令和 5 年度　グローバルゼミ・論文紹介　前刷　　令和 5 年 12 月 21 日

担当学生　谷澤　悠太
指導教員　滝沢　寛之 教授

1. 背景

HPC は現在，機械学習や数値シュミレーション，統計解析など様々な科学分野で地擁されている．そのため，HPC を専門としない科学者が HPC を利用する事例や情報系学生などが HPC を用いた並列計算を学習する際に利用するといった事例が多く存在する．しかしながら，このような事例の場合，HPC 利用のための環境構築や鍵認証のためのセットアップが諸学者にとって複雑であることや CLI ベースでスケジューラなどに命令を出すことの難易度の高さなどが利用者にとって障害となる．このことは，HPC を使いこなすための学習コストの大きさにより，本来の研究に費やすコストが減少してしまうため問題となっている．

このような問題に対して，米国オハイオ・スーパーコンピューティングセンターは HPC 利用者の支援を目的とする『Open OnDemand』（OOD）と呼ばれるオープンソースソフトウェアを開発した．[1][2][3]OOD は Web ポータル上から HPC システムの利用を可能としており，CLI ではなく GUI ベースで操作することが可能なため，初学者に寄り添った HPC 利用環境を提供することができる．また，ユーザ情報の登録によるシングルサインオン認証を用いることで，誰でも環境構築を行わずに利用することが可能となっている．開発当時の OOD は SLURM，Torque，PBS Pro，LSF などのジョブスケジューラに対応していたため，現在 OOD を導入している計算センターも多く存在する．しかしながら，対応していないスケジューラも多く，使える環境が限定されているといえる．

そこで先行事例として，OOD を Fujitsu_TCS（スーパーコンピュータ富岳で運用されているジョブスケジューラ）へ対応させたことによる『富岳』での OOD 利用という事例を考える．[4] これにより，『富岳』の利用者は HPC システムの視覚的理解が容易になった，利用難易度が低下した，などのメリットを得ることができた．また，Fujitsu_TCS のアダプタの開発により，Fujitsu_TCS を利用している他の計算センターでも OOD の利用が可能となったというメリットもある．

2. 目的

本研究では『ウェブインタフェースを介したスーパーコンピュータ利用環境に関する研究』という題目を研究テーマとして考えていく．具体的には，東北大学で用いられているスーパーコンピュータ『AOBA』の利用環境について考える．前述の通り，OOD はいくつかのジョブスケジューラに対応しているが，『AOBA』に用いられているジョブスケジューラ NQSV（Network Queuing System V）に対応するためのアダプタが存在しない．そのため，NQSV のアダプタを実装することで AOBA 利用者の支援を目的とする．

3. 実装手順

具体的な実装について，大きく 4 つの手順に分けて紹介する．

まず, 1 つ目の手順として SLURM スケジューラを試用している sendai サーバで OOD の動作確認を行う．この手順で OOD がどのようなものか理解したり，システムの設計について調査し，具体的な実装手順を見据える．

2 つ目の手順として，SLURM 用に書かれたアダプタファイルを自分で書き換えることで OOD にその変更内容が反映されるかどうかを確認する．

3 つ目の手順として，tokyo，paris，london の 3 つのサーバに NQSV をインストールして AOBA のテスト環境を構築する．

4 つ目の手順として，他のスケジューラのアダプタを参考にして NQSV のアダプタを作成する．このとき，NQSV と親和性の高い PBS Pro や動作確認で試用した SLURM，アダプタ作成例が紹介されている Fujitsu_TCS のアダプタなどを参考にする．

最後の手順として，手順 2 や手順 3 の過程からアダプタの構造理解やその汎用性などの検討を行う．アダプタやスケジューラの理解を深めることで，様々なスケジューラに対応できる汎用性の高いアダプタについて考察したい．

4. 進捗

4.1. テスト環境構築手順

現在の進捗として，手順 3 で示した AOBA のテスト環境構築が終了している．以降にテスト環境の詳細を示す．

テスト環境の構築は大きく 3 つのステップで手順が実行さてていく．1 ステップ目が OS のインストール，2 ステップ目が 3 つのサーバへの NQSV のインストール，3 つ目が各コンポーネントの設定である．

4.2. OS インストール

使用する tokyo，london，paris に Ubuntu22.04.3 LST が入っていたが NQSV は Ubuntu に対応していないため，NQSV の動作確認が保証されている CentOS7 を導入した．OS のインストールには boot 起動用の USB を作成し，各種設定を行った．

最初に，riken のミラーサイトから CentOS7 の実体をインストールして rufus というアプリケーションを用いて boot 起動用の USB を作成した．各種設定として，言語選択，時刻選択，Software 選択，インストール先選択，root ユーザのパスワード設定などを行った．また，ネットワークとホスト名設定に関して，もともと各サーバに割り当てられていら IP アドレスとホスト名を設定した．また，研究室で用いている AD との連携，ファイルシステムの同期が必要だったため，localwiki を参考に，AD との連携と，ciero のホームディレクトリと連携するための NFS の設定を行った．

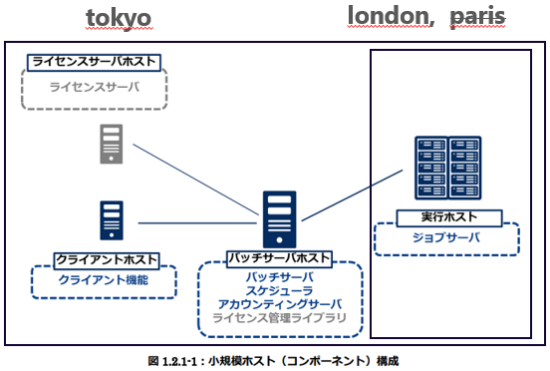

図 1. AOBA テスト環境のホスト構成

4.3. NQSV のインストール

AOBA テスト環境のホスト構成を図 1 に示す．今回のテスト環境では tokyo をマスターノードとしてスケジューラや NQSV 全体の管理を担うものとする．また，tokyo，london，paris に実行ホストをおいてジョブの割り当てを行う．

ここでホスト構成に示した各ホストの役割を説明する．ライセンスサーバホストは，ライセンス製品の利用に必要なライセンスの管理を行う．バッチサーバホストにはバッチサーバ，スケジューラ，アカウンティングサーバが置かれ，バッチサーバが NQSV の中心コンポーネントであり，スケジューラがジョブリクエストの管理を行い，アカウンティングサーバではアカウント管理や，予算管理を行う．また，クライアントホストはリクエストの投入や NQSV 全体の管理操作を行う．これらの 3 つのホストは今回のテスト環境ではすべて tokyo 上に設置することとする．また，各サーバに置く実行ホストは，各サーバのジョブ実行の管理をする．

4.3.1. ライセンスサーバの設定

JobServer（ジョブサーバ）と JobManipulator（スケジューラ）の仕様にライセンスが必要であるため，まずはじめに tokyo のライセンスサーバの設定を行う．ライセンスサーバ（aurlic-lib）のインストールを行った後，ファイヤーウォールの設定でポート 7300 番を開けておく．その後ライセンスサーバを起動する．

また，ライセンスのクライアント側として，今回使用するすべてのマシン上にライセンスアクセスライブラリをインストールする．、また，各マシンで設定ファイル（`/opt/nec/aur_license/aur_license.conf`）にホスト名を登録してライセンスサーバの情報を登録する．

4.3.2. NQSV メインコンポーネントインストール

NQSV のメインコンポーネントを各サーバホストにインストールする．ライセンスサーバは無償のトライアル提供があったが，こちらは有償ソフトウェアであったため下村先生経由でいただいたものをインストールした．バッチサーバホスト、アカウンティングサーバホスト、運用管理ホストである tokyo に NQSV-ResourceManager をインストールする．また，クライアントホスト，バッチサーバホストである tokyo に NQSV-Client，NQSV-JobManipulator をインストールする．最後に各実行ホストに NQSV-JobServer をインストールして主なコンポーネントのインストールが完了する．

4.3.3. バッチサーバの設定

まず始めに，バッチサーバのユニット（nqs-bsv）の起動を行う．続いて，バッチサーバホストに任意のマシン ID を登録してデータベースを初期化する．その後，nqs_bsvd コマンドを用いてバッチサーバを起動，ユーザマップファイル（/etc/opt/nec/nqsv/nqs_user.map）にユーザと権限を登録する．

4.3.4. クライアントホスト設定

クライアントホストである tokyo 上の設定ファイル（/etc/opt/nec/nqsv/api_client.conf）にバッチサーバホスト名を記述する．また同じく，tokyo 上の設定ファイル（/etc/opt/nec/nqsv/nqs_jmd_cmdapi.conf）にスケジューラホスト名を記述する．これらの設定により，クライアントホストの環境設定が完了した．

4.3.5. 実行ホスト設定

まず，実行ホスト上で qmgr コマンドと attach execution_host サブコマンドを用いて実行ホストとジョブサーバ番号を指定する．次に，各事項ホストでランチャーデーモン（nqs-lchd）を起動する．最後に，qmgr コマンドの start job_server all コマンドですべてのジョブサーバを起動して実行ホストの設定を完了する．

図 2 に示すように qstat -Et コマンドを用いて各実行ホストのステータスを確認することができる．JSVNO（ジョブサーバ番号）が 0 のものが tokyo で，1 のものが london になっている．どちらのサーバも接続状態（LINK UP）になっており正常に動作していることがわかる．

```
[shibuya@tokyo ~]$ qstat -Et
ExecutionHost   JSVNO JSV      S OS         Release    Hardware     VE  Load   Cpu STT
--------------- ----- -------- - ---------- ---------- ---------- ---- ----- ----- ---
tokyo.sc.cc.toh     0 LINKUP   - Linux      3.10.0-116 x86_64        0   0.0   0.0 ACT
172.20.2.216        1 LINKUP   - Linux      3.10.0-116 x86_64        0   0.0   0.0 ACT
```

図 2. 各実行ホストのステータス

4.3.6. スケジューラ設定

まず，インストールした JobManipulator を起動する．続いて，smgr -Po コマンドで start scheduling を指示してスケジューリングを開始する．

4.3.7. キュー設定

qmgr コマンドの create execution_queue サブコマンドを使用して，キューの名前（execque1）とキューのプライオリティを指定してキューを作成する．続いて，bind execution_queue job_server サブコマンドを用いてキュー名とジョブサーバ番号を指定することで，バッチキューとジョブサーバをバインドする．さらに，bind execution_queue scheduler サブコマンドを用いて，キュー名とスケジューラの ID を指定することで，バッチキューとスケジューラをバインドする．最後に，enable execution_queue サブコマンドと start execution_queue サブコマンドによってバッチキューを開始する．

図 3 に示すように qstat -Qe コマンドを用いてバッチキューのステータスを確認することができる．4 列目，5 列目の ENA，STS がキューのステータスを表していて，それぞれ enable，active となっていて正常に動作していることがわかる．

```
[EXECUTION QUEUE] Batch Server Host: tokyo.sc.cc.tohoku.ac.jp
============================================================
QueueName       SCH JSVs ENA STS  PRI  TOT ARR WAI QUE PRR RUN POR EXT HLD HOL SUS MIG STG
--------------- --- ---- --- ---- ---- ---- --- --- --- --- --- --- --- --- --- --- ---
execque1          -    2 ENA ACT   10    0   0   0   0   0   0   0   0   0   0   0   0
--------------- --- ---- --- ---- ---- ---- --- --- --- --- --- --- --- --- --- --- ---
 <TOTAL>                             0   0   0   0   0   0   0   0   0   0   0   0   0
--------------- --- ---- --- ---- ---- ---- --- --- --- --- --- --- --- --- --- --- ---
```

図 3. バッチキューのステータス

4.3.8. アカウンティングサーバ設定

アカウンティングサーバ（nqs-asv）とアカウンティングモニタ（nqs-acm）の起動を行う．続いて，予約管理機能について設定ファイル（/etc/opt/nec/nqsv/asvd.conf）において $SBU\_CHECK = ON$ とする．また，バッチサーバ側の環境設定として，バッチサーバ上で qmgr -Pm コマンドを用いて予算管理機能，リソース予約区間に対する課金設定，リクエストアカウントとジョブアカウントの設定，アカウンティングサーバ情報の設定を行った．

4.3.9. リクエスト実行の確認

```
qsub -q execque1 -l elapstim_req=200 --cpunum-lhost=1
uname -a
sleep 100
```

とコマンドを打つことでキューにリクエストを投入する．投入状態は qstat コマンドで確認することができ，図 4 に qstat の出力結果を示す．ジョブの 100 秒間のスリープをリクエストしたため，ステータスは Running 状態になっていて，経過時間も 100 秒間経つまで増加していることがわかる．

```
[shibuya@tokyo ~]$ qstat
RequestID       ReqName  UserName Queue     Pri STT S   Memory      CPU   Elapse R H M Jobs
--------------- -------- -------- --------- ---- --- - --------- --------- --------- - - - ----
24.tokyo.sc.cc. STDIN    shibuya  execque1    0 RUN -    0.00B     0.00         2 Y Y Y    1
[shibuya@tokyo ~]$ qstat
RequestID       ReqName  UserName Queue     Pri STT S   Memory      CPU   Elapse R H M Jobs
--------------- -------- -------- --------- ---- --- - --------- --------- --------- - - - ----
24.tokyo.sc.cc. STDIN    shibuya  execque1    0 RUN -    0.00B     0.00         7 Y Y Y    1
[shibuya@tokyo ~]$ qstat
RequestID       ReqName  UserName Queue     Pri STT S   Memory      CPU   Elapse R H M Jobs
--------------- -------- -------- --------- ---- --- - --------- --------- --------- - - - ----
24.tokyo.sc.cc. STDIN    shibuya  execque1    0 RUN -    0.00B     0.00        14 Y Y Y    1
[shibuya@tokyo ~]$ qstat
RequestID       ReqName  UserName Queue     Pri STT S   Memory      CPU   Elapse R H M Jobs
--------------- -------- -------- --------- ---- --- - --------- --------- --------- - - - ----
24.tokyo.sc.cc. STDIN    shibuya  execque1    0 RUN -    6.63M     0.00        15 Y Y Y    1
```

図 4. qstat 出力結果

5. 今後の予定

短期目標として，

・テスト環境の整備

・コマンドの出力形式を AOBA と一致させる

・NQSV のアダプタを書き始める

中長期目標として，

・SSHFS を用いた OOD 経由のリモート HPC クラスタへのジョブ投下について検討

・卒論構成についての深掘り

を考えている．

参考文献

[1] Robert Settlage, Eric Franz, Doug Johnson, Steve Gallo, Edgar Moore, and David Hudak. Open OnDemand: HPC for Everyone. *ISC 2019 Workshops*, No. 11887, pp. 504–513, 12 2019.

[2] OpenOnDemand 3.0.3 Docuentation. *https://osc.github.io/ood-documentation/latest/*.

[3] install Open OnDemand — Open OnDemand. *https://openondemand.org/install-open-ondemand*.

[4] Masahiro Nakao, Masaru Nagaku, Shinichi Miura, Hidetomo Kaneyama, Ikki Fujiwara, Keiji Yamamoto, and Atsuko Takefusa. Introducing Open OnDemand to Supercomputer Fugaku. *SC-W 2023*, No. 1, pp. 720–727, 11 2023.