NEC

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

[操作編]

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）により、一般ユーザがリクエストを操作する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、システム管理者および一般利用者を対象読者として書かれたものです。とくに一般利用者が構築済みのシステムを利用する際には、本書は有効な説明書となります。

本書は、対象読者がLinuxの一般操作を知っていることを前提として記述されています。

本書の読み進め方

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

章	タイトル	内容	対象読者
1	バッチリクエストの操作	バッチリクエストの操作について説明しています	一般利用者(*)
2	パラメトリックリクエストの操作	パラメトリックリクエストの操作について説明しています	一般利用者(*)
3	会話リクエストの操作	会話リクエストの操作について説明しています	一般利用者(*)
4	ジョブの操作	ジョブの操作について説明しています	一般利用者(*)
5	ネットワークリクエストの操作	ネットワークリクエストの操作について説明しています	一般利用者(*)
6	システム情報の表示	システム情報の表示	一般利用者(*) システム管理者(*)
7	ワークフロー	ワークフローについて説明しています	一般利用者
8	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	一般利用者
9	グループごと・ユーザごとの制限	グループごと・ユーザごとの制限について説明しています	一般利用者(*) システム管理者(*)
10	GPUを使用するリクエストの投入	GPUを使用するリクエストの投入について説明しています	一般利用者
11	ユーザプリ・ポストスクリプトを指定するリクエストの投入	ユーザプリ・ポストスクリプトを指定するリクエストの投入について説明しています	一般利用者
12	OpenStackと連携したプロビジョニング環境を使用するリクエストの投入	OpenStackと連携したプロビジョニング環境を使用するリクエストの投入について説明しています	一般利用者

章	タイトル	内容	対象読者
13	Docker と連携したプロビジョニング環境を使用するリクエストの投入	Docker と連携したプロビジョニング環境を使用するリクエストの投入について説明しています	一般利用者
14	カスタムリソース機能を使用するリクエストの投入	カスタムリソース機能を使用するリクエストの投入について説明しています	一般利用者
15	制約事項	制約事項について説明しています	一般利用者(*) 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システム全体を管理します。
ジョブサーバ（JSV）	実行ホスト上に常駐し、ジョブの実行を管理します。
JobManipulator（JM）	NQSVの提供するスケジューラ機能です。計算リソースの空き容量を管理し、ジョブの開始時刻を割り当てます。
アカウンティングサーバ	アカウント情報の収集・管理、予算管理を行います。
リクエスト	NQSV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

は　し　が　き i
用語定義・略語 vii
目　次 viii
表目次 xiv
図目次 xv
第 1 章　　バッチリクエストの操作 1
1.1　バッチリクエストの作成 1
1.1.1　ジョブスクリプトの作成 1
1.1.2　ジョブスクリプトの制限 1
1.1.3　入力データの指定方法 2
1.1.4　投入オプションの指定方法 2
1.2　バッチリクエストの投入 4
1.2.1　基本的な投入方法 4
1.2.2　投入するキューの指定 7
1.2.3　リクエスト名の指定 7
1.2.4　ジョブ数の指定 7
1.2.5　VE ノード総数の指定 8
1.2.6　HCA の指定 9
1.2.7　ジョブの種類の指定 10
1.2.8　出力結果ファイルの指定 12
1.2.9　資源制限値の指定 12
1.2.10　開始時刻の指定 15
1.2.11　実行開始、終了のメール通知の指定 16
1.2.12　プライオリティの指定 16
1.2.13　アカウントコードの指定 17
1.2.14　リラン可否の指定 17
1.2.15　ホールド可否の指定 17
1.2.16　環境変数の利用 17
1.2.17　スクリプトの実行シェルの指定 19
1.2.18　ジョブの実行条件（ジョブコンディション）の指定 20
1.2.19　転送ファイルの指定 22
1.2.20　ジョブマイグレーションに関する指定 22

1.2.21 投入時メッセージ表示の変更 22
1.2.22 リクエスト連携機能 23
1.2.23 排他実行リクエスト 23
1.2.24 ハイブリッドリクエスト 26
1.2.25 ユーザ指定のリクエスト投入 26
1.2.26 ジョブ実行時の SIGTERM 捕捉 27
1.3 バッチリクエストの状態確認 27
1.3.1 基本情報の確認 27
1.3.2 詳細情報の確認 28
1.3.3 出力情報のカスタマイズ 31
1.4 バッチリクエストの属性変更 34
1.5 バッチリクエストの削除 36
1.6 バッチリクエストのホールド 36
1.7 バッチリクエストのホールド解除 36
1.8 バッチリクエストのサスペンド 37
1.9 バッチリクエストのレジューム 37
1.10 バッチリクエストのリラン 37
1.11 バッチリクエストのキューの移動 38
1.12 バッチリクエストの終了確認 39
1.12.1 メールによるリクエスト終了の確認 39
1.12.2 リクエスト終了の待ち合わせ 39
1.13 バッチリクエストの出力ファイル 39
1.14 MPI リクエストの実行 40
1.14.1 NEC MPI 環境での実行 40
1.14.2 OpenMPI 環境での実行 41
1.14.3 Intel MPI 環境での実行 42
1.14.4 MVAPICH2 環境での実行 44
1.14.5 Platform MPI 環境での実行 45
1.15 リクエスト実行中の出力ファイル表示 46
1.16 リクエストログ 47
1.17 リクエストへのアタッチ 47
第 2 章 パラメトリックリクエストの操作 49
2.1 パラメトリックリクエストの投入 49
2.1.1 投入方法 49

2.1.2 パラメータ、入力ファイル 50
2.2 パラメトリックリクエストの状態確認 50
2.3 パラメトリックリクエストの操作 52
2.4 パラメトリックリクエストの開始・終了と出力ファイル 53
2.4.1 メール送信機能 53
2.4.2 終了待ち合わせ 54
2.4.3 出力ファイル 54
第 3 章 会話リクエストの操作 56
3.1 会話リクエストの投入 56
3.1.1 qlogin による投入方法 56
3.1.2 qlogin の投入オプション 57
3.2 会話リクエストの状態確認 59
3.3 会話リクエストの属性変更 62
3.4 会話リクエストの操作 62
3.5 会話リクエストの開始・終了と出力ファイル 62
3.6 会話リクエストでの MPI プログラムの実行 62
3.6.1 NEC MPI 環境での実行 62
3.6.2 OpenMPI 環境での実行 63
3.6.3 Intel MPI 環境での実行 64
3.6.4 MVAPICH2 環境での実行 65
3.6.5 Platform MPI 環境での実行 65
第 4 章 ジョブの操作 67
4.1 ジョブの状態確認 67
4.1.1 基本情報の確認 67
4.1.2 詳細情報の確認 68
4.1.3 ハイブリッドリクエストのジョブ情報の確認 69
4.1.4 出力情報のカスタマイズ 69
4.2 シグナルの送信 71
第 5 章 ネットワークリクエストの操作 72
5.1 ネットワークリクエストの状態確認 72
5.2 ネットワークリクエストの状態遷移 74
5.3 ネットワークリクエストの削除 74
第 6 章 システム情報の表示 75
6.1 システム情報の表示方法 75

6.2 出力情報のカスタマイズ 76
6.3 出力情報のソート 82
6.4 時間単位項目の表示 83
6.5 全内容の表示 84
6.6 全ジョブサーバ（実行ホスト）表示 85
第 7 章 ワークフロー 87
7.1 ワークフローの実行環境 87
7.2 ワークフローの記述 88
7.2.1 簡単な例 88
7.2.2 実行順序の指定方法 89
7.2.3 同時実行開始の指定方法 90
7.3 ワークフローの実行 91
7.4 ワークフローの参照 91
7.5 ワークフローの操作 92
7.5.1 ワークフローの削除 92
7.5.2 エラー発生時の後続リクエストキャンセル機能 93
第 8 章 リクエストのグループ 95
8.1 リクエストのグループと投入方法 95
8.2 グループに関する情報の表示 95
8.2.1 リクエスト情報 95
8.2.2 キュー情報 96
第 9 章 グループごと・ユーザごとの制限 97
9.1 グループごと・ユーザごとの制限 97
9.2 グループごと・ユーザごとの制限情報の参照 98
第 10 章 GPU を使用するリクエストの投入 100
第 11 章 マルチインスタンス GPU(MIG)を使用するリクエストの投入 101
11.1 マルチインスタンス GPU(MIG) 101
11.1.1 GI 1 つを指定するリクエストの投入 101
11.1.2 GI 複数を指定するリクエストの投入 101
11.1.3 GI を CI に分割するリクエストの投入 102
11.2 qstat での MIG 確認 102
11.3 MIG ジョブ実行 103
第 12 章 ユーザプリ・ポストスクリプトを指定するリクエストの投入 105
12.1 ユーザ PP スクリプト 105

12.2 リクエスト投入時の指定方法 106
12.3 ユーザ PP スクリプトの出力 107
第 13 章 OpenStack と連携したプロビジョニング環境を使用するリクエストの投入 109
13.1 テンプレート情報の参照 109
13.2 テンプレートを指定したリクエストの投入 110
13.3 テンプレートを指定したリクエストの情報 110
13.3.1 リクエスト情報 110
13.3.2 ジョブ情報 111
第 14 章 Docker と連携したプロビジョニング環境を使用するリクエストの投入 113
14.1 テンプレート情報の参照 113
14.2 テンプレートを指定したリクエストの投入 114
14.3 テンプレートを指定したリクエストの情報 114
14.3.1 リクエスト情報 114
14.3.2 ジョブ情報 116
第 15 章 カスタムリソース機能を使用するリクエストの投入 117
15.1 カスタムリソース情報の参照 117
15.2 リクエスト投入時の指定方法 118
15.3 リクエストのカスタムリソース利用量の参照 119
第 16 章 ハイブリッドリクエスト 121
16.1 ハイブリッドリクエストとは 121
16.2 ハイブリッドリクエストの投入 121
16.3 ハイブリッドリクエスト情報の参照 125
16.4 ハイブリッドリクエストの属性変更 127
第 17 章 singularity と連携したプロビジョニング環境を使用するリクエストの投入 129
17.1 ジョブ実行方法 129
17.1.1 MPI ジョブの実行 129
17.1.2 分散ジョブの実行 130
第 18 章 制約事項 132
18.1 NQSV システムで使用する番号などの範囲・最大値 132
18.2 コマンドとバッチサーバのバージョン 134
18.3 ユーザレベルチェックポイント 135
18.3.1 ユーザレベルチェックポイントを使用するための準備 135
18.3.2 リクエスト投入の仕方 135
18.3.3 チェックポイントをとる場合 135

付録 A　　NQSV リクエストの投入方法 1
A.1　VE を使用したリクエスト 1
A.2　x86 を使用したリクエスト 2
A.3　GPU を使用したリクエスト 2
A.4　マルチインスタンス GPU を使用したリクエスト 3
付録 B　　発行履歴 4
B.1　発行履歴一覧表 4
B.2　追加・変更点詳細 4
索　引 6

表目次

表 1-1 VE ノード毎の資源制限 14
表 1-2 論理ホスト毎の資源制限 15
表 1-3 リクエストの状態 28
表 1-4 qstat(1) -F オプションのアイテム 32
表 2-1 パラメトリックリクエストの操作 52
表 2-2 パラメトリックリクエストのメール送信 53
表 3-1 会話リクエストの操作 62
表 4-1 qstat(1) -F オプションの指定 70
表 5-1 ネットワークリクエストの情報 73
表 6-1 qstat(1)の表示オプション 75
表 6-2 出力情報のカスタマイズ 77
表 12-1 ユーザ PP スクリプトの環境変数 105
表 12-2 ユーザ PP スクリプトの結果 108

図目次

図 1.2-1：シングルジョブの実行 10
図 1.2-2：分散ジョブの実行 11
図 1.2-3：MPI ジョブの実行 11
図 7.1-1：ワークフローの実行 88
図 11.1-1　GI を CI に分割する例 102

第1章　　バッチリクエストの操作

バッチリクエストはプログラムの実行を依頼するリクエストのことです。このリクエストは、NQSVにシェルスクリプトを投入することにより実現されます。

1.1　バッチリクエストの作成

1.1.1　ジョブスクリプトの作成

バッチリクエストを作成するためには、まずバッチリクエスト用のシェルスクリプトファイルを作成してください。このシェルスクリプトがバッチリクエストとして NQSV に投入され実行されます。

通常のシェルスクリプトと同じように、このジョブスクリプトには、任意のコマンドを組み合わせて記述できます。また、このジョブスクリプトを実行するシェルは、自由に指定できます。したがって、sh 用、csh 用、その他のシェル用のどのシェルスクリプトでもかまいません。

以下にバッチリクエスト用ジョブスクリプトの例を示します。

```
#
# sample batch request
#

f90 -o prog1 prog1.f90
if [ $? -ne 0 ]
then echo "prog1.f90 compile error"
    exit 1
fi
prog1 < in_data
```

1.1.2　ジョブスクリプトの制限

バッチリクエスト用のジョブスクリプトには、以下の制限があります。

- 端末装置に対する入出力を要求するようなコマンド(例えば、stty(1)コマンドなど)は、使用できません。なぜなら、ジョブスクリプトの実行は、NQSV によってどの端末装置とも無関係に実行されるからです。
- 処理の途中でインタラクティブな操作を必要とするようなジョブスクリプトは、バッチリクエストとして実行できません。なぜなら、ジョブスクリプトの実行中は、その標準出力、標準エラー出力はそれぞれ任意のファイルに既に結び付けられているからです。

1.1.3　入力データの指定方法

バッチリクエスト用のジョブスクリプトでは、 前述のとおり対話的なデータの入力ができません。シェルコマンドが入力すべきデータがある場合は、次の 2 つの方法で行ってください。

- コマンドが入力するデータを格納したデータファイルを用意する方法

```
sort < input_data
```

この例では、ソートすべきデータは input_data というファイルに格納されています。

- ヒアドキュメントを使用する方法

```
sort << EOF
Rebert Cohn was
once middleweight boxing
champion of Princeton
EOF
```

1.1.4　投入オプションの指定方法

バッチリクエスト用ジョブスクリプトには、そのコメント部に NQSV の投入時オプション（バッチリクエスト実行のための設定情報）を埋め込むことができます。これにより、リクエスト投入時のわずらわしいオプション指定を軽減できます。

オプションは、最初のシェルコマンドが現れる前のコメント部分に、"#PBS"という文字列に続けて指定してください。"#PBS"は NQSV の投入時オプションであることを示します。"#PBS"は既定値であり、qsub(1)コマンドのコマンドラインで指定するオプションによって変更できます。

埋め込みオプションには、バッチリクエストの投入コマンドである qsub(1)コマンドのすべてのオプションが記述できます。この埋め込みオプションは、通常のシェルにとってはコメント行とみなされます。したがって、ジョブスクリプトの実行には影響しません。

```
#
# BATCH request script
#
#PBS -a 1130 -l cputim_prc="21:10"        11時30分に開始、CPU使用時間を21分10秒に制限
#PBS -q batch1                          batch1というキューに投入する
make
```

上記の例のように 1 行に複数のオプションを埋め込むこともできます。

また、埋め込みオプションをコメントとして扱う場合は、以下のように行の先頭が"#PBS" で始ま

らないようにしてください。

```
##PBS -q batch
# Comment Out #PBS -a 1130 -l cputim_prc="21:10"
```

1.2　バッチリクエストの投入

1.2.1　基本的な投入方法

バッチリクエストの投入は、qsub(1)コマンドで行ってください。qsub コマンドの主な仕様は、次のとおりです。

- コマンド行にオプションを指定できます。またジョブスクリプトファイル内に埋め込みオプションを指定できます。コマンド行と埋め込みオプションに同じオプションを指定した場合には、コマンド行の方を有効とみなします。
- コマンド行にジョブスクリプトファイルを指定しなければ、そのまま標準入力からジョブスクリプトを 読み込みます。

以下にジョブスクリプトファイルを指定したバッチリクエストの投入例を示します。

```
$ qsub -q batch1 script1
Request 65.host1 submitted to queue: batch1.
```

上記の例は、batch1 というキューに script1 というジョブスクリプトを実行するバッチリクエストを投入したものです。qsub コマンドの次の行に示されたメッセージは、NQSV が投入されたバッチリクエストを受理したことを示すメッセージです。このメッセージは次のことを示しています。

```
Request 65.host1 submitted to queue: batch1.
         ↑                          ↑
        (A)                        (B)
```

（A）バッチリクエストに対して NQSV が付けたリクエスト ID

（B）バッチリクエストを投入したキュー名

次に、標準入力からジョブスクリプトを読み込む場合を示します。

```
$ qsub -q batch1
make all
Ctrl-D(EOF)
Request 66.host1 submitted to queue: batch1.
$
```

上記の例は、batch1 というキューに標準入力から入力された「make all」コマンドをジョブスクリプトとして投入したものです。標準入力からの入力を終えるには Ctrl-D を入力してください。

投入時のオプション

投入時に指定できる主なオプションの種類として、以下のものがあります。

（投入時に指定できる具体的なオプションの使用方法については、次節以降で記述します。）

(1)　結果ファイル関係のオプション

バッチリクエストの出力結果(標準出力および標準エラー出力に出力されるもの)をどこに返すかを指定するオプションです。通常、バッチリクエストの出力結果は、いったん NQSV によって用意されたスプールファイルに出力されます。バッチリクエストの実行がすべて終了した時点で所定のディレクトリ配下の所定のファイルにコピーされます。特に指定しなかった場合は、バッチリクエストの出力結果はバッチリクエストを投入したディレクトリ配下に返されます（1.13.バッチリクエストの出力ファイル 参照）。

(2)　資源制限用オプション

バッチリクエストは、その実行時に資源制限を行えます。資源制限とは、そのバッチリクエストで使用できる CPU 時間やメモリサイズ、ファイルサイズの最大値を設定しておき、バッチリクエストが設定した値を超えて実行しようとした時にその実行を強制終了させる機能です。

NQSV では、この資源制限値を投入時のオプションとして指定できます。特に指定しなかった場合は、そのバッチリクエストは投入したバッチキューに管理者が設定した資源制限値が適用されます。また、オプションとして指定した場合は、投入先のバッチキューに管理者により設定されている値と比較されます。もし、バッチキューの値よりもオプションで指定した値の方が大きい場合は、そのバッチリクエストの投入は拒否されます。なぜなら、そのバッチキューではそのような資源を多く消費するバッチリクエストを実行できないからです。

(3)　メール関係のオプション

NQSV では、通常バッチリクエストの実行開始や実行終了が明示的には通知されません。ユーザは、1.3.バッチリクエストの状態確認 で説明する各コマンドを使用して、投入したバッチリクエストの状態を知ることができます。

一方、バッチリクエストの投入時オプションで、実行開始や実行終了をメールで通知するように指定できます。このオプションが指定されたバッチリクエストの実行が開始された時または終了した時は、NQSV からその旨を示すメールが届きます。また、メールの受け取り先の既定値は投入したユーザですが、変更することもできます。

バッチリクエストの実行が異常終了した場合は、オプションの有無にかかわらず必ずその内容を示

すメールが投入したユーザに送られます。

(4)　バッチジョブ関係のオプション

NQSV では、バッチリクエストが作成するジョブの数および種類をオプションで指定できます。通常、ジョブ数の指定がない場合は、実行されるジョブは１つとなります。また、ジョブの種類には以下があり、ジョブの種類の指定が無い場合は、分散ジョブとなります。

- 分散ジョブ
  各バッチジョブでリクエストに指定されたシェルスクリプトを実行します。
- necmpiジョブ
  NEC MPI プロセスを実行します。
- intmpiジョブ
  Intel MPI プロセスを実行します。
- mvapichジョブ
  MVAPICH プロセスを実行します。
- openmpiジョブ
  OpenMPI プロセスを実行します。
- pltmpiジョブ
  Platform MPI プロセスを実行します。

(5)　その他のオプション

その他のオプションとして、バッチリクエストの実行開始時刻を指定するオプション、投入先のキューを指定するオプション、バッチリクエストのプライオリティを指定するオプションなどがあります。

以降で、オプションの簡単な説明を示します。

ただし、詳細な説明および、下記以外のオプションについては、［リファレンス編］qsub(1)の項を参照してください。

1.2.2　投入するキューの指定

リクエストを登録するキューは、qsub(1)コマンドの -q オプションで指定してください。

```
-q destination
```

このオプションを指定しない場合は、環境変数 PBS_QUEUE の文字列値が、リクエストを投入するキューとなります。環境変数 PBS_QUEUE も定義されていない場合は、リクエストはキューに投入されません。エラーメッセージを出力します。

1.2.3　リクエスト名の指定

リクエスト名は、qsub(1)コマンドの -N オプションで指定してください。

```
-N name
```

このオプションでリクエスト名を指定しない場合は、リクエスト投入時のジョブスクリプト指定方法によって、以下のように自動的にリクエスト名をつけます。

1. 標準入力からスクリプトを入力した場合、リクエスト名は "STDIN" になります。
2. スクリプトファイルを用いた場合はディレクトリ部分を除いたファイル名になります。

【例】

```
$ qsub /usr/nqs/script
```

上記のようにリクエストを投入した場合、リクエスト名は"script"となります。

リクエスト名が数字で始まる場合、文字 'R' が先頭に付加されます。また、リクエスト名が 64 文字以上になる場合は 63 文字で打ち切られます。

1.2.4　ジョブ数の指定

リクエストで実行するジョブ数は、qsub(1)コマンドの -b オプションで指定してください。

```
-b job_count
```

1.2.5　VE ノード総数の指定

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。

qsub のオプション--venode を使用し、リクエストの実行に必要な VE の総数を指定します。本指定を行うと、キューに設定された既定搭載 VE ノード数に従って、ジョブ数が自動換算されてリクエストに適用されます。そのため、本オプションとジョブ数の指定（-b オプション）は同時に行うことはできません。なお、本オプションは VE 数とジョブ数を指定するものであるため、実際のリクエスト投入時にはその他資源制限値（例えば Elapse 時間制限値や CPU 台数制限値）も合わせて指定する必要があります。

本オプションのフォーマットは以下の通りです。

```
qsub --venode=<venum>
```

<venum>に、リクエストが必要とする VE ノードの総数を指定します。

<venum>には 1～2147483647 の整数が指定できますが、投入先キューに設定された VE 総数制限範囲内の値である必要があります。

投入先キューに設定された既定搭載 VE ノード数とは異なる値を使用してジョブ数を制御したい場合は、--venode オプションと組み合わせて、論理ホスト毎の VE 数制限（--venum-lhost オプション）（後述）を指定することで任意の既定搭載 VE ノード数を適用することができます。フォーマットは以下の通りです。

```
qsub --venode=<venum> --venum-lhost=<venum_lhost>
```

--venode オプションと組み合わせて--venum-lhost オプションが同時に指定されると、<venum_lhost>で指定した値を既定搭載 VE ノード数の代わりに使用して、ジョブ数の自動計算が行われます。論理ホスト数の算出方法はキューの既定搭載 VE ノード数の場合と同じです。

--venode オプションと組み合わせて--venum-lhost オプションを指定する場合、<venum_lhost>にはキューの資源制限値の最低値の設定に係わらず 1 以上の値を指定しなければなりません。また、キューの資源制限値の最大値以下でなければなりません。

<venum_lhost>に指定した値が、実際の実行ホスト上に搭載されている VE 数よりも大きい場合であってもエラーや警告は発生しません。このような指定をした場合、リクエストはスケジューリングされません。

--venode オプションを指定して投入したリクエストは qstat -f の VE Node Number 欄にてその指定値を参照できます。このとき、既定搭載 VE ノード数または--venum-lhost オプション指定値による換算後のジョブ数を Number of Jobs 欄で確認することができます。

1.2.6　HCA の指定

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。

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。--use-hca オプションで Direct 通信の利用、および論理ホスト内における割当て VE が所属するデバイスグループあたりに必要な HCA ポート数を宣言し、リクエストを投入してください。

フォーマットは以下の通りです。

```
qsub --use-hca=[<mode>:]<num>
```

<num>は論理ホストに割り当てられた VE が使用する HCA のポート数を指定します。

<mode>は HCA の利用用途を意味し、下記のいずれかが指定可能です。指定がない場合、all を指定したと扱います。

- io：I/O専用。デバイスリソース定義ファイルにIOを指定したHCAだけが割りあたる
- mpi：MPI専用。デバイスリソース定義ファイルにMPIを指定したHCAだけが割りあたる
- all：I/O、MPI兼用 <既定値>。デバイスリソース定義ファイルにIO,MPIの両方を指定したHCAだけが割りあたる

本オプションは VE を使用するリクエストにのみ有効です。VE を使用しないリクエスト、および、VE が搭載されていないホストで実行されるリクエストにも指定することが可能ですが、意味を持ちません。その場合は、どの HCA も利用される可能性があります。

本機能の詳細については、［JobManipulator 編］HCA の割当てを参照してください。

1.2.7　ジョブの種類の指定

リクエストで実行されるジョブの種類は、 qsub(1)コマンドの -T オプションで指定してください。

```
-T job_type
```

job_type に指定可能なジョブの種類には以下の種類があります。

- distrib　実行されるバッチジョブは分散ジョブです。(既定値)
- intmpi　実行されるバッチジョブはintmpiジョブです。
- mvapich　実行されるバッチジョブはmvapichジョブです。
- necmpi　実行されるバッチジョブはnecmpiジョブです。
- openmpi　実行されるバッチジョブはopenmpiジョブです。

【シングルジョブの実行（ジョブ数=1)】

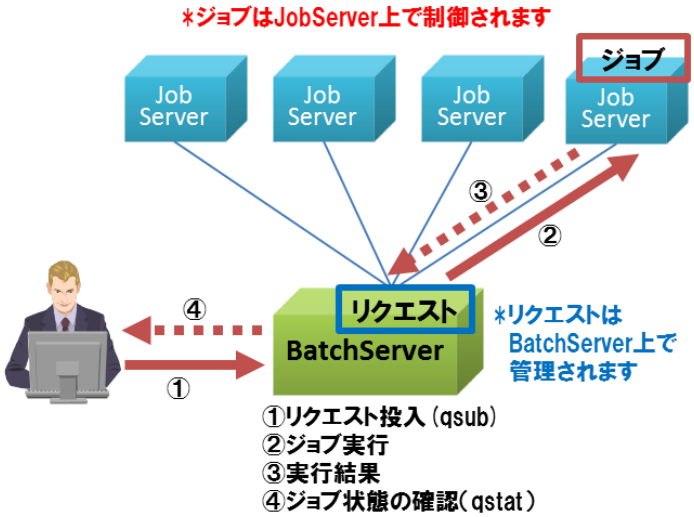


図 1.2-1：シングルジョブの実行

【分散ジョブの実行（ジョブ数＝3)】

リクエストは分散ジョブを生成することができます。

分散ジョブとは一つのリクエストをユーザが指定した数のジョブに分散させたジョブです。以下はジョブ数 3 の分散ジョブを生成した例です。

```
$ qsub -T distrib -b 3 script1
```

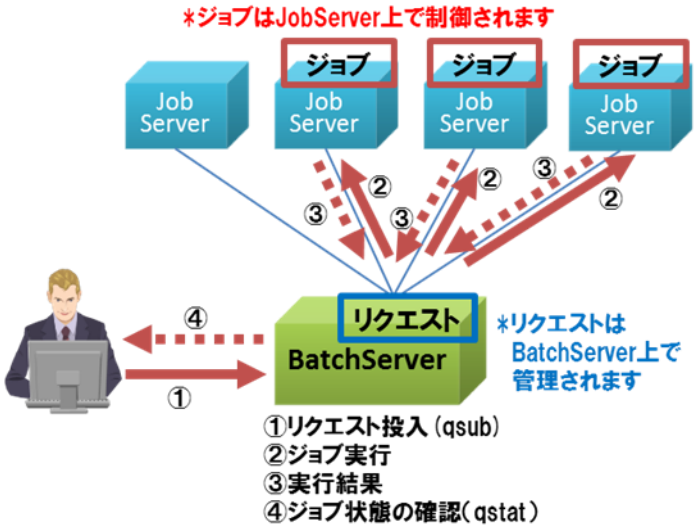


図 1.2-2：分散ジョブの実行

【MPI ジョブの実行（ジョブ数＝３）】

リクエストは MPI ジョブを生成することができます。

MPI ジョブとは 1 つのリクエストをユーザが指定した数のジョブに分散させ、MPI 通信を使用するジョブです。以下はジョブ数 3 の MPI ジョブを生成した例です。

```
$ qsub -T necmpi -b 3 --venum-lhost=1 mpi_script
$ cat mpi_script
mpirun -host 0 -host 1 -host 2 mpi_job
```

necmpi ジョブの場合は、mpirun の-host オプションにラベル名としてユニークな番号を指定してください。mpirun の -host オプションの指定は、ジョブ数分だけ必要です。

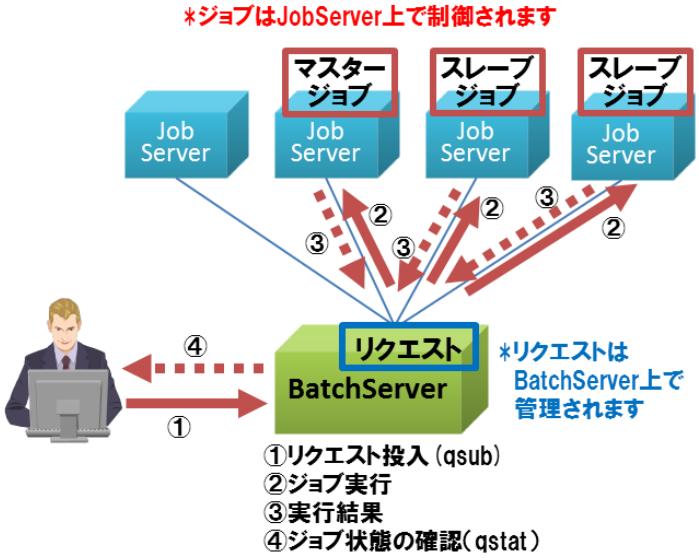


図 1.2-3：MPI ジョブの実行

1.2.8　出力結果ファイルの指定

リクエストの出力結果ファイルは、qsub(1)コマンドの -e オプション、 -o オプションで指定してください。

```
-e file_name
```

file_name に、リクエストの標準エラー出力を格納する結果ファイルを指定します。

```
-o file_name
```

file_name に、リクエストの標準出力を格納する結果ファイルを指定します。

上記の file_name に指定するファイル名は、以下の形式での指定が可能です。

```
[hostname:]path_name
```

hostname で、結果ファイルを出力するホストのホスト名を指定します。hostname が省略された場合は、リクエストを投入したホスト上に出力されます。また、path_name には、出力する結果ファイルのパス名を指定します。相対パス名を指定した場合は、リクエスト投入時のカレントディレクトリからの相対パスとなります。

また、-j オプションを用いることで、リクエストの標準出力と標準エラー出力をマージすることも可能です。

1.2.9　資源制限値の指定

リクエストの資源制限値は、qsub(1)コマンドの -l オプションで指定してください。主なものだけをここに挙げます。（その他の資源制限は、［導入編］リクエストの資源制限 参照）

```
-l filesz_prc=["]max_limit[,warn_limit"]
```

プロセスごとのファイルサイズの最大値や警告値を設定します。

```
-l cputim_prc=["]max_limit[,warn_limit"]
```

プロセスごとの CPU 時間制限の最大値や警告値を設定します。

資源制限値を設定する際の上記 max_limit、および、warn_limit は、以下の形式で指定します。

- 時間制限

時間に関する制限値は、以下の形式で指定してください。

[[hours:]minutes:]seconds[.milliseconds]
【指定例】

1234:58:21.29	1234時間 58 分 21.29秒
59:01	59分 1秒
12345	12345 秒
121.1	121.1 秒

- サイズ制限

サイズに関する制限値は、以下の形式で指定してください。
[integer][.fraction][units]
units に指定できるものは、以下のとおりです。

b	バイト
kb	キロバイト
mb	メガバイト
gb	ギガバイト
tb	テラバイト
pb	ペタバイト
eb	エクサバイト

units を指定しない場合は、バイトと解釈されます。また、1024 倍しても integer 部が 1 以上でない数字は指定できません。
【指定例】

1234	1234バイト
1234kb	1234キロバイト
1234.5gb	1234.5ギガバイト

資源制限には、最大値と警告値の 2 つを指定できるものがあります。警告値を指定する場合は、最大値以下の値でなければなりません。警告値を省略した場合は、最大値と同じとみなされます。
リクエストの使用資源量が警告値を超えた場合はそれぞれの資源制限で定められたシグナル（プロセス毎の資源制限値の場合は setrlimit(2)の man に記載されたシグナル、論理ホスト毎の CPU 時間資源制限値の場合は SIGXCPU、論理ホスト毎のその他の資源制限値は SIGTERM）が リクエストに送信されます。最大値を超えた場合は直ちにリクエストの実行が中断されます。

また、NUMA アーキテクチャのスカラマシン（Linux）の実行ホストがバインドされソケットスケジューリング機能が ON となっているキューに、リクエストを投入する場合、qsub –l オプションに cpunum_job（ジョブ毎 CPU 台数制限）の代わりに、socknum_job（ジョブ毎のソケット数制限）を指定することができます。

socknum_job によってソケット数を指定した場合、投入先キューにバインドされている実行ホストの搭載資源量（ソケット当たりの CPU コア数）に応じた CPU コア数が cpunum_job に指定されたものとして動作します。

【例】1 ソケット 10 コアが搭載されている実行ホストがバインドされているキューに、

qsub -l socknum_job=2 を指定して投入した場合、qsub -l cpunum_job=20 を指定したものとして動作します。

【注意】

- ソケットスケジューリング機能が有効なキューにリクエストを投入した場合、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。
  なお、OpenMP と組み合わせた MPI を実行する場合、リクエスト投入時に-v オプションで OMP_NUM_THREADS 環境変数の値を指定してください。

VE ノード毎の資源制限値

VE ノード毎に資源使用量の制限値を設定することができます。

```
--<資源名称>-venode=resource
resource : ["]max_limit[,warn_limit"]
```

以下に示すような VE ノード毎の資源制限が指定できます。

表 1-1 VE ノード毎の資源制限

オプション	意味	指定可能な値の範囲
--vecputim-venode= max_limit[,warn_limit]	VE ノード毎の VE の CPU時間	0～2147483647の整数 またはunlimited
--vememsz-venode= max_limit[,warn_limit]	VEノード毎のVEのメモリサイズ	0～2147483647の整数 またはunlimited

※資源制限値を指定する際の上記 max_limit、および、warn_limit は、前述の-l オプション指定と同じ形式で指定します。

※実行ホストに搭載されていない資源を指定すると、そのリクエストは実行できません。

論理ホスト毎の資源制限値

また、上記とは異なる指定方法で、リクエスト投入時に -b job_count 指定で作成された各ジョブ（論理ホスト）について、論理ホスト毎に資源使用量の制限値を設定することができます。

```
--venum-lhost=num
```

各論理ホストに対して割当てを必要とする VE ノード数（論理ホスト毎の VE ノード数制限値）を設定します。num には、には 0～256 の整数または unlimited が指定可能です。

```
--<資源名称>-lhost=resource
resource : ["]max_limit[,warn_limit"]
```

論理ホスト毎の資源量（資源制限）を設定します。

本オプションは -l <資源名称>_job に相当するものです。以下に示すような論理ホスト毎の資源制限が指定できます。

表 1-2 論理ホスト毎の資源制限

オプション	意味	指定可能な値の範囲
--cpunum-lhost=*max_limit*	論理ホスト毎CPU台数	1～2147483647の整数 またはunlimited
--cputim-lhost=*max_limit*[,*warn_limit*]	論理ホスト毎CPU時間	0～2147483647の整数 またはunlimited
--gpunum-lhost=*max_limit*	論理ホスト毎GPU台数	0～256の整数 またはunlimited
--memsz-lhost= *max_limit*[,*warn_limit*]	論理ホスト毎メモリサイズ	0～2147483647の整数 またはunlimited
--venum-lhost=*max_limit*	論理ホスト毎VE台数	0～256の整数 またはunlimited
--vecputim-lhost= *max_limit*[,*warn_limit*]	論理ホスト毎のVEのCPU時間	0～2147483647の整数 またはunlimited
--vememsz-lhost= *max_limit*[,*warn_limit*]	論理ホスト毎のVEのメモリサイズ	0～2147483647の整数 またはunlimited
--vmemsz-lhost=*max_limit*[,*warn_limit*]	論理ホスト毎仮想メモリサイズ	0～2147483647の整数 またはunlimited
--stderrsz-lhost=*max_limit*[,*warn_limit*]	論理ホスト毎の標準エラー出力の結果ファイルサイズ	0～2147483647の整数 またはunlimited
--stdoutsz-lhost=*max_limit*[,*warn_limit*]	論理ホスト毎の標準出力の結果ファイルサイズ	0～2147483647の整数 または unlimited

※資源制限値を指定する際の上記 max_limit、および、warn_limit は、前述の-l オプション指定と同じ形式で指定します。

※実行ホストに搭載されていない資源を指定すると、そのリクエストは実行できません。

1.2.10 開始時刻の指定

リクエストの開始時刻は、qsub(1)コマンドの -a オプションで指定してください。

```
-a date-time
```

指定した時刻までリクエストの実行を待ちあわせます。date-time の指定例は以下の通りです。

```
1200              リクエストを投入した日(過ぎていれば翌日)の 12:00
10100900     リクエストを投入した西暦年(過ぎていれば翌年)の 10月10日 09:00
201701010900     2017年 1月1日 09:00
```

1.2.11 実行開始、終了のメール通知の指定

リクエストの実行開始、終了をメールで通知するには、qsub(1)コマンドの -m オプション、 -M オプションで指定してください。

```
-m mail_options
```

mail_options には以下の文字が指定可能です。

b　　バッチリクエストの実行を開始した時にメールを発信します。

e　　バッチリクエストの実行が終了(異常終了を含む)した時にメールを発信します。

a　　バッチリクエストの実行が異常終了した時にメールを発信します。

異常終了とは、実行されていたジョブのうち少なくとも一つがシグナル終了したか、またはハードウェア障害によって強制終了した場合を指します。

n メールを発信しません。

また、mail_options には、次のように上記の文字を組み合わせて指定することもできます。

-m eb バッチリクエストの実行を開始した時と終了した時にメールを発信します。

```
-M mail-list
```

mail-list に、メールの送信先を指定します。mail-list は以下の形式で指定してください。

mail_address[,mail_address, ...]

指定できる mail_list の最大長は 1023 バイトです。

本オプションが指定されない場合、ジョブを投入したホスト上の投入ユーザにメールを送信します。

1.2.12 プライオリティの指定

リクエストのキュー内のプライオリティは、qsub(1)コマンドの -p オプションで指定してください。

```
-p priority
```

リクエストのキュー内のプライオリティを設定します。priority に指定可能な値は［-1024...1023］の範囲の整数です。値が大きいほど優先度が高くなります。ユーザがリクエストプライオリティを指定しなかった場合は、システムが 0 を割り当てます。

1.2.13 アカウントコードの指定

リクエストのアカウントコードは、qsub(1)コマンドの -A オプションで指定してください。

```
-A account_string
```

リクエストのアカウントコードを指定します。

1.2.14 リラン可否の指定

リクエストのリランの可否は、qsub(1)コマンドの -r オプションで指定してください。

```
-r { y | n }
```

"y"を指定するとリクエストがリラン可能、"n"を指定するとリラン不可であることを宣言します。
（リクエストのリランの詳細については、1.10.バッチリクエストのリラン を参照）

1.2.15 ホールド可否の指定

リクエストのホールドの可否は、qsub(1)コマンドの -H オプションで指定してください。

```
-H { y | n }
```

"y"を指定するとリクエストがホールド可能、"n"を指定するとリクエストがホールド不可であることを宣言します。
（リクエストのホールドの詳細については、1.6.バッチリクエストのホールド を参照）

1.2.16 環境変数の利用

環境変数を利用するには、以下の場合があります。それぞれ、使用する qsub(1)コマンドのオプションは、次のとおりです。

・リクエスト投入時のすべての環境変数をリクエストの実行環境にエクスポートする場合は -V オプションを指定してください。

・リクエストを実行するときに使用する環境変数は、-v オプションで指定してください。

```
-v variable_list
```

【例】

```
#!/bin/sh

#PBS -v NP=4          #The number of MPI processes
#PBS -T necmpi
#PBS -b ${NP}
#PBS -I "bin/mpi_prog"
#PBS --venum-lhost=1

cd ${STGDIR}
mpirun -np ${NP} mpi_prog
```

リクエスト実行時、以下の環境変数が既定値として設定されています。PBS_ENVIRONMENT、PBS_JOBID、PBS_SUBREQNO、PBS_JOBNAME、PBS_NODEFILE 以外の環境変数の値は-v オプションによって変更することができます。

環境変数名	説明
PBS_ENVIRONMENT	バッチリクエストがバッチ処理であることを示す、"PBS_BATCH"が設定されます。
PBS_JOBID	バッチリクエストのジョブIDが設定されます。
PBS_SUBREQNO	サブリクエスト番号が設定されます（パラメトリックリクエストの場合）
PBS_JOBNAME	バッチリクエスト名が設定されます。
PBS_NODEFILE	リクエストを構成するジョブが実行されているホスト一覧が記載されたファイルへのパスが設定されます。
PBS_O_HOME	クライアントホスト上の環境変数 HOME が設定されます。
PBS_O_HOST	クライアントホスト名が設定されます。
PBS_O_LANG	クライアントホスト上の環境変数 LANG が設定されます。
PBS_O_LOGNAME	クライアントホスト上の環境変数 LOGNAME が設定されます。
PBS_O_MAIL	クライアントホスト上の環境変数 MAIL が設定されます。
PBS_O_PATH	クライアントホスト上の環境変数 PATH が設定されます。
PBS_O_SHELL	クライアントホスト上の環境変数 SHELL が設定されます。
PBS_O_TZ	クライアントホスト上の環境変数 TZ が設定されます。
PBS_O_WORKDIR	クライアントホスト上の作業用ディレクトリを設定します。

・スクリプトファイルの埋め込みオプション行でのみ使用する環境変数は、-w オプションで指定してください。

```
-w script_variable_list
```

【例】

```
#!/bin/sh
```

```
#PBS -N Sample_JOB
#PBS -w ELAPS=600
#PBS -m b -m e
#PBS -q batch1 #default
#PBS -l elapstim_req=${ELAPS}
```

-v,-w で指定された環境変数およびスクリプト変数は、埋め込みオプション行内で参照することができます。参照方法は次のとおりです。

${variable}

同じ変数名が存在する場合の優先順位は以下のとおりです。

1. スクリプト内の埋め込みオプション行で使用できる変数　（-w指定）
2. qsubのコマンドオプションで指定された環境変数（-v指定）
3. 環境変数

以下の環境変数は NQSV が生成したものをジョブに渡すため、-V または-v オプションで指定したものをジョブに渡すことはできません。

CUDA_VISIBLE_DEVICES	ENVIRONMENT
FLMOD	HOME
HYDRA_LAUNCHER	HYDRA_LAUNCHER_EXEC
I_MPI_HYDRA_BOOTSTRAP	I_MPI_HYDRA_BOOTSTRAP_EXEC
JMM_COMM_PORT	LAM_MPI_SESSION_PREFIX
LAM_MPI_SESSION_SUFFIX	LOGNAME
MPICH_PROCESS_GROUP	MPI_BATCH_CMD
MPI_MAX_REMSH	MPI_REMSH
NMPI_LAUNCHER_EXEC	NMPI_TTY_COMPAT
NQSII_ELAPSE_OVER_FILE	NQSII_MPINODES
NQSII_MPIOPTS	NQSII_MPI_OUTPUT
NQSII_MPI_OUTPUT_STDERR	NQSII_MPI_OUTPUT_STDOUT
NQSV_MPIOPTS	OMPI_MCA_plm_rsh_agent
OMPI_MCA_plm_rsh_no_tree_spawn	P4_RSHCOMMAND
PATH	PBS_ENVIRONMENT
PBS_EXCLUSIVE	PBS_JOBID
PBS_JOBNAME	PBS_JSVNO
PBS_NODEFILE	PBS_SUBREQNO
QUEUENAME	SHELL
STGDIR	TERM
USER	VE_LIMIT_OPT
_MPI_RNKFL	_NECMPI_JOBTYPE
_NECMPI_MID	_NECMPI_REQID
_NECMPI_VH_NODEFILE	_NECMPI_VH_NODENUM
_NEC_NQSV_JOB	

1.2.17 スクリプトの実行シェルの指定

リクエストを実行するシェルは、qsub(1)コマンドの -S オプションで指定してください。

```
-S path_name
```

path_name に、リクエストのスクリプトを実行するシェルの絶対パス名を指定します。このオプションを指定しない場合は、以下の方式でシェルを選択します。

リクエストを実行する際に、まずリクエストのユーザのログインシェルを起動します。次にそのログインシェルが、リクエストの内容から適切なシェルを選択し、そのシェル（ログインシェルとして起動されません）がリクエストを実行します。つまり、あたかもインタラクティブな処理と同様な形態でリクエストを実行します。

1.2.18 ジョブの実行条件（ジョブコンディション）の指定

ジョブコンディションとは、ジョブをどういった条件の実行ホスト（ジョブサーバ）で 実行させるのかといった実行条件をリクエスト投入時に指定するものです。

ジョブコンディションの機能とその指定によるスケジューリングの詳細は、［JobManipulator 編］を参照ください。

ジョブコンディションは、 qsub(1) 、qlogin(1)、qrsh(1)コマンドの -B オプションで指定してください。

```
-B ["]job_condition[","job_condition"...]
```

job_condition に、スケジューラがジョブをジョブサーバに割り当てるための条件を指定します。

job_condition の形式は、以下のとおりです。

[job_number:] condition

- job_number には、ジョブコンディションを適用するジョブのジョブ番号を指定します。以下の指定方法が可能です。

(1) 数字の単独指定　　例）0:

(2) ","で区切って複数の数字指定　　0,2,5:

(3) "-"による連続の番号指定　　0-4:

(4) (2),(3)の組み合わせ　　0,2,4-6:

(5) "ALL"による全ジョブ指定　　ALL:

(6) 指定なしの場合、(5)"ALL"による全ジョブ指定 と同義

- conditionには、スケジューラへの条件文を指定します。

以下の指定が可能です。

種別=指定内容[;種別=指定内容...]

種別と指定内容は以下です。

(1)　JSV番号（ジョブサーバ番号）

JSV=jsv_no

ジョブサーバ番号をjsv_noに指定します。以下の指定方法が可能です。

(1) 数字の単独指定　　例）0

(2) ","で区切って複数の数字指定　0,2,5

(3) "-"による連続の番号指定　　0-4

(4) (2),(3)の組合せ　　0,2,4-6

(2)　HWタイプ

HW=name[,name....]

HWタイプ名を指定します。　　例）HW=x86_64

ノードグループ

NGRP=name[,name....]

ノードグループ名を指定します。　　例）NGRP=grp1,grp2

【注意】

・job_numberの指定方法一致しない文字列は、conditionとして扱われ、job_numberはALL:となります。
・-B job_conditionを複数回指定する、または、-B "job_condition","job_condition"... のようにして、同じジョブ番号（job_number）に対して、job_conditionを複数回指定した場合、最後に指定した条件（condition）のみが有効になります。
・同じジョブ番号（job_number）に対して条件の種別を複数指定したい場合は、condition内でセミコロン（;）を用いて区切ってください。
・job_conditionに指定できる文字列は255バイトまでです。

[指定例]

JSV 番号(ジョブサーバ番号）の指定

```
-B ”0:JSV=100”
-B ”1-2:JSV=101-102,105”
```

HW タイプの指定

```
-B ”ALL:HW=x86_64”
```

ノードグループの指定

```
-B ”ALL:NGRP=node_group”
```

1.2.19 転送ファイルの指定

ファイルステージング機能で転送するファイルは、 qsub(1)コマンドの -I オプション、 -O オプションで指定してください。

```
-I "stage_in"[,"stage_in"...]
```

ジョブ実行に関係するファイル群を、ジョブ実行に先立ってクライアントホスト上から実行ホスト上へ転送します。

```
-O "stage_out"[,"stage_out"...]
```

ジョブの出力であるファイル群を、ジョブ実行後に実行ホスト上からクライアントホスト上に転送します。

（ファイルステージングの詳細は、［管理編］ファイルステージング を参照）

1.2.20 ジョブマイグレーションに関する指定

リクエストのジョブマイグレーションの可否は、qsub(1)コマンドの -J オプションで指定してください。

```
-J { y | n }
```

"y"を指定するとリクエストがジョブマイグレーション可能、"n"を指定するとリクエストがジョブマイグレーション不可であることを宣言します。

ジョブマイグレーション時に転送先ホストへ同時に転送したいユーザファイルは、 qsub(1)コマンドの -G オプションで指定してください。

```
-G ["]migration_file_path[","migration_file_path"...]
```

ユーザファイル migration_file_path は、絶対パスで指定してください。

（ジョブマイグレーションの詳細は、［管理編］ジョブマイグレーション を参照）

1.2.21 投入時メッセージ表示の変更

リクエストの投入に成功したときのメッセージ表示を変更するには、qsub(1)コマンドの以下のオプションを指定してください。

- -zオプション　メッセージを何も表示しません。

- -Zオプション　リクエストIDのみを表示します。

1.2.22 リクエスト連携機能

リクエスト投入時に、複数のスクリプトファイルを指定することができます。この場合、リクエスト間に関連性を持たせて実行順序を制御するリクエスト連携機能が動作します。

リクエスト連携には直列連携と並列連携があります。

直列連携: 先行リクエストの実行完了を待って後続リクエストが実行を開始します。qsub(1)コマンドでは、script1 script2 ...のようにスクリプトファイルをスペースで区切って指定します。

並列連携: 連携している全リクエストが一斉に実行を開始します。qsub(1)コマンドでは、script1:script2:...のようにスクリプトファイルをコロン":"で区切って指定します。

リクエスト連携機能では、投入された全ての連携対象リクエストは一旦、管理者権限による HELD 状態となりスケジューリング対象から外されます。全ての連携対象リクエストの投入が完了した時点で先頭のリクエストがリリースされます。したがって、リクエスト投入途中にオプション解析エラー等が発生した場合、すでに投入されているリクエストは HELD 状態のまま残ります。

リクエスト連携機能の動作時、qsub(1)コマンドに指定されたオプションは全てのリクエストに対して有効となります。スクリプトファイル内に指定された埋め込みオプションはそのリクエストに対してのみ有効となります。

他のバッチサーバに転送する転送キューに連携リクエストを投入しないでください。他のバッチサーバにリクエストが転送された段階で、そのリクエストは投入元バッチサーバの管理外となるため、リクエスト間の連携関係を保つことができません。

1.2.23 排他実行リクエスト

リクエストに対して、1 ホストあたり 1 ジョブで実行することを指定できます。

このような実行方法を排他実行と呼び、この指定で投入されたリクエストを排他実行リクエストと呼びます。

(1) 排他実行リクエストの投入

排他実行リクエストを投入するには、qsub 時に--exclusive オプションを指定してください。

--exclusive オプションには引数として"host"を指定します。

引数の"host"は省略可能です。投入時のオプションのフォーマットは以下の通りです。

```
qsub --exclusive[=host]
```

本オプションとプロビジョニング機能のテンプレート指定（--template）は同時に使用することが

できず、同時指定した場合は投入エラーになります。

排他実行リクエストには、論理ホスト毎およびジョブ毎の資源制限値（CPU、メモリ、仮想メモリ、GPU、VE）を同時に指定が可能です。

ただし、指定された値はスケジューリングに利用されず、また、実行ホスト上での制限超過チェックも行われないため、意味を持ちません。

排他実行リクエストにおいて、資源制限値の扱いは以下となります。

対象制限値	扱い
論理ホスト（ジョブ）毎の CPU 台数制限	実行ホストに搭載されている全 CPU を割り当てる。ソケットスケジューリング機能ではホスト内の全 CPU を割り当てる
論理ホスト（ジョブ）毎のメモリ量制限	実行ホストに搭載されている全メモリ（実メモリ・仮想メモリ）を割り当てる。ソケットスケジューリング機能では全メモリノードを割り当てる
論理ホスト（ジョブ）毎の CPU 時間制限	そのまま適用する
プロセス毎の制限値	そのまま適用する
論理ホスト（ジョブ）毎の GPU 台数制限	実行ホストに搭載されている全 GPU を割り当てる
VE および HCA	実行ホストに搭載されている全 VE&HCA を全て割り当てる

また、排他実行リクエストでは、CPUSET 機能による RSG は利用することができません。キューに 0 番以外の RSG 番号が指定されていても、常に RSG0(全資源）で資源割当てを行います。

本機能利用時は、キューに対する、CPU 台数制限値、GPU 台数制限値、および VE ノード数の資源既定値として 1 以上の値を設定してください。

(2)　排他実行リクエスト機能を利用したジョブ実行

排他実行リクエスト機能を利用して、リクエスト内の特定のジョブを高性能な実行ホストに割り当てて実行したり、各ジョブに実行ホストの全資源を占有して割り当てて実行したりすることができます。

- 特定ジョブの特定実行ホストへの割当て実行

MPI によるマルチノード実行において、スレーブジョブの計算結果をランク 0 に集約し、ランク 0 のマスタージョブで集約した結果をまとめて処理するような使い方が考えられます。

同一クラスタ中に CPU やメモリが多く搭載されている実行ホスト(以後リッチノード)が存在する場合、そのようなリッチノードで上記のランク 0 の重い処理を実行することにより処理全体の高速化や効率的な資源利用が見込まれます。

このような実行形態を、排他実行リクエスト機能とジョブコンディション機能を用いて実現することができます。

(1) ノードグループの作成

まず、マスタージョブを実行するノード（リッチノード）が属するノードグループを作成します

```
$ qmgr -Pm
Mgr: create node_group = richnode
```

そのうえで、マスタージョブを実行するノード（リッチノード）の JSV をノードグループに追加します。

```
Mgr: edit node_group add job_server_id = 100-200 richnode
Add Job_Server to Node_group (richnode).
```

次に、スレーブジョブを実行するノード（非リッチノード）が属するノードグループを作成します。

```
$ qmgr -Pm
Mgr: create node_group = slavenode
```

そのうえで、スレーブジョブを実行するノードの JSV をノードグループに追加します。

```
Mgr: edit node_group add job_server_id = 300-400 slavenode
Add Job_Server to Node_group (slavenode).
```

(2) リクエストの投入

排他実行リクエストとしてリクエストを投入します。そのためには、ノードグループ richnode に登録されているノードのいずれかで 0 番目のジョブ（マスタージョブ）を実行し、ノードグループ slavenode に登録されているノードのいずれかで 1 番目以降のジョブ（スレーブジョブ）を実行するようジョブコンディションを指定します。ここでは全部で 64 ジョブのリクエストであるとします。

```
qsub --exclusive -b 64 -B "0:NGRP=richnode","1-63:NGRP=slavenode"
```

- 実行ホストの資源占有実行

ジョブの高速実行や性能ブレ防止のために、マルチノードリクエストの各ジョブに対し、実行ホストの全資源を割り当てて実行することが考えられます。排他実行リクエスト機能を利用すればこのような実行が可能です。

実のところ排他実行リクエスト機能は、実行ホストの資源占有実行そのものです。従って、リクエスト投入時に --exclusive を指定するだけです。

```
qsub --exclusive script
```

1.2.24 ハイブリッドリクエスト

通常のリクエストでは複数のジョブ（＝論理ホスト）が存在する場合は、それぞれのジョブが使用できる資源量は全て同じです。一方、リクエストに内包されるジョブ毎に異なる資源量を使用する実行形態をハイブリッドリクエストと呼びます。

ハイブリッドリクエストの詳細につきましては第 16 章 ハイブリッドリクエストを参照してください。

1.2.25 ユーザ指定のリクエスト投入

qsub 時に--user=*username* オプションを指定してリクエストを投入することで、username に指定されたユーザーをオーナーとしてリクエストを投入することができます。ジョブは指定したユーザーで実行され、そのときの実行グループは、指定したユーザーのプライマリグループになります。

ユーザー指定のリクエスト投入ができるのはグループ管理者権限があるユーザのみで、本オプションに指定可能なユーザーは、グループ管理者として管理対象となっているグループをプライマリグループとするユーザーのみです。

また、ユーザーマッピングによりバッチサーバーホストとリモートホスト（投入ホスト、実行ホスト）で異なるユーザーにマッピングしている場合には、--user オプションは指定できません。

本オプションと--group オプションを併用する場合、--group オプションには投入ユーザーの補足グループではなく、以下の条件に合致するグループが指定可能です。

- --userオプションで指定されたユーザーの補足グループであること
- 投入ユーザーの（グループ管理者としての）管理対象のグループに含まれていること

本オプションを指定して投入したリクエストに対して、投入ユーザが qalter, qdel などのリクエスト操作を行う場合、操作コマンドに対してグループ管理者権限である-Pg オプションの指定が必要です。

アクセス制限が設定されたキューでは、本オプションの指定の有無にかかわらず、投入操作を行ったユーザーおよびプライマリグループでアクセスのチェックが行われます。その他のユーザ指定を伴う各種制限は、すべて--user オプションで指定したユーザーに対して行われます。

ステージング対象のファイルは全て、--user オプションで指定したユーザーとそのプライマリグループの権限で作成されます。

1.2.26 ジョブ実行時の SIGTERM 捕捉

qsub 時に sigterm 対応オプション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 SIGTERM を補足し、trap コマンドが利用できるようにします。

sigterm 対応オプションは qsub 時に--accept-sigterm オプションで以下のように指定します。

```
qsub --accept-sigterm=<val>
```

<val>に yes を指定した場合は有効となります。<val>に no を指定すると無効になります。

本オプションを指定しない場合のデフォルトの動作は no（無効）です。

1.3　バッチリクエストの状態確認

1.3.1　基本情報の確認

投入したリクエストの状態は、qstat(1)コマンドで確認してください。リクエストの指定は、リクエスト ID で行います。

以下に表示例を示します。

【リクエスト ID 指定時】

```
$ qstat 72.host.example.com
RequestID       ReqName  UserName Queue     Pri STT S   Memory      CPU   Elapse R H M Jobs
--------------- -------- -------- -------- ---- --- - -------- -------- -------- - - - ----
72.host.example STDIN    user1    batch1      0 RUN -    1.93M     0.23       26 Y Y Y    1
```

【リクエスト ID 指定なし】

```
$ qstat
RequestID       ReqName  UserName Queue     Pri STT S   Memory      CPU   Elapse R H M Jobs
--------------- -------- -------- -------- ---- --- - -------- -------- -------- - - - ----
72.host.example STDIN    user1    batch1      0 RUN -    1.93M     0.23       26 Y Y Y    1
73.host.example STDIN    user1    batch1     20 QUE -    0.00B     0.00        0 Y Y Y    1
74.host.example STDIN    user1    batch2     20 QUE -    0.00B     0.00        0 N N N    2
```

STT のカラムにリクエストの状態が表示されます。ここで表示される意味は、それぞれ以下のとおりです。

表 1-3 リクエストの状態

RUN	実行中
QUE	実行待ち状態
WAT	開始時刻の待ち合わせ中
HLD	ホールド中
HOL	ホールド要求中
SUS	実行一時中断の要求中，実行一時中断中，実行一時中断から実行再開中(*いずれもサマリ表示では "SUS" と表示されます。)
ARI	転送キューからの受信中
TRS	転送キューからの送信中
EXT	実行結果ファイルの転送中
PRR	実行前処理中（MPIリクエストの場合、スレーブジョブの実行開始待ちの状態を含む）
POR	実行後処理中（MPIリクエストの場合、スレーブジョブの終了待ちの状態を含む）
MIG	動的ジョブマイグレーションによるリクエスト移動中
STG	ジョブの生成中、およびステージイン対象ファイルの実行ホスト上への転送中

STT 以外の項目の内容については、qstat(1)を参照してください。

【注意】

・ /etc/opt/nec/nqsv/nqsd.conf に enable_memory_cgroup on を設定している場合、Memory に表示する値にはファイルキャッシュの使用量を含みます。

1.3.2　詳細情報の確認

各リクエストについての詳細情報を表示するには、qstat(1)コマンドに -f オプションを付けて実行してください。

```
$ qstat -f 72.host1
Request ID: 72.host1
    Request Name = STDIN
    User  Name = user1
    User  ID   = 111
    Group ID   = 100
    Current State          = Running
    Previous State         = Pre-running
```

```
State Transition Time   = Mon Aug 26 19:25:13 2017
State Transition Reason = PRERUN_SUCCESS
Queue = batch1@host1 (Execution Queue)
Job Topology = Distribute Job
Request Privilege = Level 0
Request Priority = 0
Request Loglevel = 0
Rerunable = Yes
Holdable  = Yes
Hold Type = (none)
Migratable   = Yes
Suspend Type = (none)
Account Code = (none)
Stdout = host1:/usr/nec/STDIN.o%s
Stderr = host1:/usr/nec/STDIN.e%s
Reqlog = host1:/usr/nec/STDIN.l%s
Shell = (none)
Mail Address = (none)
Mail Option  = (none)
Job Condition:
   Job NO: 0 ""
Number of Jobs = 1
Created Request Time = Mon Aug 26 19:23:30 2017
Entered Queue Time   = Mon Aug 26 19:23:30 2017
Planned Start Time   = Mon Aug 26 19:23:32 2017
Execute Request Time = (none)
Started Request Time = Mon Aug 26 19:23:32 2017
Ended Request Time   = Mon Aug 26 19:25:13 2017
Requested Start Time = (none)
Deadline Time       = (none)
UMASK = 022
Checkpoint Interval = 0
Restart File Directory = (none)
Reservation ID = (none)
qattach command = Enable
Attach = No
Cluster Type Select = NONE
UserPP Script = (none)
Exclusive = (none)
VE Node Number = 1
```

```
  HCA Number = (none)
  Accept Sigterm = No
Execution Hosts(JSVNO):
  ehost101(101)
Resources Information:
  Memory    = 1.932587MB
  CPU Time  = 0.225019S
  Accumulated CPU Time = 0.225019S
  Elapse    = 11S
  Remaining Elapse = 446S
  Virtual Memory = 355.789062MB
VE Node Resources:
  VE CPU Time          = Max: UNLIMITED Warn: UNLIMITED
  VE Memory Size       = Max: UNLIMITED Warn: UNLIMITED
Logical Host Resources:
  VE Node Number       = Max:       1 Warn:      ---
  CPU Number           = Max:       2 Warn:      ---
  GPU Number           = Max:       0 Warn:      ---
  CPU Time             = Max:       0 Warn:      ---
  Memory Size          = Max: UNLIMITED Warn: UNLIMITED
  Virtual Memory Size  = Max: UNLIMITED Warn: UNLIMITED
  VE CPU Time          = Max: UNLIMITED Warn: UNLIMITED
  VE Memory Size       = Max: UNLIMITED Warn: UNLIMITED
  Stdout Size          = Max: UNLIMITED Warn: UNLIMITED
  Stderr Size          = Max: UNLIMITED Warn: UNLIMITED
VE Node Resources:
  VE CPU Time          = Max: UNLIMITED Warn: UNLIMITED
  VE Memory Size       = Max: UNLIMITED Warn: UNLIMITED
Resources Limits:
  (Per-Req) Elapse Time Limit      = Max:     600S Warn:     600S
  (Per-Job) CPU Time               = Max: UNLIMITED Warn: UNLIMITED
  (Per-Job) CPU Number             = Max:       1 Warn:      ---
  (Per-Job) Memory Size            = Max: UNLIMITED Warn: UNLIMITED
  (Per-Job) Virtual Memory Size    = Max: UNLIMITED Warn: UNLIMITED
  (Per-Job) GPU Number             = Max:       0 Warn:      ---
  (Per-Prc) CPU Time               = Max: UNLIMITED Warn: UNLIMITED
  (Per-Prc) Open File Number       = Max: UNLIMITED Warn:      ---
  (Per-Prc) Virtual Memory Size    = Max: UNLIMITED Warn: UNLIMITED
  (Per-Prc) Data Segment Size      = Max: UNLIMITED Warn: UNLIMITED
  (Per-Prc) Stack Segment Size     = Max: UNLIMITED Warn: UNLIMITED
```

```
    (Per-Prc) Core File Size         = Max: UNLIMITED Warn: UNLIMITED
    (Per-Prc) Permanent File Size     = Max: UNLIMITED Warn: UNLIMITED
    (Per-Prc) VE CPU Time            = Max: UNLIMITED Warn: UNLIMITED
    (Per-Prc) VE Memory Size         = Max: UNLIMITED Warn: UNLIMITED
  Kernel Parameter:
    Resource Sharing Group     = 0
    Nice Value               = 0
  User Attributes:
    (none)
```

各項目の内容については、qstat(1)を参照してください。

ハイブリッドリクエストの扱い

ハイブリッドリクエストを参照する場合は特別な操作が必要です。

詳細については 16.3 ハイブリッドリクエスト情報の参照を参照してください。

【注意】

・ /etc/opt/nec/nqsv/nqsd.conf に enable_memory_cgroup on を設定している場合、Resources Information:の Memory に表示する値にはファイルキャッシュの使用量を含みます。

1.3.3　出力情報のカスタマイズ

(1)　時間表示

qstat(1)コマンドの実行結果のうち、時間データはデフォルトでは秒単位での表示ですが、-d オプションを指定することによって、d+hh:mm:ss（日+時:分:秒）の形式で表示することが可能です。本オプションは下記の項目で有効となります。

- CPU時間使用量（CPU）
- 積算CPU時間使用量（ACCPU）
- Elapse time (Elapse)

下記は、-d オプションを指定した場合のコマンド例です。

```
$ qstat -d
RequestID       ReqName  UserName Queue     Pri STT S   Memory        CPU      Elapse R H M Jobs
--------------- -------- -------- -------- ---- --- - -------- ----------- ----------- - - - ----
80536.host1     STDIN    user1    batch1      0 QUE -    7.62M  1+05:21:10  1+16:30:12 Y Y Y    1
```

(2)　ロング表示

qstat(1)コマンドの基本情報表示(-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場合があります。-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。

下記は、-l オプションを指定した場合のコマンド例です。

```
$ qstat -l
RequestID                          ReqName             UserName        Queue           Pri  STT
S       Memory          CPU       Elapse R H M Jobs
------------------------------------ -------------------- --------------- --------------- ---- ---
- ------------ ------------ ------------ - - - ----
80333.host1.example.com               STDIN               nqstest1        que1              0 QUE
-        0.00B        0.00           0 Y Y Y    1
```

-l オプションによる出力は、表示文字数も拡大されますが、表示内容の文字数がその文字数を超える場合も、全内容を表示します。このとき、ヘッダと表示するカラムがずれることがあります。

(qstat(1)コマンド-l オプションの詳細については、7.5.全内容の表示 を参照)

(3)　選択表示とソート

リクエストの情報表示の際、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。-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。

リクエストの情報表示で-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。

表 1-4 qstat(1) -F オプションのアイテム

-F指定アイテム	内容	サマリ表示形式
rid	リクエスト識別子	RequestID
reqnm	リクエスト名	ReqName
own	所有者	UserName
quenm	投入キュー名	Queue
pri	リクエストプライオリティ	Pri
stt	リクエスト状態	STT
stall	ストール状態	S
mem	リクエストの使用メモリ量	Memory
cpu	リクエストの現在実行中のプロセスのCPU使用量	CPU
acpu	リクエストの積算CPU使用量(終了したプロセスを含む)	ACCPU
elaps	経過時間	Elapse
relaps	残経過時間	RElapse
rflg	再実行の可否	R

hflg	ホールドの可否	H
mflg	ジョブマイグレーションの可否	M
jobs	リクエストに存在するジョブ数	Jobs
ehost*	ジョブの実行ホスト	ExecutionHost
sdate*	リクエスト投入時刻	Date(SUBMIT)
qdate*	スケジューリング開始時刻	Date(QUEUED)
rdate*	リクエスト実行開始時刻	Date(RUNNING)
att*	アタッチ接続の有無	Att

* 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。

以下に、リクエスト情報を、投入キュー名(quenm)、バッチリクエスト識別子(rid)、リクエスト状態(stt)、ジョブの実行ホスト(ehost)の順にサマリ表示した例を示します。

```
$ qstat -F quenm,rid,stt,ehost
Queue     RequestID       STT ExecutionHost
--------- --------------- --- ---------------
nqstest  80536.host1       RUN exec1.example.c
nqstest  80542.host1     QUE -
nqstest  80543.host1     QUE -
```

（qstat(1)コマンド-F オプションの詳細については、7.2.出力情報のカスタマイズ を参照）

また、qstat(1)の-O, -o オプションにより、上記アイテムをキーとしてリクエスト情報の出力をソートすることができます。

下記の例は、リクエスト ID で降順にソートした場合の例です。

```
$ qstat -O rid
RequestID       ReqName  UserName Queue     Pri STT S   Memory      CPU R H Jobs
--------------- -------- -------- --------- ---- --- - -------- -------- - - ----
74.host1        STDIN    user1    batch2     20 QUE -    0.00B     0.00 N N    2
73.host1        STDIN    user1    batch1     20 QUE -    0.00B     0.00 Y Y    1
72.host1        STDIN    user1    batch1      0 RUN -    1.93M     0.23 Y Y    1
```

（qstat(1)コマンド-o オプション、-O オプションの詳細（エラー時の出力等）については、7.3.出力情報のソート を参照）

1.4　バッチリクエストの属性変更

バッチリクエストにはさまざまな属性がついています。それらの属性は、リクエストを投入した後に変更できます。

バッチリクエストの属性変更は、qalter(1)コマンドで行ってください。

qalter コマンドでは、変更したい属性に対応するオプションを使用し、変更対象のリクエストのリクエスト ID を指定して実行します。たとえば、72.host1 というリクエスト ID を持つリクエストの、プロセスごとの CPU 時間制限値を変更したい場合は、 次のように実行します。

```
$ qalter -l cputim_prc=1000 72.host1
Attribute of Request is altered.
```

属性変更に関しては、以下のことに注意してください。

- リクエストが実行中の場合は、変更できない属性があります。
  - リクエスト実行中にElapse時間制限値を延長した場合、後続にスケジューリングされている全てのリクエストをリランする必要があります。
- リクエストが転送キュー上で転送中の場合は、変更できない属性があります。
- -投入されているキューに設定されている資源制限値を越えるような値への変更はできません。
  - システムがサポートしていない資源制限値に関しては、値を変更できません。

    （実行ホストがLinuxの場合、プロセス毎の資源制限値は変更できません。）

  サポートされている資源制限値の詳細については、[管理編] バッチキューの設定（資源制限値, カーネルパラメータ） を参照してください。

以下に qalter コマンドで変更できる属性に対するオプションのうち 主なものについて説明します。

-a　　リクエスト実行時刻を変更します。

【実行例】

```
$ qalter -a 1720 72.host1
Attribute of Request is altered.
```

-e　　標準エラー出力結果ファイルを変更します。

【実行例】

```
$ qalter -e host1:/usr/result.e 72.host1
```

```
Attribute of Request is altered.
```

-o　　標準出力結果ファイルを変更します。

【実行例】

```
$ qalter -o host1:/usr/result.o 72.host1
Attribute of Request is altered.
```

-p　　リクエストのプライオリティを変更します。

【実行例】

```
$ qalter -p 25 72.host1
Attribute of Request is altered.
```

-r　　バッチリクエストリラン可不可モードを変更します。

【実行例】

```
$ qalter -r n 72.host1
Attribute of Request is altered.
```

-S　　リクエストの実行シェルを変更します。

【実行例】

```
$ qalter -S /bin/csh 72.host1
Attribute of Request is altered.
```

ハイブリッドリクエスト属性の変更

ハイブリッドリクエストにおける属性値の変更については 16.4 ハイブリッドリクエストの属性変更を参照してください。

1.5　バッチリクエストの削除

バッチリクエストの削除は、qdel(1)コマンドで行ってください。

```
$ qdel 72.host1
Request 72.host1 has been deleted.
```

実行中のバッチリクエストは、-g で grace タイムの秒数を指定することにより、リクエストの削除のために SIGKILL を送信するまでの待ち時間を指定することができます（-g オプションにより指定がなければ待ち時間は 5 秒です）。

なお、結果ファイル転送中のバッチリクエストを指定すると、そのリクエストと関連するネットワークリクエストも削除されます。

1.6　バッチリクエストのホールド

バッチリクエストをホールドすることにより、リクエストをスケジューリング対象からはずし、実行が開始されない状態（HELD 状態）にすることができます。

バッチリクエストのホールドは、qhold(1)コマンドで行ってください。

```
$ qhold 72.host1
Hold request 72.host1 is accepted.
```

バッチリクエストのホールドはリクエストの状態が QUEUED（実行待ち）、WAITING（実行遅延）状態で実行可能です。

ただし、ホールドが許可されていないバッチリクエストの場合は、ホールドできません。また、ホールドを行うには、リクエストを投入したキューに設定されているホールド要求権限以上の権限が必要です。

1.7　バッチリクエストのホールド解除

バッチリクエストのホールド解除(リリース)は、qrls(1)コマンドで行ってください。ホールドしたリクエストのホールドを解除することにより、リクエストはホールド前の状態に戻り、再度スケジューリング対象となります。

```
$ qrls 72.host1
Release request 72.host1 is accepted.
```

リクエストのホールド解除は、リクエストのホールド操作を実行した際のユーザ権限より低い権限では実行できません。(ただし、スケジューラ権限でホールドされたリクエストを NQSV 管理者権限でホールド解除することは可能です。)

1.8　バッチリクエストのサスペンド

実行ホスト上で実行中のバッチリクエストは、ジョブに対して SIGSTOP シグナルを送信してリクエストの実行を一時停止(サスペンド)させることができます。サスペンドされたリクエストは SUSPENDED 状態になります。

バッチリクエストの一時停止(サスペンド)は、qsig(1)コマンドで行ってください。バッチリクエストのサスペンドは RUNNING（実行中)状態でのみ可能です。

```
$ qsig -s SIGSTOP 72.host1
Request 72.host1 was sent signal SIGSTOP.
```

サスペンド中(SUSPENDED 状態)のバッチリクエストに対しても経過時間制限は有効です。

また、リクエストのサスペンドを行うには、リクエストを投入したキューに設定されているサスペンド要求権限以上の権限が必要です。

1.9　バッチリクエストのレジューム

バッチリクエストのサスペンド状態からの実行再開(レジューム)は、qsig(1)コマンドで SIGCONT シグナルを送信することで実行します。

```
$ qsig -s SIGCONT 72.host1
Request 72.host1 was sent signal SIGCONT.
```

リクエストのレジュームは、リクエストのサスペンド操作を実行した際のユーザ権限より低い権限では実行できません。(ただし、スケジューラ権限でサスペンドされたリクエストを NQSV 管理者権限でレジュームすることは可能です。)

1.10 バッチリクエストのリラン

バッチリクエストは、実行ホストが割り当てられてジョブが生成された後、リクエストが消滅するまでの任意の状態から、リラン操作によって QUEUED 状態へ戻すことができます。実行ホスト上で実行中のリクエストをリランすると、ジョブの実行を中止して QUEUED 状態へ戻します。このとき、

リクエスト ID は変更されません。

リクエストのリランは、qrerun(1)コマンドで行います。

```
$ qrerun 72.host1
Request 72.host1 has been rerun.
```

ジョブが未生成のリクエストに対してはリランを行うことはできません。

```
$ qrerun 73.host1
NQSrqureq: [BSV EWRNGSTS] Request state isn't suitable. (QUEUED).
```

一つのリクエストがリランできる回数の上限は、アカウントに記録可能な SHRT_MAX(32676)回です。

1.11 バッチリクエストのキューの移動

バッチキューに投入したリクエストを qmove(1)コマンドにより、別のバッチキューに移動する（投入しなおす）ことができます。リクエストに対してキューの移動を行った場合は、リクエスト投入時と同様に、移動するリクエストの資源制限属性が移動先のキューの資源制限値を超えていないかチェックを行います。もし、リクエストが移動先のキューの制限値を超えている場合、キューの移動はエラーとなります。

バッチリクエストの移動は、QUEUED 状態（ジョブが存在しない場合のみ）、WAITING 状態、HELD 状態のリクエストに対してのみ実行可能です。ただし、qmove コマンドに-f オプションを指定した場合、ジョブが生成された後の状態でもリクエストを強制的に削除した上でキューの移動を行います。

```
$ qmove batch1 72.host1
Request 72.host1 has been moved.
```

また、qmove(1)コマンドによりキューを指定することで、特定のキューに投入されている全リクエストを一度に別のキューへ移動することができます。ただし、特別ユーザ権限以下のユーザ権限では、自分のリクエストのみが移動可能です。

```
$ qmove batch1 -q batch2
Request 72.host1 has been moved.
Request 73.host1 has been moved.
Request 74.host1 has been moved.
```

1.12 バッチリクエストの終了確認

リクエストが終了すると、qstat コマンドでは表示されなくなります。

その他、リクエストの終了を確認する方法には、メールによる方法と、qwait(1)コマンドによる待ち合わせを行う方法があります。

1.12.1 メールによるリクエスト終了の確認

バッチリクエストの投入時に、qsub(1)に-m e オプションを指定することで、リクエストの終了時に投入ユーザ宛てにメールを送信することができます。

1.12.2 リクエスト終了の待ち合わせ

qwait(1)コマンドにより、バッチリクエストの終了を待ち合わせることができます。qwait(1)コマンドは、指定したリクエストの終了を待ち合わせて、リクエストが終了した際にその終了状態を表示します。

【例】リクエストが終了コード 45 で終了した場合

```
$ qwait 123.host1
exited 45
```

【例】リクエストが QUEUED 状態で削除された場合

```
$ qwait 124.host1
deleted in the QUEUED state
```

1.13 バッチリクエストの出力ファイル

バッチリクエストの実行が終了すると、そのリクエストの実行結果ファイルとして、リクエストの実行中に標準出力に出力された内容を格納した標準出力ファイルと、標準エラー出力に出力された内容を格納した標準エラー出力ファイルとが作成されます。

標準出力ファイル、および、標準エラー出力ファイルのファイル名は以下のように命名されます。

【標準出力ファイル】

```
＜リクエスト名＞.o＜シーケンス番号＞
```

[例] リクエスト名が batreq でリクエスト ID のシーケンス番号が 72 の場合のファイル名は、batreq.o72 となります。

【標準エラー出力ファイル】

＜リクエスト名＞.e＜シーケンス番号＞

[例] リクエスト名が batreq でリクエスト ID のシーケンス番号が 72 の場合のファイル名は、batreq.e72 となります。

リクエスト名は、リクエスト投入時に qsub(1)の-N オプションで指定することができますが、-N オプションを指定しなかった場合は、リクエストのジョブスクリプトファイル名がリクエスト名となり、リクエストのジョブスクリプトを qsub(1)コマンドの標準入力から入力した場合は、リクエスト名は"STDIN"となります。

1.14 MPIリクエストの実行

NQSV は、以下の MPI に対応しています。

- NEC MPI
- OpenMPI （OpenMPI Version 4.1.x. Version 5.0.x）
- Intel MPI （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, Version2.3.6, Version 2.3.7）
- Platform MPI (Platform MPI Version 09.01.04.03)

1.14.1 NEC MPI 環境での実行

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 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。

※キューの設定は、qstat -Qf の出力（NEC MPI Process Manager）にて確認できます。

NEC MPI を使用したジョブを実行する場合は、以下のようにしてリクエストを投入してください。

- 投入オプション

qsub(1)コマンドの-T オプションに necmpi を指定します。

以下に、MPI プログラム mpi_prog について、4 論理ホストを使用し、それぞれの論理ホストの 8VE で 1 プロセスずつ実行する場合の実行例を示します。

```
#! /bin/sh
#PBS -T necmpi
#PBS -b 4
#PBS -I "bin/mpi_prog,"
#PBS --venum-lhost=8

cd ${STGDIR}

source /opt/nec/ve/mpi/1.0.1/bin/necmpivars.sh
mpirun –nnp 8 -nn 4 -ve 0-7 mpi_prog
```

1.14.2 OpenMPI 環境での実行

- 投入方法

NQSV において OpenMPI を使用したジョブを実行する場合は、以下のようにしてリクエストを投入してください。

- 投入オプション

qsub(1)コマンドの-T オプションに openmpi を指定します。

- mpirunの指定

MPI プログラムの実行には OpenMPI の mpirun コマンドを使用します。

mpirun の引数に${NQSV_MPIOPTS}または${NQSII_MPIOPTS}を指定してください。（この指定をとおして、ジョブが実行されるホストを MPI に渡すようになっています。）

【参考】

ジョブ内から使用できるネットワーク I/F が複数ある場合、-mca btl_tcp_if_include オプションで、使用したいネットワーク I/F を指定できます。

例えば、Docker コンテナ内でジョブを実行する場合、-mca btl_tcp_if_include で、オーバーレイネットワークのインタフェースを指定してください。

例えば、バッチリクエストにて、2 ジョブ（ノード）で、合計 64 個の MPI プロセスを生成して、OpenMPI のプログラムを実行する場合には、ジョブスクリプトを以下のように記述します。

```
#! /bin/bash
#PBS -T openmpi
```

```
#PBS -b 2         # The number of Batch Job

mpirun ${NQSV_MPIOPTS} -np 64 ${HOME}/mpi_prog
```

- OpenMPと組み合わせた実行

OpenMPI プログラムと OpenMP と組み合わせて実行をする場合は、リクエスト投入時に、ジョブの環境変数として、OMP_NUM_THREADS に 1 プロセス内の多重度を設定してください。

なお、ソケットスケジューリング機能が有効なキューにリクエストを投入した場合、論理ホストごとの 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。本環境変数は、qsub -v オプションで値を指定することで上書き可能です。

ジョブスクリプト内で qsub -V を指定して新しいリクエストを投入すると、1 番目のリクエストが持つ環境変数が 2 番目のリクエストに継承されます。そのため、2 番目のリクエストで使用する値を環境変数 OMP_NUM_THREADS に設定してください。

例えば、バッチリクエストにて、2 ジョブ（ノード）で各ノード上に 1 個ずつ MPI プロセスを生成し、OpenMP により各プロセスで 32 個のスレッドを生成できるようにして、OpenMPI のプログラムを実行する場合には、ジョブスクリプトを次のように記述します。

```
#! /bin/bash
#PBS -T openmpi
#PBS -b 2              # The number of Batch Job
#PBS --cpunum-lhost=32    # The number of CPUs per logical host
#PBS -v OMP_NUM_THREADS=32

mpirun ${NQSV_MPIOPTS} -np 2 ${HOME}/mpi_prog
```

【注意】

- OpenMPI では一つの実行ホスト上で複数のジョブを実行することはできません。
- 資源制限超過時やノード障害発生時などでジョブが正常終了しなかったときにマスターノードの /tmp 配下に一時ファイルが残る場合があります。削除して問題ありません。
- OpenMPI では-hostfile ${PBS_NODEFILE}を使用しないでください。NQSV のリソース管理外の CPU を使用する可能性があります。

1.14.3 Intel MPI 環境での実行

- 投入方法

NQSV において Intel MPI を使用したジョブを実行する場合は、以下のようにしてリクエストを投

入してください。

- 投入オプション

  qsub(1)コマンドの-T オプションに intmpi を指定します。

- mpirunの指定

  MPI プログラムの実行には Intel MPI の mpirun コマンドを使用します。

  mpirun の引数に${NQSV_MPIOPTS}または${NQSII_MPIOPTS}を指定してください。(この指定をとおして、ジョブが実行されるホストを MPI に渡すようになっています。)

例えば、バッチリクエストにて、2 ジョブ（ノード）で、合計 64 個の MPI プロセスを生成して、Intel MPI のプログラムを実行する場合には、ジョブスクリプトを以下のように記述します。

```
#! /bin/bash
#PBS -T intmpi
#PBS -b 2         # The number of Batch Job

mpirun ${NQSV_MPIOPTS} -np 64 ${HOME}/mpi_prog
```

また、複数のノードに均等に MPI プロセスを分散させることもできます。例えば、2 ノードで合計 64 個の MPI プロセスを 32 プロセスずつ割り当てるには、ジョブスクリプトを以下のように記述します。

```
#! /bin/bash
#PBS -T intmpi
#PBS -b 2         # The number of Batch Job

mpirun -hostfile ${PBS_NODEFILE} -np 64 -ppn 32 ${HOME}/mpi_prog
```

- OpenMPと組み合わせた実行

  Intel MPI プログラムを OpenMP と組み合わせて実行をする場合は、リクエスト投入時に、ジョブの環境変数として、OMP_NUM_THREADS に 1 プロセス内の多重度を設定してください。

  なお、ソケットスケジューリング機能が有効なキューにリクエストを投入した場合、論理ホストごとの 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。本環境変数は、qsub -v オプションで値を指定することで上書き可能です。

  ジョブスクリプト内で qsub -V を指定して新しいリクエストを投入すると、1 番目のリクエストが持つ環境変数が 2 番目のリクエストに継承されます。そのため、2 番目のリクエストで使用する値を

環境変数 OMP_NUM_THREADS に設定してください。

例えば、バッチリクエストにて、2 ジョブ（ノード）で各ノード上に 1 個ずつプロセスを生成し、OpenMP により各プロセスで 32 個のスレッドを生成できるようにして、Intel MPI のプログラムを実行する場合には、ジョブスクリプトを次のように記述します。

```
#! /bin/bash
#PBS -T intmpi
#PBS -b 2              # The number of Batch Job
#PBS --cpunum-lhost=32    # The number of CPUs per logical host
#PBS -v OMP_NUM_THREADS=32

mpirun ${NQSV_MPIOPTS} -np 2 ${HOME}/mpi_prog
```

【注意】

- IntelMPI では NQSV のソケットスケジューリング機能が有効な場合、一つの実行ホスト上で複数のジョブを実行することはできません。
- ジョブ実行中に qdel でジョブを削除した場合、ジョブの実行結果ファイル(標準エラー出力)に下記のようなメッセージが出る場合があります。動作としては特に問題はありません。
  [proxy:0:1@host1] proxy_upstream_control_cb (../../../../../src/pm/i_hydra/proxy/proxy_cb.c:69): assert (!closed) failed
  [proxy:0:1@host1]HYDI_dmx_poll_wait_for_event (../../../../../src/pm/i_hydra/libhydra/demux/hydra_demux_poll.c:80): callback returned error status
  [proxy:0:1@host1]main (../../../../../src/pm/i_hydra/proxy/proxy.c:964): error waiting for event

1.14.4 MVAPICH2 環境での実行

NQSV において MVAPICH2 を使用したジョブを実行する場合は、以下のようにしてリクエストを投入してください。

- 投入オプション

  qsub(1)コマンドの-T オプションに mvapich を指定します。

- mpiexecの指定投入

  MPI プログラムの実行には MVAPICH2 の mpiexec コマンドを使用します。

  mpiexec の引数に${NQSV_MPIOPTS}または${NQSII_MPIOPTS}を指定してください。

  （この指定をとおして、ジョブが実行されるホストを MPI に渡すようになっています。）

例えば、バッチリクエストにて、2 ジョブ（ノード）で、合計 64 個の MPI プロセスを生成して、

MVAPICH のプログラムを実行する場合には、ジョブスクリプトを以下のように記述します。

```
#! /bin/bash
#PBS -T mvapich
#PBS -b 2       # The number of Batch Job
mpiexec ${NQSV_MPIOPTS} -np 64 ${HOME}/m
```

1.14.5 Platform MPI 環境での実行

- 投入方法

NQSV において Platform MPI を使用したジョブを実行する場合は、以下のようにしてリクエストを投入してください。

- 投入オプション

qsub(1)コマンドの-T オプションに pltmpi を指定します。

- mpirunの指定

MPI プログラムの実行には Platform MPI の mpirun コマンドを使用します。mpirun の引数に${NQSV_MPIOPTS}または${NQSII_MPIOPTS}を指定してください。この指定で MPI にジョブが実行されるホストを渡しています。また、PATH や LD_LIBRARY_PATH 等の環境変数を各ジョブに渡す必要がある場合は、mpirun のオプション-envlist に指定してください。

例えば、2 ジョブ（ノード）のバッチリクエストで、合計 64 個の MPI プロセスを生成して Platform MPI のプログラムを実行する場合は、ジョブスクリプトを以下のように記述します。

```
#! /bin/bash
#PBS -T pltmpi
#PBS -b 2         # The number of Batch Job

mpirun ${NQSV_MPIOPTS} -np 64 –envlist PATH,LD_LIBRARY_PATH ${HOME}/mpi_prog
```

【注意】

- ジョブ実行中に qdel でジョブを削除した場合、ジョブの実行結果ファイル(標準エラー出力)に下記のようなメッセージが出る場合があります。動作としては特に問題はありません。

  -15: -c: line 0: unexpected EOF while looking for matching `""

  -15: -c: line 1: syntax error: unexpected end of file

  また、/tmp 配下にジョブ実行時に作成された下記のような一時ファイルが残ることがありま

す。手動で削除してください。
mpijob_user1_0、mpiafQeIuGG

1.15 リクエスト実行中の出力ファイル表示

リクエストの実行中に、qcat(1)コマンドによって、ジョブの標準出力／標準エラー出力の内容を表示することができます。qcat(1)コマンドでは、その他、ジョブスクリプトファイルの内容や、ジョブとして実行中のプロセスのリストを表示することができます。

以下に、qcat(1)コマンドのオプションについて記載します。

-e	標準エラー出力の内容を表示します。
-i （既定値）	ジョブスクリプトを表示します。
-o	標準出力の内容を表示します。
-p	ジョブ内で実行中のプロセスのリストを表示します。
-d	リクエストログを表示します。

-e/-o オプションとの組合せにより、下記のオプションで表示方法を指定することができます。

-f	ファイルの内容が増え続ける時、追加されたデータを出力します。

上記オプションとの組合せにより、下記のオプションで表示方法を指定することができます。

-n	指定した行数分を表示します。（本オプションが指定されない場合は、10行分表示されます。）
-b	ファイルの先頭から表示します。（本オプションが指定されない場合は、ファイルの最終行から表示されます。）

【例】ジョブスクリプトをファイルの先頭から 15 行表示する場合

```
$ qcat -b -n 15 123.host1
#!/bin/sh
#
#PBS -N Sample_JOB
#PBS -m b -m e
#PBS -j o
#PBS -q batch1 #default
#PBS -l cputim_prc=10:00:00
#PBS -l cputim_job=20:00:00
#PBS -l memsz_job=200mb
#PBS -l cpunum_prc=1
```

```
#PBS -l elapstim_req=600

date
make
$
```

【例】実行中のジョブ(ジョブ ID 3:123.host1)のプロセスリストを表示する場合

```
$ qcat -p 3:123.host1
 JID   PID S     SZ  TIME COMMAND
  77  2025 S     33  0:01 ksh
  77  2030 R    523  5:02 a.out
```

1.16 リクエストログ

バッチリクエストの処理状況はリクエストログとして記録することができます。リクエストログは qsub(1)コマンドの-D オプションによってログレベルが 1 以上に設定された場合に記録されます(ログレベルの既定値は 0 で、リクエストログは出力されません)。

リクエストログのログレベルとして、以下の 3 段階が指定可能です。

0　　リクエストログは出力されません(既定値)。

1　　リクエストに関するログを出力します。

2　　リクエスト、およびジョブに関するログを出力します。

リクエストログは qcat(1)コマンドの-d オプションによって参照することができます。

また、リクエストの処理完了後に qsub(1)コマンドの-d オプションで指定した出力先へステージアウトすることが可能です。

1.17 リクエストへのアタッチ

リクエストが実行ホスト上で実行中に、実行ホスト上で実行中のジョブに対して、qattach(1)コマンドにより、実行中のジョブスクリプト以外に外部から別のプログラムを実行することができます。

qattach は、実行するプログラムを-c オプションで指定し、そのプログラムを実行する対象のジョブのジョブ ID を指定して起動します。ジョブ ID の代わりにジョブ番号を省略してリクエスト ID を指定することも可能です。この場合、-c で指定したプログラムは、リクエスト内の全ジョブに対して実行されます。

qattach の-c で指定したプログラムは、実行ホスト上で実行中のジョブの一部として実行されます。したがって、qattach によって実行ホスト上で起動したプログラムが使用する CPU 時間、メモリ量は、実行中のジョブに使用量に加算されます。

また、qattach で実行するプログラムに標準入出力は、qattach コマンドの標準入出力として入力、および出力ができます。ただし、qattach により実行ホスト上でプログラムを実行する際、生成されるプロセスはターミナルを持ちません。したがって、qattach では、ターミナルを必要とするプログラム、または、機能は利用できません。

以下に、リクエストとして実行中のプログラムに対して、qattach によりデバッグする場合の実行例を示します。

```
$ qcat -p 0:16303
  SID   PID STAT    VSZ   RSS     TIME COMMAND
10073 10073 Ss    22628  2400 00:00:00 nqs_shpd
10073 10074 S    119772  2276 00:00:00 bash
10073 10113 R+     4164   340 00:00:07 a.out

$ qattach -c "gdb --quiet" 0:16303
(gdb) attach 10113
Attaching to process 10113
. . .
```

qattach によるリクエストへのアタッチ操作は、１つのジョブに対して同時に実行できるのは１つだけです。

第2章　　パラメトリックリクエストの操作

2.1　パラメトリックリクエストの投入

パラメトリックリクエストは、入力となるパラメータや、入力ファイルを変えながら、同一のジョブスクリプトをサブリクエストとして複数回実行するリクエストです。

2.1.1　投入方法

パラメトリックリクエストで実行するジョブスクリプトは、バッチリクエストと同様のスクリプトの記述、投入オプションの指定が行えます。

パラメトリックリクエストは、バッチリクエストと同様、qsub コマンドを使用し、投入先のキューとしてバッチキューを指定して投入します。

このとき、-t オプションで、パラメータとして使用するサブリクエスト番号の組み合わせを指定します。

以下にサブリクエスト番号として 1 から 5 の連続した番号を指定してパラメトリックリクエストを投入する例を示します。

```
$ qsub -q batch1 -t 1-5 script1
Request 65[].bsv1 submitted to queue: batch1.
```

-t オプションによるサブリクエスト番号の指定においては、以下の指定ができます。

カンマで区切って複数の番号を指定（例： -t 1,3,5）

マイナス記号で連続した番号を指定（例： -t 1-5、これは、-t 1,2,3,4,5と同じです）

マイナス記号利用時に、後に:*n*を指定することで*n*番おきの番号を指定

（例： -t 1-5:2、これは、 -t 1,3,5と同じです）

パラメトリックリクエストの投入が成功すると、バッチリクエストの投入と同様、標準出力にリクエスト ID と投入先のキュー名を含むメッセージが表示されます。

パラメトリックリクエストの投入の場合、表示されるリクエスト ID は、パラメトリックリクエストの ID として、通常のバッチリクエストのシーケンス番号の後に[]が付いたリクエスト ID が表示されます。

上記の例に示した投入により、NQSV システム上で、1 から 5 のサブリクエスト番号を持つ 5 つのサブリクエストが生成され、サブリクエスト単位で、通常のバッチリクエストと同様にスケジューリング、ジョブ実行を行います。

パラメトリックリクエストの投入時に指定した、ジョブ数、資源制限等の指定は、各サブリクエストに対する指定値として扱います。

2.1.2　パラメータ、入力ファイル

パラメトリックリクエストの投入時に-t オプションで指定したサブリクエスト番号は、各サブリクエストの実行時の入力パラメータとして、PBS_SUBREQNO 環境変数により参照できます。

また、qsub で-I, -O オプションで指定するファイルステージングの転送ファイルの設定において、%t により転送ファイルのパス名にサブリクエスト番号を埋め込むことができます。
サブリクエストごとにそれぞれ、/DATA/IN_DATA.001, /DATA/IN_DATA.002, /DATA/IN_DATA.003, ･･･の入力ファイルをステージインする場合、以下のように指定を行います。

```
$ qsub -q batch1 -t 1-5 -I ¥"/DATA/IN-DATA.%03t,filein¥"  script1
Request 65[].bsv1 submitted to queue: batch1.
```

上記のように指定することで、各サブリクエストの実行時に、対応する入力ファイルが実行スクリプト内において$STGDIR/filein のパス名で参照できます。

2.2　パラメトリックリクエストの状態確認

投入したパラメトリックリクエストの状態は、qstat(1)コマンドで確認することができます。

```
$ qstat
RequestID       ReqName  UserName Queue    Pri STT S   Memory     CPU   Elapse R H M Jobs
--------------- -------- -------- -------- ---- --- - -------- -------- -------- - - - ----
71.bsv1         STDIN    user1    batch1     0 RUN -    1.93M      0.23      46 Y Y Y    1
72[].bsv1       STDIN    user1    batch1     0 QUE -       -       -       - Y Y Y    1
```

qstat コマンドでのリクエスト参照において、オプションなしの場合は、通常のリクエストと、パラメトリックリクエストが存在する場合は、パラメトリックリクエスト自身の情報が表示されます。リクエスト ID を指定して表示する場合、パラメトリックリクエストに対しては、シーケンス番号に[]を付加したパラメトリックリクエストのリクエスト ID を指定してください。

```
$ qstat 72[].bsv1
RequestID       ReqName  UserName Queue    Pri STT S   Memory     CPU     Elapse R H M Jobs
--------------- -------- -------- -------- ---- --- - -------- -------- -------- - - - ----
72[].bsv1       STDIN    user1    batch1     0 QUE -       -       -       - Y Y Y    1
```

また、パラメトリックリクエストとして実際に実行が行われるサブリクエストの状態を確認する場合は、qstat コマンドに-s オプションを指定してください。-s オプションを指定することで、パラ

メトリックリクエストに対しては、パラメトリックリクエスト自身の情報の代わりに、バッチサーバ内に存在するサブリクエストの情報を表示します。

```
$ qstat -s
RequestID      ReqName  UserName Queue    Pri STT S   Memory     CPU   Elapse R H M Jobs
-------------- -------- -------- -------- ---- --- - -------- -------- -------- - - - ----
71.bsv1        STDIN    user1    batch1     0 RUN -    1.93M    0.23      46 Y Y Y    1
72[1].bsv1     STDIN    user1    batch1     0 RUN -    3.87M    0.00      29 Y Y Y    1
72[2].bsv1     STDIN    user1    batch1     0 QUE -    0.00B    0.00       0 Y Y Y    1
72[3].bsv1     STDIN    user1    batch1     0 QUE -    0.00B    0.00       0 Y Y Y    1
72[4].bsv1     STDIN    user1    batch1     0 QUE -    0.00B    0.00       0 Y Y Y    1
72[5].bsv1     STDIN    user1    batch1     0 QUE -    0.00B    0.00       0 Y Y Y    1
```

サブリクエストのリクエスト ID は、パラメトリックリクエストのリクエスト ID の[]の中にサブリクエスト番号を挿入した形式となります。

パラメトリックリクエストの実行において、バッチサーバ上で生成するサブリクエストの数には上限が設定されています（既定値は 100）。

そのため、qstat -s では、qsub コマンドの-t オプションで指定したサブリクエストが全て表示されるとは限りません。

また、qsub コマンドで-h オプションや -a オプションで実行開始を保留した場合は、投入時点ではバッチサーバ上でサブリクエストの生成が行われないため、qstat -s では投入したパラメトリックリクエストの情報が表示されないことがあります。

その他、qstat コマンドの -R オプションによって、パラメトリックリクエストのサマリが表示されます。

```
$ qstat -R
RequestID       ReqName  UserName Queue    Pri STT R H M Jobs  TOTAL ACTIVE   DONE
--------------- -------- -------- -------- ---- --- - - - ---- ------ ------ ------
72[].bsv1       STDIN    user1    batch1     0 RUN Y Y Y    1      5      2      3
```

パラメトリックリクエストのサマリ表示では、TOTAL に、全サブリクエスト数、ACTIVE に、バッチサーバ上で保持しているサブリクエスト数、DONE に、終了したサブリクエスト数(削除された数を含みます)を表示します。

【注意】

- /etc/opt/nec/nqsv/nqsd.conf に enable_memory_cgroup on を設定している場合、Memory に表示する値にはファイルキャッシュの使用量を含みます。

2.3　パラメトリックリクエストの操作

パラメトリックリクエストに対する操作には、全サブリクエストを含むパラメトリックリクエスト全体に対する操作と、サブリクエスト単位での操作があります。

パラメトリックリクエスト全体に対する操作の際には、パラメトリックリクエスト ID、サブリクエスト単位で操作する場合は、サブリクエスト ID で、対象のリクエストを指定します。

パラメトリックリクエスト全体を削除する場合、

```
$ qdel 72[].bsv1
Request 72[].bsv1 was deleted.
```

サブリクエスト 72[2]のみを削除する場合は、

```
$ qdel 72[2].bsv1
Request 72[2].bsv1 was deleted.
```

のように指定します。

パラメトリックリクエストに対する各操作でのリクエストの動作は以下のようになります。

表 2-1 パラメトリックリクエストの操作

操作	パラメトリックリクエスト全体	サブリクエスト単位
削除 （qdel）	全サブリクエストを削除し、パラメトリックリクエスト自体も削除	指定されたサブリクエストのみを削除
属性変更 （qalter）	全サブリクエスト共通で属性を変更 ※変更結果は、変更後に実行されるサブリクエストに対して有効	不可
ホールド （qhold）	全サブリクエストに対してホールド操作を実行 ※各サブリクエストのホールド結果の保証はなし	指定したサブリクエストのみに対してホールドを実行
ホールド解除 （qrls）	全サブリクエストに対してホールド解除の操作を実行	指定したサブリクエストのみに対してホールド解除を実行
サスペンド （qsig -s STOP）	全サブリクエストに対してサスペンドの操作を実行	指定したサブリクエストのみに対してサスペンドを実行

操作	パラメトリックリクエスト全体	サブリクエスト単位
レジューム （qsig -s CONT）	全サブリクエストに対してレジュームの操作を実行	指定したサブリクエストのみに対してレジュームを実行
リラン （qrerun）	全サブリクエストを削除した後、パラメトリックリクエストとして再度キューへの投入を実行	指定されたサブリクエストに対して再登録を実行
キューの移動 （qmove）	パラメトリックリクエスト内のすべてのサブリクエストが実行開始前の場合のみ実行可能で、パラメトリックリクエスト全体の投入キューを変更する ※サブリクエストがジョブを持つ場合は、forceオプションが必要	不可
アタッチ （qattach）	不可	指定されたサブリクエスト内のジョブに対してアタッチ操作を実行

【注意】パラメトリックリクエスト全体をホールドする場合、ホールド操作はサブリクエスト単位で実行するため、サブリクエストの状態によっては、HELD 状態とならない場合があります。したがって、パラメトリックリクエスト全体が HELD 状態でも、HELD 状態になっていないサブリクエストが存在することがあります。

2.4　パラメトリックリクエストの開始・終了と出力ファイル

2.4.1　メール送信機能

パラメトリックリクエストにおいても、qsub コマンドの-m オプションで、リクエストの開始、終了時にメールを送信する機能が利用可能です。パラメトリックリクエストの場合、パラメトリックリクエスト全体の開始、終了、および、サブリクエスト単位の開始、終了時にメールを送信するよう指定することができます。

表 2-2 パラメトリックリクエストのメール送信

メールオプション	メールの送信タイミング
b	パラメトリックリクエストが実行開始した時点で送信
e	パラメトリックリクエストが実行終了した時点で送信
a	サブリクエストが異常終了した時点で送信
s	各サブリクエストが実行開始した時点で送信

メールオプション	メールの送信タイミング
d	各サブリクエストが実行終了した時点で送信

2.4.2　終了待ち合わせ

パラメトリックリクエストの場合も、qwait(1)コマンドによって終了の待ち合わせを行うことができます。パラメトリックリクエストの場合は、qwait にパラメトリックリクエストのリクエスト ID を指定することで、パラメトリックリクエスト全体の終了の待ち合わせを行い、サブリクエストの ID を指定することで、サブリクエスト単位での終了の待ち合わせを行います。

【例】パラメトリックリクエスト全体の待ち合わせ

```
$ qwait 72[].bsv1
exited (3 subrequests)
```

【例】サブリクエスト単位での待ち合わせ

```
$ qwait 72[2].bsv1
exited 0
```

2.4.3　出力ファイル

パラメトリックリクエストの実行は、サブリクエスト単位で行うため、リクエストの出力もサブリクエスト単位に出力ファイルが作成されます。サブリクエストごとに作成する標準出力ファイルのファイル名は次のように命名されます。

```
＜リクエスト名＞.o＜シーケンス番号＞.＜サブリクエスト番号＞
```

[例] リクエスト名が preq でリクエスト ID のシーケンス番号が 72 の場合、サブリクエスト番号が 2 のサブリクエストの標準出力ファイル名は、

preq.o72.2

となります。

ただし、qsub コマンドの-o オプションで標準出力ファイルを指定した場合のファイル名は、次のように命名されます。

```
＜指定ファイル名＞.＜サブリクエスト番号＞
```

また、サブリクエストごとに作成する標準エラー出力ファイルのファイル名は次のように命名されます。

```
＜リクエスト名＞.e＜シーケンス番号＞.＜サブリクエスト番号＞
```

[例] リクエスト名が preq でリクエスト ID のシーケンス番号が 72 の場合、サブリクエスト番号が 2 のサブリクエストの標準エラー出力ファイル名は、

preq.e72.2

となります。

ただし、qsub コマンドの-e オプションで標準エラー出力ファイルを指定した場合のファイル名は、以下のようになります。

```
＜指定ファイル名＞.＜サブリクエスト番号＞
```

パラメトリックリクエストの場合、qsub コマンドの-o, -e オプション、および、-O オプションによるステージアウトファイルの指定時に、%t により、ファイル名中にサブリクエスト番号を展開することができます。

また、%r によるリクエスト ID の展開において、パラメトリックリクエストの場合は、各サブリクエストにおいて、以下の形式で展開されます。

```
＜シーケンス番号＞[＜サブリクエスト番号＞].＜BSVホスト名＞
```

ただし、外部ステージングを使用する場合、-O によるステージアウトファイルの指定において上記%r の展開を使用すると、転送用プログラムによっては、[や] の展開がファイル名の一部として認識されない等の問題が発生することがあります。その場合は、[と]を使用せずに、_を使用してリクエスト ID を展開する、%P を使用してください。

%P は、各サブリクエストのリクエスト ID を以下の形式に展開します。

```
＜シーケンス番号＞_＜サブリクエスト番号＞.＜BSVホスト名＞
```

第3章　　会話リクエストの操作

会話リクエストは、会話的に実行ホストを利用するリクエストです。

3.1　会話リクエストの投入

会話リクエストを投入するには、qlogin コマンドを利用する方法と、リモート実行型会話機能（qrsh コマンド）を利用する方法の 2 種類があります。本節では、qlogin コマンドを利用したセッション接続タイプの会話リクエストの投入方法について、説明します。

リモート実行型会話機能（qrsh コマンド）については、［管理編］リモート実行型会話機能 を参照してください。

3.1.1　qlogin による投入方法

セッション接続タイプの会話リクエストの投入は、qlogin(1)コマンドで行ってください。qlogin(1)コマンドでは、コマンド行にオプションを指定できます。指定できるオプションは、基本的に qsub(1)コマンドと共通です。qlogin(1)固有のオプション等については、3.1.2qlogin の投入オプションで説明します。

以下は、qlogin コマンドによる会話リクエストの投入例です。

```
$ qlogin  -q ique --cpunum-lhost=4 -l elapstim_req=1800
Request 111.bsv1 submitted to queue: ique.     ←①
Waiting for 111.bsv1 to start.      ←②
-sh-4.1$          ←③
-sh-4.1$ hostname
ehost001
```

上記例は、ique という会話キューに、資源制限を指定して会話リクエストを投入しています。

① qloginコマンドの次の行に示されたメッセージは、NQSV が投入された会話リクエストを受理したことを示すメッセージで、会話リクエストにNQSVが付けたリクエストIDと、投入した会話キューを表示しています。

② ″Waiting for …″ のメッセージ②を表示し、実行ホストとのセッションの接続待ちを行います。実行ホストとqloginを実行したユーザ端末（クライアントホスト）のセッション接続を行い、双方の標準入出力・エラー出力の接続が完了すると、会話リクエストが開始となります。

③ 実行ホスト上のシェルのプロンプトが表示され、実行ホストのインタラクティブな利

用が可能となります。

qlogin で会話リクエストを投入すると、実行ホスト上のジョブサーバから qlogin コマンドに対してセッションの接続を行います。このとき、接続先のホストとして、qlogin を実行したホストのホスト名を使用します。

実行ホスト上で、qlogin を実行したホストのホスト名が参照できない環境の場合は、qlogin の実行時に、環境変数、NQSV_INTERACTIVE_IP または NQSII_INTERACTIVE_IP を使用して、ジョブサーバから qlogin コマンドへのセッションを接続に使用する IP アドレスを指定することで、ホスト名を使用せずにセッションの接続を行うことができます。

NQSV_INTERACTIVE_IP と NQSII_INTERACTIVE_IP のどちらに値を指定しても効果は同じです。両方の環境変数に値が設定された場合は NQSV_INTERACTIVE_IP の値が採用されます。

NQSV_INTERACTIVE_IP 環境変数の設定例は以下の通りです。

```
$ export NQSV_INTERACTIVE_IP=192.168.0.1
$ qlogin  -q ique --cpunum-lhost=4 -l elapstim_req=1800
Request 111.bsv1 submitted to queue: ique.
  :
```

3.1.2　qlogin の投入オプション

(1)　キュー名

```
-q queue
```

queue に、会話キューを指定します。

(2)　ジョブ数

```
-b N
```

-T オプションで指定するジョブの種類が、distrib(分散ジョブ)の場合は、N=1 のみ指定可能です。N に 2 以上の数字を指定しても、実行するジョブの数は 1 つのみです。

(3)　実行ホスト割当ての待ち合わせ

```
-W
```

投入する会話キューの実行ホスト割当ての待ち合わせ設定が、"manual"に設定されている場合、

qlogin に-W を指定することにより、実行ホストの割り当てを待ち合わせるようになります。-W を指定しない場合は、即時に実行開始可能な実行ホストの割当てができない場合は、投入をキャンセルします。
ただし、投入する会話キューの、実行ホスト割当ての待ち合わせ設定が、"manual"以外に設定されている場合は、-W の指定は無効です。
（会話キューの待合わせ設定の詳細は、［管理編］会話キューの設定（待ち合わせ指定）参照）

(4)　アイドルタイマー

```
-l idle_timer=max_limit
```

qlogin の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 max_limit 分経過した場合に、自動的にセッションを切断し、リクエストを削除します。
（会話キューのアイドルタイマーの詳細は、[管理編] 会話キューの設定（アイドルタイマー） 参照）

(5)　その他

会話リクエストには、以下の特徴があります。

- ジョブの入出力がqloginの標準入力／標準出力／標準エラー出力と接続されているため、ステージングを行いません。
- 会話的に利用するため、バッチ実行のようなプライオリティ、リラン、パラメトリック実行等の機能はありません。

そのため、qlogin では以下の指定をする機能はありません。

- スクリプトファイル中の埋め込みオプション指定
- リクエストログ指定
- 標準出力／標準エラー出力ファイル指定
- ジョブマイグレーション関連の指定
- ホールド可否、リラン可否の指定
- ステージング関連の指定
- リクエストプライオリティ

3.2　会話リクエストの状態確認

qstat(1)コマンドにより会話リクエストの状態を表示します。

【リクエスト ID 指定時】

```
$ qstat 82.host
RequestID      ReqName  UserName Queue    Pri STT S   Memory     CPU   Elapse R H M Jobs
---------------- -------- -------- -------- ---- --- - -------- -------- -------- - - - ----
82.host.example QLOGIN   user1    ique       0 RUN -    1.93M    0.23     128 N N N    1
```

【リクエスト ID 指定なし】

```
$ qstat
RequestID      ReqName  UserName Queue    Pri STT S   Memory     CPU   Elapse R H M Jobs
---------------- -------- -------- -------- ---- --- - -------- -------- -------- - - - ----
73.host.example STDIN    user1    batch1    20 QUE -    0.00B    0.00       0 Y Y Y    1
74.host.example STDIN    user1    batch2    20 QUE -    0.00B    0.00       0 N N N    2
82.host.example QLOGIN   user1    ique       0 RUN -    1.93M    0.23     128 N N N    1
```

【詳細情報の確認】

```
$ qstat -f 82.host
Request ID: 82.host.example.com
    Request Name = QLOGIN
    User  Name = user1
    User  ID   = 100
    Group ID   = 100
    Current State          = Running
    Previous State         = Pre-running
    State Transition Time   = Sat Feb  8 15:26:35 2017
    State Transition Reason = PRERUN_SUCCESS
    Queue = ique@ host.example.com (Interactive Queue)
    Job Topology = Distribute Job
    Request Privilege = Level 0
    Request Priority  = 0
    Rerunable    = No
    Holdable     = No
    Migratable   = No
    Suspend Type = (none)
```

```
  Account Code = (none)
  Shell = (none)
  Restrict shell = (none)
  Mail Address = user1@client.example.com
  Mail Option  = (none)
  Job Condition:
     Job NO: 0 ""
  Number of Jobs = 1
  Created Request Time = Sat Feb  8 15:26:22 2017
  Entered Queue Time   = Sat Feb  8 15:26:22 2017
  Planned Start Time   = Sat Feb  8 15:26:35 2017
  Execute Request Time = (none)
  Started Request Time = Sat Feb  8 15:26:35 2017
  Ended Request Time   = (none)
  Requested Start Time = (none)
  UMASK = 022
  Interactive Host = client.example.com
  Interactive Port = 54467
  Idle Timer = 0
  qattach command = Enable
  Attach = No
  Cluster Type Select = NONE
  UserPP Script = (none)
  Exclusive = (none)
  HCA Number = (none)
  Accept Sigterm = No
Execution Hosts(JSVNO):
  ehost100(100)
Resources Information:
  Memory    = 3.011719MB
  CPU Time  = 0.010000S
  Accumulated CPU Time = 0.010000S
  Elapse    = 766S
  Remaining Elapse = 2834S
  Virtual Memory = 355.789062MB
Logical Host Resources:
  VE Node Number       = Max:       1 Warn:      ---
  CPU Number           = Max:       1 Warn:      ---
  GPU Number           = Max:       0 Warn:      ---
  CPU Time             = Max:       0 Warn:      ---
```

```
  Memory Size          = Max: UNLIMITED Warn: UNLIMITED
  Virtual Memory Size   = Max: UNLIMITED Warn: UNLIMITED
  VE CPU Time          = Max: UNLIMITED Warn: UNLIMITED
  VE Memory Size       = Max: UNLIMITED Warn: UNLIMITED
VE Node Resources:
  VE CPU Time          = Max: UNLIMITED Warn: UNLIMITED
  VE Memory Size       = Max: UNLIMITED Warn: UNLIMITED
Resources Limits:
  (Per-Req) Elapse Time Limit      = Max:    3600S Warn:    3600S
  (Per-Job) CPU Time              = Max: UNLIMITED Warn: UNLIMITED
  (Per-Job) CPU Number            = Max:        1 Warn:      ---
  (Per-Job) Memory Size           = Max: UNLIMITED Warn: UNLIMITED
  (Per-Job) Virtual Memory Size    = Max: UNLIMITED Warn: UNLIMITED
  (Per-Job) GPU Number             = Max:        0 Warn:      ---
  (Per-Prc) CPU Time              = Max: UNLIMITED Warn: UNLIMITED
  (Per-Prc) Open File Number       = Max: UNLIMITED Warn:      ---
  (Per-Prc) Virtual Memory Size    = Max: UNLIMITED Warn: UNLIMITED
  (Per-Prc) Data Segment Size      = Max: UNLIMITED Warn: UNLIMITED
  (Per-Prc) Stack Segment Size     = Max: UNLIMITED Warn: UNLIMITED
  (Per-Prc) Core File Size         = Max: UNLIMITED Warn: UNLIMITED
  (Per-Prc) Permanent File Size    = Max: UNLIMITED Warn: UNLIMITED
  (Per-Prc) VE CPU Time           = Max: UNLIMITED Warn: UNLIMITED
  (Per-Prc) VE Memory Size         = Max: UNLIMITED Warn: UNLIMITED
Kernel Parameter:
  Resource Sharing Group    = 0
  Nice Value              = 0
User Attributes:
  (none)
```

各項目の内容については、qstat(1)を参照してください。

【注意】

・ /etc/opt/nec/nqsv/nqsd.conf に enable_memory_cgroup on を設定している場合、基本情報の Memory 及び詳細情報の Resources Information:の Memory に表示する値にはファイルキャッシュの使用量を含みます。

3.3　会話リクエストの属性変更

会話リクエストの属性の変更も、バッチリクエストと同様に、qalter(1)コマンドで行うことができます。ただし、会話リクエストに対して指定できない属性の変更はできません。

3.4　会話リクエストの操作

会話リクエストに対しても、バッチリクエストと同様、以下の操作が可能です。

表 3-1 会話リクエストの操作

操作	コマンド
削除	qdel
属性変更	qalter
サスペンド	qsig -s STOP
レジューム	qsig -s CONT
アタッチ	qattach

会話リクエストのキューの移動、ホールド、リランはできません。

バッチキューに投入されたリクエストを会話キューに移動することもできません。

【注意】OpenMPI 環境で実行中の会話リクエストに対しサスペンド・リジュームを行った場合、実行中の MPI プロセスが強制停止する場合があります。

3.5　会話リクエストの開始・終了と出力ファイル

会話リクエストに対しても、バッチリクエストと同様に、qlogin(1)によるリクエスト投入時に指定により、リクエストの開始、終了時のメール送信、および、qwait（１）コマンドによるリクエスト終了の待ち合わせをおこなうことが可能です。

ただし、会話リクエストの出力は qlogin の出力として表示されるため、出力ファイルを生成する機能はありません。

3.6　会話リクエストでのMPIプログラムの実行

3.6.1　NEC MPI 環境での実行

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 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。

※キューの設定は、qstat -Qf の出力（NEC MPI Process Manager）にて確認できます。

会話リクエスト上で NEC MPI を使用したプログラムを実行する場合は、以下のようにしてリクエストの投入、および、mpirun の実行をしてください。

- 投入オプション

qlogin(1)コマンドの-T オプションに necmpi を指定します。

以下に、MPI プログラム mpi_prog を 2 論理ホストで使用し、それぞれの論理ホストの 4VE で 1 プロセスずつ実行する場合の実行例を示します。

```
client $ qlogin -q ique -T necmpi -b 4 --venum-lhost=8
Request 28.host1 submitted to queue: ique.
Waiting for 28.host1 to start.
ehost $ source /opt/nec/ve/mpi/1.0.1/bin/necmpivars.sh
ehost $ mpirun -nnp 4 -nn 2 -ve 0-3 ./mpi_prog
```

【注意】上記の例では、実行ホスト上のユーザのホームディレクトリに mpi_prog が配置されていることを想定しています。

3.6.2　OpenMPI 環境での実行

会話リクエスト上で OpenMPI を使用したプログラムを実行する場合は、以下のようにしてリクエストの投入、および、mpirun の実行をしてください。

- 投入オプション

qlogin(1)コマンドの-T オプションに openmpi を指定します。

- mpirunの指定

MPI プログラムの実行には OpenMPI の mpirun コマンドを使用します。

mpirun の引数に${NQSV_MPIOPTS}または${NQSII_MPIOPTS}を指定してください。（この指定をとおして、ジョブが実行されるホストを MPI に渡すようになっています。）

【参考】

ジョブ内から使用できるネットワーク I/F が複数ある場合、-mca btl_tcp_if_include オプションで、使用したいネットワーク I/F を指定できます。

例えば、Docker コンテナ内でジョブを実行する場合、-mca btl_tcp_if_include で、オーバーレイネットワークのインタフェースを指定してください。

例えば、2 ジョブ（ノード）で、合計 64 個の MPI プロセスを生成して、OpenMPI のプログラム mpi_prog を実行する場合には、以下のように実行します。

```
client $ qlogin -q iq1 -T openmpi -b 2
Request 173.bsv submitted to queue: iq1.
Waiting for 173.bsv to start.
-bash-$ mpirun ${NQSV_MPIOPTS} -np 64 ${HOME}/mpi_prog
```

【注意】

- OpenMPI では一つの実行ホスト上で複数のジョブを実行することはできません。
- 資源制限超過時やノード障害発生時などでジョブが正常終了しなかったときにマスターノードの /tmp 配下に一時ファイルが残る場合があります。削除して問題ありません。
- OpenMPI では-hostfile ${PBS_NODEFILE}を使用しないでください。NQSV のリソース管理外の CPU を使用する可能性があります。

3.6.3　Intel MPI 環境での実行

会話リクエスト上で Intel MPI を使用したプログラムを実行する場合は、以下のようにしてリクエストの投入、および、mpirun の実行をしてください。

- 投入オプション

  qlogin(1)コマンドの-T オプションに intmpi を指定します。

- mpirunの指定

  MPI プログラムの実行には Intel MPI の mpirun コマンドを使用します。

  mpirun の引数に${NQSV_MPIOPTS}または${NQSII_MPIOPTS}を指定してください。（この指定をとおして、ジョブが実行されるホストを MPI に渡すようになっています。）

例えば、2 ジョブ（ノード）で、合計 64 個の MPI プロセスを生成して、Intel MPI のプログラム mpi_prog を実行する場合には、以下のように実行します。

```
client $ qlogin -q iq2 -T intmpi -b 2
Request 174.bsv submitted to queue: iq2.
Waiting for 174.bsv to start.
-bash-$ mpirun ${NQSV_MPIOPTS} -np 64 ${HOME}/mpi_prog
```

【注意】

- ジョブ実行中に qdel でジョブを削除した場合、ジョブの実行結果ファイル(標準エラー出力)に下記のようなメッセージが出る場合があります。動作としては特に問題はありません。

```
[proxy:0:1@host1] proxy_upstream_control_cb (../../../../../src/pm/i_hydra/proxy/proxy_cb.c:69):
assert (!closed) failed
[proxy:0:1@host1]HYDI_dmx_poll_wait_for_event
(../../../../../src/pm/i_hydra/libhydra/demux/hydra_demux_poll.c:80): callback returned error
status
[proxy:0:1@host1]main (../../../../../src/pm/i_hydra/proxy/proxy.c:964): error waiting for event
```

3.6.4　MVAPICH2 環境での実行

会話リクエスト上で MVAPICH2 を使用したジョブを実行する場合は、以下のようにしてリクエストを投入してください。

- 投入オプション

  qlogin(1)コマンドの-T オプションに mvapich を指定します。

- mpiexecの指定投入

  MPI プログラムの実行には MVAPICH2 の mpiexec コマンドを使用します。

  mpiexec の引数に${NQSV_MPIOPTS}または${NQSII_MPIOPTS}を指定してください。

  （この指定をとおして、ジョブが実行されるホストを MPI に渡すようになっています。）

例えば、2 ジョブ（ノード）で、合計 64 個の MPI プロセスを生成して、MVAPICH のプログラムを実行する場合には、ジョブスクリプトを以下のように記述します。

```
client $ qlogin -q iq3 -T mvapich -b 2
Request 175.bsv submitted to queue: iq3.
Waiting for 175.bsv to start.
-bash-$ mpiexec ${NQSV_MPIOPTS} -np 64 ${HOME}/mpi_prog
```

3.6.5　Platform MPI 環境での実行

会話リクエスト上で PlatformMPI を使用したジョブを実行する場合は、以下のようにしてリクエストを投入してください。

- 投入オプション

  qlogin(1)コマンドの-T オプションに pltmpi を指定します。

- mpirunの指定

  MPI プログラムの実行には Platform MPI の mpirun コマンドを使用します。

  mpirun の引数に${NQSV_MPIOPTS}または${NQSII_MPIOPTS}を指定してください。（この指定をとおして、ジョブが実行されるホストを MPI に渡すようになっています。）

例えば、2 ジョブ（ノード）で、合計 64 個の MPI プロセスを生成して、Platform MPI のプログラム mpi_prog を実行する場合には、以下のように実行します。

```
client $ qlogin -q iq4 -T pltmpi -b 2
Request 174.bsv submitted to queue: iq4.
Waiting for 174.bsv to start.
-bash-$ mpirun ${NQSV_MPIOPTS} -np 64 ${HOME}/mpi_prog
```

【注意】

・　ジョブ実行中に qdel でジョブを削除した場合、ジョブの実行結果ファイル(標準エラー出力)に下記のようなメッセージが出る場合があります。動作としては特に問題はありません。

-15: -c: line 0: unexpected EOF while looking for matching `"'

-15: -c: line 1: syntax error: unexpected end of file

また、/tmp 配下にジョブ実行時に作成された下記のような一時ファイルが残ることがあります。手動で削除してください。

mpijob_user1_0、mpiafQeIuGG

第4章　　ジョブの操作

4.1　ジョブの状態確認

ジョブは、実行ホスト上でのリクエストの実行単位で、リクエストに対して実行ホストが割り当てられてステージングを行う際に、実行ホスト上に生成されます。

4.1.1　基本情報の確認

ジョブの情報を確認するには、qstat(1)コマンドの -J オプションを使用します。表示するジョブの指定は、ジョブ ID（リクエスト ID＋ジョブ番号）で行います。

以下に表示例を示します。

【ジョブ ID 指定時】

```
$ qstat -J 0:72.host.example.com
 JNO  RequestID       EJID   Memory     CPU    JSVNO ExecutionHost   UserName Exit
---- --------------- ----- -------- -------- ----- --------------- -------- ----
   0 72.host.example 11366    1.93M    0.23      10 exechost.exampl user1      -
```

【ジョブ ID 指定なし】

```
$ qstat -J
JNO   RequestID       EJID   Memory     CPU    JSVNO ExecutionHost   UserName Exit
---- --------------- ----- -------- -------- ----- --------------- -------- ----
   0 72.host.example 11366    1.93M     0.23     10 exechost.exampl  user1      -
   1 72.host.example 11366    2.23M     3.15     11 exechost.exampl  user1      -
   2 72.host.example 11366    1.01M    2.45    12 exechost.exampl user1      -
```

SX-Aurora TSUBASA システムの VH で実行されているジョブの VE に関する情報は、qstat -J コマンドに-e オプションをつけることで参照できます。ただし、-e オプション指定時は、VH 上のジョブのみを表示します。なお、以降の文中では、VE 上の CPU を VECPU、VE 上のメモリを VE メモリと表記します。

以下に表示例を示します。

```
$ qstat -Je
 JNO RequestID        EJID VEMemory    VECPU JSVNO VectorIsland    UserName Exit
---- --------------- ----- -------- -------- ----- --------------- -------- ----
   0  72.host.example 11366   1.00GB     2.00     10 exechost.exampl user1      -
```

【注意】

・　/etc/opt/nec/nqsv/nqsd.conf に enable_memory_cgroup on を設定している場合、Memory に表示する値にはファイルキャッシュの使用量を含みます。

4.1.2　詳細情報の確認

各ジョブについての詳細情報を表示するには、qstat(1)コマンドに -f オプションをつけて実行してください。

```
$ qstat -J -f 0:72.host.example.com
  Request ID : 72.host.example.com
    Batch Job Number = 0
    Execution Job ID = 11366
    User Name = user1
    User ID   = 111
    Group ID  = 100
    Job Server Number = 10
    Job Server Name   = JobServer0033
    Execution Host = exechost.example.com
    Exit Code = (none)
  Resources Information:
    Memory    = 1.932587MB
    CPU Time  = 0.225019S
    Accumulated CPU Time = 0.225019S
    Remaining CPU Time = UNLIMITED
```

VH で実行されているジョブの VE に関する詳細情報は、qstat -Je コマンドに-f オプションをつけることで参照できます。

以下に表示例を示します。

```
$ qstat -J -f 0:72.host.example.com
  Request ID : 72.host.example.com
    Batch Job Number = 0
    Execution Job ID = 11366
    User Name = user1
    User ID   = 111
    Group ID  = 100
    Job Server Number = 10
```

```
    Job Server Name   = JobServer0033
    Vector Island = exechost.example.com
    VE Node = 0-3,5
    Exit Code = (none)
  Resources Information:
    VEMemory    = 1.000000GB
    VECPU Time  = 2.000000S
```

qstat(1) -f で Scheduler Message を表示します。このメッセージにより、ジョブが実行されない理由を確認することが可能です。

以下に表示例を示します。

```
$ qstat –f [Request-ID]
Scheduler Message:
    Not enough resources for 1 of 2jobs in a request on available 1 hosts.
    1 of 2 hosts cannot be used due to the scheduling condition mismatching.
       Reason:
          - JSV LINKDOWN
```

【注意】

・ /etc/opt/nec/nqsv/nqsd.conf に enable_memory_cgroup on を設定している場合、Resources Information:の Memory に表示する値にはファイルキャッシュの使用量を含みます。

4.1.3　ハイブリッドリクエストのジョブ情報の確認

ハイブリッドリクエストの情報を参照する際は特別な操作が必要です。

詳細については 16.3 ハイブリッドリクエスト情報の参照を参照してください。

4.1.4　出力情報のカスタマイズ

時間表示

qstat(1)コマンドの実行結果のうち、時間データはデフォルトでは秒単位での表示ですが、-d オプションを指定することによって、d+hh:mm:ss（日+時:分:秒）の形式で表示することが可能です。本オプションはジョブ情報の表示において下記の項目で有効となります。

- CPU時間使用量（CPU）
- 積算CPU時間使用量（-c 1 オプション指定時、ACCPU）
- 残CPU時間使用量（-mオプション指定時、RCPU）

下記は、-d オプションを指定した場合のコマンド例です。

```
$ qstat -J -d
JNO RequestID         EJID   Memory         CPU    JSVNO  ExecutionHost  UserName Exit
---- --------------- ----- -------- ------------ ----- --------------- -------- ----
   0 1249.host1.exam  19710    1.91M   1+05:21:10   1020 host1.example.c user1      -
   1 1249.host1.exam 19710     1.90M   1+05:01:10   1030 host2.example.c user1      -
```

（qstat(1)コマンド-d オプションの詳細については、6.4.時間単位項目の表示 を参照）

ロング表示

qstat(1)コマンドの基本情報表示（-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、実行ホスト名などが途中で切られてしまう場合があります。この場合、-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 オプションによる出力は、表示文字数も拡大されますが、表示内容の文字数がその文字数を超える場合も、全内容を表示します。このとき、ヘッダと表示するカラムがずれることがあります。

（qstat(1)コマンド-l オプションの詳細については、6.5.全内容の表示 を参照）

選択表示とソート

ジョブ情報表示の際、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。-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。

ジョブの情報表示で-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。

表 4-1 qstat(1) -F オプションの指定

-F指定アイテム	内容	サマリ表示形式
jno	ジョブ番号	JNO
rid	リクエストID	RequestID
jid	ジョブID	JobID
ejid	実行ジョブID（実行ホスト上のジョブID）	EJID
mem	ジョブの使用メモリ量	Memory
cpu	ジョブの実行中プロセスのCPU使用量	CPU
acpu	ジョブの積算CPU使用量（終了したプロセスを含む）	ACCPU
rcpu	ジョブの残CPU使用量	RCPU
jsvno	ジョブサーバ番号	JSVNO
ehost	ジョブの実行ホスト	ExecutionHost
euser	ジョブの実行ユーザ	UserName
ecode	ジョブ終了コード(16進表示)	Exit

qstat(1)コマンド-F オプションの使用方法詳細については、6.2.出力情報のカスタマイズ を参照してください。

また、qstat(1)の-O, -o オプションにより、上記アイテムをキーとしてリクエスト情報の出力をソートすることができます。qstat(1)コマンド-o オプション、-O オプションの詳細（エラー時の出力など）については、6.3.出力情報のソート を参照してください。

4.2　シグナルの送信

ジョブ ID を指定して qsig(1)コマンドを実行することにより、ジョブに対してシグナルを送ることができます。シグナルは指定したジョブ内の全プロセスに対して送信します。

```
$ qsig -s SIGKILL 0:72.host1
Job 0:72.host1 was sent signal SIGKILL.
```

-s signal を省略すると SIGTERM を送信します。

ジョブを強制的に終了させたい場合などに利用可能です。

第5章　　ネットワークリクエストの操作

ネットワークリクエストとは、バッチリクエストの入出力ファイルや実行結果ファイルをクライアントホスト、実行ホスト間で転送するためのリクエストです。
バッチサーバにより自動生成されるものです。

5.1　ネットワークリクエストの状態確認

ネットワークリクエストの状態を確認するには、qstat(1)コマンドの-T オプションを実行してください。ネットワークリクエストの指定は、ファイルステージング対象であるバッチリクエストのリクエスト ID で行います。(以下、親リクエスト ID と言います。)

qstat(1)コマンドに、-T オプションと、親リクエスト ID を指定してください。
以下にその場合の例を示します。

```
$ qstat -T 92.host1
RequestID        UserName  Pri Dir STGNO StagingFile                 STT
---------------- -------- ---- --- ----- ------------------------------ ---
92.host1         user1      0 IN     0    host1:/home/user1/dir1/infile1 QUE
92.host1         user1      0 IN     1    host1:/home/user1/dir1/infile1 QUE
```

STT のカラムにリクエストの状態が表示されます。ここで表示される意味は、それぞれ以下のとおりです。

- RUN ‥‥‥ 実行中
- QUE ‥‥‥ 実行待ち状態
- WAT ‥‥‥ 開始時刻の待ち合わせ中

STT 以外の項目の内容については、qstat(1)の項を参照してください。

qstat(1)コマンドで、-T オプションのみを指定して実行した場合は、現在登録されているすべてのネットワークリクエストの情報を表示します。

```
$ qstat -T
RequestID        UserName  Pri Dir STGNO StagingFile                 STT
---------------- -------- ---- --- ----- ------------------------------ ---
90.host1         user1      0   IN     0 host1:/home/user1/dir0/infile0 QUE
90.host1         user1      0   IN     1  host1:/home/user1/dir0/infile0 QUE
```

```
91.host1        user1      0   IN     0  host1:/home/user1/dir1/infile1 QUE
92.host1        user1      0   IN     0  host1:/home/user1/dir1/infile1 QUE
92.host1        user1      0   IN     1  host1:/home/user1/dir1/infile1 QUE
```

ネットワークリクエストの詳細情報が欲しい場合は、qstat(1)コマンドに-T -f オプションを付けて実行してください。

```
$ qstat -T -f 92.host1
Request ID: 92.host1
    User  Name = user1
    User  ID   = 500
    Group ID   = 501
    Network Request Priority = 0
    Staging Direction = STAGE_IN
    Staging Number    = 0
    Staging File     = host1:/home/user1/dir1/infile1.txt
    Staging Method    = Internal
    State           = Queued
```

各項目の内容については、qstat(1)の項を参照してください。

qstat(1)コマンドで出力される各項目（アイテム）を、必要なものだけ選択し、それらを組み合わせて表示することが可能です。qstat(1)コマンドの-F オプションを使用して表示するアイテムを指定してください。指定可能なアイテムは、ネットワークリクエスト情報に関しては以下のようになっています。

表 5-1 ネットワークリクエストの情報

-F指定アイテム	内容	サマリ表示形式
rid	ネットワークリクエストの親リクエストID	RequestID
own	所有者	UserName
pri	ネットワークリクエストプライオリティ	Pri
stgdir	ステージング方向	Dir
stgno	ステージングファイル番号	STGNO
stgf	ステージングファイル	StagingFile
stt	ネットワークリクエスト状態	STT

-F オプションの詳細については、6.2.出力情報のカスタマイズ を参照してください。

また、qstat(1)の-O, -o オプションにより、上記アイテムをキーとしてリクエスト情報の出力をソートすることができます。qstat(1)コマンド-o オプション、-O オプションの詳細（エラー時の出力な

ど）については、6.3.出力情報のソート を参照してください。

qstat(1)コマンドの-T の基本情報表示(-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、文字数の多い情報が途中で切られる場合があります。この様な場合は -l オプションを指定することで、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 オプションの詳細については、6.5.全内容の表示 を参照してください。

5.2　ネットワークリクエストの状態遷移

ネットワークリクエストはバッチリクエストが STAGING 状態、または EXITING 状態になった時点でバッチサーバによって自動的に生成され、ネットワークキューに投入されます。

投入された直後のネットワークリクエストは、QUEUED 状態です。その後、バッチサーバによってスケジューリングされ RUNNING 状態に移行します。ネットワークリクエストが RUNNING 状態になったら、ファイル転送を実行します。（[管理編] ネットワークキューの設定（転送先ホスト）参照）

ネットワークリクエストがファイル転送に失敗した場合、WAITING 状態に遷移します。WAITING 状態になった場合、バッチサーバやネットワークキューに設定された転送待ち時間後に再転送を試みます。（[管理編] ネットワークキューの設定（転送待ち時間）参照）

5.3　ネットワークリクエストの削除

ネットワークリクエストは、親リクエストの削除に伴って削除されます。ネットワークリクエストを削除するには、その親となるバッチリクエストを削除してください。ネットワークリクエストのみを削除することは、できません。（1.5 **バッチリクエストの削除** 参照）

第6章　　システム情報の表示

6.1　システム情報の表示方法

NQSV システムの情報を確認するには、qstat(1)コマンドを実行してください。

qstat(1)コマンドに、表示対象を指定する表示情報オプション（B, C, D, E, G, J, M, Q, R, S, T, オプションなし, --limit, --custom, --template,--venode）を付けて実行すると、以下の各情報を表示します。

表 6-1 qstat(1)の表示オプション

表示される情報	表示情報オプション	その他追加オプション
バッチサーバホスト情報	-B	-l, -F, -o, -O, -d, -x 指定可
スケジューラ情報	-D	-l, -F, -o, -O, -x 指定可
ジョブサーバ情報	-S	-l, -F, -o, -O, t, -x 指定可
実行ホスト情報	-E	-l, -F, -o, -O, -t 指定可
ノードグループ情報	-G	-l, -F, -o, -O, -x 指定可
キュー情報	-Q	-l, -F, -o, -O, -d, -t, --group=*group*名 指定可
リクエスト情報	オプションなし	-l, -F, -o, -O, -d, -x, --group[=*group*名] 指定可
パラメトリックリクエスト情報	-R	-l, -F, -o, -O, -x 指定可
ジョブ情報	-J	-l, -F, -o, -O, -d, 指定可
ネットワークリクエスト情報	-T	-l, -F, -o, -O 指定可
グループ毎・ユーザ毎の制限情報	--limit	--group=*group*名 指定可
カスタムリソース情報	--custom	
テンプレート情報	--template	-l 指定可
VEノード情報	--venode	

各情報の基本表示は、情報の項目や表示文字数が規定されたサマリ表示ですが、-f オプションを指定すると、詳細情報が表示されます。

その他に、情報の種類により、以下のオプションを合わせて使用できます。

出力項目をカスタマイズする-F オプション　（6.2.出力情報のカスタマイズ 参照）

出力情報を任意の項目でソートする-o, -O オプション　（6.3.出力情報のソート 参照）

時間データの表示フォーマットを変更する-d オプション　（6.4.時間単位項目の表示 参照）
基本表示から文字数制限だけ解除する-l オプション　（6.5.全内容の表示 参照）
ダウンしている場合も、ジョブサーバ（実行ホスト）の情報を表示する-t オプション　（6.6.全ジョブサーバ（実行ホスト）表示 参照）
グループに関する--group オプション　（8.2.グループに関する情報の表示 参照）

なお、他ユーザのリクエスト情報を参照するには特別利用者権限以上、その他の情報を参照するには利用者権限以上が必要です。

上記以外の各情報の固有オプションについては、qstat(1)コマンドのリファレンスを参照してください。

6.2　出力情報のカスタマイズ

qstat(1)コマンドの各情報のサマリ表示で出力される各項目（アイテム）を、必要なものだけを選択して表示することが可能です。前述の表示対象オプションに重ねて、-F オプションを使用して、表示するアイテムを指定してください。各アイテムは-F オプションで指定された任意の順番で表示されます。

指定方法は以下のとおりです。
qstat <表示情報オプション> -F item[,item, ...]

下記に、バッチリクエスト情報を、投入キュー名(quenm)、バッチリクエスト識別子(rid)、リクエスト状態(stt)、ジョブの実行ホスト(ehost)の順にサマリ表示した例を示します。

```
$ qstat -F quenm,rid,stt,ehost
Queue     RequestID       STT ExecutionHost
--------- --------------- --- ---------------
nqstest   80536.host1     RUN exec.example.co
nqstest   80542.host1     QUE -
nqstest  80543.host1     QUE -
```

各表示オプションで指定可能なアイテムは決まっており、以下のようになっています。
* 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。

表 6-2 出力情報のカスタマイズ

表示情報オプション	-F指定 アイテム	サマリ表示形式	内容
-B （バッチサーバ ホスト情報）	bsvhost	BatchServer	バッチサーバホスト名
	mid	MachineID	マシンID
	umax	UMAX	ユーザ毎のリクエスト投入数制限値
	gmax	GMAX	グループ毎のリクエスト投入数制限値
	rrlm	RRL	転送キュー同時実行数制限値
	nrlm	NRL	ネットワークキュー同時実行数制限値
	tot	TOT	バッチサーバが管理しているリクエスト数
	arr	ARR	各状態のリクエスト数
	wai	WAI	
	gqd	GQD	
	que	QUE	
	run	RUN	
	ext	EXT	
	hld	HLD	
	sus	SUS	
	fwd	FWD	
	bstt	Status	バッチサーバの状態
-D （スケジューラ情報）	schid	SCHID	スケジューラID
	schnm	SCHName	スケジューラ名
	schost	ExecutionHost	スケジューラ実行ホスト
	ques	BindQueue	バインドしている実行キュー数
	quenm*	BindQueueName	バインドしているキュー名
-S （ジョブサーバ情報）	jsvno	JSVNO	ジョブサーバ番号
	jsvnm	JobServerName	ジョブサーバ名
	bsvhost	BatchServer	バッチサーバホスト名
	ehost	ExecutionHost	ジョブサーバ実行ホスト
	link	LINK	リンク状態
	bind	BIND	バインド状態
	quenm	Queue	バインドしているキュー名

表示情報オプション	-F指定 アイテム	サマリ表示形式	内容
	jobs	Jobs	バッチジョブ数
	ldavg1	Load	過去1分間のロードアベレージ
	cpuavg1	CPU	過去1分間のCPUアベレージ
-E （実行ホスト情報）	ehost	ExecutionHost	実行ホスト
	bsvhost	BatchServer	バッチサーバホスト名
	osnm	OS	オペレーティングシステム名
	rel	Release	OSリリース番号
	hwnm	Hardware	ハードウェア名
	ldavg1	Load	過去1分間のロードアベレージ
	cpuavg1	CPU	過去1分間のCPUアベレージ
	ucpu*	Used_CPUs	使用CPU数
	fcpu*	Free_CPUs	未使用CPU数
	umem1*	Used_MEM1	使用メモリサイズ
	fmem1*	Free_MEM1	未使用メモリサイズ
	uswap1*	Used_SWAP1	使用スワップサイズ
	fswap1*	Free_SWAP1	未使用スワップサイズ
	quenm*	QueueName	バインドしているキュー名
	stt	STT	実行ホストの状態
-G （ノードグループ情報）	ngrpnm	NodeGroup	ノードグループ名
	ngrptype	Type	ノードグループのタイプ
	bsvhost	BatchServer	バッチサーバホスト
	comment	Comment	コメント
	jsvs	JSVs	ジョブサーバ数
	quenm	BindQueue	バインドしているキュー名
-Q [-e] （実行キュー情報）	quenm	QueueName	キュー名
	schid	SCH	バインドされているスケジューラID
	jsvs	JSVs	バインドされているジョブサーバ数
	stt1	ENA	キューの状態（投入受付可否）
	stt2	STS	キューの状態（実行可否）

表示情報オプション	-F指定 アイテム	サマリ表示形式	内容
	pri	PRI	キュープライオリティ
	qtot	TOT	キューに登録されているバッチリクエスト数
	arr	ARR	キューに登録されている各状態のリクエスト数
	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*	ExecutionHost	バインドされている実行ホスト
	attbl*	ATTBL	リクエストへのアタッチの可否
-Q -r （転送キュー情報）	quenm	QueueName	キュー名
	stt1	ENA	キューの状態（投入受付可否）
	stt2	STS	キューの状態（実行可否）
	pri	PRI	キュープライオリティ
	rqrlm	RLM	同時実行数制限値
	qtot	TOT	キューに登録されているバッチリクエスト数
	arr	ARR	キューに登録されている各状態のリクエスト数
	wai	WAI	
	que	QUE	
	hld	HLD	
	trs	TRS	
-Q -N （ネットワーク	quenm	QueueName	キュー名
	stghost	StagingMachine	ステージングホスト

表示情報オプション	-F指定 アイテム	サマリ表示形式	内容
キュー情報）	stt1	ENA	キューの状態（投入受付可否）
	stt2	STS	キューの状態（実行可否）
	pri	PRI	キュープライオリティ
	nqrlm	RLM	同時実行数制限値
	qtot	TOT	キューに登録されているバッチリクエスト数
	wai	WAI	キューに登録されている各状態のリクエスト数
	que	QUE	
	run	RUN	
-Q -i （会話キュー情報）	quenm	QueueName	キュー名
	schid	SCH	バインドされているスケジューラID
	jsvs	JSVs	バインドされているジョブサーバ数
	stt1	ENA	キューの状態（投入受付可否）
	stt2	STS	キューの状態（実行可否）
	pri	PRI	キュープライオリティ
	qtot	TOT	キューに登録されているバッチリクエスト数
	arr	ARR	キューに登録されている各状態のリクエスト数
	wai	WAI	
	que	QUE	
	prr	PRR	
	run	RUN	
	por	POR	
	ext	EXT	
	hld	HLD	
	sus	SUS	
	ehost*	ExecutionHost	バインドされている実行ホスト
	attbl*	ATTBL	リクエストへのアタッチの可否
なし （リクエスト情報）	rid	RequestID	リクエストID
	reqnm	ReqName	リクエスト名
	own	UserName	所有者
	group	GrpName	グループ名
	quenm	Queue	投入キュー名

表示情報オプション	-F指定 アイテム	サマリ表示形式	内容
	pri	Pri	リクエストプライオリティ
	stt	STT	リクエスト状態
	stall	S	ストール状態
	mem	Memory	リクエストの使用メモリ量
	cpu	CPU	リクエストの実行中プロセスのCPU使用量
	acpu	ACCPU	リクエストの積算CPU使用量(終了したプロセスを含む)
	elaps	Elapse	経過時間
	relaps	RElapse	残経過時間
	rflg	R	再実行の可否
	hflg	H	ホールドの可否
	mflg	M	ジョブマイグレーションの可否
	jobs	Jobs	リクエストあたりのジョブ数
	ehost*	ExecutionHost	ジョブの実行ホスト
	sdate*	Date(SUBMIT)	リクエスト投入時刻
	qdate*	Date(QUEUED)	スケジューリング開始時刻
	rdate*	Date(RUNNING)	リクエスト実行開始時刻
	att*	Att	アタッチ接続の有無
-R （パラメトリック リクエスト情報）	rid	RequestID	パラメトリックリクエストID
	reqnm	ReqName	リクエスト名
	own	UserName	所有者
	group	GrpName	グループ名
	quenm	Queue	投入キュー名
	pri	Pri	リクエストプライオリティ
	stt	STT	リクエスト状態
	rflg	R	再実行の可否
	hflg	H	ホールドの可否
	mflg	M	ジョブマイグレーションの可否
	jobs	Jobs	サブリクエストあたりのジョブ数
	tot	TOTAL	全サブリクエスト数
	active	ACTIVE	存在するサブリクエスト数

表示情報オプション	-F指定 アイテム	サマリ表示形式	内容
	done	DONE	終了したサブリクエスト数
-J （ジョブ情報）	jno	JNO	ジョブ番号
	rid	RequestID	リクエストID
	jid*	JobID	ジョブID （ジョブ番号とリクエストIDを合わせたもの）
	ejid	EJID	実行ジョブID
	mem	Memory	ジョブの使用メモリ量
	cpu	CPU	ジョブの実行中プロセスのCPU使用量
	acpu	ACCPU	ジョブの積算CPU使用量(終了したプロセスを含む)
	rcpu	RCPU	ジョブの残CPU使用量
	jsvno	JSVNO	ジョブサーバ番号
	ehost	ExecutionHost	ジョブの実行ホスト
	euser	UserName	ジョブ実行ユーザ
	ecode	Exit	ジョブ終了コード(16進表示) 詳細な値は、[リファレンス編]の1.16 qstat(1)を参照してください。
-T （ネットワーク リクエスト情報）	rid	RequestID	ネットワークリクエストの親リクエストID
	own	UserName	所有者
	pri	Pri	ネットワークリクエストプライオリティ
	stgdir	Dir	ステージング方向
	stgno	STGNO	ステージングファイル番号
	stgf	StagingFile	ステージングファイル
	stt	STT	ネットワークリクエスト状態

なお、-F オプションと-f オプション（詳細表示）が同時に指定された場合は、エラーは出力されず、-f オプションの指定が優先され、-F オプションが無視されます。（結果として、詳細情報が出力されます。）

6.3　出力情報のソート

qstat(1)コマンドの実行結果に対して、任意の項目をキーとしてソートすることが可能です。

前述の表示情報オプションに重ねて、qstat(1)コマンドの-o オプション、または-O オプションを使用して、ソート対象項目（アイテム）を指定してください。

ソート対象項目名は、前節の 6.2.出力情報のカスタマイズ の表にて示されるアイテムと共通です。

ソート対象のアイテムは、","で区切ることで複数指定することができます。

-o item[,item, ...]
item で指定されたアイテムをキーとして昇順にソートします
-O item[,item, ...]
item で指定されたアイテムをキーとして降順にソートします

item が複数指定された場合、並びに本オプションが複数指定された場合、順番に、アイテムでソートし、その結果をさらにソートします。

-o オプションと-O オプションを組み合わせて、最初のアイテムは昇順、次のアイテムは降順といった指定をすることも可能です。

また、-F オプションと同時に使用することも可能です。

ソート方法は、対象が数字の場合は数の大小を比較、文字の場合はアルファベット順となります。

【例】

① qstat -o rid,quenm,stt

② qstat -o rid -o quenm -o stt

③ qstat -o rid -O quenm -o stt

①,②の出力結果は同じです。rid,quenm,stt の順に、昇順でソートします。

③は、rid を昇順でソートした後に、quenm で降順にソートし、さらに stt で昇順にソートします。

下記の場合はエラーとなります。

ソート対象項目名が指定されていない場合

ソート対象項目名に指定形式以外のものを指定した場合

各表示形式に存在しないソート対象項目名を指定した場合

なお、-o, -O オプションと-f オプション（詳細表示）が同時に指定された場合は、エラーは出力されず、-f オプションの指定が優先され、-o, -O オプションが無視されます。（結果として、詳細情報が出力されます。）

6.4　時間単位項目の表示

qstat(1)コマンドの実行結果のうち、時間データは秒単位での表示ですが、-d オプションを指定することによって、d+hh:mm:ss（日+時:分:秒）形式での表示にすることが可能です。

時間データの桁数が大きくなり、秒単位では瞬時に時間を理解するのが困難となる場合などに活用することができます。

-d オプションは、下記の出力項目で有効となります。

Elaps time（Elaps）

リクエスト状態（オプションなし）の CPU 時間使用量(CPU), 積算 CPU 時間使用量（ACCPU),
バッチジョブ情報（-J オプション）の CPU 時間使用量(CPU), 積算 CPU 時間使用量（ACCPU）

また、-f オプションにて詳細表示を行った場合にも、-d オプションを指定すると、上記項目の表示が d+hh:mm:ss（日+時:分:秒）形式になります。

【例】

リクエスト情報の CPU 時間使用量(CPU)と Elaps time(Elapse)

```
$ qstat -d
RequestID       ReqName  UserName Queue     Pri STT S   Memory          CPU       Elapse R H M Jobs
--------------- -------- -------- -------- ---- --- - -------- ------------ ------------ - - - ----
80536.host1     STDIN    nqstest  que1       0 QUE -    7.62M   1+05:21:10   1+16:30:12 Y Y Y    1


バッチジョブ情報のCPU時間使用量(CPU)
$ qstat -J -d
JNO RequestID        EJID   Memory          CPU JSVNO ExecutionHost   UserName Exit
---- --------------- ----- -------- ------------ ----- --------------- -------- ----
0 1249.bsvhost    19710    1.91M   1+05:21:10  1020          host1  nqstest    -
1 1249.bsvhost    28094    1.90M   1+05:01:10  1030          host2  nqstest    -
```

6.5　全内容の表示

qstat(1)コマンドでの各情報の基本表示は、サマリ表示で、出力する内容が規定の文字数で表示されるため、文字数の多い情報が途中で切られる場合があります。

このような場合は、-l オプションを指定することにより、情報の内容を全て表示することが可能です。

例えば、qstat コマンドの表示結果を使用してスクリプトを実行する場合、-l オプションを指定した表示を利用すると、問題なく動作させることができます。

なお、-l オプションと-f オプション（詳細表示）が同時に指定された場合は、エラーは出力されず、-f オプションの指定が優先され、-l オプションが無視されます。（結果として、詳細情報が出力されます。）

下記は、-l オプション指定なしと指定ありのコマンド実行例です。

```
$ qstat
RequestID       ReqName  UserName Queue     Pri STT S   Memory      CPU   Elapse R H M Jobs
--------------- -------- -------- -------- ---- --- - -------- -------- -------- - - - ----
80333.host1     STDIN    nqstest  que1       0 QUE -   0.00B     0.00       0 Y Y Y    1
```

```
$ qstat -l
RequestID                              ReqName             UserName        Queue
  Pri STT S       Memory          CPU       Elapse R H M Jobs
---------------------------------------- -------------------- --------------- ---------------
 ---- --- - ------------ ------------ ------------ - - - ----
80333.host1.example.com                STDIN               nqstest1        que1
    0 QUE -        0.00B        0.00           0 Y Y Y    1
```

-l オプションを指定した場合、規定の文字数からは、ずれる形になりますが、リクエスト ID などが途中で切られることなく全て表示されます。

6.6　全ジョブサーバ（実行ホスト）表示

qstat(1)コマンドでは、通常、ジョブサーバや実行ホストの情報は、ジョブサーバがリンクアップしているものだけを対象としていますが、-t オプションを指定すると、リンクダウンしているものも含めて、情報を表示するようになります。

ジョブサーバ情報表示（-S オプション）、並びに、実行ホスト情報表示（-E オプション）では、-t を指定した場合、リンクダウンしているものも表示対象として表示します。

実行ホスト情報表示（-E オプション）では、ジョブサーバ番号とジョブサーバの状態の情報も追加して表示します。

また、-f オプションを合わせて指定した場合も同様に、リンクダウンしているものも表示対象としますが、リンクダウン状態で取得できる情報のみとなります。

キュー情報表示（-Q オプション）では、通常バインドしているジョブサーバの数は、リンクアップしているものだけの数ですが、-t を指定した場合、リンクアップしている数に、並べてリンクダウン

しているものも含めた総数も合わせて表示します。

第7章　　ワークフロー

NQSV システムのワークフロー機能は、リクエストの投入・実行を含む一連の処理をシェルスクリプトとして記述し、クライアントホスト上で、そのシェルスクリプトをワークフローとして NQSV システムで実行する機能です。

NQSV のワークフロー機能により、NQSV システムにおけるリクエストの実行において以下のことが行えます。

- ワークフロー内で、投入する複数のリクエスト間の実行順序関係を簡単に記述して制御することができます。
- ワークフロー内で、シェルスクリプトによる繰り返し、条件判定を利用して、リクエストの結果などによる、リクエスト実行の繰り返し、条件分岐が行えます。
- リクエスト実行を含む一連の処理を、ワークフローとして繰り返し実行することができます。
- ワークフロー単位で、投入・実行したリクエストの状態の参照、制御が行えます。

7.1　ワークフローの実行環境

NQSV のワークフロー機能を使用してリクエストを投入する際に、ワークフロー内で投入可能なリクエストの投入環境には、以下の条件があります。

- ワークフロー内で投入するリクエストの投入先のバッチサーバが同じであること
- ワークフロー内で投入するリクエストの投入先のキューが同じJobManipulatorにバインドされていること

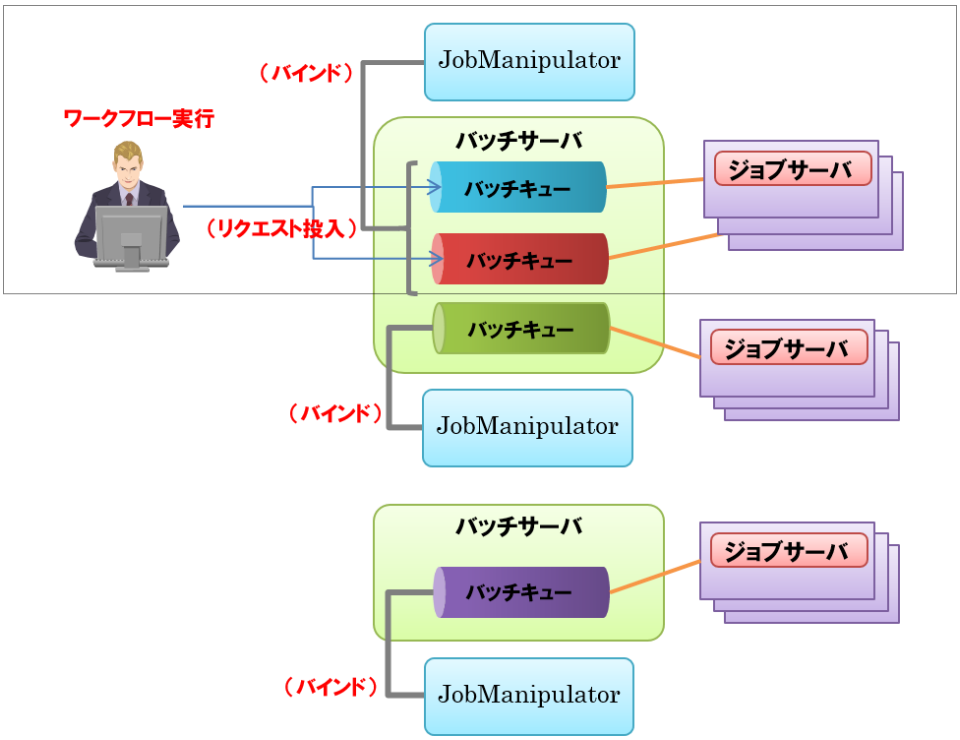


図 7.1-1：ワークフローの実行

7.2　ワークフローの記述

ワークフローは、クライアントホスト上のシェルスクリプトとして記述します。ワークフローとして記述するシェルスクリプトを、以降ワークフロースクリプトと呼びます。

ワークフロースクリプト内では、リクエストの投入、参照、制御のために、qsub, qstat, qdel などの NQSV の CUI コマンドを使用することができます。

7.2.1　簡単な例

たとえば、リクエストを投入してその実行の終了を待ち合わせる、ワークフロースクリプトは以下のように記述することができます。

```
#!/bin/bash
qsub -q batch1 ./req_A
qwait req_A
```

ここで、上記のワークフロースクリプト内の req_A は、バッチキューbatch1 へ投入するリクエストのジョブスクリプトファイルです。

ジョブスクリプトファイルには、#PBS 行に投入オプションを記述することができます。（1.1.バッチリクエストの作成 参照）

また、ワークフロースクリプト内で、リクエストを指定した各種の操作を行う際には、リクエス

ト ID の代わりに、リクエスト名を指定することができます。

上記の例では、qwait コマンドで待ち合わせを行う対象リクエストの指定において、リクエスト名(req_A)を指定しています（※リクエスト投入の際に-N オプションにより明示的にリクエスト名を指定しない場合、ジョブスクリプトのファイル名がリクエスト名となります)。

リクエストの指定において、リクエスト ID の代わりにリクエスト名が指定できるのは、以下のコマンドです。

- qdel
- qhold
- qrls
- qalter
- qmove
- qrerun
- qstat
- qwait
- qwait2

ワークフロースクリプト内でリクエストの指定の際にリクエスト名を使用する場合は、ワークフロースクリプト内で投入するリクエストに対して、リクエスト名が一意となるようにしてください。

7.2.2　実行順序の指定方法

ワークフロー内でリクエストの実行順序を指定する際は、qsub コマンドの--after オプションを使用します。

たとえば、リクエスト A（req_A）の実行が終了した後に、リクエスト B（req_B）が実行開始するように実行順序の指定を行う場合は、ワークフロースクリプト内で qsub コマンドを以下のように記述します。

```
#!/bin/bash
qsub -q batch1 ./req_A
qsub -q batch2 --after req_A ./req_B
```

上記のようにしてリクエスト A とリクエスト B を投入することで、リクエスト A とリクエスト B を同時に投入した状態で、実行開始の順序がスケジューラによって制御されます。

また、ワークフロースクリプト内で、qsub --after オプションによりリクエストを指定する際にも、リクエスト ID の代わりにリクエスト名を使用することができます。

さらに、投入したリクエストの実行開始の条件として、複数のリクエストの実行終了を待ち合わ

せる場合は、qsub コマンドの--after オプションでリクエストを指定する際に、待ち合わせたいリクエストをカンマで区切って複数記述します。

```
#!/bin/bash
qsub -q batch1 ./req_A
qsub -q batch2 ./req_B
qsub -q batch1 --after req_A,req_B ./req_C
```

上記のようにワークフロースクリプトを記述した場合、リクエスト C（req_C）は、リクエスト A（req_A）とリクエスト B（req_B）の両方が実行終了した後で実行開始するようにスケジューリングされます。

リクエスト間で実行順序を指定してリクエストを実行する際に、先行リクエストの出力ファイルを後続のリクエストの入力とする場合は、実行ホスト間で共有されているファイルシステムを使用して、リクエスト間でファイルの受け渡しを行うようにしてください。

7.2.3　同時実行開始の指定方法

ワークフローによるリクエストの実行の際に、複数のリクエストを同時に実行開始するようにスケジューリングする場合、qsub --parallel オプションを使用します。

```
#!/bin/bash
qsub -q batch1 --parallel ./req_A ./req_B
```

qsub コマンドで--parallel オプションを指定した場合、qsub コマンドの最後の引数として 2 つ以上のジョブスクリプトファイルを指定することで、指定したジョブスクリプトをそれぞれ別のリクエストとして投入し、それらのリクエストを同時に実行開始するようにスケジューリングします。

上記の場合、req_A と req_B で指定されてジョブスクリプトがそれぞれ、リクエストとして投入され、この 2 つのリクエストが同時に実行開始するようにスケジューリングされます。

qsub コマンドで--parallel オプションを指定した場合は、リクエスト連携機能（直列連携、並列連携）の投入は行えません。

ハイブリッドリクエスト（qsub コマンドでのリクエスト投入時に--job-separator あるいは---オプションを用いてジョブ毎に異なる資源を要求して投入したリクエスト）を同時実行リクエストとして投入した場合（qsub コマンドで--parallel オプションを指定した場合）は、当該リクエストはスケジューリングされません。

7.3　ワークフローの実行

上記のように作成したワークフロースクリプトは、クライアントホスト上で、wstart コマンドを使用して実行します。

ワークフロースクリプトの実行は、以下のように、wstart コマンドの引数に、実行するワークフロースクリプトのファイルを指定して、wstart を起動することにより行います。

```
$ /opt/nec/nqsv/bin/wstart wfl.sh
Workflow 84_bsvhost is start.
Request 163.bsvhost submitted to queue: batch1.
Request 164.bsvhost submitted to queue: batch2.
   :
```

wstart コマンドは、ワークフローの起動をバッチサーバに通知して、ワークフローID を取得し、取得したワークフローID を標準出力に表示した後、引数に指定されたワークフロースクリプトを NQSV のワークフローとして実行します。

wstart コマンドから起動されたワークフロースクリプトから qsub コマンドにより投入されたリクエストは、wstart で取得したワークフローID を持つワークフローの一部としてバッチサーバにより管理されます。

wstart が実行するワークフローのために取得したワークフローID は、ワークフロースクリプト内で、環境変数 NQSV_WFID または NQSII_WFID（どちらも同じ値が入ります）により参照することができます。

wstart コマンドからのワークフロースクリプトの実行による標準出力、標準エラー出力は、そのまま wstart を起動した端末上に出力されます。

また、wstart の起動後、起動した端末からの標準入力は、ワークフロースクリプトを実行するシェルの標準入力として渡されます。

ワークフロースクリプトからのリクエスト投入によりワークフローの一部として管理できるのは、バッチキューに投入したバッチリクエストのみです。会話リクエストは、ワークフロー内のリクエストとして管理されません。

wstart コマンドにより生成されるワークフローは、wstart で起動したワークフロースクリプトが実行中、または、ワークフロースクリプトから NQSV へ投入されてリクエストが存在する間ワークフローとしてバッチサーバにより管理され、以下の参照、操作を行うことができます。

7.4　ワークフローの参照

wstart コマンドで生成したワークフローは、wstat コマンドにより参照することができます。wstat コマンドを引数なしで実行すると、自分が実行しているワークフローの一覧を表示します。

```
$ wstat
WFL-ID        Request Owner    Stat
------------ ------- -------- ----
84_bsvhost          3 user1    RUN
86_bsvhost        2 user1     RUN
```

ワークフローの一覧表示では、ワークフローID、ワークフロー内で投入したリクエスト数、実行ユーザ名、ワークフローの状態（RUN）が表示されます。

また、wstat の引数に、wstart の実行時に表示されるワークフローID を指定することで、指定したワークフローで投入された全リクエストの状態を表示します。

```
$ wstat 84_bsvhost
WFL-ID      RequestID    RequestName  Stat Exit
------------ ------------ ------------ ---- ----
84_bsvhost   163.bsvhost  Req_A       EXT    0
84_bsvhost   164.bsvhost  Req_B         EXT    9
84_bsvhost   165.bsvhost  Req_C         RUN    -
```

ワークフローID 指定の表示では、ワークフロー内の各リクエストについて、ワークフローID、リクエスト ID、リクエスト名、リクエストの状態、および、リクエストの終了コードを表示します。リクエストの終了コードは、リクエスト内のジョブ番号 0 のジョブのジョブスクリプトの終了コードで、終了していないリクエストに対しては、"-"が表示されます。

7.5　ワークフローの操作

7.5.1　ワークフローの削除

wstart により実行中のワークフロー（ワークフロースクリプトが実行中、またはワークフロースクリプト内で投入したリクエストが存在する間）に対しては、wdel コマンドを使用して削除することができます。

wdel コマンドを引数にワークフローID を指定して実行することで、指定したワークフローを削除します。

```
$ wdel 84_bsvhost
Workflow 84_bsvhost was deleted.
```

このとき、ワークフロー内で投入されたリクエストがすべて削除されます。また、ワークフロースクリプトが実行中の場合は、同時にワークフロースクリプトも終了します。

7.5.2　エラー発生時の後続リクエストキャンセル機能

ワークフロースクリプト内で、リクエストが異常終了した際に、後続指定されたリクエストを自動的にキャンセルすることができます。この機能を利用するには、先行するリクエスト投入時に qsub --cancel-after オプションを指定します。なお、リクエストが異常終了したとするのは、以下の場合です。

- 終了ステータスが0以外
- qdel(1)で削除された
- HW障害によって終了した

たとえば、リクエスト A（req_A）が終了した後、リクエスト B（req_B）が実行開始するように実行順序の指定を行い、リクエスト A が異常終了した際にはリクエスト B を自動的にキャンセルしたい場合、ワークフロースクリプト内で qsub コマンドを以下のように記述します。

```
#!/bin/bash
qsub -q batch1 --cancel-after ./req_A
qsub -q batch2 --after req_A ./req_B
```

上記ワークフロースクリプトでリクエストを投入すると、以下のような動作になります。

1. リクエストAとリクエストBを投入
2. リクエストAを実行開始
3. リクエストAの終了を待つ
   - リクエストAが正常終了の場合：リクエストBを実行開始
   - **リクエストAが異常終了の場合：リクエストBを削除**

なお、自動キャンセルする後続リクエストは、--cancel-after オプションを付けて投入したリクエストの直後に実行されるリクエストだけではなく、そのリクエストの後続となるリクエスト全てを対象とします。

また、wstart コマンドは、--cancel_after オプションを使用して、ワークフロースクリプト内で実行される全ての qsub コマンドに対して、一括して自動で --cancel-after オプションを指定するこ

とができます。

実行例）

```
$ /opt/nec/nqsv/bin/wstart --cancel-after wfl.sh
```

wfl.sh

```
#!/bin/bash
qsub -q batch1 ./req_A                          ← 自動で--cancel-afterが指定される
qsub -q batch2 --after req_A ./req_B            ← 自動で--cancel-afterが指定される
```

第8章　　リクエストのグループ

8.1　リクエストのグループと投入方法

リクエストを投入すると、リクエスト情報にはユーザと共にグループの情報が保持されます。
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ます。
また、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されます。

一方、**バッチサーバの設定でリクエストのグループ指定実行機能が ON** となっている場合には、リクエストの投入コマンド（qsub, qlogin, qrsh）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、ジョブ実行時には、そのグループで実行されることになります。
なお、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した場合も同様です。
また、リクエストの投入コマンド（qsub, qlogin, qrsh）で、--group=<group_name> で、グループ名を指定してリクエストを投入することも可能です。
但し、指定できるグループ名はユーザが所属するグループに限ります。

8.2　グループに関する情報の表示

8.2.1　リクエスト情報

qstat でリクエスト情報を表示する際、qstat の実行ユーザがオーナであるリクエストを全て表示します。この時、qstat の実行グループや、リクエストのグループは問いません。

リクエスト情報の参照で、--group=<group_name> を指定した場合には、自リクエストの内、指定したグループのリクエストのみを選択表示します。

また、qstat でリクエストのサマリ情報を表示する場合に、--group オプションを指定すると、リクエストのグループが表示されます。

```
$ qstat --group
RequestID     ReqName  UserName GrpName  Queue    Pri STT S   Memory     CPU  Elapse R H M Jobs
------------ -------- -------- -------- ------- ---- --- - -------- ------- ------ - - - ----
46.bsv1       a.sh     user1    grpA     bq0        0 HLD -   0.00B    0.00       0 Y Y Y    1
47.bsv1       a.sh     user1    grpA     bq0        0 QUE -   0.00B    0.00       0 Y Y Y    1
52.bsv1       STDIN    user1    grpX     bq2        0 RUN -   1.93M    0.23      26 Y Y Y    1
```

8.2.2　キュー情報

qstat -Q でキュー情報を参照する際、アクセス権のあるキューのみを表示しますが、このアクセス権チェックは、通常、qstat 実行時のユーザ名とそのプライマリグループ名で行います。

一方、バッチサーバの設定でリクエストのグループ指定実行機能が ON となっている場合には、qstat コマンド実行時のユーザ名とグループ名でアクセス権チェックを行います。

なお、--group=<group_name> で、グループ名を指定してキューの情報参照することも可能です。

第9章　　グループごと・ユーザごとの制限

9.1　グループごと・ユーザごとの制限

リクエスト投入時、バッチサーバ並びにキューで設定された制限値を超えないかをチェックしますが、バッチサーバ並びにキューで設定された各種制限の内、以下の制限については、グル―プ名・ユーザ名を個別指定して制限値が設定される場合があります。

- バッチサーバの投入数制限
    - バッチサーバ全体での投入数制限値
    - グループごとの投入数制限
        - ✧　１グループあたりの投入数制限値（個別指定が無い場合に適用）
    - グループ名個別指定での投入数制限値
    - ユーザごとの投入数制限
        - ✧　１ユーザあたりの投入数制限値（個別指定が無い場合に適用）
        - ✧　ユーザ名個別指定での投入数制限値
- バッチ・会話・転送キューの投入数制限
    - キュー全体での投入数制限値
    - グループごとの投入数制限
        - ✧　１グループあたりの投入数制限値（個別指定が無い場合に適用）
        - ✧　グループ名個別指定での投入数制限値
    - ユーザごとの投入数制限
        - ✧　１ユーザあたりの投入数制限値（個別指定が無い場合に適用）
        - ✧　ユーザ名個別指定での投入数制限値
- バッチ・会話キューの生成ジョブ数制限
    - キューとしての生成ジョブ数制限値
    - グループ名個別指定での生成ジョブ数制限値
    - ユーザ名個別指定での生成ジョブ数制限値
- バッチ・会話キューの経過時間制限

- キューとしての経過時間制限値
- グループ名個別指定での経過時間制限値
- ユーザ名個別指定での経過時間制限値

上記のいずれかの制限を超えると投入エラーとなります。なお、グループ毎の制限については、リクエストのグループでチェックします。リクエストのグループについては、8.1.リクエストのグループと投入方法 を参照してください。

9.2　グループごと・ユーザごとの制限情報の参照

上記制限の設定値については、qstat --limit にて確認できます。バッチサーバと、バッチキュー、会話キュー、転送キューについて、通常、qstat コマンド実行時のユーザ、並びにプライマリグループでの制限情報を表示します。（但し、アクセス権の無いキューは表示されません。）

一方、**バッチサーバの設定でリクエストのグループ指定実行機能が ON** となっている場合には、qstat コマンド実行時のユーザ名とグループ名での制限情報を表示します。

なお、--group=*group*名 で、グループ名を指定して制限を確認することも可能です。

例）ユーザ名 user1, グループ名 grpA の場合：**太字の値**で制限チェックされます

```
$ qstat --limit
Batch Server: bsv1
  Submit Number Limitation Value      = 1000
  Submit User  Number Limitation Value = UNLIMITED
       user1      : Limit = 200
  Submit Group Number Limitation Value = 300
       grpA       : Limit = 100
Execution Queue: bq0
  Submit Number Limit      = 100
  Submit User  Number Limit = 10
       user1      : Limit = 30
  Submit Group Number Limit = 50
       grpA       : Limit = 40
  Range of Jobs Limit per Batch Request (min,max) = 1,10240
     User  Limit:
          user1      : Limit = 1,512
     Group Limit:
          grpA       : Limit = 512,1024
```

```
  Resources Limits:
    (Per-Req) Elapse Time Limit = Max: UNLIMITED Warn: UNLIMITED
      User Limit:
           user1      : Limit = Max:    3000S Warn:    2940S
      Group Limit:
           grpA       : Limit = Max:     600S Warn:     540S
Interactive Queue: iq
  Submit Number Limit      = UNLIMITED
  Submit User  Number Limit = UNLIMITED
  Submit Group Number Limit = UNLIMITED
  Range of Jobs Limit per Batch Request (min,max) = 1,10240
  Resource Limits:
    (Per-Req) Elapse Time Limit = Max: UNLIMITED Warn: UNLIMITED
Routing Queue: rq
  Submit Number Limit      = UNLIMITED
  Submit User  Number Limit = UNLIMITED
  Submit Group Number Limit = UNLIMITED
```

第10章　GPU を使用するリクエストの投入

qsub -l オプションに gpunum_job=*max_limit* 引数を指定することにより、GPU 台数制限値を設定できます。GPU 台数制限値には警告値はありません。

本オプションにより GPU 台数制限値を設定しなかった場合、投入先キューの同時実行 GPU 台数既定値の値がリクエストに対して設定します。

【例】

```
$ qsub -l gpunum_job=1 -q bq -l elapstim_req=1000
```

GPU 台数制限値を 1 以上に設定したリクエストの実行開始時に、実行ホスト内で各ジョブが利用できるGPUデバイスをJSVが選択し、選択したGPUデバイス番号をCUDA_VISIBLE_DEVICES環境変数にセットしてジョブを起動します。

GPU 台数制限値が 0 または UNLIMITED の場合、CUDA_VISIBLE_DEVICES 環境変数は設定しません。

GPU-CPU Affinity 機能を使用する場合は、qmgr(1M)にて GPU-CPU Affinity 機能を ON に設定して、--gpunum-lhost オプションまたは-l gpunum_job オプションにて GPU 台数制限値を指定してリクエストを投入します。この場合は、CPU 台数制限値が自動計算されますので、リクエスト投入時に指定しないでください。GPU-CPU Affinity 機能の詳細は、[管理編]18.3 GPU-CPU Affinity 機能を参照してください。

第11章　マルチインスタンス GPU(MIG)を使用するリクエストの投入

11.1 マルチインスタンスGPU(MIG)

マルチインスタンス GPU（以後 MIG とする）を使用するリクエストを投入する場合、qsub --mig オプションを使用します。MIG を使用する場合、GPU を使用するための qsub --gpunum-lhost は使用できません。qsub --mig と --gpunum-lhost を同時に指定した場合、--gpunum-lhost は無視されます。

書式

```
$ qsub --mig=gi-name[:ci-slice-count[,ci-slice-count...]]
```

qsub --mig オプションに *gi-name* 引数を指定することにより、*gi-name* に対応した GPU インスタンス(以後 GI)を割り当てます。

qsub --mig オプションに *ci-slice-count* 引数を指定することにより、GI を更に複数のコンピュートインスタンス(以後 CI)に分割し、複数のプロセスから 1 つの GI を使用できます（CI を分割した場合、メモリ領域は GI 内のプロセスで共有されます）。

また、--mig オプションを複数指定することで複数の GI をリクエストに割り当てることができます。

実行例は以下の通りです。

11.1.1　GI 1つを指定するリクエストの投入

【例】GI プロファイル 4g.20gb を 1 つ使用する

```
$ qsub --mig=4g.20gb -l cpunum_job=1,elapstim_req=1000
```

11.1.2　GI 複数を指定するリクエストの投入

リクエストの投入時に複数の GI を設定する方法は以下の通りです。

【例】GI プロファイル 4g.20gb を 2 つ使用する

```
$ qsub --mig=4g.20gb --mig=4g.20gb -l cpunum_job=1,elapstim_req=1000
```

【例】GI プロファイル 4g.20gb、3g.20gb　を使用する

```
$ qsub --mig=4g.20gb --mig=3g.20gb -l cpunum_job=1,elapstim_req=1000
```

11.1.3　GI を CI に分割するリクエストの投入

GI は複数のコンピュートインスタンス(CI)に分割可能です。qsub --mig オプションの *ci-slice-count* に分割する CI 数を指定します。

【例】GI プロファイル 4g.20gb を 1 つ使用し、GI を下図の赤枠の様に 3 つの CI に分割する
この場合は *ci-slice-count* に 2,1,1 と指定する。

```
$ qsub --mig=4g.20gb:2,1,1 -l cpunum_job=1,elapstim_req=1000
```

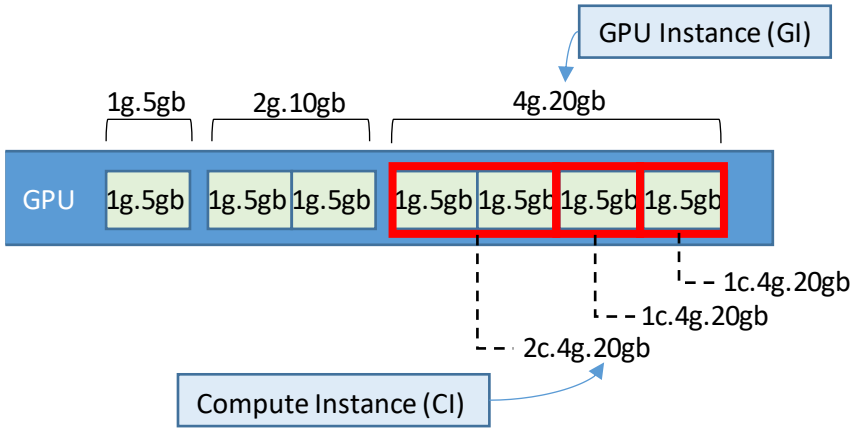


図 11.1-1　GI を CI に分割する例

11.2 qstatでのMIG確認

リクエスト投入可能な GI は qstat -Ef コマンド（実行ホストの情報）で確認できます。

```
$ qstat -Ef
Execution Host: mig1.exsample.local
    Batch Server = bsv.exsample.local
   （省略）
  Resource Information:
    Memory          = Assign:  255846400 Using:     889856 Maximum:  255846400
    Swap            = Assign:    1047552 Using:          0 Maximum:    1047552
    Number of Cpus = Assign:        128 Using:          0 Maximum:        128
    Multi Instances GPU = {
      GPU0:
        MIG 1g.5gb      = Using: 0  Total : 0
        MIG 1g.5gb+me   = Using: 0  Total : 0
        MIG 2g.10gb     = Using: 0  Total : 0
        MIG 3g.20gb     = Using: 0  Total : 1 (4:4)
        MIG 4g.20gb     = Using: 0  Total : 1 (0:4)
        MIG 7g.40gb     = Using: 0  Total : 0
```

```
    GPU1:
      MIG 1g.5gb      = Using: 0  Total : 0
      MIG 1g.5gb+me   = Using: 0  Total : 0
      MIG 2g.10gb     = Using: 0  Total : 0
      MIG 3g.20gb     = Using: 0  Total : 1 (4:4)
      MIG 4g.20gb     = Using: 0  Total : 1 (0:4)
      MIG 7g.40gb     = Using: 0  Total : 0
  }
...
```

11.3 MIGジョブ実行

ジョブサーバはジョブ実行時、指定された MIG デバイスを CUDA_VISIBLE_DEVICES 環境変数に設定してジョブスクリプトを起動します。

CUDA_VISIBLE_DEVICES 環境変数にはカンマ区切りで複数の MIG デバイスが設定されます。しかし、MIG 対応 GPU 環境では CUDA ライブラリの制限により、CUDA アプリケーションからは先頭の MIG デバイスしか使用されません。複数の MIG デバイスを 1 つのリクエストで使用する場合、カンマで区切られた内容ごとに CUDA_VISIBLE_DEVICES 環境変数を再設定して、各プロセスに渡す必要があります。

例：

CUDA_VISIBLE_DEVICES=device1,device2,device3 の場合、CUDA_VISIBLE_DEVICES を下記のように再設定して、各プロセスを実行します。

```
export CUDA_VISIBLE_DEVICE=device1
program1

export CUDA_VISIBLE_DEVICE=device2
program2

export CUDA_VISIBLE_DEVICE=device3
program3
```

ジョブスクリプトの例:

```
#! /bin/bash
#PBS -q bq
#PBS -l elapstim_req=1000
#PBS -l cpunum_job=1
#PBS --mig=4g.20gb
#PBS --mig=4g.20gb

function cudamprun(){
    pids=""
    # Replace "," with a space so that it can be used in the for statement
```

```
        tmpcuda="${CUDA_VISIBLE_DEVICES//,/ }"
        # Redefine the CUDA_VISIBLE_DEVICES environment variable
        for CUDA_VISIBLE_DEVICES in $tmpcuda;do
            "$@" &
            pids="$pids $!"
        done
        wait $pids
    }

    cudamprun cudaprogram
```

第12章　ユーザプリ・ポストスクリプトを指定するリクエストの投入

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。

12.1 ユーザPPスクリプト

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。

ユーザ PP スクリプトが参照可能な環境変数は下記のとおりです。

表 12-1 ユーザ PP スクリプトの環境変数

環境変数	説明	変数値
UPP_LOCATION	ユーザPPスクリプトを起動したロケーション	"PRERUNNING" または "POSTRUNNING"
UPP_JOBID	ジョブID	ジョブ番号:シーケンス番号.ホスト名
UPP_NJOBS	ジョブ数	1以上の整数
UPP_USER	リクエスト所有者の実行ホスト上でのユーザ名	文字列
UPP_GROUP	リクエスト所有者の実行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がONの場合のリクエスト投入時に指定したグループ名	文字列
UPP_UID	リクエスト所有者の実行ホスト上でのユーザID	0以上の整数値
UPP_GID	リクエスト所有者の実行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が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	0以上の整数値
UPP_HOME	ホームディレクトリの絶対パス名	文字列
UPP_STGDIR	ステージングファイルの格納ディレクトリ	絶対パス名

環境変数	説明	変数値
UPP_RSTPREV	ひとつ前のリクエスト状態	"QUEUED" （QUEUED状態） "RUNNING" （RUNNING状態） "CHKPNTING" （CHKPNTING状態） "HOLDING" （HOLDING状態） "SUSPENDING" （SUSPENDING状態） "SUSPENDED" （SUSPENDED状態） "RESUMING" （RESUMING状態）
UPP_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）
UPP_JTPLGY	ジョブトポロジー１．の	"JTPLGY_DISTRIB" （分散ジョブ） "JTPLGY_INTMPI" （intmpiジョブ） "JTPLGY_MVAPICH" （mvapichジョブ） "JTPLGY_NECMPI" （necmpiジョブ） "JTPLGY_OPENMPI" （openmpiジョブ）
その他ジョブに設定する環境変数		

ユーザ PP スクリプトのカレントディレクトリは、ジョブ実行ユーザ（リクエスト所有者の実行ホスト上のユーザ）のホームディレクトリです。

ユーザ PP スクリプトは、その終了ステータスにより、実行の成否を判断します。

終了ステータスが 0 の場合、ユーザ PP スクリプトの実行は成功とみなします。

そうではない場合、ユーザ PP スクリプトの実行は失敗とみなします。

PRE-RUNNING でユーザ PP スクリプトの実行に失敗した場合、ジョブを実行開始せず、リクエストは QUEUED 状態に遷移します。

POST-RUNNING でユーザ PP スクリプトの実行に失敗した場合、後続の処理（ステージアウトなど）を実行し、リクエストは終了します

12.2 リクエスト投入時の指定方法

ユーザ PP スクリプトは、qsub(1),qlogin(1), qrsh(1)コマンドの--userpp-script オプションで指定してください。

```
--userpp-script=<loc>:<script_path>[,<loc>:<script_path>]
```

<loc>には、ユーザ PP スクリプトの実行位置を指定します。指定可能な実行位置は下記のとおり

です。

- prr　PRE-RUNNINGでユーザPPスクリプトを実行します。
- por　POST-RUNNINGでユーザPPスクリプトを実行します。
  <script_path>には、実行するスクリプトのパスを指定します。

以下に、ジョブ実行前にカレントディレクトリの test.sh を実行する場合の例を示します。

```
$ qsub --userpp-script=prr:test.sh
```

リクエスト投入時に指定したユーザ PP スクリプトは、qstat(1)コマンドの -f オプションで、確認できます。

【表示例】

```
$ qstat -f 123.bsv1
Request ID: 123.bsv1
    Request Name = STDIN
    User  Name = user1
:
    Attach = No
Cluster Type Select = NONE
UserPP Script = {
      Pre-running  = /home/user1/test.sh
    }
:
```

ユーザ PP スクリプトは qalter(1)の --userpp-script オプションで変更または削除できます。ただし、リクエストが実行開始した場合は、変更および削除はできません。

12.3 ユーザPPスクリプトの出力

ユーザ PP スクリプトの実行結果（標準出力と標準エラー出力）は、qcat(1)の--userpp-script オプションで参照できます。

また、バッチリクエストのみ、リクエストログのログレベルを 2 に設定することで、詳細な結果（実行結果、標準出力、標準エラー出力など）をリクエストログで参照できます。

リクエストログの出力例は下記のとおりです。

```
～省略～
----- UserPP Information -----
Location: PRE-RUNNING
Path: clhost:/home/user1/test.sh
===== Job 0 =====
Result: Success (exited 0)
Session ID: 8739
Stdout:
<ユーザPPスクリプトの標準出力の内容>
Stderr:
<ユーザPPスクリプトの標準エラー出力の内容>
```

Location は、ユーザ PP スクリプトを実行した位置（PRE-RUNNING または POST-RUNNING）です。

Path は、リクエスト投入時に指定したユーザ PP スクリプトの投入ホスト上のパスです。

Result は、ユーザ PP スクリプトの結果です。以下のいずれかとなります。

表 12-2 ユーザ PP スクリプトの結果

Resultの出力	意味
Success (exited 0)	ユーザPPスクリプトの実行が成功した。
Failure (exited+終了コード)	ユーザPPスクリプトの実行が失敗した。
Killed by signal+シグナル番号	シグナルにより終了した。
Deleted	ユーザPPスクリプト実行をqdelで中断した。
Timeout	タイムアウトにより終了した。
Not Running	ジョブサーバがLINKDOWNし、実行できなかった。
System Error	NQSVシステムエラーが発生した。

Session ID は、ユーザ PP スクリプトのプロセスのセッション ID です。(ユーザ PP スクリプトが実行した場合のみ)

Stdout, Stderr は、ユーザ PP スクリプトの標準出力、標準エラー出力の内容です。

第13章　OpenStack と連携したプロビジョニング環境を使用するリクエストの投入

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 では、OpenStack と連携したプロビジョニング環境下でジョブが使用する 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します。

リクエスト投入時、投入コマンド qsub(1), qlogin(1), qrsh(1) で、--template オプションにてこのテンプレートを指定することにより、OpenStack と連携したプロビジョニング環境を使用してジョブを実行することができます。

13.1 テンプレート情報の参照

システムで定義されているテンプレート情報は qstat(1)コマンドの --template オプション にて確認できます。

【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
    :
```

13.2 テンプレートを指定したリクエストの投入

リクエスト投入コマンド qsub(1), qlogin(1), qrsh(1) で、--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ごとの資源制限値（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（ジョブごと CPU 数制限、ジョブごとメモリサイズ制限、ジョブごと 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。

13.3 テンプレートを指定したリクエストの情報

13.3.1 リクエスト情報

リクエスト投入時に--template でテンプレートを指定した場合、qstat -f にて、以下のように表示されます。

【例】

```
$ qstat -f
Request ID: 3.bsvhost
    Request Name = STDIN
    User Name = user1
```

```
     :
  OpenStack Template = os_70_small
    CPU Number  = 1
    Memory Size = unused
    GPU Number  = 0
     :
Resources Information:
  Memory    = 0.000000B
     :
Resources Limits:
  (Per-Req) Elapse Time Limit      = Max:    1000S Warn:    1000S
     :
  (Per-Job) CPU Number             = Max:        1 Warn:      ---
     :
  (Per-Job) Memory Size            = Max:       1G Warn:       1G
     :
  (Per-Job) GPU Number             = Max:        0 Warn:      ---
     :
```

OpenStack Template にて、指定されたテンプレート情報が表示されます。

また、ジョブごとの資源制限値（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。

13.3.2 ジョブ情報

--template 指定により投入されたリクエストのジョブが仮想マシン上で実行している場合、qstat -J の ExcutionHost の項目の先頭に「*」を表示します。

この情報はリクエストが PRERUNNING～POSTRUNNING 状態で、仮想マシンが起動している間のみ表示します。

【例】

```
$ qstat -J
NO    RequestID       EJID   Memory    CPU     JSVNO ExecutionHost   UserName Exit
---- --------------- ----- -------- -------- ----- --------------- -------- ----
  0      3.bsvhost   100    0.00B    0.00       100        *exec1    user1    -
```

上記ジョブを qstat -Jf で詳細表示すると以下のように実行ホスト名の後ろに、仮想マシンのホスト名を表示します。

【例】

$qstat -Jf

```
Request ID: 3.bsvhost
    Batch Job Number = 0
    Execution Job ID = (none)
    User  Name = user1
    User  ID   = 111
    Group ID   = 111
    Job Server Number = 100
    Job Server Name = JobServer0100
    Execution Host = exec1(vmhost1)
       :
```

一方、--template 指定により投入されたリクエストのジョブがベアメタルサーバ上で実行している場合、qstat –J では、ExecutionHost の項目の先頭に"[B]"を表示します。

【例】

```
$ qstat -J
NO    RequestID       EJID   Memory     CPU    JSVNO ExecutionHost      UserName Exit
---- --------------- ----- -------- -------- ----- --------------- -------- ----
  0      4.bsvhost   100    0.00B        0.00  1000    [B]bhost    user1    -
```

上記ジョブを qstat -Jf で詳細表示すると以下のように実行ホスト名の後ろに、[Baremetal]を表示します。

【例】

```
$ qstat -Jf
Request ID: 4.bsvhost
    Batch Job Number = 0
    Execution Job ID = (none)
    User  Name = user1
    User  ID   = 111
    Group ID   = 111
    Job Server Number = 1000
    Job Server Name = JobServer1000
    Execution Host = bhost [Baremetal]
       :
```

第14章　Docker と連携したプロビジョニング環境を使用するリクエストの投入

NQSV では、Docker と連携したプロビジョニング環境下でジョブが使用する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します。

リクエスト投入時、投入コマンド qsub(1), qlogin(1), qrsh(1) で、--template オプションにてこのテンプレートを指定することにより、Docker と連携したプロビジョニング環境を使用してジョブを実行することができます。

14.1 テンプレート情報の参照

システムで定義されているテンプレート情報は qstat(1)コマンドの --template オプション にて確認できます。

【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
  VE Number    = 2
  HCA Number   = {
    For I/O = 0
    For MPI = 0
```

```
    For ALL = 1
  }
  Boot Timeout = 900
  Stop Timeout = 900
  Custom       = (none)
  Comment      = For App_A
  Requests     = 5
```

14.2 テンプレートを指定したリクエストの投入

リクエスト投入コマンド qsub(1), qlogin(1), qrsh(1) で、--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 batch1 -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ごとの資源制限値（CPU 数、メモリサイズ、GPU 数、VE 数、HCA ポート数）は、テンプレートで定義された CPU 数、メモリサイズ、GPU 数、VE 数、HCA ポート数になります。これらの値は、投入時にキューの資源制限値（ジョブごと CPU 数制限、ジョブごとメモリサイズ制限、ジョブごと 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。

14.3 テンプレートを指定したリクエストの情報

14.3.1 リクエスト情報

リクエスト投入時に--template でテンプレートを指定した場合、qstat -f にて、以下のように表示されます。

【例】

```
$ qstat -f
Request ID: 443.bsvhost
    Request Name = STDIN
    User Name = user1
       :
    Container Template = App_A
      CPU Number  = 2
```

```
    Memory Size = 1GB
    GPU Number  = 0
    VE  Number = 2
    HCA Number   = {
      For I/O = 0
      For MPI = 0
      For ALL = 1
    }
     :
Resources Information:
     :
Logical Host Resources:
  VE Node Number       = Max:        2 Warn:      ---
  CPU Number          = Max:        2 Warn:      ---
     :
Resources Limits:
  (Per-Req) Elapse Time Limit      = Max:    1000S Warn:    1000S
     :
  (Per-Job) CPU Resident Time      = Max: UNLIMITED Warn: UNLIMITED
  (Per-Job) CPU Number            = Max:        2 Warn:      ---
  (Per-Job) Tape Number           = Max: UNLIMITED Warn:      ---
     :
  (Per-Job) Process Number         = Max: UNLIMITED Warn:      ---
  (Per-Job) Memory Size            = Max:       1G Warn:       1G
  (Per-Job) Virtual Memory Size    = Max: UNLIMITED Warn: UNLIMITED
     :
  (Per-Job) GPID Number           = Max: UNLIMITED Warn:      ---
  (Per-Job) GPU Number            = Max:        0 Warn:      ---
  (Per-Prc) CPU Time              = Max: UNLIMITED Warn: UNLIMITED
     :
```

Container Template にて、指定されたテンプレート情報が表示されます。また、ジョブごとの資源制限値（CPU 数、メモリサイズ、GPU 数、VE 数、HCA ポート数）は、テンプレートで定義された CPU 数、メモリサイズ、GPU 数、VE 数、HCA ポート数になります。

14.3.2 ジョブ情報

--template 指定により投入されたリクエストのジョブがコンテナ内で実行している場合、qstat -J の ExcutionHost の項目の先頭に「*」を表示します。この情報はリクエストが PRERUNNING～POSTRUNNING 状態で、コンテナが起動している間のみ表示します。

【例】

```
$qstat -J
NO    RequestID       EJID   Memory        CPU JSVNO ExecutionHost   UserName Exit
---- --------------- ----- -------- -------- ----- --------------- -------- ----
  0    443.bsvhost   100    0.00B    0.00    100        *exec1    user1    -
```

上記ジョブを qstat -Jf で詳細表示すると以下のように実行ホスト名の後ろに、コンテナのホスト名を表示します。

【例】

```
$ qstat -Jf
Request ID: 443.bsvhost
   Batch Job Number = 0
   Execution Job ID = (none)
   User  Name = user1
   User  ID   = 111
   Group ID   = 111
   Job Server Number = 100
   Job Server Name = JobServer0100
   Execution Host = exec1(NQS-0-443-10)
     :
```

第15章　カスタムリソース機能を使用するリクエストの投入

カスタムリソース機能とは、定義されたカスタムリソース情報に基づき、スケジューリングにて同時に使用するカスタムリソースの利用量を制御する機能です。

カスタムリソース機能を使用している環境では、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、キューには、カスタムリソース利用量制限情報が設定されています。これは、各カスタムリソースについて、リクエストでの利用量の既定値、指定範囲、利用量制御対象外指定の可否が設定されています。

【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。

15.1 カスタムリソース情報の参照

システムで定義されているカスタムリソース情報は qstat(1)コマンドの --custom オプション にて確認できます。

【例】

```
$ qstat --custom
Custom Resource : Power
  Consumer = job
  Check Mode = Moment
  Terminate Job = Off
  Unit = W
  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 = Off
  Terminate Job = Off
  Unit = (none)
  Type = bsv :                          Available Resource Limit = 50

Custom Resource : Virtual
```

```
  Consumer = job
  Check Mode = Off
  Terminate Job = Off
  Unit = (none)
  Type = bsv :                          Available Resource Limit = 100
  Type = host: Target = (default)       Available Resource Limit = 1
```

また、各キューのカスタムリソース利用量制限情報については qstat -Qf にて確認できます。各カスタムリソースについて、リクエストでの利用量の指定範囲、既定値、利用量制御対象外指定の可否の情報が表示されます。

【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
      :
```

15.2 リクエスト投入時の指定方法

リクエスト投入コマンド qsub(1), qlogin(1), qrsh(1) でのカスタムリソースの指定方法の詳細は、以下です。

```
--custom cr_name=n[,cr_name=n…]
```

cr_name にはカスタムリソース名を、n にはその利用量を指定します。カンマで区切って（スペースなしで）複数のカスタムリソースの利用量を指定することができます。

n には 1～2147483647 の整数、または unused （または 0）が指定できます。unused （または 0）は、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。但し、投入先のキューで利用量制御対象外指定不可となっている場合、unused （または 0）の指定はできません。また、整数で指定された利用量が、投入先のキューに設定されている指定範囲を超えている場合は投入エラーとなります。

リクエスト投入時に、カスタムリソースの利用量を指定しなかった場合には、投入キューに設定されている既定値が適用されます。

【例】

```
$ qsub -custom Power=20,License=unused -q bq script
Request 6.bsv submitted to queue: bq.
```

15.3 リクエストのカスタムリソース利用量の参照

リクエストに指定したカスタムリソースの利用量は、qstat -f にて、確認できます。

【例】

```
$ qstat -f
Request ID: 6.bsv
    Request Name = STDIN
    User Name = user1
      :
  Custom Resources:
    Power           = 20 [W]
    License         = unused
    Virtual         = 1
  Execution Hosts(JSVNO):
    ehost100(100)
  Resources Information:
    Memory   = 1.932587MB
    CPU Time = 0.225019S
      :
```

また、リクエストのカスタムリソースの利用量は、ジョブの環境変数に、以下のように設定されます。2 種類の命名規則の環境変数が設定されますが両方とも同じ情報が入ります。これにより、ジョブ内、および、ユーザ PP スクリプト（詳細は 12.ユーザプリ・ポストスクリプトを指定するリクエストの投入 参照）にて、リクエストの全カスタムリソースの利用量を参照することができます。

NQSV_CR_<カスタムリソース名>=<利用量>

または

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
```

第16章　ハイブリッドリクエスト

16.1 ハイブリッドリクエストとは

通常のリクエストは一つ以上のジョブ（＝論理ホスト）を内包していました。しかし、複数のジョブが存在する場合でも、それぞれのジョブが使用できる資源量は同じでした。ハイブリッドリクエストでは内包するジョブ毎に異なる資源量を使用することができます。例えば、六つのジョブを持つリクエストが、ジョブ 0 とジョブ 1 は CPU×1＋VE×2、ジョブ 2 とジョブ 3 は CPU×1＋GPU×4、ジョブ 4 とジョブ 5 は CPU×8 を使用して実行する、ということができます。
なお、ハイブリッドリクエストで実行できる MPI は NEC MPI のみです。

16.2 ハイブリッドリクエストの投入

ハイブリッドリクエストでは一つのリクエスト内に二つ以上のジョブのグループが存在し、各グループは一つ以上のジョブを持ちます。一つのグループ内のジョブは全て同じ資源制限値を持ちます。異なるグループは異なる資源制限値を持つことができます。なお、グループが一つだけのリクエストは通常のリクエストとして扱われます。

通常のリクエストと同様にバッチリクエストは qsub、会話リクエストは qlogin または qrsh で投入します。これらのコマンドにジョブのグループ毎に異なるオプションを指定するため、--job-separator オプションを使用します（--job-separator オプションは---と略記することができます）。qsub を例にとると、下記のように記述します。

```
qsub 〈リクエスト全体に共通なオプション〉 --job-separator ¥
     〈グループ 0 に対するオプション〉 --job-separator ¥
     〈グループ 1 に対するオプション〉 --job-separator ¥
       :
     〈ジョブスクリプト〉
```

ジョブのグループ毎に異なる値を指定できるオプションは下記のとおりです。--job-separator で区切られた各グループに対する個別のオプションとして指定することができます。

オプション	意味
--cpunum-lhost	論理ホストごとの CPU 台数。
--cputim-lhost	論理ホストごとの CPU 時間。
--gpunum-lhost	論理ホストごとの GPU 台数。
--memsz-lhost	論理ホストごとのメモリサイズ。

オプション	意味
--venum-lhost	論理ホストごとの VE ノード数。
--vmemsz-lhost	論理ホストごとの仮想メモリサイズ。
--use-hca	同一 PCIeSW 内の VE が使用する、HCA のポート数。
--venode	VE ノード総数。
-l cpunum_job	ジョブ（論理ホスト）ごとの CPU 台数。
-l cputim_job	ジョブ（論理ホスト）ごとの CPU 時間。
-l gpunum_job	ジョブ（論理ホスト）ごとの GPU 台数。
-l memsz_job	ジョブ（論理ホスト）ごとのメモリサイズ。
-l vmemsz_job	ジョブ（論理ホスト）ごとの仮想メモリサイズ。
-l coresz_prc	プロセスごとのコアファイルサイズ。
-l cputim_prc	プロセスごとの CPU 時間。
-l datasz_prc	プロセスごとのデータセグメントサイズ。
-l filenum_prc	プロセスごとの同時オープンファイル数。
-l filesz_prc	プロセスごとの生成ファイルサイズ。
-l stacksz_prc	プロセスごとのスタックサイズ。
-l vecputim_prc	VE プロセスごとの CPU 時間
-l vememsz_prc	VE プロセスごとのメモリサイズ
-l vmemsz_prc	プロセスごとの仮想メモリサイズ。
-b	ジョブ数。
-B	ジョブコンディション。
-v	実行時に使用する環境変数。
-y	予約区間 ID（※）

（※）ハイブリッドリクエストを予約区間に投入する場合は、全てのジョブに対して予約区間 ID を指定してください。一部のジョブのみ予約区間 ID を指定し、その他のジョブは予約区間 ID を指定しないハイブリッドジョブの投入はできません。

上記のオプションは--job-separator で区切られた各グループに対するオプションの領域に記述することができます。また、リクエスト全体に共通なオプションの領域に記述することもできます。その場合は、各グループにオプションが指定されなかった場合の既定値として使用されます。

上記以外のオプションはリクエスト全体に共通なオプションの領域にのみ記述できます。各グループに対するオプションの領域に記述すると、エラーになります。

具体例を挙げます。例の中で背景がオレンジ色部分はリクエスト全体に共通なオプションの領域、背景が水色部分は各グループに対する個別のオプションの領域を表します。

例 1：

```
qsub -q exeq -l elapstim_req=600 -Tnecmpi -b2 ¥
--job-separator --venum-lhost=2 --cpunum-lhost=1 --use-hca=1 ¥
--job-separator --cpunum-lhost=1 --gpunum-lhost=4 ¥
--job-separator --cpunum-lhost=8 ¥
job_script
```

この例ではジョブスクリプト job_script をキューexeq に投入し、最大実行時間は 600 秒、ジョブトポロジーは necmpi です。各ジョブのグループに対する個別オプションとしてジョブ数（-b）が指定されていないため、共通なオプションとして指定されている-b2 が全てのジョブのグループに適用されます。

よって、この例では、ジョブのグループとして以下の 3 種類が投入されます。

- CPU台数×1＋VE台数×2＋HCA×１のジョブが二つ
- CPU台数×1＋GPU台数×4のジョブが二つ
- CPU台数×8のジョブが二つ

例 2：

```
qsub -q exeq -l elapstim_req=600 --cpunum-lhost=2 -Tnecmpi ¥
--job-separator --memsz-lhost=1GB ¥
--job-separator --venum-lhost=3 --use-hca=1 -b4 ¥

job_script
```

この例ではジョブスクリプト job_script をキューexeq に投入し、最大実行時間は 600 秒、ジョブトポロジーは necmpi です。各ジョブのグループに対する個別オプションとして論理ホスト毎の CPU 台数（--cpunum-lhost）が指定されていないため、共通なオプションとして指定されている--cpunum-lhost=2 が全てのジョブのグループに適用されます。

よって、この例では、ジョブのグループとして以下の 2 種類が投入されます。

- CPU台数×2＋最大メモリ1GBのジョブが１つ
- CPU台数×2＋VE台数×3＋HCA×1のジョブが４つ

例 3：

```
qsub -q exeq -l elapstim_req=600 ¥
--cpunum-lhost=2 -Tnecmpi --venum-lhost=2 -use-hca=1 -b4 ¥
job_script
```

この例ではジョブスクリプト job_script をキューexeq に投入し、最大実行時間は 600 秒、ジョブトポロジーは necmpi です。このリクエストには--job-separator が指定されていないため、ハイブリッドリクエストではない通常のリクエストです。

よって、この例では、CPU×2＋VE×2 のジョブが４つ投入されます。

例 4：

```
qsub -q exeq -l elapstim_req=600 --cpunum-lhost=1 -Tnecmpi ¥
--job-separator --cpunum-lhost=4 --venum-lhost=1 --use-hca=1 ¥
--job-separator --use-hca=1 --venode=8 ¥
job_script
```

この例ではジョブスクリプト job_script をキューexeq に投入し、最大実行時間は 600 秒、ジョブトポロジーは necmpi です。共通なオプションにジョブ数が指定されておらず、2 つ目のジョブのグループには--venode オプションが指定されていることから、2 つ目のジョブのグループの VE 数とジョブ数はキューexeq に設定されている既定搭載 VE ノード数に依存します。

よって、この例では、ジョブのグループとして以下の 2 種類が投入されます。

- CPU台数×4＋VE台数×1＋HCA×1のジョブが一つ

- CPU台数×1＋HCA×1で、VE台数とジョブ数はキューexeqに設定されている既定搭載VEノード数に依存

例 5：

```
qsub -q exeq -Tnecmpi ¥
--job-separator --cpunum-lhost=4 --venum-lhost=1 --use-hca=1 ¥
                -l  elapstim_req=300 ¥
--job-separator --cpunum-lhost=1 -b2 ¥
job_script
```

この例では-l elapstim_req が一つ目のジョブのグループに対するオプションの領域に記述されているため、投入エラーとなります。

ハイブリッドリクエストを同時実行リクエスト（qsub コマンドの--parallel オプションを用いて投入したリクエスト）として投入した場合は、当該リクエストはスケジューリングされません。

16.3 ハイブリッドリクエスト情報の参照

通常の同様に qstat -f でリクエスト情報の詳細が表示されます（パラメトリックリクエストの場合は qstat -Rf も可能です）。このうち下記の項目はハイブリッドリクエストについてはジョブ 0 の値が表示されます。

ジョブグループ番号		Job Group Number
予約区間 ID		Reservation ID
VE ノード総数		VE Node Number
HCA ポート数		HCA Number
論理ホストごとの資源制限値情報		Resources Limits per Logical Host
	VE ノード数	VE Node Number
	CPU 台数	CPU Number
	GPU 台数	GPU Number

ジョブグループ番号		Job Group Number
	CPU 時間	CPU Time
	メモリサイズ	Memory Size
	仮想メモリサイズ	Virtual Memory Size
	VE CPU 時間	VE CPU Time
	VE メモリサイズ	VE Memory Size
プロセスごとの資源制限値情報		Resources Limits per Process
	CPU 時間	CPU Time
	同時オープンファイル数	Open File Number
	仮想メモリサイズ	Virtual Memory Size
	データセグメントサイズ	Data Segment Size
	スタックサイズ	Stack Segment Size
	コアファイルサイズ	Core File Size
	生成ファイルサイズ	Permanent Files Size
	VE CPU 時間	VE CPU Time
	VE メモリサイズ	VE Memory Size

VE ノード総数（VE Node Number）は、各ジョブのグループに対して指定された--venode の値（VE ノード総数）の合計を表示します。ジョブ番号 0 のジョブが属するジョブグループに対して指定された--venode の値ではないことにご注意ください。

ジョブ 0 以外の値を参照するには qstat -Jt で資源制限情報を表示してください。表示例を以下に示します。

```
JNO         RequestID              VE       CPU       GPU   CPUTime   Memory  VMemory
----------- --------------- -------- -------- -------- --------- -------- --------
0           117.host.exampl         1        1         0      ULIM     ULIM     ULIM
1           117.host.exampl         0        4         0      ULIM       1G     ULIM
2-3         117.host.exampl         0        1         1      ULIM     ULIM     ULIM
```

資源制限情報として上記の項目が表示されます。ハイブリッドでない通常のリクエストが持つジョブについても上記の情報は表示されますが、その場合は一つのリクエストが持つジョブは全て同じ値が表示されます。

また、-f と-t を同時に指定することもできます。この場合は資源制限情報を詳細な形で表示されます。

```
$ qstat -Jft
Request ID: 117.host.example.com
    Batch Job Number = 0
    HCA Number = (none)
    Reservation ID = (none)
  Resources Limits per Logical Host:
    VE Node Number         = Max:          1 Warn:        ---
    CPU Number             = Max:          1 Warn:        ---
    GPU Number             = Max:          0 Warn:        ---
    CPU Time               = Max: UNLIMITED Warn: UNLIMITED
    Memory Size            = Max: UNLIMITED Warn: UNLIMITED
    Virtual Memory Size    = Max: UNLIMITED Warn: UNLIMITED
    VE CPU Time            = Max: UNLIMITED Warn: UNLIMITED
    VE Memory Size         = Max: UNLIMITED Warn: UNLIMITED
  Resources Limits per VE Node:
    VE CPU Time            = Max: UNLIMITED Warn: UNLIMITED
    VE Memory Size         = Max: UNLIMITED Warn: UNLIMITED
  Resources Limits per Process:
    CPU Time               = Max: UNLIMITED Warn: UNLIMITED
    Open File Number       = Max: UNLIMITED Warn:        ---
    Memory Size            = Max: UNLIMITED Warn: UNLIMITED
    Virtual Memory Size    = Max: UNLIMITED Warn: UNLIMITED
    Data Segment Size      = Max: UNLIMITED Warn: UNLIMITED
    Stack Segment Size     = Max: UNLIMITED Warn: UNLIMITED
    Core File Size         = Max: UNLIMITED Warn: UNLIMITED
    Permanent File Size    = Max: UNLIMITED Warn: UNLIMITED
    VE CPU Time            = Max: UNLIMITED Warn: UNLIMITED
    VE Memory Size         = Max: UNLIMITED Warn: UNLIMITED
```

16.4 ハイブリッドリクエストの属性変更

通常のリクエストと同様にハイブリッドリクエストの属性変更も qalter コマンドで行います。ジョブ毎の資源制限値のようにジョブのグループ毎に異なる値を持つ属性を変更する場合は、下記のようにリクエスト ID に変更するジョブ番号を付加して指定します。

```
[job_number:]<request_identifier>
```

ここで指定されたジョブ番号と同じグループに属する全てのジョブで対象属性値が同様に変更されます。

例えば、117.host.example.com というリクエスト ID を持つリクエストのジョブ 1 に対して VE ノード数制限値を 4 に変更する場合は、下記のように実行します。この時、ジョブ 1 と同じグループに属する全てのジョブの VE ノード数制限値も同じ値に変更されます。

```
$ qalter --venum-lhost=4 1:117.host.example.com
Attribute of Request is altered.
```

ジョブ番号は省略可能です。省略した場合は、全てのジョブに対して変更します。また、リクエスト全体で共通な値を持つ属性を変更する場合は、ジョブ番号を省略しなければなりません。この場合に、ジョブ番号を指定すると、エラーが発生します。

第17章　singularity と連携したプロビジョニング環境を使用するリクエストの投入

NQSV では、singularity と連携してジョブの実行ができます。singularity イメージファイルを事前に用意して、ジョブ実行時にジョブスクリプトの中に singularity コマンドを呼び出してイメージファイルやジョブプログラムを指定することで singularity コンテナによるジョブは実行可能です。

NQSV では、利用可能なバージョンは、singularity version 3.6.4 となります。サポートする MPI は、NEC MPI と OpenMPI 4.0.3 です。

NEC MPI を利用する singularity コンテナのビルド方法については、下記のページを参照してください。

https://github.com/veos-sxarr-NEC/singularity

17.1 ジョブ実行方法

17.1.1 MPI ジョブの実行

NEC MPI の場合

NEC MPI ジョブを singularity コンテナにて実行する場合、ジョブスクリプトには下記のように記載します。

```
source /opt/nec/ve/mpi/<NEC MPIバージョン>/bin64/necmpivars.sh
mpirun –np <並列数> singularity exec --bind /var/opt/nec/ve/veos <コンテナイメージ名> <実行プログラム名>
```

ジョブスクリプトの例：

```
#!/bin/bash
#PBS -q bq
#PBS -T necmpi
#PBS -b 2
#PBS -l elapstim_req=300
#PBS --venum-lhost=8 --cpunum-lhost=2

source /opt/nec/ve/mpi/2.11.0/bin64/necmpivars.sh
mpirun -np 16 /usr/bin/singularity exec --bind /var/opt/nec/ve/veos ~/necmpi.sif ~/a.out
```

qlogin(1)や qrsh(1)で会話リクエストを実行する場合は、バッチリクエストと同様なイメージで singularity コンテナによる NEC MPI ジョブの実行が可能です。

OpenMPI の場合

OpenMPI ジョブを singularity コンテナにて実行する場合、ジョブスクリプトには下記のように記載します。

```
mpirun –np <並列数> singularity exec <コンテナイメージ名> <実行プログラム名>
```

ジョブスクリプトの例：

```
#!/bin/bash
#PBS -q bq
#PBS -T openmpi
#PBS -b 2
#PBS -l elapstim_req=300
#PBS --cpunum-lhost=2

mpirun -np 2 /usr/bin/singularity exec ~/openmpi.sif ~/a.out
```

qlogin(1)や qrsh(1)で会話リクエストを実行する場合は、バッチリクエストと同様なイメージで singularity コンテナによる OpenMPI ジョブの実行が可能です。

17.1.2 分散ジョブの実行

分散ジョブを singularity コンテナにて実行する場合、ジョブスクリプトには下記のように記載します。

```
singularity exec <コンテナイメージ名> <実行プログラム名>
```

ジョブスクリプトの例：

```
#!/bin/bash
#PBS -q bq
#PBS -T distribute
#PBS -b 2
#PBS -l elapstim_req=300
```

```
#PBS --cpunum-lhost=2


/usr/bin/singularity exec ~/distribute.sif ~/a.out
```

qlogin(1)や qrsh(1)で会話リクエストを実行する場合は、バッチリクエストと同様なイメージで singularity コンテナによる分散ジョブの実行が可能です。

第18章　制約事項

18.1 NQSVシステムで使用する番号などの範囲・最大値

NQSV システムで使用する番号や指定値などについての、指定可能な値の範囲や、最大値、最大長は以下となっています。これらについては、nqsv.h にて、定数として定義されているものもあります。

項目	属性	値
ジョブサーバ	1つのバッチサーバに接続できるジョブサーバ最大値	10240
	指定可能なジョブサーバ番号の範囲	0～10239
	定数名 NQS_MAX_JSVNO	10239 （ジョブサーバ番号の最大値）
	指定可能なジョブサーバ名の最大長	15
	定数名 NQS_LEN_JSVNAME	15
スケジューラ （スケジューラの詳細は 、[JobManipulator 編]参照）	指定可能なスケジューラ番号の範囲	0～15
	定数名 NQS_MAX_SCHNO	15 （スケジューラ番号の最大値）
	指定可能なスケジューラ名の最大長	15
	定数名 NQS_LEN_SCHNAME	15
ホスト名	NQSV 内で指定可能なホスト名の最大長	255
	定数名 NQS_LEN_HOSTNAME	255
ノードグループ	作成できるノードグループの数	2048
	指定可能なノードグループ名の最大長	15
	定数名 NQS_LEN_NGRPNAME	15
	指定可能なコメントの最大長	63
	定数名 NQS_LEN_COMMENT	63
リクエスト名	NQSV 内で指定可能なリクエスト名の最大長	63
	定数名 NQS_LEN_REQNAME	63
ジョブ番号	ジョブ番号の範囲	0～10239
	定数名 NQS_MAX_JOBNO	10239 （ジョブ番号の最大値）
ユーザ名	NQSV 内で指定可能なユーザ名の最大長	47

項目	属性	値
ジョブサーバ	1つのバッチサーバに接続できるジョブサーバ最大値	10240
	指定可能なジョブサーバ番号の範囲	0～10239
	定数名 NQS_MAX_JSVNO	10239 （ジョブサーバ番号の最大値）
	指定可能なジョブサーバ名の最大長	15
	定数名 NQS_LEN_JSVNAME	15
スケジューラ （スケジューラの詳細は 、[JobManipulator 編]参照）	指定可能なスケジューラ番号の範囲	0～15
	定数名 NQS_MAX_SCHNO	15 （スケジューラ番号の最大値）
	指定可能なスケジューラ名の最大長	15
	定数名 NQS_LEN_SCHNAME	15
	定数名 NQS_LEN_USERNAME	47
グループ名	NQSV 内で指定可能なグループ名の最大長	47
	定数名 NQS_LEN_GROUPNAME	47
キュー名	NQSV 内で指定可能なキュー名の最大長	15
	定数名 NQS_LEN_ QUENAME	15
パス名	NQSV 内で指定可能なパス名の最大長	1023
	定数名 NQS_LEN_PATHNAME	1023
ファイル名	NQSV 内で指定可能なファイル名の最大長	255
	定数名 NQS_LEN_FILENAME	255
アカウントコード	NQSV 内で指定可能なアカウントコードの最大長	15
メールアドレス	NQSV 内で指定可能なメールアドレスの最大長	1023
	定数名 NQS_LEN_MAILADDR	1023
ジョブ実行条件式（ジョブコンディション）	NQSV 内で指定可能なジョブコンディションの最大長	255
	定数名 NQS_LEN_JOBCOND	255
テンプレート	指定可能なテンプレート名の最大長	47
	定数名 NQS_LEN_TEMPLATENAME	47
	指定可能なテンプレートの OS イメージ名の最大長	47
	定数名 NQS_LEN_VMIMGNAME	47
	指定可能なテンプレートのフレーバ名の最大長	47

項目	属性	値
ジョブサーバ	1つのバッチサーバに接続できるジョブサーバ最大値	10240
	指定可能なジョブサーバ番号の範囲	0～10239
	定数名 NQS_MAX_JSVNO	10239 （ジョブサーバ番号の最大値）
	指定可能なジョブサーバ名の最大長	15
	定数名 NQS_LEN_JSVNAME	15
スケジューラ （スケジューラの詳細は 、[JobManipulator 編]参照）	指定可能なスケジューラ番号の範囲	0～15
	定数名 NQS_MAX_SCHNO	15 （スケジューラ番号の最大値）
	指定可能なスケジューラ名の最大長	15
	定数名 NQS_LEN_SCHNAME	15
	定数名 NQS_LEN_FLAVORNAME	47
	指定可能なテンプレートの独自定義の最大長	400
	定数名 NQS_LEN_TEMPLATECUSTOM	400
	指定可能なテンプレートのコメントの最大長	255
	定数名 NQS_LEN_TEMPLATECOMMENT	255
カスタムリソース	1つのバッチサーバで定義できるカスタムリソースの最大数	20
	定数名 NQS_MAX_CRNUM	20
	指定可能なカスタムリソース名の最大長	15
	定数名 NQS_LEN_CRNAME	15
	指定可能なカスタムリソースの単位の長さ	5
	定数名 NQS_LEN_CRUNIT	5

18.2 コマンドとバッチサーバのバージョン

ユーザコマンド、管理者コマンドでバッチサーバに接続して参照、設定などの操作行う際、コマンドとバッチサーバのバージョンを合わせてください。

接続するバッチサーバのバージョンが、コマンドの NQSV/Client のバージョンより古い場合は、コマンドの実行中にエラーが発生する場合があります。

18.3 ユーザレベルチェックポイント

チェックポイントは実行中のジョブの内部状態を保存しておく機能です。対になるリスタート機能によって保存しておいたジョブを再開することができます。ユーザレベルチェックポイントはユーザプロセスによってチェックポイントの保存を行います。NQSV ではユーザレベルのチェックポイントを使用するためのサポート機能を提供します。

注意：本機能は試験的な機能です。使用する場合は NEC サポートにご相談ください。

18.3.1 ユーザレベルチェックポイントを使用するための準備

チェックポイントをとるためのスクリプトはユーザ側で作成します。NQSV には含まれません。このスクリプトをチェックポイントスクリプトと呼びます。チェックポイントスクリプトは実行ホストに配置することが必要です。また、全ての実行ホストにおいて同じパスで参照できる必要があります。

18.3.2 リクエスト投入の仕方

チェックポイントをとりたいリクエストは、qsub で投入する際にチェックポイントスクリプトを指定する必要があります。チェックポイントスクリプトは--use-custom-suspend オプションで指定します。書式は以下のとおりです。

```
--use-custom-suspend=chkpnt:script
```

script にチェックポイントスクリプトを実行ホスト上の絶対パスで指定します。例えば、/home/user1/checkpoint.sh を指定する場合は下記のように指定します。

```
--use-custom-suspend=chkpnt:/home/user1/checkpoint.sh
```

詳細は NEC Network Queuing System V (NQSV) 利用の手引［リファレンス編］の qsub の項目を参照してください。

18.3.3 チェックポイントをとる場合

チェックポイントをとるタイミングは、通常リクエストが緊急リクエストまたは特別リクエストに割り込まれた場合です。緊急リクエスト、特別リクエストについては NEC Network Queuing System V (NQSV) 利用の手引［JobManipulator 編］の第 4 章　高度なスケジューリング機能を参照してください。割り込まれた通常リクエストが、--use-custom-suspend オプションでチェックポイントスクリプトを指定されている場合、NQSV はこのチェックポイントスクリプトを実行します。実

行後、割り込まれた通常リクエストは終了します。

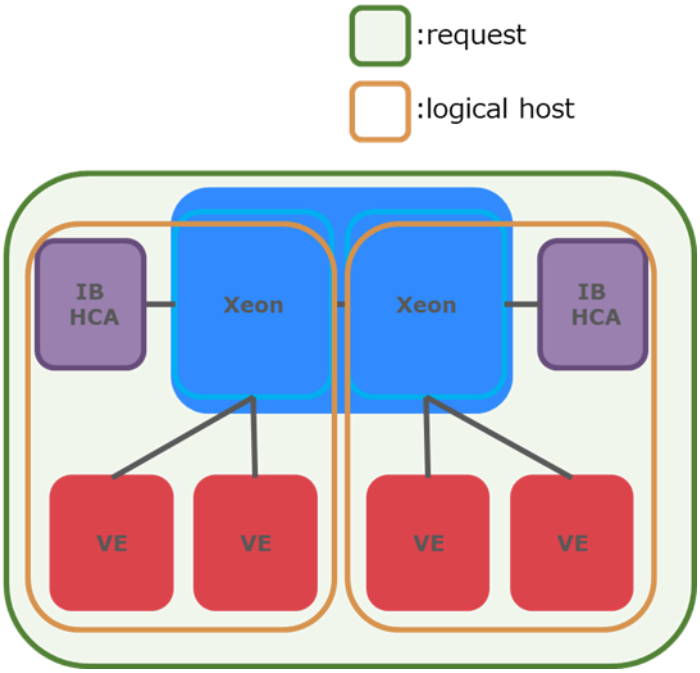
付録 A　NQSV リクエストの投入方法

A.1　VEを使用したリクエスト

VE を 4 台使用するリクエストのジョブスクリプトの例を以下に示します。

```
#PBS -b 2
#PBS -T necmpi
#PBS -l elapstim_req=300
#PBS --cpunum-lhost=2
#PBS --memsz-lhost=10gb
#PBS --use-hca=1
#PBS --venum-lhost=2

mpirun -nn 2 -np 8 ./ve.out
```

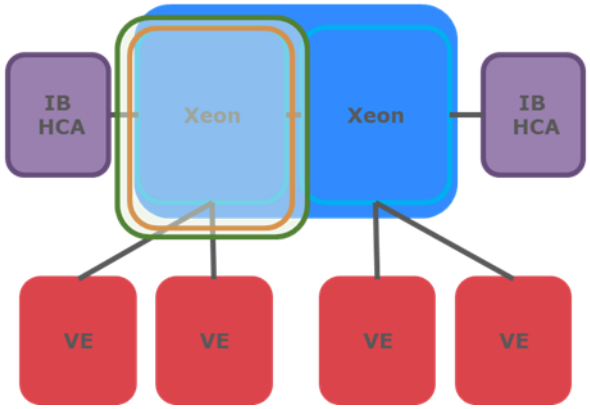


A.2　x86を使用したリクエスト

VH の x86 CPU のリソースのみを使用するリクエストのジョブスクリプトの例を以下に示します。

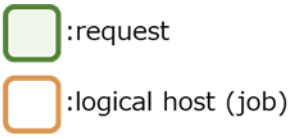
```
#PBS -b 1
#PBS -l elapstim_req=300
#PBS --cpunum-lhost=8
#PBS --memsz-lhost=10gb

export OMP_NUM_THREADS="8"
./vh.out
```



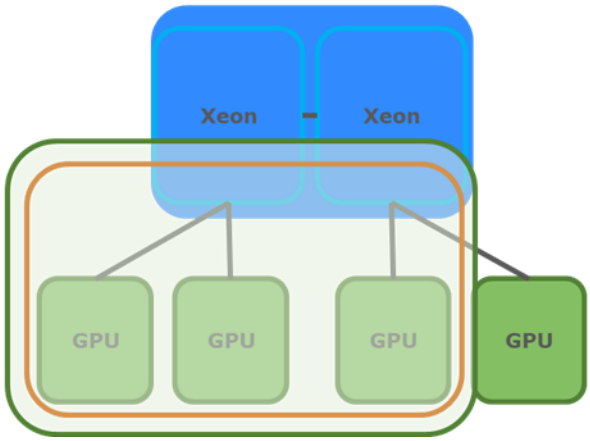
A.3　GPUを使用したリクエスト

GPGPU のリソースを使用するリクエストのジョブスクリプトの例を以下に示します。




```
#PBS -b 1
#PBS -l elapstim_req=300
#PBS --cpunum-lhost=8
#PBS --gpunum-lhost=3

./cuda.out
```

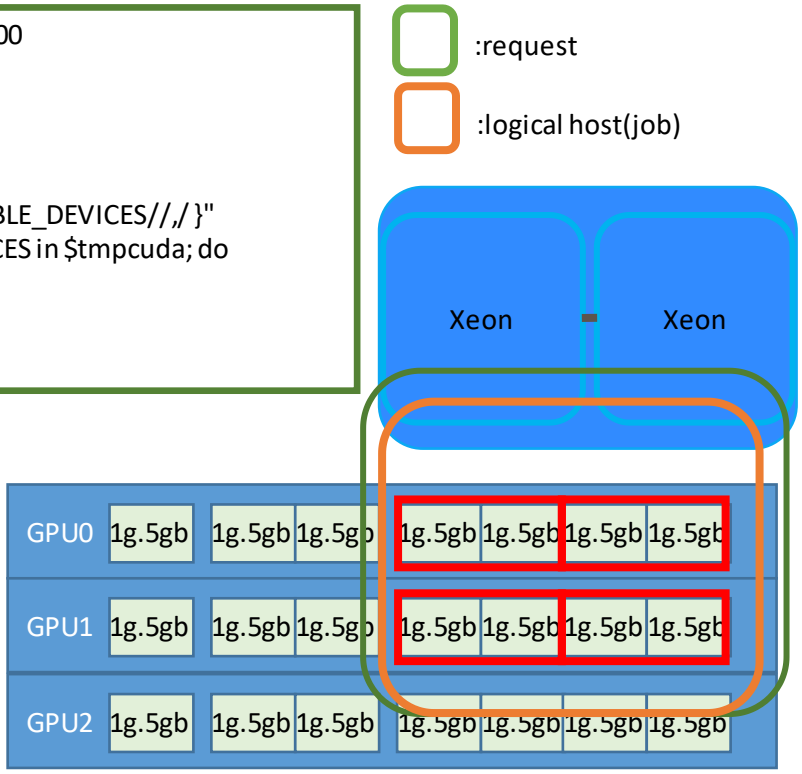


A.4　マルチインスタンスGPUを使用したリクエスト

マルチインスタンス GPU のリソースを使用するリクエストのジョブスクリプトの例を以下に示します。

```
#PBS -l elapstim_req=1000
#PBS --cpunum-lhost=8
#PBS --mig=4g.20gb:2,2
#PBS --mig=4g.20gb:2,2

tmpcuda="${CUDA_VISIBLE_DEVICES//,/ }"
for CUDA_VISIBLE_DEVICES in $tmpcuda; do
   cuda.out &
done
```



付録 B　発行履歴

B.1　発行履歴一覧表

2018 年	2 月	初版
2023 年	1 月	第 16 版
2023 年	3 月	第 17 版
2023 年	5 月	第 18 版
2023 年	6 月	第 19 版
2025 年	1 月	第 20 版

B.2　追加・変更点詳細

第 16 版

- 11章 マルチインスタンスGPU(MIG)を使用するリクエストの投入を追加
- 付録A.4 マルチインスタンスGPUを使用したリクエストを追加

第 17 版

- 1章 1.2のバッチリクエストの投入 資源制限値の指定に標準出力の結果ファイルサイズ、および、標準エラー出力の結果ファイルサイズを追加
- 1章 1.3のバッチリクエストの状態確認 詳細情報の確認に標準出力の結果ファイルサイズ、および、標準エラー出力の結果ファイルサイズを追加

第 18 版

- 1章 1.2.9のバッチリクエストの投入 資源制限値の指定にOpenMPと組み合わせたMPIを実行する場合の注意事項を追加
- 1章 1.2.16のバッチリクエストの投入 環境変数の利用における-vオプションで変更できる環境変数の説明に関する誤りを修正
- 1章 1.14.2のMPIリクエストの実行 OpenMPI環境での実行、および、1章 1.14.3のMPIリクエ

ストの実行 IntelMPI環境での実行に、OpenMPと組み合わせた実行に説明を追加

- 1章 1.14.3のMPIリクエストの実行 IntelMPI環境での実行に、NQSVのソケットスケジューリング機能が有効な場合の注意事項を追加

第 19 版

- 1章 1.2.16のバッチリクエストの投入 環境変数の利用に-Vまたは-vオプションでジョブに渡せない環境変数のリストを追加

第 20 版

- 1章 1.14のMPIリクエストのOpenMPIのサポートバージョンを更新

- 1章 1.14.2のMPIリクエスト OpenMPI環境での実行、および、3章 3.6.2の会話リクエストでのMPIプログラムの実行 OpenMPI環境での実行に、OpenMPI使用時の注意事項を追加

索　引

B

BMC vii

D

Docker 109

G

GPU 100

H

HCA ..... vii, 9

I

IB vii

M

MPI vii

N

NIC vii

O

OpenStack 105

Q

qlogin 56

V

VE ..... vii, 8
VI クラスター vii

か

会話リクエスト 56

し

出力情報のカスタマイズ 30, 69, 76
出力情報のソート 82
情報の表示 75
ジョブ 67

ね

ネットワークリクエスト 72

は

パラメトリックリクエスト 49

ふ

プロビジョニング 105, 109

へ

ベクトルエンジン vii
ベクトルホスト vii

め

メール送信機能 53

ゆ

ユーザ PP スクリプト 101
ユーザプリ・ポストスクリプト 101

り

リクエストのグループ 95
リクエストへのアタッチ 47
リクエスト連携 22
リラン 37

わ

ワークフロー 87
ワークフロースクリプト 88

NEC Network Queuing System V (NQSV)

利用の手引[操作編]

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）

© NEC Corporation 2018

日本電気株式会社の許可なく複製・改変などを行うことはできません。

本書の内容に関しては将来予告なしに変更することがあります。