令和 5 年度　グローバルゼミ・論文紹介　前刷　　令和 6 年 1 月 18 日

担当学生　谷澤　悠太
指導教員　滝沢　寛之 教授

1. 背景・課題

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利用されているため，HPC 利用の需要は年々高まってきている．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 の利用においてシェルを用いたコマンドラインインタフェースや SSH の鍵設定などによるセットアップの複雑さ，ジョブスケジューラごとに異なる操作方法などが初学者にとって大きな障害となり得る．このように，HPC を使いこなすための前提知識は幅広く，学習コストも大きいため，HPC 利用者が本来のタスクに費やすコストが減少してしまうといった問題がある．

2. 先行研究

2.1. Open OnDemand

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][4]．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，コマンドラインインタフェースではなくグラフィカルユーザインタフェ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った懸念点も考えられる．

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 [5]．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り，利用難易度が低下したという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挙げられる．

2.2. PSI/J

『PSI/J（Portable Submission Interface for Jobs）』[6] は様々な HPC のジョブスケジューラに対して HPC システムとの統合的なインタフェースの役割を果たす．つまり，PSI/J はスケジューラの抽象化レイヤーとして機能し，スケジューラの種類に依らずジョブの投入や削除などの一元管理が可能となる．実際に，Parsl，RADICAL-Pilot，Swift/T などのライブラリやフレームワークはすでに PSI/J に対応しており，その利点を十分に享受している．また，PSI/J は SLURM や LSF，PBS Pro，cobalt などに対応しているが，OOD と同様にこれらも数あるジョブスケジューラのうちの一つであり，未対応スケジューラには使用することができないといった懸念点が考えられる．

3. 目的・提案手法

そこで本研究では『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，OOD と PSI/J を通じて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 や PSI/J 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）には未対応である．そこで，ユーザと HPC システムのインタフェースとして OOD を用いることで，HPC システム操作方法の簡易化に伴う利用難度の低下を試みる．さらに，OOD と HPC システムのジョブスケジューラのインタフェースとして PSI/J を用いることで，ジョブスケジューラの仕様が異なるために発生するジョブスケジューラ依存の問題を PSI/J に役割の細分化を図る．

そのため，本研究では『AOBA』の利用環境を想定しているため，図 1 に示すようなシステムを想定している．OOD が動作しているサーバとジョブスケジューラが動作するサーバは別マシンであり，HPC システムの利用に用いられる Active Directory（AD）も独立したサーバとして稼働している．OOD は各スケジューラに定義された関数を用いてジョブの管理を行っていたが，この工程に PSI/J で用いられている各メソッドを用いることで OOD はどのスケジューラに対しても PSI/J のメソッドを用いることになり，PSI/J がスケジューラに依存したコマンドの相違などを解消することになる．したがって，実装は大きく分けて，PSI/J の NQSV 対応，OOD の PSI/J 対応の 2 つが挙げられる．

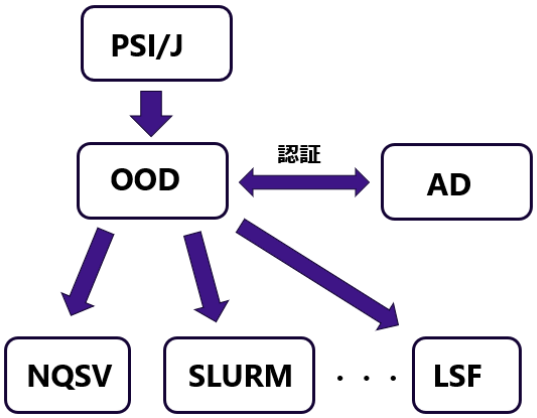

図 1. 『AOBA』の仮想利用環境

4. 進捗

4.1. PSI/J ⇔ NQSV の接続

まず始めに PSI/J と NQSV の接続を考える．PSI/J にはジョブの投入や削除などを行う JobExecutor クラスと対象のジョブ自身を表す Job クラスなど複数のクラスとメソッドで構成されている．特に，JobExecutor クラスは選択するスケジューラごとにサブクラスが構成され,PSI/J は現在 SLURM や PBS Pro，LSF などに対応しているため，それぞれ個別の JobExecutor クラスが存在している．

今回は NQSV との接続を実装するため，NQSV 用の Executor クラスを作成することを目指す．Executor クラスには大きく分けて 3 つの動作を行う関数が定義されていて，ジョブの投入，ジョブの削除，ジョブのステータスの取得を行うメソッドが定義されている．

まず始めに，ジョブの投入を行うメソッドを作成した．ジョブ投入に関するメソッドは NQSV の前身である PBS Pro の JobExecutor の内部とほとんど変わらずに作成することができたが，PBS Pro と異なり，NQSV では投入するキューを明示的に指定しなければジョブを投入できないといった特徴があるため，ジョブスクリプトを作成するために用いられるスクリプト中でキューの指定を行うための「#PBS -q execque1」という行を挿入する必要があった．

続いて，ジョブのステータスを取得するメソッドを作成した．前述の通り，大半の構成は PBS Pro とほとんど同じだが，大きく異なる点として両者の qstat の仕様の違いが挙げられる．PBS Pro では qstat コマンドに-x オプションを付けることで実行が完了した過去のジョブのステータスも確認することができ，得られたステータスからジョブの完了を認識することができたが，NQSV では終了してしまった

ジョブは qstat コマンドでステータスを確認することができないため，ジョブの完了を示すステータスがそもそも存在しない．そのため現在，PSI/J ⇔ NQSV では完了したジョブに COMPLEATED のステータスをマッピングすることができていない状態になっており，実際にはジョブが完了しているのにもかかわらず，ジョブの投入中を示す QEAEAD のステータスが永続的に割り振られている．この状態を避けるために qstat コマンドを投げて出力がかえって来ない場合に COMPLEATED ステータスを割り振るなどのアイデアがあるが現段階では未実装である．

最後にジョブの削除を行うメソッドを作成した．ジョブの削除に関しても構成はほとんど PBS Pro と同じものであるが，PBS Pro ではジョブが削除された際に返される終了コードの値を用いてジョブが正常終了したかコマンドにより削除されたのかを判別していたが，NQSV では PBS Pro における終了コードの役割を果たすものが存在しないと考えられる．そのため，NQSV ではキューに入っていないジョブに対してステータスが COMPLETED の状態であるのか CANCELED の状態であるのか区別することができなくなってしまっている．

先に示した三つのメソッドをテストした．実行コードと出力結果を図 2，図 3 に示す．図 2 では NQSV の Excutor クラスを作成した後，テスト用に用いられている実行スクリプトを用いて標準出力先とエラー出力先を指定した Job クラスを作成し，submit メソッドを実行した後と cancel メソッドを実行した後に qstat コマンドの出力結果を表示した．これにより，各メソッド後のキュー内部のジョブの状態を観測することができる．図 3 より結果を確認すると確かに submit メソッド後はジョブが投入されているのに対して cancel メソッド後はキューの中身が何も入っていないことがわかり，PSI/J ⇔ NQSV の実装が予想通りに動作していることがわかる．なお，テスト結果の STT で示されているステータス PRR はジョブの実行前処理中を表すステータスである．

```
from pathlib import Path
import subprocess
import time

ex = JobExecutor.get_instance("nqsv")
job = Job(
    JobSpec(
        executable="/home/gp.sc.cc.tohoku.ac.jp/tanizawa/psij_test/run.sh",
        # directory="/home/gp.sc.cc.tohoku.ac.jp/tanizawa/psij_test/",  run.shのhello.pyが相対パスの場合はdirectoryの指定が必要
        stdout_path=Path("/home/gp.sc.cc.tohoku.ac.jp/tanizawa/psij_test/result/stdout.txt"),
        stderr_path=Path("/home/gp.sc.cc.tohoku.ac.jp/tanizawa/psij_test/result/stderr.txt")
    )
)
command = "qstat"


ex.submit(job)
# print(job.status.state)
submit_result = subprocess.run(command, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, text=True)
print("after qsub:\n", submit_result.stdout)

ex.cancel(job)
# print(job.status)
time.sleep(1)
cancel_result = subprocess.run(command, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, text=True)
print("after qdel:\n", cancel_result.stdout)
```

図 2. テスト実行コード

```
(base) [tanizawa@tokyo psij_test]$ python3 psij_test.py
after qsub:
  RequestID       ReqName  UserName Queue     Pri STT S  Memory      CPU   Elapse R H M Jobs
  --------------- -------- -------- --------- --- --- - --------  -------- ------- - - - ----
  338.tokyo.sc.cc f39dfd0f tanizawa execque1    0 PRR -    0.00B     0.00       0 Y Y Y    1

after qdel:
```

図 3. テスト出力結果

4.2. OOD ⇔ PSI/J の接続

続いて OOD と PSI/J の接続を考える．設計として OOD のジョブの投入や削除を行う各メソッドからそれぞれ役割を果たすための PSI/J メソッドを呼び出すことができるようにしたい．これにより，ユーザの依頼を受けた OOD はスケジューラに直接命令を出すのではなく，PSI/J を仲介して命令を送ることになり，前章で示した，スケジューラの抽象化機構を挟むことができる．また，この実装の中で，OOD が Ruby スクリプトで記述されているのに対して，PSI/J は python で記述されているため，個々のメソッド用のスクリプトを書き，OOD が作成された python スクリプトを必要に応じて呼び出すといった形式をとる．

現在は OOD ⇔ PSI/J での submit メソッド，cancel メソッドのの実装途中である．OOD 上の JobComposer で submit ボタンをクリックすると PSI/J 経由でジョブが投入されるようになったが，図 4 に示すように，OOD 内部で Job のパスを明示しないと動作しない状態であり，様々なジョブを選択して実行することができていない．submit メソッドは引数が OOD 内で定義されている Script クラスのインスタンスであるため，この Script クラスからジョブスクリプトのパスを得ることができれば正常に動作する．

また，ジョブの削除を行う delete メソッドも実装途中であり，submit メソッドと同様に OOD によって与えられる引数と PSI/J の submit メソッドを行うために必要な引数が違うために問題が生じている．図 5 に示すように，引数には id のみが与えられるが，ジョブの削除を PSI/J で行うためには PSI/J で定義される Job クラスの情報が必要なため，submit メソッドを PSI/J で実行した際に，用いた Job クラスのデータを残しておき，delete メソッドの実行時に与えられた id と一致する id をもつ Job クラスを取り出す必要がある．現在は，Python スクリプト間で Job クラスを受け渡す機能の実装中である．

```
def submit(script, after: [], afterok: [], afternotok: [], afterany: [])
  #scriptから実行するジョブのパスを抜き取ってPSIJでjobクラスを作成してex.submit(job)を実行したい

  job_path = "/home/gp.sc.cc.tohoku.ac.jp/tanizawa/psij_test/run.sh"
  psij_submit = "python3 /opt/ood/gems/gems/ood_core-0.23.5/lib/ood_core/job/adapters/psij/submit_script.py" + " " + @psij.executor + " " + job_path
  system(psij_submit) #psij_submitコマンドを実行
end
```

図 4. OOD の submit メソッド

```
def delete(id)
  #PSIJからidと一致するjobクラスを持ってきてex.cancel(job) or job.cancel()を実行する

  psij_delete = "python3 /opt/ood/gems/gems/ood_core-0.23.5/lib/ood_core/job/adapters/psij/delete_script.py" + " " + id
  system(psij_delete) #psij_deleteコマンドを実行
end
```

図 5. OOD の delete メソッド

5. 今後の予定

来週までに，

・OOD ⇔ PSI/J での submit メソッド，cancel メソッドのの実装

短期目標として，

・OOD ⇔ PSI/J での qstat メソッドの実装

・OOD ⇔ PSI/J の動作確認

中長期目標として，

・本番発表用の前刷り・スライドの作成

を考えている．

参考文献

[1] David E. Hudak, Thomas Bitterman, Patricia Carey, Douglas Johnson, Eric Franz, Shaun Brady, and Piyush Diwan. OSC OnDemand: A Web Platform Integrating Access to HPC Systems, Web and VNC Applications. *XSEDE '13*, No. 49, pp. 1–6, 7 2013.

[2] 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.

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

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

[5] 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.

[6] Mihael Hategan-Marandiuc, Andre Merzky, Nicholson Collier, Ketan Maheshwari, Jonathan Ozik, Matteo Turilli, Andreas Wilke, Justin M. Wozniak, Kyle Chard, Ian Foster, Rafael Ferreira da Silva, Shantenu Jha, and Daniel Laney. PSI/J : A Portable Interface for Submitting, Monitoring, and Managing Jobs. *IEEE 19th International Conference on e-Science*, 2023.