TSUBAME2におけるジョブスケジューリング効率化への取り組みと検証

野村 哲弘[1,2,a)]　佐々木 淳[1]　三浦 信一[1,2]　遠藤 敏夫[1,2]　松岡 聡[1,2]

概要：
スーパーコンピュータの資源利用の効率化のためには，投入されるジョブの情報を正確に把握し，ジョブのスケジューリングを最適化することが重要である．東京工業大学学術国際情報センターの TSUBAME2 では，ユーザの実行時間指定が不正確であったためにバックフィルを用いた効率的なスケジューリングができず，ジョブが流れない事態が頻発していた．本報告では，TSUBAME において行われているユーザの資源指定を正確なものにするための取り組みと，その成果を確認するための各種ログ情報・センサー情報の解析・活用について報告する．

1. 背景

東京工業大学学術国際情報センターでは，2010 年 11 月よりスーパーコンピュータ TSUBAME 2.0 [1] を，2013 年 9 月よりそのアップグレード版である TSUBAME2.5(以下，両システムの総称として TSUBAME2 を用いる) を運用し，学内外のユーザから利用されている．TSUBAME2 の運用中は，TSUBAME2 全体を単一のアプリケーションが利用するグランドチャレンジ制度等の実行時などの例外を除き，計算資源をノード単位もしくは仮想マシン単位で分割し，複数のユーザの大小さまざまなアプリケーション (計算ジョブ) が同時に実行されている．

一般にこのようなシステムにおいては，バッチジョブスケジューラを用いて，各ジョブの推定実行時間・要求資源量 (ノード数) の情報をもとに実行順を制御し，複数の計算ジョブがオーバラップしないよう順番に実行させている．計算資源の利用効率はすなわちジョブスケジューリングの効率であり，多数のジョブスケジューリングアルゴリズムに関する研究がなされてきており，商用のジョブスケジューラにもその成果が反映されているものと考えられるが，これらのアルゴリズムにおける原理的な限界として，実際の実行時間はジョブの終了まで分からず，推定実行時間はユーザにとってはあくまでも最大実行時間でしかない点である，一方でスケジューラは常にジョブが指定された実行時間まで実行されることを前提としているため，事後的に実際に実行時間に基づいた最適な詰め込みが発見できる場合でも，オンラインでは最大実行時間しか分からないため，そのような解を選択することはできない．

TSUBAME2 においてもスケジューリングの限界に起因するマシンのアイドル時間の発生が問題となっており，繁忙期におけるユーザの実行機会損失の一要因となっていた．本報告では，スケジューリングアルゴリズム・ポリシーの変更ではなく，ユーザへの広報および課金ポリシーの変更によって緩和する試みについて報告する．また，著者ら TSUBAME2 の運用チームはこのような問題に定性的には気づいていたものの，定量的な分析は今まで行えていなかった，本報告では TSUBAME2 に蓄積されているジョブ実行ログおよびモニタリングログを活用し，上記ポリシーの変更についての検証結果について報告するとともに，どのようなデータが利用可能となり，どのようなデータの蓄積が必要であるかを議論する．

2. TSUBAME2 のジョブスケジューリング

2.1 キュー構成

TSUBAME2 では，ユーザのジョブ特性に合わせて主に以下のジョブクラスを定義し，それぞれについて一定台数の実行キューを作成している，夏季節電期間中の消費電力に応じたノード数の動的増減 [2] および後述の U/V キュー間のノード数動的制御はあるものの，基本的にはクラスごとに固定台数のノードを持つ PBS Professional のキューを作成してそれぞれのキューの中でスケジューリングを行っている．

[1] 東京工業大学 学術国際情報センター
Global Scientific Information and Computing Center, Tokyo Institute of Technology
[2] JST CREST
[a)] nomura.a.ac@m.titech.ac.jp

2.1.1 予約ジョブ (H)

TSUBAME2 では大規模ジョブの実行を希望するユーザのために 24 時間単位でのノードの予約が可能であり，予約用の資源として 420 台の Thin ノードを確保している．スケジューラの観点からは，予約ごとに対応する固定台数の計算ノードからなる PBS キューが作成され，そのキューを予約したユーザグループで独占して利用する，もしくは割り当てられた計算ノードにスケジューラを迂回して直接アクセスして，ユーザの責任においてタスクを実行するという形態がとられる．予約用資源である 420 ノードのうち予約に用いられなかった部分は，後述する X キューとして再利用される．

2.1.2 ベアメタルノード (S 系, X, L 系)

TSUBAME2 の Thin 計算ノードは 2 基の CPU(Intel Xeon X5670, 計 12 物理コア), 3 基の GPU(TSUBAME2.5 では NVIDIA TESLA K20X)，56GB(一部 96GB) の物理メモリ，2 つの InfiniBand QDR ネットワーク (インジェクションバンド幅 80Gbps)，120〜240GB のローカル SSD などからなり，そのすべてをジョブの実行に用いるユーザのための TSUBAME における最も基本的なジョブクラスである，S キューには 300 台の Thin 計算ノードを割り当てており，単一の PBS キューの上で多数のユーザがそれぞれのジョブを実行している．

X キューは前節の予約ノードに用いられなかったノードを用いて毎日動的に作成されるキューであり，S キューに投入されたジョブが以下の条件を満たす時に自動的に X キューに転送されるバイパス的な役割を果たしているキューである．

- ジョブが即時開始することができる
  - X キューに要求台数以上の空きがある
  - ジョブに実行順依存関係などの即時開始と反する実行条件が付されていない
- ジョブの開始から指定実行時間経過までの間に予約キューの切り替え時間帯 (9:00〜10:00) をまたがない

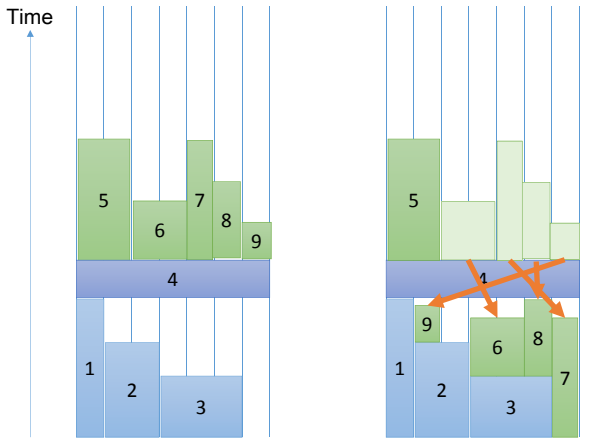
最初の条件より，X キューに転送されたジョブは即時実行開始されるので，X キューにおけるスケジューラの役割は空き資源の管理だけであり，スケジューリングの諸問題はこのキューにおいては発生しない．なお，上記 X キューでの実行条件は公開されており，即時の実行を望むユーザは実行時間条件を上記に合うように調整してジョブを投入している．

S96 キューは 96GB のメモリを持つ Thin ノード，L 系 (L128, L128F, L256, L512) キューはより多くのメモリを持つ Medium/Fat ノードと呼ばれるノードからなるメモリ要求量別のキューである，これらについては X キューへの転送が行われない点を除いてスケジューリングの観点からは S キューと同等である．

2.1.3 仮想マシンによるノード共有 (G, U, V)

TSUBAME2 の Thin ノードは計 12 コアの CPU と 3 基の GPU からなるが，多くのジョブについては，そのうちの片方のみを主に利用し，使われない側のプロセッサがアイドルとなる．そのため，Thin ノードのうち 480 台については，ノード内に一定量の CPU コアおよびメモリからなる KVM 仮想マシンを作成し，ホスト側の残り資源 (4CPU コア，3GPU) からなる GPU ジョブクラス (G キュー) と仮想マシンによる CPU ジョブクラス (U キュー，V キュー) を作成し，ユーザにに提供している．U キューおよび V キューの差異は課金方式の違い (V キューのみ購読方式の月額課金，それ以外が従量課金) によるもので，ジョブの性質によるものではないので，負荷に応じて計算資源をキュー間で動的に融通しあっている．

2.2 ジョブのバックフィル処理



(a) バックフィル無効　(b) バックフィル有効

図 1 ジョブスケジュール順序の例: 図中の数字はジョブ投入順

TSUBAME2 では，**図 1**(a) に示すようなジョブ投入順による早い者勝ちのスケジューリングではなく，(b) に示すように，先行ジョブの実行を阻害しない範囲で空いているノードを用いた後続ジョブの実行を許すバックフィル処理を有効にしている．図 1 の例では，4 番のジョブがスケジュールされると後続の 5 番〜9 番のジョブが 4 番の終了を待たない限り実行できないところ，5 番を除く 6 番〜9 番のジョブを先に実行しても 4 番のジョブの開始時刻には影響がないために，これらを先に実行することができる．なお，この例においては 5 番のジョブはどのノードに配置しても 4 番のジョブの開始時刻までに終了しないため，実行を許してしまうと 4 番のジョブの実行開始が遅れるため，バックフィル処理の対象にはならない．以上の仕組みによって計算資源の有効活用とジョブの待ち時間の削減が実現される．

2.2.1 バックフィル処理における問題点

図 1 の例において，仮に 1 番〜4 番までのジョブについては指定終了時間と実際の終了時刻が一致しており，5 番〜9 番のジョブについては実際の実行時間の 3 倍の実行時間が指定されていると仮定すると，6 番〜9 番のジョブをバックフィル処理で先行させようとしたときにそれらが 4 番のジョブの開始時刻までに終了することがスケジュール時点では保証できない．このため，バックフィル処理が行われず，ユーザにとっては終了時間の遅延，資源提供者にとっては利用効率の低下という問題が発生する．この問題の唯一の解決方法は，ユーザにより正確に実行時間を見積もってもらい，正確なデータでスケジューリングを行うことである．しかしながら，指定実行時間を超過したジョブは強制終了され，多くの場合結果がえられなくなってしまうため，ユーザは可能な限り長い実行時間指定をする傾向にあると考えられる．

2.3 従量利用の課金計算式

TSUBAME2 では運用開始時より，以下の計算式により従量課金の各キューの利用料 P を算出してきた．

$$P = q \times T_u \times N \times e \times p \tag{1}$$

上式において q はジョブクラスに応じて変化する係数，T_u はジョブの実実行時間 (秒)，N は利用したノード (もしくは VM) 数，p はスケジューリング優先度に応じて変化する係数，e は下式に示すとおり指定実行時間 T_s に応じて変化する係数である．

$$e = \begin{cases} 1 & (T_s \leq 24\text{hours}) \\ 2 & (24\text{hours} < T_s \leq 48\text{hours}) \\ 4 & (48\text{hours} < T_s) \end{cases} \tag{2}$$

このように，指定実行時間が課金額に与える影響は係数 e のみであるので，経済的に実行するという観点からは，指定実行時間として 24 時間を指定することが最適となり，多くのジョブの指定実行時間が 24 時間であり，バックフィルが全く効かない状態となっていた．

3. 指定実行時間最適化への取り組み

3.1 利点アナウンスと課金係数の変更

このように，バックフィル処理があるにもかかわらず有効に利用されていない現状を打開するために，2014 年 2 月に短い実行時間指定によるスケジュール順の面における利点をアナウンスするとともに，2014 年 4 月より課金係数 e の計算式を次式のように変更した．[3]

$$e = \begin{cases} 0.9 & (T_s \leq 1\text{hour}) \\ 1 & (1\text{hour} < T_s \leq 24\text{hours}) \\ 2 & (24\text{hours} < T_s \leq 48\text{hours}) \\ 4 & (48\text{hours} < T_s) \end{cases} \tag{3}$$

指定実行時間 T_s が 1 時間以下のジョブについては従来の 90%の課金額で実行できるようにインセンティブを与える一方，それ以外のジョブについては従来通りの課金額とする変更である，

3.2 課金計算式の変更

また，上記係数 e の変更に加えて，2014 年 8 月からさらなる課金計算式の変更を行うことを 2014 年 2 月に予告し，2014 年 4 月より新計算式による推定課金額の表示を開始，2014 年 8 月より予告通り新計算式の適用を開始した．[3] 変更後の課金計算式は式 (1) で定義されていた課金額の計算式を下式のように変更するものである．

$$P = q \times (0.7T_u + 0.1T_s) \times N \times e \times p \tag{4}$$

本式では，ジョブの実実行時間 T_u だけではなく，スケジュールに使われる指定実行時間 T_s についても明示的に課金対象としている．同じ実実行時間で比較すると，指定実行時間が実実行時間の 3 倍以下であったとき，すなわち実行時間の予測が 3 倍よりも高い精度でできた場合には従来より課金額が低く，逆に実行時間の予測精度が 3 倍よりも悪いときには従来よりも課金額が高くなる．このような課金計算式を用いることで，安易に T_s が 24 時間に設定されたジョブが少なくなることを期待している．なお，実実行時間が T_u が 5 分に満たないジョブについては，起動時の初期化やパラメータ設定に失敗したと見做し，式 (4) によらず，式 (1) を適用している．

4. 取り組みの検証

著者ら TSUBAME2 運用チームでは，上記計算式の変更後については，バックフィルが効かずにジョブが滞留する現象が軽減されたことを体感してはいたが，その効果について定量的には検証を行ってこなかった．また，大学設置のスーパーコンピュータの負荷は学事歴および予算年度の関係で 4 月から 10 月頃までの閑散期 (この間には 7 月〜9 月の節電運用期間も含まれる) と 11 月から 3 月にかけての繁忙期では大きく異なり，制度変更の前後の負荷状況の変化が制度変更によるものなのか年周期の他要因によるものなのかが分からない．そこで，TSUBAME2 の供用開始から 2015 年 3 月までの PBS の実行ログを解析することにより，本取り組みの効果について検証を行った．

4.1 ユーザによる実行時間予測精度の検証

2014 年 4 月および 8 月の課金計算ポリシーの変更の前後

で，ユーザのジョブ実行時間予測にどのような変化が発生したかを検証した．**図 2** にキュー毎のジョブの T_u/T_s 比の T_u による加重平均値の推移を示す．加重平均値の算定に当たっては，ノードのハングアップ等で T_u が T_s の 1.2 倍を超えたケースを異常値として，また 5 秒以内にジョブの実行が終了したケースを初期化エラーに起因するものとして除外している．グラフ中の縦実線はそれぞれ課金計算式の変更が行われた 2014 年 4 月および 8 月を，破線は他年度の 4 月および 8 月を示す．

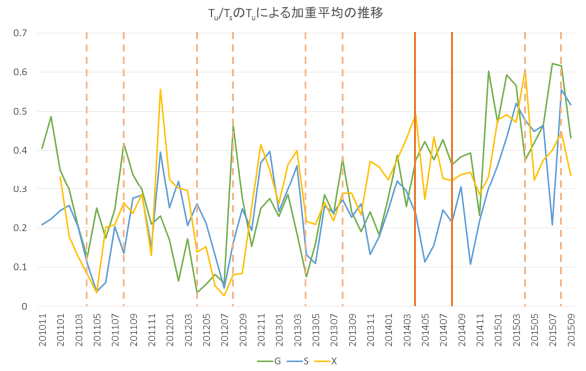


図 2 キュー毎の平均 T_u/T_s 比

年度間の同月どうしを比較すると，2014 年 4 月の計算式変更によって例年低い値を示している閑散期の T_u/T_s 比が大きい (予測精度が高い) 値で推移し，2014 年 8 月の変更後も例年の繁忙期の水準と比しても高い T_u/T_s 比を維持したことがわかる．

また，キュー間の比較では，同じ従量課金でも，S キューに振り分けられたジョブに比べて X キューに振り分けられたジョブや，G キューに投入されたジョブのほうが T_u/T_s 比が高くなる傾向が 2014 年 4 月以降に明確に現れている，これは，即時の実行を望みジョブの実行条件を X キューに合わせて適応させるユーザや，GPU をメインで使うユーザ，すなわち TSUBAME2 のアーキテクチャやスケジューリングポリシーに積極的に適応するユーザが我々の狙い通りにジョブの実行時間の予測を詳細化したものと考えられる．また，2015 年 1 月頃から，S キューに振り分けられたジョブについても従来に比べて T_u/T_s 比が高くなる傾向が現れており，これらの施策が一部の先進ユーザだけではなく，広範囲のユーザの行動を変化させていることが分かる．

なお，これらの施策を導入する際や導入後も，アプリケーションユーザからは実行時間が事前には予測ができないとの意見を受けた，ユーザレベルチェックポイントなど，計算途中でジョブの実行時間が尽きることが問題とならない手法の普及に努めたい．

4.2 ジョブの待ち時間の変化の検証

本施策の最終目的は，ジョブの実行効率を改善して各ジョブの待ち時間 (ジョブが実行可能になってからジョブが実行開始されるまでの時間) を短縮することであり，本来は待ち時間の変化も検証の対象とすべきである．しかしながら，TSUBAME2 に記録されていた実行ログを精査したところ，ジョブの間にユーザが依存関係 (特定のジョブが実行終了を条件として実行開始可能とする) を設定した場合，実行開始可能となった時刻，依存関係の指定の存否およびその内容がログファイルに記録されておらず，検証を行うことができなかった．このような検証に必要なデータを失わないよう，十分な情報量をもつログを保存することは次期システムにおける課題である．

5. モニタリング情報とジョブ実行履歴の連携

5.1 TSUBAME2 における Ganglia による環境モニタリング

TSUBAME2 では，計算ノードや，配電盤において，以下に例示する様々な指標を 1 分単位で計測し，Ganglia を用いて記録・公開している．[4]

- ノード
  - CPU 負荷
  - Load average
  - メモリ使用量
  - GPU 負荷
  - Ethernet I/O
  - 各コンポーネントの温度
- 各分電盤の電力
- ストレージサーバの負荷

5.2 ジョブ実行履歴との連携

これらは単なる時系列データとして現在公開されているが，バッチジョブスケジューラの実行ログ情報と組み合わせることで，ジョブ単位の負荷や I/O 等の統計情報を得ることができると考えられる．しかし，公開されている測定データは計測点 (ノード等) および計測項目ごとに RDD ファイルとして保存されており，400 日分のデータを格納するためには約 450GB のストレージ領域を使用している，今回はこの RRD ファイルの年次バックアップデータをもとに解析を行うので，5 年度分のデータを検索対象とすると探索範囲が 2TB 以上となり，ここから統計情報をリアルタイムに生成することは現実的ではない．

そのため，データ量を削減するためにあらかじめ RRD ファイルから必要な指標を事前にノード単位・ジョブ単位で統計処理 (平均もしくは加算) を行うことで，データベースのサイズ (探索範囲) を削減することとした．

5.3 GPU 利用率の検証

TSUBAME2 では，ジョブの実行時に資源指定として GPU の利用台数の入力を求めているが，実際には資源指定

をせずとも GPU が利用可能であり，また資源指定をしていても実際には GPU が使われていないケースが考えられ，GPU の利用率や利用状況を確認するうえで，資源指定の値をそのまま使うことは正確性の面で問題がある．また，GPU ジョブのキューとして作成している G キューに GPU を用いないジョブが過度に投入されていると，GPU を利用するユーザの機会損失につながるため，適切なキューにユーザを誘導する必要があるが，そのようなジョブを検出するうえで精度の低い資源指定を用いることに問題がある．一方，GPU の負荷情報は Ganglia によるモニタリングログとして収集しており，ジョブの実行期間中に負荷が 0 を超えた GPU の台数を調べることで，実際のジョブ毎・ノード毎の GPU 使用台数を算出することができる．

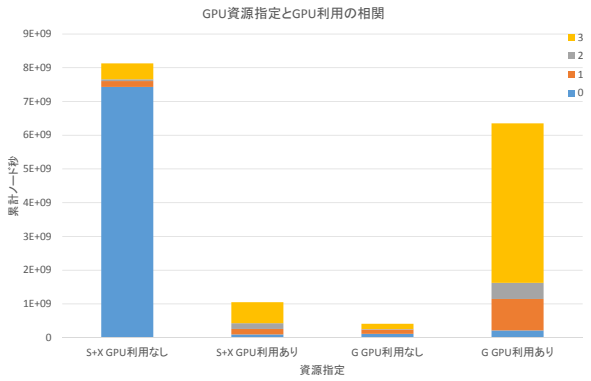


図 3　GPU 資源指定と実際のノードあたり GPU 利用台数の関係

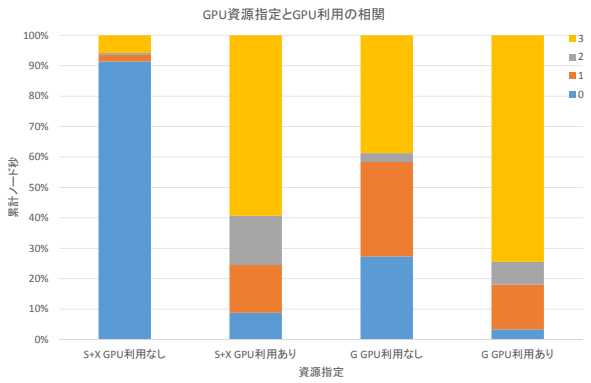


図 4　図 3 の各棒を 100%に拡大したもの

図 3 および**図 4** に 2014 年度の S，X，G の各キューにおける GPU 資源指定の有無と実際の 1 ノードあたりの GPU 利用数の相関を示す．それぞれのキューにおいて，不適切な資源指定が一定の割合で存在することおよび GPU ジョブ専用の G キューにおいても，ノード時間において約 5%のジョブは実際には GPU を全く利用していないことが分かった．

5.4　今後に向けて

今回は，モニタリングログとスケジューラログを事後的に組み合わせることで，TSUBAME2 の使われ方の検証を行った．そのため，スケジューラログをもとにしたモニタリングログの後処理が必要となり，事後的に検証を行うことはできるものの，ジョブを実行したユーザにモニタリング情報を即時で提供するなどの使いかたは現システムでは行うことができない．次期システムの構築においては，モニタリング情報をリアルタイムでジョブ情報と関連付けて，システムの使われ方の把握と，ユーザへのより踏み込んだ情報提供へとつなげたい．

6.　関連研究

ユーザの資源消費動向を経済的インセンティブによって誘導する試みは FOCUS スーパーコンピュータシステムにおいても行われている．[5] では，ジョブの並列数を向上させるために課金計算式を年ごとに変更する施策と，運用期間中のジョブ並列数分布の変化を報告しているが，施策による消費動向の変化と，大口ユーザの移動やシステムの追加などの要因が課金計算式変更を行った年度境界で発生しているため，原因と結果の因果関係が明らかではない．

7.　おわりに

本報告では，TSUBAME2 の運用におけるジョブスケジューラの運用効率化のために，ユーザに経済的インセンティブを与える施策について紹介し，その効果を測定するためのに行ったスケジューラ実行ログの解析について述べた．施策導入時には定量的に評価されていなかったユーザ動向の変化について，実際に効果があることが定量的に示されただけでなく，モニタリング情報と組み合わせることで，TSUBAME2 の運用および次期システムの設計の資料となりうるユーザジョブの計算機上での振る舞いについて，把握できる事例を示した．今後は得られたジョブ実行履歴とモニタリング情報をもとにどのような情報が得られるかを詳細に検討し，日々の運用と次期システムの設計に生かしていきたい．また，4.2 節に示したジョブが実行可能になった時刻等，運用資料として必要な項目が収集されていないことが明らかになったので，次期システムでは，必要とされるジョブの実行履歴情報およびモニタリング情報の項目を同定し，記録するように設計する必要がある．

参考文献

[1] Global Scientific Informantion and Computing Center, Tokyo Institute of Technology: TSUBAME Computing Serices.
[2] 遠藤敏夫，額田　彰，松岡　聡：ウルトラグリーンスパコン TSUBAME2.5/TSUBAME-KFC，大学 ICT 推進協議会 2013 年度年次大会 論文集 (2013).
[3] 東京工業大学学術国際情報センター：正確な walltime 指

定を促進し、TSUBAME の稼働率を高めるための消費ポイント計算式の変更について.
[4] Global Scientific Informantion and Computing Center, Tokyo Institute of Technology: TSUBAME 2.5 Monitoring Portal.
[5] 西川武志：FOCUS スーパーコンピュータシステムにおける並列課金インセンティブの効果, 情報処理学会研究報告, Vol. 2015-HPC-149, No. 2, pp. 1–5 (2015).