Vol.2020-HPC-175 No.23
2020/7/31

TSUBAME3のインタラクティブ利用の利便性向上にむけた取り組み

野村 哲弘[1]　遠藤 敏夫[1,2]　三浦 信一[1,3]　朝倉 博紀[4]　越野 俊充[4]　草間 俊博[4]

概要：東京工業大学の TSUBAME3 では，ノード使用率の上昇によるノード枯渇のためにビジュアライズやデバッグ等のインタラクティブな利用が困難な状況となっていた．本稿ではインタラクティブ利用ユーザやスパコン初心者に対する利便性向上施策として行った 1) インタラクティブジョブ実行専用キューの創設 2) 利用者ポータル経由での SSH 等を経由しない Web アプリケーション (Jupyter Lab 等) 実行基盤の構築 について報告する．

1. 背景

東京工業大学学術国際情報センター (以下,「本センター」という) では,「みんなのスパコン」を合言葉とし，使いやすさと高性能を両立したスーパーコンピュータ TSUBAME シリーズを構築・運用しており，2017 年 8 月からは現行の TSUBAME3.0 [1] を供用している．2006 年に導入された TSUBAME シリーズの初代である TSUBAME1.0 以来，TSUBAME は学内外で約 2,000 名のユーザ（2019 年度 SSH ログインユーザ数）に利用されるシステムとなっている．

TSUBAME3.0 [2] の全体構成は**図 1** に示す通りであり，540 台の計算ノードと各種ストレージおよびそれらを高速に結合する OmniPath による相互結合網などからなる．

全ての計算ノードは，フルバイセクション・ファットツリートポロジで接続されており，任意の計算ノードから任意の計算ノードおよびストレージサーバにポート当たり 100Gbps のスピードで接続することができる．

ノード構成は**図 2** に示すとおりであり，1 ノードあたり 4 つの GPU，4 つの OmniPath HFI が接続されている非常に大きなノードとなっている．このように複数の GPU 等を擁する巨大なノードは，ノード内の GPU 間相互結合網である NV-Link などを活用し，高度に最適化されたプログラムに対しては有効ではあるが，単一の GPU や CPU のみを想定したプログラムでは，ノード性能の大半を持て余してしまうことから，**図 3** に示すようにジョブスケジューラである Univa Grid Engine [3](以下,「UGE」という) 経由で Linux cgroups を活用したノード分割を積極的に行うことで，有休資源の最小化に努めている．

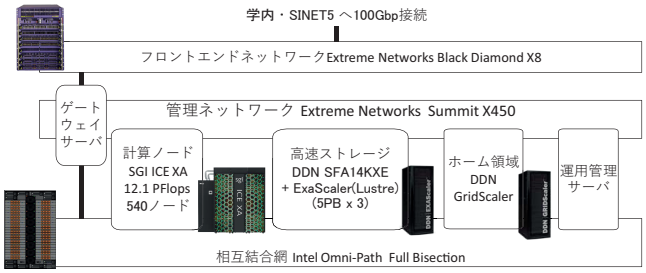

図 1　TSUBAME3.0 全体構成

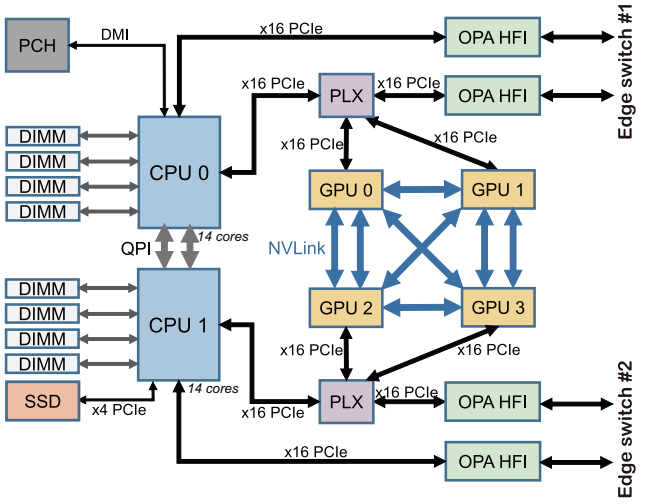

図 2　TSUBAME3.0 計算ノードのブロックダイアグラム

このような努力にもかかわらず，近年の「京」コンピュー

1　東京工業大学
2　産総研・東工大 実社会ビッグデータ活用 オープンイノベーションラボラトリ
3　理化学研究所
4　日本ヒューレット・パッカード

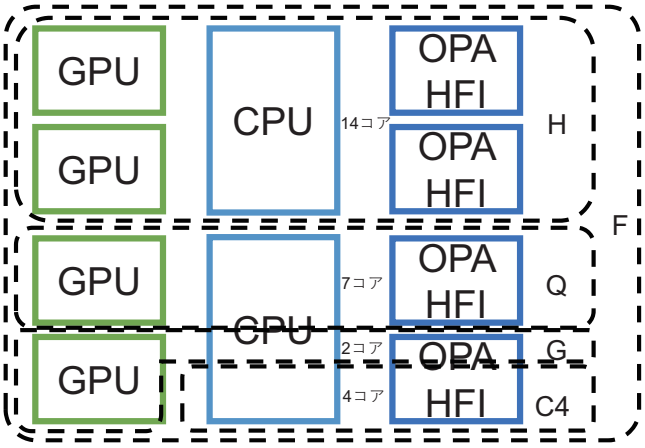

図 3　TSUBAME3.0 計算ノードの資源分割イメージ

タからスーパーコンピュータ「富岳」への移行期に伴う HPCI 課題などの学内外の計算需要の高まりを受けて，TSUBAME3.0 の利用率は**図 4** に示すように，年度全体で 80%超，最繁忙期では 96%を上回る非常に高い水準で推移している [4]．このようなジョブの滞留傾向により，TSUBAME3.0 ではジョブの投入から実行開始までに 24 時間以上の待ち時間がかかり，デバッグや計算結果のビジュアライズなどのインタラクティブ処理を満足に行うことができない状態が続いていた．

一方で，インタラクティブ処理の需要は従来のデバッグや計算結果のビジュアライズだけではなく，IoT 分野におけるセンサデータ類のオンタイム処理や，AI 分野を中心に広く用いられている Jupyter Lab [5] などの統合開発環境経由での利用など，広がりを見せており，大学における「みんなのスパコン」としてユーザ層を拡げていくうえでは解決が必須な課題である．

先代の TSUBAME2.0/2.5 においては，外部から ssh を受け付けるログイン用の計算機はインタラクティブノードと呼ばれる，計算ノードと同一構成のノードを 20 台割り当て，ロードバランサ経由で割り当てることで，ログイン用ノードとインタラクティブ処理用ノードを兼ねさせる形にすることで，この問題に対応してきた．一方で，計算ノード数が TSUBAME2.0/2.5 の約 40%しかない TSUBAME3.0 では，計算ノードを全てバッチジョブに配分し，ログインノードは計算ノード群の外に，GPU を備えない比較的非力なノードを 2 台のみ設置し，ユーザには高負荷処理をログインノード上で行わないようアナウンスする [6] など，インタラクティブ処理の需要に十分に応えられていない実情があった．

本報告では，このような状況を鑑みて，TSUBAME3.0 において 2020 年初頭に導入したインタラクティブジョブ実行専用キューおよび Web アプリケーション実行基盤について，その設計および実装を示すとともに，制約や問題点についても共有する．

2. インタラクティブジョブ実行専用キュー

2.1 従来のインタラクティブジョブ実現方法

従来，インタラクティブジョブはバッチジョブの特別な形として，ジョブ実行中のノードへの ssh 接続を許可する，もしくはジョブ実行時のターミナルを確保した計算ノードに接続する形で提供されてきた．TSUBAME3.0 における UGE においても `qrsh` コマンドおよび，スケジューラへの`-now` オプションによって，確保した計算ノードへの接続や，即時に実行されず，実行キューで待ち状態になったジョブの自動キャンセルが行われ，またノードの資源分割を行わなかった場合には，ジョブ実行期間中のみログインノードから各計算ノードへの ssh を許可することで，このようなインタラクティブ操作の機能を提供してきた．

しかし，このような形で提供されるインタラクティブジョブも，ジョブスケジューラの観点からは単なるバッチジョブの一類型に過ぎず，システムの混雑時にはまとまったノード時間を即時に確保することは難しく，ジョブをキューに入れて数時間後に実行が開始するまで待たされるなど，インタラクティブな操作ができるまでの待ち時間が許容できない状況となっていた．

2.2 バッチジョブおよびインタラクティブジョブの特性

通常のバッチジョブとインタラクティブジョブでは，求められるジョブの特性が異なる．

通常のバッチジョブでは，確保した計算資源を安定して最大限に活用できるよう，ジョブが開始した後は外乱がなくノード性能の 100%がジョブに費やされ，同じワークロードであれば同じ経過時間 (≃ 課金額) で終了することが期待される一方で，ジョブが投入されてから即時に実行されることは，望ましい性質ではあるものの混雑時には実現不可能であり，ユーザからも前者の性質ほどには期待されていない．

他方，インタラクティブ処理においては，ユーザの操作 (ジョブの投入やコンソールでの入力) に対して，即時に反応が返ってくることが重要である一方，計算リソースの使用は間欠的であり，計算ノードを占有していたとしてもその大半の時間はユーザの操作待ちのアイドル時間というジョブが多く，前章の TSUBAME2.0/2.5 におけるインタラクティブノードの例のように，他のインタラクティブジョブと資源を共有しても問題にならない．

このような差異は，計算資源に余裕があるうちは両者の性質が同時に満たされるために顕在化しないが，計算資源が逼迫してくるに従って，従来のシステムではインタラクティブジョブを通常のバッチジョブと同じ方法でスケジューリングした結果，インタラクティブジョブを実質的に利用できない状況に陥った．

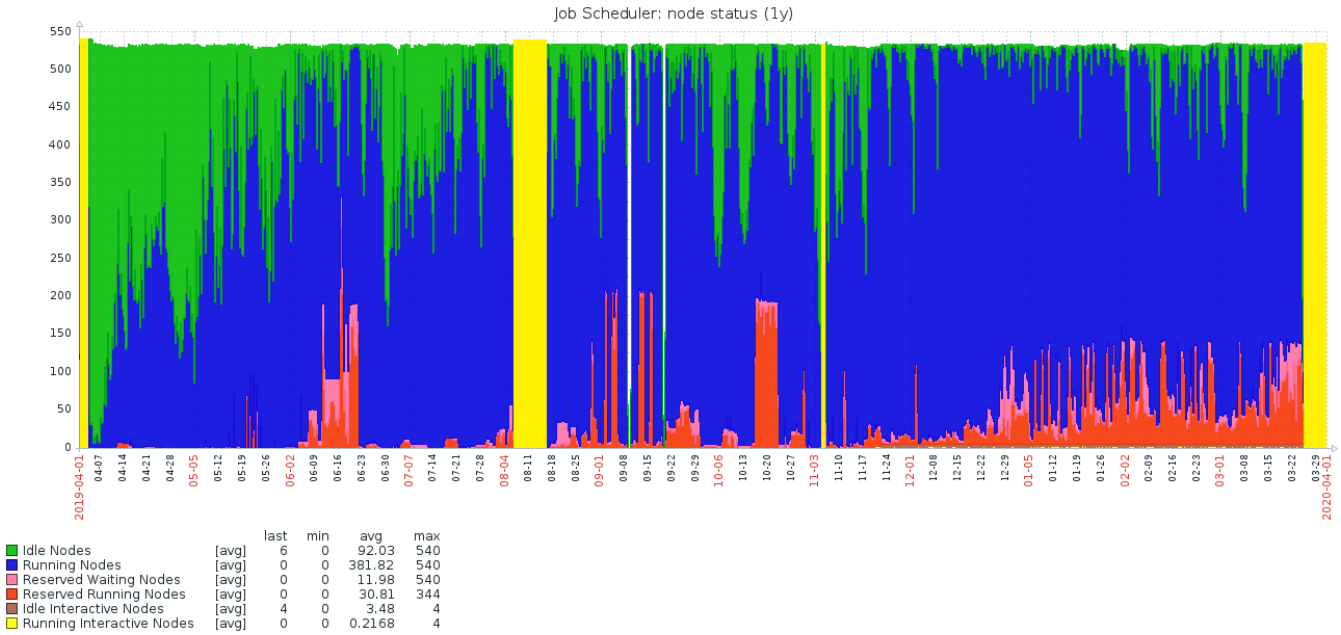

図 4　TSUBAME3.0 計算ノード利用率 (2019 年度) 黄色はメンテナンス期間

2.3　インタラクティブジョブ専用キューの設計

前節のような検討の結果，インタラクティブジョブの実行に求められる最高優先度の要件は，実行要求時に即時に実行されることであり，この実現のためには，システム内にインタラクティブジョブ実行のための計算資源を常時確保しておくことが必要となった．他方で，インタラクティブジョブ同士は資源の共有を行っても問題にならず，少数の計算ノードで多数のインタラクティブジョブを処理することは現実的である．そこで，本センターでは 50 人〜100 人規模の計算科学・高性能計算に関する講義・実習時に受講者が同時にインタラクティブジョブを利用できることを目標に，以下の方針でインタラクティブジョブ実行専用キューを設計した．

- インタラクティブジョブ専用に計算ノードを切り出し，バッチジョブ用とは独立したキューを構成する
- インタラクティブジョブ専用ノードでは多数のジョブが 1 ノード内で同時に実行される (オーバーコミット状態)
- 1 つのインタラクティブジョブがノードの全資源を使いつくさないよう，ジョブ当たりの利用可能資源量に制限をかける
- 1 ユーザあたり同時に投入・実行可能なインタラクティブジョブ数は 1 に制限する
- インタラクティブジョブに対しては性能の保証を一切行わない
- 試行期間中はインタラクティブジョブ実行専用キューに対して課金を行わない

上記指針に基づき，UGE 上で通常のバッチジョブ用のキュー (`all.q`) とは別のキュー `interactive.q` を作成し，540 ノード中，4 ノードを `interactive.q` に移動した．`interactive.q` においては，ノードを論理分割し，図 3 における Q 相当の論理ノードが 16 台，各論理ノードにつき最大 7 ジョブのインタラクティブジョブが実行されるよう設定した．なお，7 ジョブの制限は，UGE におけるジョブスケジューリング計算時に CPU コアを 1 コア以上占有することが必要であったための，実装上の制約から来たものである．各論理ノードにおけるリソース制限は，従来と同様に cgroups で行い，同じ論理ノードで実行される最大 7 ジョブが 7CPU コア (Hyper-Threading を含めると 14 論理コア)，1GPU を共有する．

ジョブの最大実行時間は 24 時間とし，長時間のインタラクティブ処理をジョブ中断によるストレスなく実施できるようにするとともに，ジョブの終了忘れによる残留が長期間にわたって発生することがないようにした．

CPU メモリに関してはプロセッサコアと違い時分割で共有することができないため，1 ジョブ当たりの CPU メモリ使用量を 60GB に制限した．通常のバッチジョブ用のノードでは性能低下につながるためスワップファイルを用いていないが，インタラクティブジョブ用のノードにおいてはメモリ需要が逼迫することを想定し，ノードのローカル SSD の大部分 (1.5TB) をスワップファイルに廻すことで使用メモリ量の増大に備えた．なお，実際にすべてのジョブがメモリを最大量まで使うことは想定しておらず，利用者数が少ない間はスワップ領域の利用による性能低下は起こらないと期待している．

他方で，GPU メモリに関しては現時点のシステムソフトウェアおよびプログラムの書かれ方では，ジョブ当たりのメモリ量を制限したりスワップデバイスに退避したりす

ることが現実的ではなく，「早い者勝ち」という状況になっており，今後必要に応じて制約手法を検討する必要がある課題となっている．

2.4 利用状況

2019 年 11 月にインタラクティブジョブ実行専用キューの試行サービスを学内ユーザに向けて開始し，2019 年度中は 54 名，2020 年 4 月から 5 月の期間は 120 名の利用があった．これは同期間の利用歴のあるユーザ数のそれぞれ 2.6%，15.8%に相当し，2020 年度になってから利用者が急増していることがわかる．利用者増加の主因は，高性能計算に関する講義が同期間に行われ，講義内で本機能についての紹介が行われたことと考えられ，ノードの混雑期である 2019 年度末以上の伸びを見せていることから，機能の存在をいかに利用者にアピールするかが課題として残る結果となった．

今回，性能保証を行わないサービスにおける課金ポリシの策定が間に合わず，学内ユーザ限定の試行という形で導入したが，今後は利用状況の詳細な把握を行い，課金モデルの設計をすすめ，学外からの利用者に向けての開放の是非についても議論する予定である．

3. Web アプリケーション実行機能

3.1 Linux 初心者がスパコン利用で求められる前提知識

TSUBAME3.0 を含む現在の Linux ベースのスパコンの利用のためには，以下のような膨大な前提知識が要求され，このことが今まで Linux を利用していなかった初学者および非 HPC ユーザの参入障壁となっている．

- cd, mv 等の Linux の基礎的なコマンド
- SSH クライアント・ターミナルのインストールおよび操作法
- SSH 公開鍵認証方式および鍵ペアの作成・管理・アップロード
- ジョブスケジューラの概念・ジョブスクリプト
- 実際に動かしたいドメイン・アプリの知識

このうち，最後のアプリケーション自体の知識以外は，長期的に知っておくことが望ましい内容ではあるが，本来のスパコン利用においては枝葉末節にあたり，これらをいかに簡便にできるかが「みんなのスパコン」としての裾野の広さに直結する．

加えて，ビジュアライズを行う X アプリケーションにおいては，ローカル X Window System サーバおよび X 転送，Jupyter Lab のような Web ベースのアプリケーションでは，SSH のポートフォワーディングに関する知識も要求されるようになり，そのハードルはさらに大きなものとなってしまう．

3.2 プロトタイプ: jupyterrun スクリプト

本センターでは，2019 年 6 月に，Jupyter Notebook の計算ノードでの利用法を例示するために，jupyterrun というスクリプトを作成し，公開した [7]．本ツールはコマンド本体とそこから起動されるジョブスクリプトの 2 つのスクリプトからなり，後者のスクリプトで Jupyter Notebook を起動しつつ，そこへの接続に必要なポートフォワーディングオプションを含めた TSUBAME3.0 への ssh コマンドを表示し，ユーザに別ターミナルから当該 ssh コマンドの実行を促すだけのものであった．しかしながら，Web 上にはスパコン上での Jupyter Notebook(もしくは Jupyer Lab) の起動方法に関する公式 [8] もしくはユーザ作成の非公式な記事が林立しており，このような知識に関する需要がかなり大きいことがうかがえる．

3.3 設計

前節の jupyterrun スクリプトでは，Linux のコマンド操作や SSH 関係についてはユーザの習熟を要するものであった．そこで我々は，対象 Web アプリケーション (Jupyter Lab を想定) の起動から終了までを全てブラウザ上の TSUBAME 利用者ポータル上の操作で完結させることを目標に，Web アプリケーション実行機能を設計した．

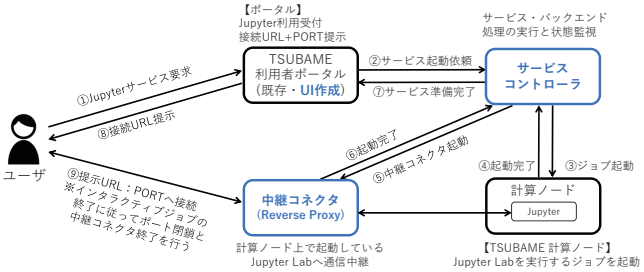

図 5 Web アプリケーション実行機能の構成

図 5 に Web アプリケーション実行機能の全体像を示す．図中のサービスコントローラ，中継コネクタの 2 つが本機能の実装のために新たに作成したコンポーネントであり，TSUBAME 管理ネットワークおよび外部ネットワークに接続された専用ノードで動作する．ユーザが利用者ポータル上で Web アプリケーションの起動を指示すると，その情報はサービスコントローラに引き継がれ，スケジューラ経由で計算ノード上に専用のジョブを起動し，ジョブの起動を受けて中継コネクタ内にジョブと HTTP ポートのマッピングを作成する．利用者にはジョブに対応する中継コネクタの専用 URL を提示し，ユーザが中継コネクタにアクセスすることで，中継コネクタが HTTPS Reverse Proxy として動作し，計算ノード内のジョブで動作する Web アプリケーションに外部ネットワークからアクセスすることができるようになる．

図 5 に Web アプリケーション起動画面のイメージを示

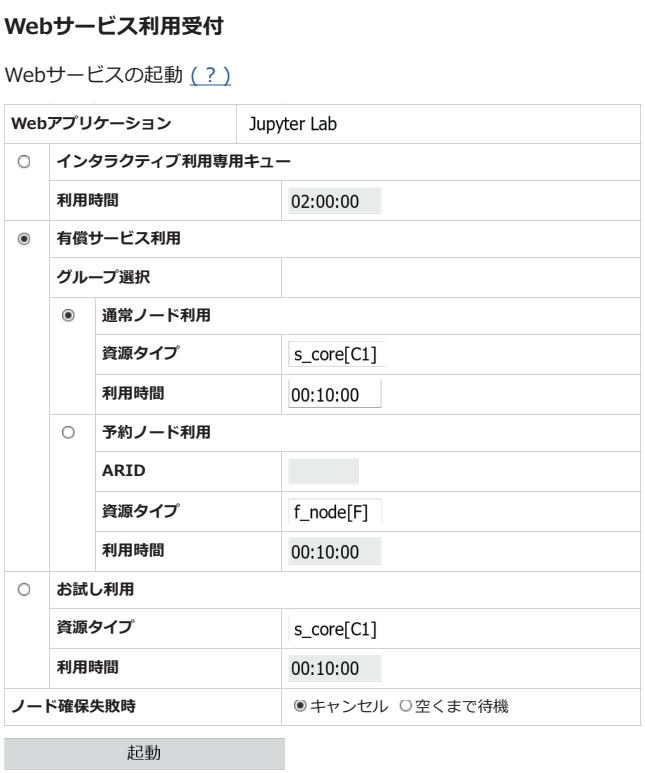

図 6 Web アプリケーション起動画面

す．ここでは，利用する Web アプリケーションイメージ (本稿執筆時点では Jupyter Lab のみ) および，ジョブ作成時に `qsub` コマンドもしくはジョブスクリプトのオプションで指定する各種パラメータを設定できる．ジョブは本報告の前半で導入したインタラクティブジョブ実行専用キューでも実行することができる．

インタラクティブジョブ実行専用キューを用いない場合は，(論理) ノードを占有できるのでより安定した性能を期待できる一方で，混雑時にはジョブの実行開始まで待ちが発生することが起こりうる．待ち時間が発生したときにジョブを継続すべきかどうかは利用者の状況によって異なるため，起動画面の最下部のオプションによって待ち発生時の動作を選択できるようにした．(UGE の `qsub` の`-now y` オプションに相当)

3.4 Jupyter Lab の機能と制約事項

Jupyter Notebook の発展形である Jupyter Lab では，ノートブック形式で Python を対話的に利用することに加えて，簡易的なファイルブラウザおよび Linux のコマンドラインシェルを起動する機能も備えている．このため，Web アプリケーション実行機能で起動する Jupyter Lab を SSH およびローカルの端末エミュレータの代替として利用することができる．

Jupyter Lab(および Notebook) で利用する Python カーネルは，原則的に Jupyter Lab を起動する際に利用した Python の環境を継承する．TSUBAME3.0 では CUDA を含む各種ライブラリは Environment Modules(`module` コマンド) で明示的に読み込む設計となっており，Jupyter Lab 起動時の環境では CUDA 等のライブラリへのパスが通っておらず，CuPy などの GPU を利用する機械学習関連パッケージの利用ができない．このため，Jupyter Notebook から利用できる `t3jpttools` Python モジュールを提供し，この中で各種ライブラリの Environment module を指定して読み込んだ Python カーネルを定義できるようにした．CUDA ライブラリへのパスが通っている Python 環境において，Jupyter Notebook のシェルエスケープなどを経由して `pip` コマンドを実行することで，CuPy などのサードパーティの Python モジュールをインストールすることができる．

Python の実行環境は，virtualenv, pyenv, Anaconda など様々な手段で作成され利用されているが，本稿執筆時点では上記の制約により環境の選択に対応していない．また，本機能は Python 3.6.5 で実装されているが，それ以外の任意のバージョンの Python をカーネルとして指定する機能も実装されていない．これらは，変更先の環境においても Jupyter が必要とする Python モジュール (ipykernel_launcher など) がインストールされていることを保証する必要があり，現状では個別の Python バージョン等に対して必要なモジュールを都度構築する対応が必要となっている．

3.5 ユーザの動向

2020 年 5 月末日時点での，TSUBAME3.0 の年度内アクティブユーザ数 (ログインノードもしくは Web アプリケーション実行機能のいずれかの利用歴があるユーザ数) は 760 名であり，うち 99 名が Web アプリケーション実行機能を利用していた．また，このうちの 30 名のユーザは SSH によるログインノードへの接続歴がなく，本機能の実装により 2 か月間で新たに 4%のユーザを獲得できたといえる．

3.6 関連研究

中田らは，自動ポートフォワーディング手法および Jupyter Hub をクラウド上に設置する手法で ABCI 上の Jupyter Notebook の外部からの利用を実現した [9]．

自動ポートフォワーディング手法は，我々の `jupyterrun` の手法をさらに自動化すべく推し進めた形となっているが，そのために利用者の手元にも小規模なスクリプトを保存することを要求し，利用者の手元環境に一定の仮定を置いている．

後者の Jupyter Hub 設置手法では，我々の手法と同様にターミナルの操作を極力省く設計となっているが，ブラウザの接続先の Jupyter Hub を対象システムである ABCI の外に置く設計となっているために，ユーザ管理が Jupyter Hub 側と ABCI 側で多重に必要となり，その間をパスフ

レーズのない SSH 鍵ペアで結ぶ必要があり，セキュリティ面および初学者の参入コストの面で問題がある．我々の手法では，Web アプリケーション実行機能を TSUBAME3.0 の利用者ポータル内に組み込むことで，ユーザ認証の問題を解決し，ジョブの投入も直接 Web アプリサーバが実行する形となっており，SSH 鍵ペアの管理に関する問題も発生しない．

また，同文献にて記載されていることではあるが，1 つの計算機アカウントを Jupyter Hub の利用者で共有する構造となっているため，1 アカウントの利用者は自然人 1 名に限ると利用細則で厳格に定められている TSUBAME3.0 においては，利用者ごとに別アカウントを利用するようにする改変を行わない限り，複数人での利用は難しい．

Google Colab [10] は，Google の機械学習指向の Jupyter Notebook 環境であり，Jupyter カーネルの切り替えの要領でアクセラレータなし・GPU・TPU のそれぞれのインスタンスへの切り替えが実行可能になっている．Notebook のフロントエンドを実行するノードと，Python カーネルを実行するノードを別にする設計を行うことで実装可能で，このことにより計算ノードの利用効率が向上する可能性もあるように思えるが，一度 Python カーネルを起動した後はいずれの方式においても Python カーネルの実行のために資源が占有された状態でユーザの入力待ちとなるため，大きな差はないものと考えられる．文献 [9] においては，Colab 方式のほうがインスタンス切り替え時にユーザの Notebook を開き直す必要があるためにユーザ経験が大きく劣ると結論付けているが，Python カーネル切り替え時にはいずれにせよ内部状態が失われてしまうため，インタラクティブジョブ実行専用キューのようにリソースのオーバーコミットを行うことで，サイズの違う資源とのマイグレーションを行う必要がない (課金方式を含めた) 設計を行うほうが良いと考える．

4. おわりに

TSUBAME3.0 におけるインタラクティブ利用の利便性向上のために行った，インタラクティブジョブ実行専用キューの導入と Web アプリケーション実行機能による Jupyter Lab の提供を通じて，「みんなのスパコン」としてのユーザ体験の向上について報告した．

インタラクティブジョブ実行専用キューを導入することで，資源の占有よりも応答性が優先されるインタラクティブジョブを通常のバッチジョブから隔離し，インタラクティブタスクの実行者のユーザ体験を向上させることができた．

Web アプリケーション実行機能の導入により，初学者等に対して SSH クライアントのインストールや公開鍵，Linux のコマンド類の学習を行うことなくスパコン上の GPU を含む計算資源への対話的なアクセスを直接提供することができるようになり，GPU スパコンの利用の敷居を下げることができた．

いずれの施策においても，利用され始めたばかりの段階であり，現時点では効果を定量的に示すには至っていないが，TSUBAME3.0 の利用者のストレスの軽減に資することができていれば幸いであり，利便性向上のための改良を続けていきたい．

謝辞　本研究の一部は，JSPS 科研費 (JP19H04121)，JST CREST(JPMJCR1501, JPMJCR19F5) の助成を受けたものである．また，TSUBAME3.0 の設計には，東京工業大学学術国際情報センターが推進してきた文部科学省「スパコン・クラウド情報基盤におけるウルトラグリーン化技術」および「スマートコミュニティ実現のためのスパコン・クラウド情報基盤のエネルギー最適化の研究推進」，JST CREST(JPMJCR1303, JPMJCR1501) などのプロジェクトの研究成果が活用されている．

参考文献

[1] 東京工業大学学術国際情報センター：TSUBAME 計算サービス，https://www.t3.gsic.titech.ac.jp/.
[2] 松岡　聡，遠藤敏夫，額田　彰，三浦信一，野村哲弘，佐藤　仁，實本英之，Drozd, A.：HPC とビッグデータ・AI を融合するグリーン・クラウドスパコン TSUBAME3.0 の概要，情報処理学会研究報告，Vol. 2017-HPC-160, No. 29, pp. 1–6 (2017).
[3] Univa Corporation: Univa Grid Engine, http://www.univa.com/products/.
[4] 東京工業大学学術国際情報センター：TSUBAME3.0 モニタリングページ，https://www.t3.gsic.titech.ac.jp/monitoring.
[5] Project Jupyter: Home, https://jupyter.org/.
[6] 東京工業大学学術国際情報センター：ログインノードで CPU を占有する利用は行わないでください，https://www.t3.gsic.titech.ac.jp/loginnode-limit.
[7] 東京工業大学学術国際情報センター：計算ノードでの Jupyter Notebook の起動，https://www.t3.gsic.titech.ac.jp/jupyter_notebook.
[8] 産業技術総合研究所：ABCI User Guide: Jupyter Notebook の利用，https://docs.abci.ai/ja/tips/jupyter-notebook/.
[9] 中田秀基，滝澤真一朗，小川宏高：大規模計算クラスタにおけるユーザ利便性向上，情報処理学会研究報告，Vol. 2020-HPC-174, No. 1, pp. 1–6 (2020).
[10] Google: Colaboratory, https://colab.research.google.com/.