.2(2).起動デーモンの設定 にて説明します。

1. 起動環境の構築用スクリプト(/opt/nec/nqsv/sbin/btu_prog/btup_configure.sh)
2. 冗長化対象の起動用スクリプト(/opt/nec/nqsv/sbin/btu_prog/btup_bootup.sh)
3. 起動環境の停止用スクリプト(/opt/nec/nqsv/sbin/btu_prog/rlbk_configure.sh)
4. 冗長化対象の停止用スクリプト(/opt/nec/nqsv/sbin/btu_prog/rlbk_bootup.sh)

20.1.2 冗長化機能の設定

(1) 事前準備

バッチサーバホストにおいて冗長化機能を使用する場合、以下の設定を行ってください。

- データベースが存在する/var/opt/nec/nqsv 配下が運用系・待機系で共有されるよう共有ディスクに構成してください。
    - 定義ファイル等が置かれる /etc/opt/nec/nqsv 配下も共有してください。
    - ただし、待機系・運用系の間で同期が取れるのであれば、それぞれのローカルディスク上に/etc/opt/nec/nqsvがあってもかまいません。
    -
- 運用系・待機系ともに、冗長化されるコンポーネント（バッチサーバやJobManipulatorなど）は、systemdによる自動起動および終了を無効にしてください。

・

・　冗長化構成で運用する場合、運用系のバッチサーバはnqs_bsvd の-a オプションで仮想IPアドレスを指定して起動してください。

```
# /opt/nec/nqsv/sbin/nqs_bsvd -a <Virtual_IP_Address>
```

仮想 IP アドレスが指定された場合、バッチサーバは以下の動作を行います。

バッチサーバ、およびバッチサーバから起動されるルーティングサーバ、ステージングサーバがリモートプロセスへTCP 接続を行う際、 ローカル側アドレスとして仮想IPアドレスを用います。これにより、リモートプロセスから見たバッチサーバのIP アドレスは、常に仮想IP アドレスとなります。 したがって、フェールオーバーを通して常に同じIPアドレスに保たれます。

NQSV/API が返すバッチサーバホスト名は、仮想IP アドレスに対応したホスト名になります。よって、NQSV/API を使ってバッチサーバ名を取得しているクライアント（CUIコマンド、スケジューラ等）が 表示するホスト名は、仮想IP アドレスに対するホスト名となります。 したがって、フェールオーバーを通して常に同じホスト名となります。

・

・　待機系のバッチサーバホストでは、バッチサーバ起動設定ファイル（/etc/opt/nec/nqsv/nqs_bsv.env）において、BSV_PARAMを以下のように設定してください。この指定により仮想IPアドレスを引き継ぎます。

・　BSV_PARAM="-a ${VIRTUAL_IP_ADDRESS}"

・

(2)　起動デーモンの設定

起動デーモンは代替ホスト上に常駐して、ノード管理エージェントからの代替指示を待ち受けます。代替指示を受け付けると、ホスト上にあらかじめ設置されたシェルスクリプトを使用して、代替対象コンポーネントの起動・停止を行います。

a)　起動デーモンのオプション

起動デーモンの起動時オプションは以下の通りです。

```
# nqs_btud [-p <port>] [-c <1_path>] [-b <2_path>] [-C <3_path>] [-B <4_path>]
```

<port>は起動デーモンが使用する TCP ポート番号を指定します。

指定可能な範囲は 0 から 65535 です。

省略した場合やこの範囲から外れた値を指定した場合は、49600 を使用します。

<1_path> <2_path> <3_path> <4_path>は、起動デーモンが使用するシェルスクリプトのパス名を指定します。

パス名の最大長は 1023 文字です。

省略した場合や、パス名が最大長を超過した場合はデフォルトのパスを使用します。

指定項目	スクリプト種別	デフォルトのパス
<1_path>	起動環境の構築用スクリプト	/opt/nec/nqsv/sbin/btu_prog/btup_configure.sh
<2_path>	冗長化対象の起動用スクリプト	/opt/nec/nqsv/sbin/btu_prog/btup_bootup.sh
<3_path>	起動環境の停止用スクリプト	/opt/nec/nqsv/sbin/btu_prog/rlbk_configure.sh
<4_path>	冗長化対象の停止用スクリプト	/opt/nec/nqsv/sbin/btu_prog/rlbk_bootup.sh

b)　起動デーモンが使用するシェルスクリプト

起動デーモンが使用する各シェルスクリプトは、実際の運用に合わせて作成してください。以下に各スクリプトの詳細を説明します。

・　起動環境の構築用スクリプト

冗長化対象コンポーネントの起動に先立って、代替ホストの環境構築を行うためのスクリプトです。

仮想 IP アドレスが第一引数に渡されます。

仮想 IP アドレスの追加やデータベースが置かれている共有ファイルシステムのマウント等を行ってください。

【実装例】

```
#!/bin/sh
# sample /opt/nec/nqsv/sbin/btu_prog/btup_configure.sh
export PATH=/sbin:/bin:/usr/bin
# mount NFS for configuration files & database files
case $1 in
192.168.1.1)
    mount -t nfs nfs_server:/exports/etc/opt/nec/nqsv/host-a /etc/opt/nec/nqsv || exit $?
    mount -t nfs nfs_server:/exports/var/opt/nec/nqsv/host-a /var/opt/nec/nqsv || exit $?;;
192.168.1.2)
    mount -t nfs nfs_server:/exports/etc/opt/nec/nqsv/host-a /etc/opt/nec/nqsv || exit $?
    mount -t nfs nfs_server:/exports/var/opt/nec/nqsv/host-b /var/opt/nec/nqsv || exit $?;;
192.168.1.3)
    mount -t nfs nfs_server:/exports/etc/opt/nec/nqsv/host-c /etc/opt/nec/nqsv || exit $?
    mount -t nfs nfs_server:/exports/var/opt/nec/nqsv/host-c /var/opt/nec/nqsv || exit $?;;
esac
# add secondary ip address / subnetmask to eth0 temporarily
ip address add $1/24 dev eth0
```

・　冗長化対象の起動用スクリプト

冗長化対象のコンポーネントを起動するスクリプトです。

仮想 IP アドレスが環境変数 VIRTUAL_IP_ADDRESS で渡されます。

【実装例】

```
#!/bin/sh
# sample /opt/nec/nqsv/sbin/btu_prog/btup_bootup.sh
export PATH=/bin:/usr/bin
if ps -ef | fgrep -q '/opt/nec/nqsv/sbin/nqs_bsvd'
then
    systemctl stop nqs-bsv.service
fi
systemctl start nqs-bsv.service || exit $?
if ps -ef | fgrep -q '/opt/nec/nqsv/sbin/nqs_jmd'
then
    systemctl stop nqs-jmd.service
fi
systemctl start nqs-jmd.service
```

・　起動環境の停止用スクリプト

代替運用後、障害ホストが復旧した場合、代替ホストの環境を初期状態に戻すためのスクリプトです。なお、代替処理中にトラブルが発生した場合等にも、一度構築した代替ホストの環境を初期化して元に戻すために使用します。

仮想 IP アドレスが第一引数で渡されます。

仮想 IP アドレスの削除やデータベースが置かれている共有ファイルシステムのアンマウント等を行ってください。

【実装例】

```
#!/bin/sh
# sample /opt/nec/nqsv/sbin/btu_prog/rlbk_configure.sh
export PATH=/sbin:/bin:/usr/bin
# add secondary ip address / subnetmask
if ip address show dev eth0 | fgrep -q -w "$1"
then
    ip address del $1/24 dev eth0 || exit $?
fi
```

```
# unmount NFS for configuration files & database files
if mountpoint -q /var/opt/nec/nqsv
then
    umount /var/opt/nec/nqsv || exit $?
fi
if mountpoint -q /etc/opt/nec/nqsv
then
    umount /etc/opt/nec/nqsv
fi
```

・　冗長化対象の停止用スクリプト

代替運用後、障害ホストが復旧した場合、代替ホストで運用中の冗長化対象コンポーネントを停止するスクリプトです。なお、代替処理中にトラブルが発生した場合等にも、代替ホスト上の冗長化対象コンポーネントの停止に使用します。

仮想 IP アドレスが環境変数 VIRTUAL_IP_ADDRESS で渡されます。

【実装例】

```
#!/bin/sh
# sample /opt/nec/nqsv/sbin/btu_prog/rlbk_bootup.sh
export PATH=/bin:/usr/bin
if ps -ef | fgrep -q '/opt/nec/nqsv/sbin/nqs_jmd'
then
    systemctl stop nqs-jmd.service || exit $?
fi
if ps -ef | fgrep -q '/opt/nec/nqsv/sbin/nqs_bsvd'
then
   systemctl stop nqs-bsv.service stop
fi
```

(3)　ノード管理エージェントの設定

冗長化機能では、運用系・待機系の各ホストの状態管理や代替ホストへの切り替えの制御にノード管理エージェントを使用します。

ノード管理エージェントにおいては、運用系ホストを監視対象ホストと呼びます。

また、待機系ホストのことを代替ホストと呼びます。

冗長化機能でノード管理エージェントを運用するために下記を設定する必要があります。

1) 代替ホストで、BMC が搭載されている場合は利用できるように設定

※BMC にアクセスするためのパスワードを設定した場合には、パスワードを記載したファイルを作成してください。3) 4)の設定でこのパスワードファイルのパスを指定します。

2) 代替ホストに BMC が搭載されている場合、運用管理ホストに ipmitool をインストール

3) ノード管理エージェントに関する設定（/etc/opt/nec/nqsv/nag.conf）

4) 監視対象ホストおよび代替ホストに関する設定（/etc/opt/nec/nqsv/nag_backup）

本節では上記 3）と 4）について説明します。

・　/etc/opt/nec/nqsv/nag.conf（ノード管理エージェント設定ファイル）

ノード管理エージェントの一般的な設定をするファイルです。

冗長化機能に特有な設定項目は以下です。

これ以外の項目については 20.1.ノード管理エージェントの設定 をご覧ください。

パラメータ	値 (設定可能範囲または設定可能最大文字数)	説明
BTU_MAXWAIT_MIN	数値 (0～2147483647)	代替ホストとの通信タイムアウト時間を指定します。単位は分です。指定がない場合は10とします。 この時間内に通信ができなかった場合、この代替ホストは使用不可と見なします。

・　/etc/opt/nec/nqsv/nag_backup.conf（監視対象ホストと代替ホストに関する設定ファイル）

冗長化機能において、ノード管理エージェントが管理する監視対象ホストと代替ホストを設定するファイルです。冗長化機能ではこのファイルを基に障害が発生したホストをどの代替ホストに切り替えるかを判断します。

/etc/opt/nec/nqsv/nag_backup.conf の記述形式は下記のとおりです。

```
# 監視対象ホスト行
<IP_Address> <Virtual_IP_Address>
# 代替ホスト行
 * <IP_Address>[:<Port>] [<BMC_Address> [<BMC_User> <BMC_Password_file>]]
----  # 区切り行
# 監視対象ホスト行
     :
# 代替ホスト行
     :
```

※本ファイルの IP アドレスは、IPv4 の数値とドットによる表記です。

このファイルは、区切り行で区切られた、監視対象ホスト行と代替ホスト行の組の集まりです。
区切り行は 1 文字以上の連続する "-" からなる行です（前後に空白文字があっても構いません）。
#から改行文字まではコメントとして無視されます。
空白文字しかない行は空行として無視されます。

区切られた一つの組の中では、必ず監視対象ホスト行とその後に代替ホスト行を記述します。
一つの組の中に監視対象ホスト行と代替ホスト行の両方が揃っていない場合は、冗長化できません。
監視対象ホストに障害が発生すると、その組に記述された代替ホストに切り替えられます。
一つの組の中で、監視対象ホスト行と代替ホスト行はいずれも複数記述しても構いません。
複数の監視対象ホスト行がある場合、それらで代替ホストを共有します。
また、代替ホスト行が複数ある場合、記述順に使用されます。

監視対象ホスト行の書式は下記のとおりです。

```
<IP_Address> <Virtual_IP_Address>
```

<IP_Address>は監視対象ホストの物理 IP アドレスです。物理 IP アドレスに結びついているものであれば、ホスト名を記述しても構いません。ホスト名の最大長は 255 文字です。
<Virtual_IP_Address>は監視対象ホストの仮想 IP アドレスです。仮想 IP アドレスに結びついているものであれば、ホスト名を記述しても構いません。ホスト名の最大長は 255 文字です。

代替ホスト行の書式は下記のとおりです。
* <IP_Address>[:<Port>] [<BMC_Address> [<BMC_User> <BMC_Password_file>]]
行頭の "*" は必須です（前後に空白文字があっても構いません）。
<IP_Address>は代替ホストの物理 IP アドレスです。物理 IP アドレスに結びついているものであれば、ホスト名を記述しても構いません。
ホスト名の最大長は 255 文字です。
<Port>は TCP ポート番号です。指定可能な範囲は 0 から 65535 までの整数で、:の前後に空白文字を入れてはいけません。省略時は 49600 を使用します。
<BMC_Address>は BMC の IP アドレスです。BMC のホスト名を記述しても構いません。ホスト名の最大長は 255 文字です。
本設定を行うことで、代替処理実施時に最初に BMC を通して代替ホストの起動を行います。
<BMC_User>は BMC のユーザ名です。ユーザ名の最大長は 47 文字です。
<BMC_Password_file>は BMC のパスワードを記述したファイルのパス名です。
パス名の最大長は 1023 文字です。

<BMC_User>と<BMC_Password_file>を共に省略した場合は、/etc/opt/nec/nqsv/nag.conf に記述されている IPMIUSER と IPMIPASSWD の内容を使用します。(運用環境に応じて、設定してください。)

> **【注意】**<*BMC_Address*>以降を全て省略すると、そのホストは BMC 非搭載として扱います。従って、代替ホストが BMC 非搭載の場合、あらかじめ起動しておく必要があります。

/etc/opt/nec/nqsv/nag_backup.conf の記述例を示します。

```
# sample
# for running nodes
# physical ip address   virtual ip address
192.168.1.100          host-a
192.168.1.101          host-b
# for waiting nodes
# physical ip address   BMC ip address   BMC username   BMC password
* backup-a             192.168.1.230     bmc-user     /etc/opt/nec/nqsv/pw/bmc-pw
----
192.168.1.103     host-c
* backup-b:50000       192.168.1.232     bmc-admin  /etc/opt/nec/nqsv/pw/admin-pw
* backup-c              # This host doesn't have BMC.
```

/etc/opt/nec/nqsv/nag_backup.conf を更新した場合、ノード管理エージェントを再起動してください。

```
# systemctl restart nqs-nag.service
```

20.1.3 障害検知機能の設定

(1) 冗長化機能用簡易障害検知スクリプトによる障害検知

Zabbix などの OSS の運用管理ソフトウェアを使用しない場合、冗長化機能用簡易障害検知スクリプトにて、簡易的に障害検知を行うことができます。

冗長化機能用簡易障害検知スクリプトは、実行環境に応じて編集してください。

冗長化機能用簡易障害検知スクリプトは、運用管理ホストにインストールされています。

インストールパス： /opt/nec/nqsv/sbin/check_alive.sh

処理概要：

監視対象ホストの生存を、pingコマンドで確認。

応答がない場合には、障害通知コマンドnqs_ntfrにて、ノード管理エージェントに障害を通知。ノード管理エージェントは代替ホストへの切り替えを開始。

障害となったホストの電源を落とす。(ipmitoolでBMCを通して実施。)

本スクリプトを使用して監視対象ホストの障害検知を行う場合には、運用管理ホストにて、以下の設定を行ってください。

・　本スクリプトを編集して、監視対象ホストの情報を設定します。

"# HostList"から始まるコメント行群の後に、監視対象ホストの物理 IP アドレス、BMC の IP アドレス、BMC のユーザ名、BMC のパスワードを設定してください。

/opt/nec/nqsv/sbin/check_alive.shの設定例を示します。(環境に応じて、変更してください。)

```
#!/bin/bash
export PATH=/bin:/usr/bin:/opt/nec/nqsv/sbin
tempfile=`mktemp`
sed '1,/^# HostList/d; /#/d' $0 > $tempfile
exec < $tempfile
while read host bmc user pw
do
    if ping -c 4 $host > /dev/null 2>&1
    then
        :
    else
        nqs_ntfr -s $host
        ipmitool -I lanplus -H $bmc -U $user -P $pw chassis power off
    fi
done
rm $tempfile
exit 0
# HostList
# Information of the hosts to check alive are written in following lines.
# Physical_IP_Address BMC_IP_Address BMC_User_Name BMC_Password
192.168.1.100         192.168.200.10   bmc-admin       admin.pw
192.168.1.101         192.168.200.11   root0             PwOfRoot
```

【注意】冗長化機能用簡易障害検知スクリプトでは、監視対象ホストで障害が発生した場合、障害ホスト上の全ての冗長化対象コンポーネントと仮想 IP アドレスを、代替ホストへ引き継ぐために、BMC を通して障害ホストの電源を落としています。
監視対象ホストが BMC を搭載していない場合は、代替運用中、障害ホスト上で冗長化対象コンポーネントや仮想 IP アドレスの設定が残存することがないように、シャットダウンするなどの対処をしてください。

・ 上記設定後、運用管理ホストにて、cron で本スクリプトを定期的に実行するよう設定してください。

以上の設定を行うことにより、冗長化機能用簡易障害検知スクリプトは、定期的にバッチサーバなどが動作するホストの生存を確認し、応答がない場合は、障害検知時に障害通知コマンド nqs_ntfr を実行し、ノード管理エージェントに障害を通知します。

障害通知コマンドから障害通知を受けたノード管理エージェントは、障害が発生したホストを代替ホストに切り替えます。

(2) OSSによる障害検知

Zabbix などの OSS の運用管理ソフトウェアを利用して監視対象ホストの障害検知を行う場合、障害検知時に OSS で障害通知コマンド nqs_ntfr を実行するように設定してください。

障害通知コマンドはノード管理エージェントに障害を通知します。障害通知を受けたノード管理エージェントは、障害が発生したホストを代替ホストに切り替えます。

障害通知コマンドは運用管理ホストの以下のパスにインストールされます。
障害通知コマンド：　/opt/nec/nqsv/sbin/nqs_ntfr

監視対象ホストで障害が発生したことを、ノード管理エージェントに通知するための障害通知コマンド nqs_ntfr の実行形式は以下のとおりです。

```
nqs_ntfr -s <hostname> | <IP_address>
```

<hostname> もしくは <IP_address> で指定したバッチサーバなど冗長化対象が動作するホストで、障害が発生したことをノード管理エージェントに通知します。

ノード管理エージェントは代替ホストへの切り替えを開始します。

(3) 障害の記録

ノード管理エージェントは障害通知を受けると、ノード管理エージェントのログファイル

(/var/opt/nec/nqsv/node_agent_log)にメッセージを記録します。

障害が通知され、正常に代替ホストに切り替わると、ログレベルが 1 以上の場合、下記のメッセージが記録されます。

xxx.xxx.xxx.xxx は障害が発生したホストの物理 IP アドレス、yyy.yyy.yyy.yyy は代替ホストの物理 IP アドレスです。

```
[INFO  ] IP: xxx.xxx.xxx.xxx was switched. => IP: yyy.yyy.yyy.yyy
```

障害が発生しても代替ホストが見つからなかったときは、下記のメッセージが記録されます。xxx.xxx.xxx.xxx は障害が発生した監視対象ホストの物理 IP アドレスです。

```
[ERROR ] no available waiting node. (IP: xxx.xxx.xxx.xxx)
```

切り替えようとした代替ホストの電源投入に失敗した時は、下記のメッセージが記録されます。xxx.xxx.xxx.xxx は障害が発生した監視対象ホストの物理 IP アドレスです。

この時、他に代替ホストがあれば、他の代替ホストを使用します。

```
[ERROR ] failed to power on. (IP: xxx.xxx.xxx.xxx)
```

切り替えようとした代替ホストが応答しない時は、下記のメッセージが記録されます。xxx.xxx.xxx.xxx は障害が発生した監視対象ホストの物理 IP アドレスです。

この時、他に代替ホストがあれば、他の代替ホストを使用します。

```
[ERROR ] no response waiting node. (IP: xxx.xxx.xxx.xxx)
```

切り替えようとした代替ホストの起動処理（起動環境の構築用スクリプト、冗長化対象の起動用スクリプトの実行）に失敗した場合は、自動的にそのホストの停止処理（冗長化対象の停止用スクリプト・起動環境の停止用スクリプトの実行）を行います。

停止に成功した場合は、下記のメッセージが記録されます。

xxx.xxx.xxx.xxx は障害が発生した監視対象ホストの物理 IP アドレスです。

この時、他に代替ホストがあれば、他の代替ホストを使用します。

```
[ERROR ] failed to boot up. (IP: xxx.xxx.xxx.xxx)
```

代替ホストの起動処理に失敗し、さらに、そのホストの停止処理にも失敗した場合は、下記のメッ

セージが記録されます。xxx.xxx.xxx.xxx は障害が発生したホストの物理 IP アドレスです。

この時には他に代替ホストがあっても、他の代替ホストへの切り替えは行いません。

```
[ERROR ] failed to boot up and roll back. (IP: xxx.xxx.xxx.xxx)
```

20.1.4 障害発生したホストの復旧

(1) 状況確認の方法

ノード管理エージェントは、冗長化機能による障害発生ホストの切り替え状況を、ファイルに記載します。このファイルを、状況記録ファイルと呼びます。このファイルを参照することにより、状況確認ができます。

状況記録ファイルは以下のパスに出力されます。

状況記録ファイル：　/var/opt/nec/nqsv/nag_redundancy

状況記録ファイルのパス名は、ノード管理エージェントの起動時オプションで変更できます。

nqs_nagd -R <path>

ノード管理エージェントは障害通知を受けると、状況記録ファイルを更新します。このファイルは毎回上書きしますので、常に最新の状況が記録されています。また、このファイルはログレベルに関わらず、更新されます。

【注意】状況記録ファイルは、参照のみとしてください。
ノード管理エージェントは、本ファイルを状況記録の用途以外にも利用していますので、ファイルの内容を手動で変更しないでください。

状況記録ファイルは(1)障害が発生したホストのリスト、(2)見出し、(3)代替ホストの状態 の三つが組となった形式で記録されます。

この組は、ノード管理エージェントの設定ファイル /etc/opt/nec/nqsv/nag_backup.conf (21.1.2(3). ノード管理エージェントの設定 参照) における監視対象ホストと代替ホストの組に相当します。

この監視対象ホストと代替ホストの組と同じ数だけ、繰り返し記録されます。

```
記録日時
障害が発生した監視対象ホストのリスト... (1)
見出し... (2)
代替ホストの状態... (3)
：
障害が発生したホストのリスト... (1)
```

```
見出し...(2)
代替ホストの状態...(3)
:
```

a)　障害が発生したホストのリスト

```
defected hosts=〈物理IPアドレス〉… | (none)
```

監視対象ホストと代替ホストの組において、障害が発生した監視対象ホストのリストが記録されます。

障害が発生した監視対象ホストが複数存在する場合は、空白で区切って記録されます。

障害が発生した監視対象ホストがない場合は、(none)と記録されます。

b)　見出し

```
# ip waiting host   state       ip switched host
```

代替ホストの状態を表すための見出しです。

c)　代替ホストの状態

<代替ホストの物理 IP アドレス> <状態> [<切り替えられた障害ホストの物理 IP アドレス>]

1 行に一つの代替ホストの状態が記録されます。

代替ホストが複数有る場合、代替ホストの数だけ、この行が存在します。

状態は 5 種類あり、下記の意味を持ちます。

状態を示す記述	意味
waiting	待機状態です。
switching	切り替えのための処理中です。
switched	切り替えが発生して、代替運用状態です。
recovering	復旧のための処理中です。
failed	切り替え時に問題が発生しました。 このホストは使用できません。

障害ホストと代替ホストが切り替わった場合には、状態が switched となり、行末に切り替えられた障害ホストの物理アドレスが記録されます。

状況記録ファイルの出力例を以下に示します。

この例では、監視対象ホスト 192.168.1.100 で障害が発生し、ホスト 192.168.1.200 がその代替をしています。

また、監視対象ホスト 192.168.1.101 でも障害が発生していますが、どのホストでも代替されていません。

代替ホスト 192.168.1.201 では何らかのエラーが起こり、代替ホストとして使用できません。

```
11/11 11:11:11
defected hosts= 192.168.1.100 192.168.1.101
# ip waiting host   state       ip switched host
192.168.1.200       switched    192.168.1.100
192.168.1.201       failed
192.168.1.202       waiting
defected hosts= (none)
# ip waiting host   state       ip switched host
192.168.1.203       waiting
```

(2)　障害発生ホストの復旧手順

障害が発生した監視対象ホストが復旧し、運用に戻したい場合の手順を説明します。

元の運用に戻す手順は、以下の通り大きく 2 段階に分かれます。

a)　代替ホストの運用停止

nqs_ntfr -r コマンドを使用して、ノード管理エージェントに監視対象ホストの復旧処理開始を指示し、代替ホストの運用を停止させます。

b)　監視対象ホストの運用再開

障害から復旧した、運用に戻したい監視対象ホストを起動した後、

nqs_ntfr -R コマンドを使用して、ノード管理エージェントに復旧完了を通知します。

ここでは a).状況確認の方法 で例示した状況記録ファイルの状態において、ホスト 192.168.1.100 を復旧させる場合を例に挙げて説明します。

i)　　代替ホストの運用停止

最初に、運用管理ホスト上にて、障害通知コマンド nqs_ntfr の-r オプションで運用に戻したい監視対象ホストの復旧処理開始をノード管理エージェントに指示し、代替ホストの運用を停止させます。ここで引数に指定するホスト名（または IP アドレス）は、（代替ホストではなく）運用に戻したい監視対象ホストのホスト名です。

```
# nqs_ntfr -r <ホスト名> | <IPアドレス>
```

今回の例では、下記のコマンドを実行します。

```
# nqs_ntfr -r 192.168.1.100
```

代替ホストの運用停止に成功すると、ノード管理エージェントのログファイルに、ログレベル 1 以上の場合、下記のメッセージが記録されます。yyy. yyy. yyy. yyy は代替ホストの物理 IP アドレスです。

```
[INFO  ] succeeded to stop. (IP: yyy. yyy. yyy. yyy)
```

また、同時に状況記録ファイルが更新されます。該当する代替ホストの状況が waiting に変わります。今回の例では、192.168.1.200 が waiting に変わりました。

```
11/17 17:17:17
defected hosts= 192.168.1.100 192.168.1.101
# ip waiting host   state       ip switched host
192.168.1.200       waiting
192.168.1.201       failed
192.168.1.202       waiting
defected hosts= (none)
# ip waiting host   state       ip switched host
192.168.1.203       waiting
```

ここまでの手順でなんらかの問題が発生した場合、後述の ・代替ホスト停止時のトラブルシューティング を参照してください。

ii)　監視対象ホストの運用再開

代替ホストの停止を確認した後で、運用に戻したい監視対象ホストを起動します。その後、冗長化対象となっていたコンポーネントも起動します。

運用に戻したい監視対象ホストの起動完了後、障害通知コマンド nqs_ntfr の-R オプションで、ノード管理エージェントに復旧完了を通知します。ここで引数に指定するホスト名は運用に戻したい監視対象ホストのホスト名です。

```
# nqs_ntfr -R <ホスト名> | <IPアドレス>
```

今回の例では、下記のコマンドを実行します。

nqs_ntfr -R 192.168.1.100

ノード管理エージェントのログファイルに、ログレベル 1 以上の場合、下記のメッセージが記録さ

れます。ここで xxx.xxx.xxx.xxx は運用に戻したホストの物理 IP アドレスです。

[INFO　] IP: xxx.xxx.xxx.xxx was recovered.

同時に状況記録ファイルが更新されます。

障害が発生したホストのリスト（defected hosts）から、復旧したホストが消えます。

今回の例では、192.168.1.100 が消えました。

これで復旧は完了です。

```
11/17 18:00:00:00
defected hosts= 192.168.1.101
# ip waiting host   state       ip switched host
192.168.1.200       waiting
192.168.1.201       failed
192.168.1.202       waiting
defected hosts= (none)
# ip waiting host   state       ip switched host
192.168.1.203       waiting
```

・　代替ホスト停止時のトラブルシューティング

1) 代替ホストの状況が recovering となっている

nqs_ntfr -r コマンドを実行した際、状況記録ファイルにおいて、以下のように該当するホストの状況が recovering となっている場合は、まだ処理中です。

```
11/17 17:17:17
defected hosts= 192.168.1.100 192.168.1.101
# ip waiting host   state       ip switched host
192.168.1.200       recovering
192.168.1.201       failed
192.168.1.202       waiting
```

waiting または後述の failed となるまでお待ちください。

2) 状況記録ファイルが更新されない

nqs_ntfr -r を実行しても状況記録ファイルが更新されていない場合は、

nqs_ntfr -r の引数に指定したホストが誤っているか、あるいは他のノード管理エージェントの状況記録ファイルを参照していることが考えられます。nqs_ntfr コマンドに指定するホスト名や、参照している状況記録ファイルの場所等をご確認ください。

3) 代替ホストの停止に失敗した

もし、何らかの原因で代替ホストの停止に失敗した場合、ログファイルに下記のメッセージが記録されます。ここで yyy. yyy. yyy. yyy は停止に失敗した代替ホストの物理 IP アドレスです。

```
[ERROR ] failed to stop. (IP: yyy. yyy. yyy. yyy)
```

この時､状況記録ファイルには以下のように failed と記録され､運用に戻したい監視対象ホストは､障害が発生したホストのリスト（defected hosts）に残ります。

```
11/18 17:17:17
defected hosts= 192.168.1.100 192.168.1.101
# ip waiting host   state       ip switched host
192.168.1.200       failed        192.168.1.100
192.168.1.201       failed
192.168.1.202       waiting
defected hosts= (none)
# ip waiting host   state       ip switched host
192.168.1.203       waiting
```

この場合、停止に失敗した代替ホストを手動でシャットダウンしてください。

その後、運用に戻したい監視対象ホストを起動し、障害通知コマンド nqs_ntfr の-R オプションでノード管理エージェントへの復旧完了通知を行ってください。

上記例の場合、

停止に失敗した代替ホスト192.168.1.200を手動でシャットダウン

運用に戻したい監視対象ホスト192.168.1.100 を起動

nqs_ntfr -R 192.168.1.100を実行して、復旧完了通知

となります。

これで復旧は完了です。

状況記録ファイルを確認すると、障害が発生したホストのリストから 192.168.1.100 が消え、停止に失敗した代替ホスト 192.168.1.200 の行からも 192.168.1.100 が消えます。

```
11/18 18:00:00
defected hosts= 192.168.1.101
# ip waiting host   state       ip switched host
192.168.1.200       failed
```

```
192.168.1.201       failed
192.168.1.202       waiting
defected hosts= (none)
# ip waiting host   state       ip switched host
192.168.1.203       waiting
```

代替ホスト 192.168.1.200 の状態は failed のままとなります。代替ホストの復旧については、(4).エラーが発生した代替ホストの復旧手順 を参照してください。

(3)　　代替されていない障害発生ホストの復旧手順

障害発生時に何らかの理由で代替ホストに切り替わらなかった、障害発生ホストが復旧し、運用に戻したい場合の手順を説明します。

ここでは、以下の状況記録ファイルの状態を例にとり説明します。代替ホスト 192.168.1.201 は何らかのエラーが起こり、代替ホストとして使用できない状態です。監視対象ホスト 192.168.1.101 では障害が発生していますが、どのホストでも代替されていません。

```
11/11 11:11:11
defected hosts= 192.168.1.100 192.168.1.101
# ip waiting host   state       ip switched host
192.168.1.200       switched    192.168.1.100
192.168.1.201       failed
192.168.1.202       waiting
defected hosts= (none)
# ip waiting host   state       ip switched host
192.168.1.203       waiting
```

前処理として、状況記録ファイルで failed となっている代替ホストについて、冗長化対象コンポーネントの停止や環境の初期化を確実にするため、シャットダウンすることをお勧めします。

障害から復旧した、運用に戻したい監視対象ホストを起動します。

運用に戻したい監視対象ホストの起動完了後、障害通知コマンド nqs_ntfr の-R オプションで、ノード管理エージェントに復旧完了を通知します。ここで引数に指定するホスト名は運用に戻したい監視対象ホストのホスト名です。今回の例では、下記のコマンドを実行します。

```
# nqs_ntfr -R 192.168.1.101
```

状況記録ファイルを確認すると、障害が発生したホストのリストから 192.168.1.101 が消えています。これで復旧は完了です。

```
11/19 17:17:17
defected hosts= 192.168.1.100
# ip waiting host   state       ip switched host
192.168.1.200       switched    192.168.1.100
192.168.1.201       failed
192.168.1.202       waiting
defected hosts= (none)
# ip waiting host   state       ip switched host
192.168.1.203       waiting
```

(4)　エラーが発生した代替ホストの復旧手順

状況記録ファイルに failed と記録された代替ホストのエラーが解決すると、代替ホストとして再び使用できます。ここでは、前節で例示した状況記録ファイルの状態において、ホスト 192.168.1.201 を復旧させる場合を例に挙げて説明します。

代替ホストのエラーが解決した後、障害通知コマンド nqs_ntfr の-R オプションで、ノード管理エージェントに復旧完了を通知します。

ここで引数に指定するホスト名は復旧する代替ホストのホスト名です。今回の例では、下記のコマンドを実行します。

```
# nqs_ntfr -R 192.168.1.201
```

状況記録ファイルを確認すると、ホスト 192.168.1.201 の状態が waiting に変わっています。これで復旧は完了です。

```
11/19 17:20:20
defected hosts= 192.168.1.100
# ip waiting host   state       ip switched host
192.168.1.200       switched    192.168.1.100
192.168.1.201       waiting
192.168.1.202       waiting
defected hosts= (none)
# ip waiting host   state       ip switched host
192.168.1.203       waiting
```

20.2 CLUSTERPROによる冗長化

バッチサーバ、アカウンティングサーバ、アカウンティングモニタ、JobManipulator は、CLUSTERPRO による二重化に対応しています。CLUSTERPRO によってこれらのホストを二重化することで、NQSV システムをダウンさせることなく、継続稼働させることができます。

NQSV は以下のバージョンの CLUSTERPRO による二重化を動作確認しています。

CLUSTERPRO のバージョン	動作確認 OS
CLUSTERPRO X 5.0	Red Hat Enterprise Linux 8.8 Rocky Linux 8.8

20.2.1 注意事項

CLUSTERPRO を使用して NQSV を二重化する場合は、以下の点に注意してください。

・ バッチサーバおよびJobManipulatorを二重化する場合は、フローティングIPアドレスを指定して起動する必要があります。バッチサーバおよびJobManipulatorの起動時に下記のように指定してください。

```
# /opt/nec/nqsv/sbin/nqs_bsvd -a フローティングIPアドレス
# /opt/nec/nqsv/sbin/nqs_jmd -a フローティングIPアドレス
```

・ バッチサーバのマシンIDはフローティングIPアドレスに対応するホスト名に対して割り当ててください。

・ バッチサーバおよびJobManipulatorを二重化する場合は、アカウンティングモニタも二重化してください。

・ アカウンティングサーバを二重化する場合は、アカウンティングサーバのホスト名にフローティングIPアドレスに対応するホスト名を指定してください。

・ CLUSTERPROによって二重化されるコンポーネント（バッチサーバやスケジューラなど）については、systemdによる自動起動および終了を無効にしてください。

systemdによる自動起動/終了を無効にしない場合、待機系のOSシャットダウンによって、運用系のコンポーネントがシャットダウンされてしまう場合があります。

・ データベースが存在する/var/opt/nec/nqsv配下、並びに、定義ファイル等が置かれる/etc/opt/nec/nqsv配下は、共有ディスクに構成してください。具体的には、これら２つのディレクトリを共有ディスク上に移動の上、元々の/var/opt/nec/nqsvおよび/etc/opt/nec/nqsvにシンボリックリンクを作成してください。

【CLUSTERPRO に関する注意】
CLUSTERPRO X 4.3 以前のバージョンでディスクリソースに nas をご利用されている場合、CLUSTERPRO X 5.0 ではディスクリソースに nas を使用できません。

20.2.2 設定方法

ここでは NQSV を二重化する際の設定方法を説明します。なお、記載されていないパラメータはすべて既定値のままです。以降の説明は、CLUSTERPRO に付属する「Builder」上での設定になります。

CLUSTERPRO Builder のインストールおよび使用方法については、CLUSTERPRO X の「インストール＆設定ガイド」および「リファレンスガイド」を参照してください。

(1) フェールオーバーグループの作成

CLUSTERPRO 上に NQSV 用のフェールオーバーグループを作成します。

設定内容は以下の通りです。

名前	NQSVSOFT (任意)
タイプ	フェールオーバー
起動可能サーバ	デフォルトの［全てのサーバでフェールオーバー可能(P)］のまま
グループ属性	デフォルトのまま

次にディスクリソースを以下のように NQSV 用のフェールオーバーグループに追加します。

名前	DK_NQSV(任意)
タイプ	disk resource
依存関係	既定の依存関係に従う

名前	DK_NQSV(任意)
ファイルシステム	ext4
デバイス名	/dev/sdb1　(※利用する環境に合わせて設定してください)
マウントポイント	/var/opt/nec/nqsv

次にフローティング IP リソースを以下のように NQSV 用のフェールオーバーグループに追加します。

名前	FIP_NQSV (任意)
タイプ	floating ip resource
依存関係	既定の依存関係に従う
IP アドレス	10.34.154.138　(※利用する環境に合わせて設定してください)

(2)　execリソースの追加

NQSV で利用する exec リソースを NQSVSOFT グループに追加します。ここではバッチサーバを冗長化するための設定を示します。必要に応じてアカウンティングサーバ、アカウンティングモニタ、JobManipulator 用に変更して設定してください。

BSV 起動・停止用 exec リソース

設定項目				記入欄
情報			名前	EXE_BSV
			タイプ	execute resource
依存関係				既定の依存関係に従う
詳細	スクリプト一覧		Start script	start.sh (※1)
			Stop script	stop.sh (※2)
	調整	パラメータ	開始スクリプト	■同期 □非同期
			終了スクリプト	■同期 □非同期

※1…スクリプトの内容は、(4)i) EXE_BSV の start.sh を参照

※2…スクリプトの内容は、(4)ii) EXE_BSV の stop.sh を参照

BSV プロセス監視用 exec リソース

設定項目			記入欄
情報		名前	EXE_BSVMON
		タイプ	execute resource
依存関係		依存するリソース	DK_NQSV FIP_NQSV EXE_BSV
詳細	スクリプト一覧	Start script	start.sh (※3)
		Stop script	stop.sh (※4)
	調整　パラメータ	開始スクリプト	□同期 ■非同期
		終了スクリプト	■同期 □非同期

※3…スクリプトの内容は、(4)iii) EXE_BSVMON の start.sh を参照

※4…stop.sh は置換せず、既に登録されているものそのままのスクリプトを使用してください

(3) モニタリソースの追加

NQSV で利用する PID モニタリソースを追加します。ここではバッチサーバの設定を示します。必要に応じてアカウンティングサーバ、アカウンティングモニタ、JobManipulator 用に変更して設定してください。

BSV プロセス監視用 PID モニタリソース

設定項目		記入欄
情報	名前	pidw-BSVMON
	タイプ	pid monitor
監視(共通)	対象リソース	EXE_BSVMON
回復動作	回復対象	NQSVSOFT

また、以下のモニタリソースを追加してください。

ディスクモニタリソース

設定項目		記入欄
情報	名前	diskw-NQSVSOFT
	タイプ	disk monitor

設定項目		記入欄
監視(固有)	監視方法	READ(O_DIRECT)
	監視先	/dev/sdb1 (※利用する環境に合わせて設定してください)
回復動作	回復対象	NQSVSOFT

NIC Link UP/Down モニタリソース

設定項目		記入欄
情報	名前	miiw-NQSVSOFT
	タイプ	NIC Link Up/Down monitor
監視(固有)	監視対象	eth0 (※利用する環境に合わせて設定してください)
回復動作	回復対象	NQSVSOFT

(4)　execリソース用スクリプト

設定に使用するスクリプトを以下に記載します。これらのスクリプトはバッチサーバを冗長化するための処理になっています。必要に応じてアカウンティングサーバおよび JobManipulator に書き換えて使用してください。

i)　EXE_BSV の start.sh

```
#! /bin/sh
#***************************************
#*              start.sh               *
#***************************************

if [ "$CLP_EVENT" = "START" ]
then
    if [ "$CLP_DISK" = "SUCCESS" ]
    then
          echo "NORMAL1"
          ###### for BSV software ######
          systemctl start nqs-bsv.service
          ret=$?
          if [ $ret -ne 0 ]; then
                exit $ret;
          fi;
          ##############################
```

```
        if [ "$CLP_SERVER" = "HOME" ]
        then
            echo "NORMAL2"
        else
            echo "ON_OTHER1"
        fi
    else
        echo "ERROR_DISK from START"
    fi
elif [ "$CLP_EVENT" = "FAILOVER" ]
then
    if [ "$CLP_DISK" = "SUCCESS" ]
    then
        echo "FAILOVER1"
        ###### for BSV software ######
        systemctl start nqs-bsv.service
        ret=$?
        if [ $ret -ne 0 ]; then
            exit $ret;
        fi;
        ##############################
        if [ "$CLP_SERVER" = "HOME" ]
        then
            echo "FAILOVER2"
        else
            echo "ON_OTHER2"
        fi
    else
        echo "ERROR_DISK from FAILOVER"
    fi
else
    echo "NO_CLP"
fi
echo "EXIT"
exit 0
```

ii)　EXE_BSV の stop.sh

```
#! /bin/sh
```

```
#***************************************
#*               stop.sh               *
#***************************************

if [ "$CLP_EVENT" = "START" ]
then
    if [ "$CLP_DISK" = "SUCCESS" ]
    then
          echo "NORMAL1"
          ##### For BSV Software #####
          systemctl stop nqs-bsv.service
          ret=$?
          if [ $ret -ne 0 ]; then
                exit $ret;
          fi;
          ############################
          if [ "$CLP_SERVER" = "HOME" ]
          then
                echo "NORMAL2"
          else
                echo "ON_OTHER1"
          fi
    else
          echo "ERROR_DISK from START"
    fi
elif [ "$CLP_EVENT" = "FAILOVER" ]
then
    if [ "$CLP_DISK" = "SUCCESS" ]
    then
          echo "FAILOVER1"
          ##### For BSV Software #####
          systemctl stop nqs-bsv.service
          ret=$?
          if [ $ret -ne 0 ]; then
                exit $ret;
          fi;
          ############################
          if [ "$CLP_SERVER" = "HOME" ]
          then
                echo "FAILOVER2"
```

```
        else
            echo "ON_OTHER2"
        fi
    else
        echo "ERROR_DISK from FAILOVER"
    fi
else
    echo "NO_CLP"
fi
echo "EXIT"
```

exit 0

iii)　EXE_BSVMON の start.sh

```
#! /bin/sh
#**************************************
#*              start.sh              *
#**************************************

if [ "$CLP_EVENT" = "START" ]
then
    if [ "$CLP_DISK" = "SUCCESS" ]
    then
        echo "NORMAL1"
        ##### start BSV MONITOR #####
        /var/opt/nec/nqsv/bsvmon.sh
        ret=$?
        if [ $ret -ne 0 ]; then
            exit $ret
        fi;
        ###################################
        if [ "$CLP_SERVER" = "HOME" ]
        then
            echo "NORMAL2"
        else
            echo "ON_OTHER1"
        fi
    else
        echo "ERROR_DISK from START"
    fi
```

```
elif [ "$CLP_EVENT" = "FAILOVER" ]
then
    if [ "$CLP_DISK" = "SUCCESS" ]
    then
          echo "FAILOVER1"
          ##### start BSV MONITOR #####
          /var/opt/nec/nqsv/bsvmon.sh
          ret=$?
          if [ $ret -ne 0 ]; then
                exit $ret
          fi;
          ################################
          if [ "$CLP_SERVER" = "HOME" ]
          then
                echo "FAILOVER2"
          else
                echo "ON_OTHER2"
          fi
    else
          echo "ERROR_DISK from FAILOVER"
    fi
else
    echo "NO_CLP"
fi
echo "EXIT"
exit 0
```

iv)　/var/opt/nec/nqsv/bsvmon.sh の内容

```
#!/bin/sh
#####################################################################
#
# Name: /var/opt/nec/nqsv/bsvmon.sh
#
#####################################################################

BSV_PROC[0]="/opt/nec/nqsv/sbin/nqs_bsvd"

# The interval (in seconds) between checking
```

```
# if processes of BSV are up and running.
INTERVAL=60

while :
do
        for PROC in ${BSV_PROC[@]}
        do
                pid=`ps -ef | grep "$PROC" | grep -v grep`
                if [ -z "$pid" ]; then
                        echo "NOTICE: $PROC is terminated"
                        exit 1
                fi
        done

        sleep $INTERVAL
done
```

20.2.3 NQSV サービスの起動・停止方法

ここでは CLUSTERPRO による二重化設定後に、NQSV サービスを起動・停止する操作方法について説明します。

二重化設定後は systemctl コマンドではなく、CLUSTERPRO からリソースを操作することで NQSV サービスを起動・停止してください。systemctl コマンドによる NQSV サービスの停止やリソースの操作順を誤ると、障害を誤検出しフェールオーバーに至ることがあります。

なお、以降の説明では、例として BSV サービスを起動・停止する場合について説明します。下記のように exec リソースとプロセスの監視リソースの起動・停止を行ってください。

■サービスの起動

(1) NQSVのexecリソース「EXE_BSV」を起動する。

(2) NQSVのサービスのプロセスを監視するリソース「EXE_BSVMON」を起動する。
このリソースを起動することにより、「pidw-BSVMON」も自動的に起動します。

■サービスの停止

(1) NQSVのサービスのプロセスを監視するリソース「EXE_BSVMON」を停止する。
このリソースを停止することにより、「pidw-BSVMON」も自動的に停止します。

(2)　NQSVのexecリソース「EXE_BSV」を停止する。
　　このリソースを停止することにより、BSVのサービスが停止します。

第21章　バッチジョブ連携 OSS の利用

NQSV では、R や Python から直接バッチジョブシステムへジョブを投入・実行する OSS（例えば Dask-Jobqueue, ClusterMQ など）と連動することができます。

21.1.1 環境設定

OSS からのジョブ投入に連動するために、OSS で使用している SLURM コマンドに代えて、NQSV の連携用コマンドをシンボリックリンクで使用するように設定します。これにより、OSS からジョブを投入、実行したときに、NQSV のコマンドが呼び出されてジョブの投入や表示、削除ができるようになります。

以下の通り、シンボリックリンクを作成してください。

コマンド名	シンボリックリンク先	説明
sbatch	/opt/nec/nqsv/bin/sbatch_cnv.sh	バッチジョブ投入コマンドです。 SLURMのsbatchコマンドのI/FをNQSVのqsub(1)コマンドのI/Fに変換してバッチジョブを投入します。
squeue	/opt/nec/nqsv/bin/squeue_cnv.sh	バッチジョブ表示コマンドです。 SLURMのsqueueコマンドのI/FをNQSVのqstat(1)コマンドのI/Fに変換してバッチジョブを表示します。
scancel	/opt/nec/nqsv/bin/scancel_cnv.sh	バッチジョブ削除コマンドです。 SLURMのscancelコマンドのI/FをNQSVのqstat(1)コマンドのI/Fに変換してバッチジョブを削除します。

本機能を使う場合は、上記のシンボリックリンクを手動で作成してください。シンボリックリンクファイルの場所は、OSS から利用するバッチジョブコマンドのパスとしてください。

各 OSS の利用方法については、OSS のサイトを参照ください。

付録 A　運用事例

A.1　各種スクリプト実行機能の比較

ユーザ EXIT、フックスクリプト、ユーザプリ・ポストスクリプト、資源監視スクリプトの用途を比較した表です。

	権限	実行タイミング	実行場所	使用目的など	備考
ユーザ EXIT	root	PRE-RUNNING POST-RUNNING HOLDING RESTARTING	ジョブサーバホスト	作業領域の掃除	スクリプトの完了を待つ。 スクリプト実行結果を処理する。
フックスクリプト	root	QUEUED WAITING HELD HOLDING SUSPENDING SUSPENDED STAGING ARRIVING PRE-RUNNING RUNNNING POST-RUNNING EXITING EXITED RESUMING etc	バッチサーバホスト	パラメータを間違えて指定したジョブを拒否する	スクリプトの完了を待たない。 スクリプトの実行結果を処理できない
ユーザプリ・ポストスクリプト	ユーザ	PRE-RUNNING POST-RUNNING	ジョブサーバホスト	ジョブ実行ノードのヘルスチェック	
資源監視スクリプト	root	RUNNING	ジョブサーバホスト	カスタムリソースの資源使用量を取得する	

A.2　例：標準出力・標準エラー出力ファイルの制限（ユーザEXIT）

標準出力と標準エラー出力のファイルサイズが増大した場合に、ファイル転送を防止するスクリプトの例です。

POST-RUNNING のタイミングで処理します。

ジョブ実行終了後にファイルサイズが 30MB を超える場合は、/tmp/backup_dir にファイルを保存し、以下のメッセージを出力します。

"The size of stdout(stderr) exceed 31457280 byte(s).“

```
#!/bin/sh
FILESIZELIMIT=31457280
BACKUP_DIR_STDOUT=/tmp/backup_dir
BACKUP_DIR_STDERR=/tmp/backup_dir

case "$UEX_LOCATION" in
POSTRUNNING)
    case "$UEX_PROCTYPE" in
    EXECUTION)
        JOBID=`echo $UEX_STGDIR | sed 's|.*/¥(.*¥)/.*|¥1|'`
        STDOUT=/var/opt/nec/nqsv/jsv/jobfile/$JOBID/stdout
        STDERR=/var/opt/nec/nqsv/jsv/jobfile/$JOBID/stderr

        [ -z "$PBS_JOBNAME" ] && echo 1>&2 "PBS_JOBNAME is not set" && exit 1
        [ -z "$UEX_STGDIR" ] && echo 1>&2 "UEX_STGDIR is not set" && exit 1

        if [ `wc -c "$STDOUT" | awk '{print $1}'` -gt $FILESIZELIMIT ]; then
            if [ ! -d "$BACKUP_DIR_STDOUT" ]; then
                mkdir -p "$BACKUP_DIR_STDOUT" || exit 1
            fi
            mv $STDOUT $BACKUP_DIR_STDOUT/$PBS_JOBNAME.o$JOBID || exit 1
            echo "The size of stdout exceeded $FILESIZELIMIT byte(s)." > $STDOUT
        fi

        if [ `wc -c "$STDERR" | awk '{print $1}'` -gt $FILESIZELIMIT ]; then
            if [ ! -d "$BACKUP_DIR_STDERR" ]; then
                mkdir -p "$BACKUP_DIR_STDERR" || exit 1
            fi
            mv $STDERR $BACKUP_DIR_STDERR/$PBS_JOBNAME.e$JOBID || exit 1
            echo "The size of stderr exceeded $FILESIZELIMIT byte(s)." > $STDERR
        fi
        ;;
    esac
    ;;
esac
```

A.3　例：残存プロセスの削除（ユーザEXIT）

会話ジョブノードで、ジョブ実行終了後に残ったプロセスを削除するスクリプトの例です。
この例はソケットスケジューリング機能が有効な場合のみ利用できます。
POST-RUNNING のタイミングで処理します。

```
#!/bin/bash

JID=`basename $UEX_JOBDB`

while read line
do
    kill -9 $line
done < /sys/fs/cgroup/cpuset/$JID/tasks
```

キューに RSG 番号を設定している場合

ユーザ EXIT スクリプトで参照すべき memory cgroup のパスが異なります。<cpuset_name>にはキューに設定した RSG 番号に対応する CPUSET 名が入ります。

```
#!/bin/bash

JID=`basename $UEX_JOBDB`

while read line
do
    kill -9 $line
done < /sys/fs/cgroup/cpuset/<cpuset_name>/$JID/tasks
```

CPUSET 名は下記の手順で確認できます。
まず、ユーザ EXIT を設定するキューの RSG 番号を確認します。Kernel Parameter の Resource Sharing Group に表示される番号が該当する番号です。下記の例では 1 です。もし、0 の場合は RSG 番号を設定していません。

```
$ qstat -Qf intque
Interactive Queue: intque@bsvhost01
    Run State    = Active
    Submit State = Enable
      :
  Kernel Parameter:
    Resource Sharing Group     = 1
    Nice Value                 = 0
```

実行ホストの/etc/opt/nec/nqsv/cpuset.conf を確認します。RSG 番号に対応する CPUSET 名が該当する CPUSET 名です。下記の例では forjob です。

```
$ cat /etc/opt/nec/nqsv/cpuset.conf
CPUSET  0-31  0-1  0
```

```
forjob  8-31  0-1  1
```

この例の場合、ユーザ EXIT スクリプトの内容は下記です。

```
#!/bin/bash

JID=`basename $UEX_JOBDB`

while read line
do
    kill -9 $line
done < /sys/fs/cgroup/cpuset/forjob/$JID/tasks
```

なお、CPUSET 機能については本文書の 18.2　CPUSET 機能をご覧ください。

A.4　例：パラメータ確認とジョブ削除（フックスクリプト）

間違ったパラメータを指定したジョブを拒否するスクリプトの例です。

QUEUED か PRE-RUNNING のタイミングで処理します。

“-l cpunum_job=40” is specified but “-l gpunum_job=8”が指定されていないジョブは qdel で削除されます。

```
#!/bin/bash

CPUNUM=`qstat -f ${HOOK_RID} -Pm | grep "(Per-Job) CPU Number" | awk '{print $6}'`
GPUNUM=`qstat -f ${HOOK_RID} -Pm | grep "(Per-Job) GPU Number" | awk '{print $6}'`

if (CPUNUM == 40) ; then
   if (GPUNUM != 8) ; then
      qdel -Pm ${HOOK_RID}
   fi
fi
```

A.5　例：GPU利用台数の取得（資源監視スクリプト）

ジョブが使用した GPU 利用台数を取得するスクリプトの例です。

RUNNING のタイミングで定期的に処理します。

本スクリプトを利用する場合、カスタムリソースの Consumer を job に、Check Mode を moment に設定し作成した後、/opt/nec/nqsv/sbin/custom_prog/<カスタムリソース名>に配置してください。

```
#!/bin/bash
```

```
NVIDIA_SMI=/usr/bin/nvidia-smi
NVIDIA_SMI_OPT="pmon -c 1"
SAVED_FL=${CR_TMPDIR}/gpuacct

SUM=0
declare -A GPU_HASH

if [ -e ${SAVED_FL} ]; then
    while read line
    do
        GPU=`echo $line | awk '{print $1}'`
        GPU_HASH["$GPU"]=1
        SUM=$((${SUM} + 1))
    done < $SAVED_FL
fi

SMIOUT=`${NVIDIA_SMI} ${NVIDIA_SMI_OPT}`
if [ $? -ne 0 ];then
    exit 1
fi

while read line
do
    GPU=`echo $line | awk '{print $1}'`
    PID=`echo $line | awk '{print $2}'`
    if expr "${PID}" : "[0-9]*$" >&/dev/null; then
        SID=`ps --no-header -o sid -p ${PID} | awk '{print $1}'`
        if [ "$SID" = "$CR_EJID" ];then
            if [ -z ${GPU_HASH["$GPU"]} ];then
                GPU_HASH["$GPU"]=1
                SUM=$((${SUM} + 1))
            fi
        fi
    fi
done <<EOF
$SMIOUT
EOF

echo $SUM

if [ -e ${SAVED_FL} ]; then
    rm ${SAVED_FL}
fi

for g in "${!GPU_HASH[@]}"
do
    echo $g >> $SAVED_FL
done

exit 0
```

A.6　例：OSS連携（Dask-Jobqueue）

コマンドラインから Python3 を起動し、Dask-jobqueue の SLURMCluster()/cluster.scale() API を使用したバッチジョブ投入・実行例です。各 API 及びジョブ投入・実行の詳細については、OSS の公式サイトを参照してください。

```
$ python3
>>>
>>> from dask_jobqueue import SLURMCluster
>>> cluster = SLURMCluster(
queue='regular',
project="myproj",
cores=1,
memory="1 GB"
)
>>> cluster.scale(jobs=1)
>>>
```

付録 B　発行履歴

B.1　発行履歴一覧表

2018 年　2 月	初版
2023 年　3 月	第 16 版
2023 年　6 月	第 17 版
2024 年　3 月	第 18 版
2024 年　7 月	第 19 版
2025 年　1 月	第 20 版

B.2　追加・変更点詳細

第 16 版

- 4.1 および 4.5 に標準出力の結果ファイルサイズ制限、および、標準エラー出力の結果ファイルサイズ制限を追加
- 12.3 に投入可能な VE ノード数制限の拡張 を追加
- 16.1 に Dockerfile 作成時の注意事項を追記
- 20.2.3 に NQSV サービスの起動・停止方法 を追加
- A.5 のサンプルの誤りを訂正

第 17 版

- 20.2 CLUSTERPRO X の動作確認バージョンを記載
- A.3 にキューに RSG 番号を設定している場合のユーザ EXIT スクリプトの例を追加

第 18 版

- 2.3.21 リクエスト転送時の投入リクエスト数制限超過処理変更を追加
- 5.1.7 HW 障害時の自動リラン・課金除外機能を追加

第 19 版

- 3.10 実行ホストの情報表示 の qstat イメージを更新
- 16.1 Docker によるプロビジョニング環境の設定 の対応 OS バージョン、手順更新
- 20.2 CLUSTERPRO による冗長化 の動作確認 OS を更新

第 20 版

- 第 9 章 MPI リクエスト実行環境の設定 の OpenMPI サポートバージョンを更新
- 12.5 HCA 障害の検知 に注意事項を追加

索　引

し

す

せ

そ

て

と

な

に

ね

の

は

ふ

へ

ほ

ま

ゆ

ら

り

ろ

NEC Network Queuing System V (NQSV)

利用の手引［管理編］

2025 年 1月 第20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）