論文 / 著書情報
Article / Book Information

論題(和文)	スパコンTSUBAMEシリーズにおけるリソース分割戦略
Title(English)	
著者(和文)	野村 哲弘, 遠藤 敏夫
Authors(English)	Akihiro Nomura, Toshio Endo
出典(和文)	情報処理学会研究報告, Vol. 2024-HPC-195, No. 7, pp. 1-7
Citation(English)	IPSJ SIG Technical Report, Vol. 2024-HPC-195, No. 7, pp. 1-7
発行日 / Pub. date	2024, 8
権利情報 / Copyright	

スパコンTSUBAMEシリーズにおけるリソース分割戦略

野村 哲弘[1]　遠藤 敏夫[1]

概要：東京工業大学のスーパーコンピュータ TSUBAME シリーズにおいては，1 台の計算ノードに複数の GPU が搭載されているため，全てのユーザが計算ノードの能力を十全に扱うことが困難であったため，計算ノードを仮想的に分割してユーザのニーズに合う形の計算ノードを提供することに努めてきた．計算ノードの仮想分割は，限られた台数の計算ノードをより多くのユーザが利用できるようにする点においてもその必要性が年々増している．本稿では TSUBAME2.0(2.5)，TSUBAME3.0 および TSUBAME4.0 において実施した計算リソース分割の戦略について報告する．

1. はじめに

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，2024 年 4 月からは現行の TSUBAME4.0 [1] を供用している．TSUBAME シリーズは 2006 年に導入された TSUBAME1.0 以来，学内外で 1,300 名 (2023 年度 TSUBAME3.0 アクティブユーザ数) を超えるユーザに利用されている．

TSUBAME シリーズでは 2008 年に TSUBAME1.2 として NVIDIA Tesla S1070 を既存の TSUBAME1.0 計算ノードに後付けして以来，GPU を積極的に活用したスパコンとして知られ，2010 年の TSUBAME2.0 では NVIDIA Tesla M2050(2013 年に NVIDIA Tesla K20X に交換 (TSUBAME2.5)) をノードあたり 3 基，2017 年の TSUBAME3.0 では NVIDIA Tesla P100 をノードあたり 4 基，2024 年の TSUBAME4.0 では NVIDIA H100 SXM5 HBM2e 94GB をノードあたり 4 基備え，いずれの世代においても (メモリ増強ノードなどの例外を除いて) 複数 GPU を備える単一構成の計算ノードを備えてきた．

TSUBAME シリーズにおける演算能力の大きな割合が GPU によることは明らかではあるが，TSUBAME の全てのユーザが計算ノードに備えられた複数の GPU を活用できるわけではない．また，TSUBAME シリーズでは代を経るごとに計算ノードの台数が半数以下へと減少しており，そのままではユーザに提供可能な資源量 (同時実行可能ジョブ数) も同様に激減せざるを得ない状況にあった．そのため，GPU を必要としていないユーザ等にむけて，その時代にそれぞれ利用可能であった技術を活用し，計算ノードを仮想的に分割することで，それぞれのユーザに必要十分な資源を見せるとともに，それらのジョブを 1 台の物理計算ノードに集約することによって，遊休資源の最小化とともに，計算ノードの台数以上の計算要求を満たすことに努めてきた．

以降の各章では，TSUBAME の各世代におけるリソース分割における制約条件と，その実装について説明する．

2. TSUBAME2 におけるリソース分割戦略

TSUBAME2.0 [2] および 2.5(以下，リソース分割の観点では相違がないため，TSUBAME2 と総称する) の 1408 台の Thin 計算ノードは，2 台の Intel Xeon X5670 CPU(Westmere-EP, 物理 6 コア) と 3 台の NVIDIA GPU を持ち，54GiB のメインメモリを備えたノードである．TSUBAME2.0 導入当時はまだ CUDA 以外の GPU プログラミング環境が十分ではなく，限られたユーザのみしか GPU で動作するコードを用意することができなかったため，CPU による計算需要が極めて高く，そのようなジョブを Thin 計算ノードで実行すると，大多数の GPU が使われない状況となることが予想された．

2.1 VM によるノード分割

TSUBAME2.0 の導入当時は Linux で実用的に利用可能なリソース分割手法は仮想マシン (VM) によるものしかなかった．また，当時の VM 実装では，PCI-Express デバイス (GPU および InfiniBand HCA) の仮想化が十分ではなく，特に SR-IOV を利用してホストと VM もしくは複数の VM から一つのデバイスの機能を活用することはできなかった．そのため，**図 1** に示すように，計算ノード内にて

[1] 東京工業大学学術国際情報センター

CPU ジョブ用の VM を KVM を用いて実行し，GPU プロセスは VM を実行しているノードのベアメタル環境 (ホスト環境) において動作させる設計が行われた．メインメモリは 32GiB を VM に与え，ホスト環境には 22GiB のメモリが残された．これは TSUBAME1.2 の計算ノード相当のメモリを VM で提供しつつ，当時の NVIDIA Tesla M2050 および K20X の GPU メモリ量 (それぞれ 3GB, 6GB)3 台分を超えるメモリをホスト環境に残したものである．

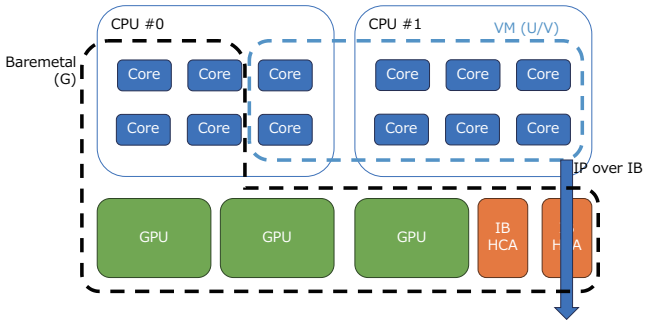

図 1 TSUBAME2 計算ノードの資源分割イメージ

VM で実現された CPU 処理用仮想計算ノードからは PCI-Express デバイスが直接扱えないため，InfiniBand のドライバおよび API を利用することはできず，KVM による IP ネットワーク仮想化機能とホストにおける IP over IB を利用した IP 通信しか行うことができない．そのため，ホスト環境で用いている InfiniBand Verbs および RDMA を利用した MPI 実装 (Open MPI および MVAPICH) を利用することはできず，別に用意した TCP/IP 通信を用いる MPI 実装 (Open MPI) を利用せざるを得なかった．そのため，ノード間通信および Lustre, GPFS ストレージのアクセス性能がホスト環境に較べて低速であった．

ノードを共有するプロセス間の情報が相互に見えないという要件は，学術利用だけではなく産業利用のユーザを抱える TSUBAME シリーズにおいて重要な要件であるが，片方が KVM による VM のため，ホスト環境からは VM があること以上のことは見えず，VM 環境からはホスト環境に関する情報は一切得られなかったため，環境のアイソレーションという側面では都合がよかった．一方で，スケジューラの視点ではホスト環境と VM 環境を別個の独立した計算ノードとして登録する必要があり，ベアメタル環境のみの場合と較べてスケジュール対象が増えることとなった．

TSUBAME2 では PBS を用いて計算ノードのスケジューリングを行っていた．1 台のスケジューラサーバでは 1408 台+VM 分のジョブをさばききることができるか性能上の懸念があったため，計算ノード群をおおよそ 3 分割し，それぞれを 1) ノード分割を行わない通常ジョブ用ノード群，2) ノード分割を行った通常ジョブ用ノード群，3) 日単位のカレンダー予約用のノード群 に分けてそれぞれ別のスケジューラインスタンスとして管理し，ユーザ向けには各インスタンスへの `qsub` コマンドのラッパーとなる `t2sub` コマンド等を提供した．スケジューラインスタンス間の計算ノードの所属関係を変更することは，関連する VM の起動・終了を含めて自動化もしくは日常的に実施する作業の作業量として現実的ではなく，それぞれのノード群におけるノード台数は固定である．3) のカレンダー予約用のノード群においては，予約が入っておらずジョブを実行していないノードに対して 1) の通常ジョブとして投入されたジョブのうち，次の予約スロットの開始までに実行可能なジョブを `t2sub` ラッパーが転送することで，遊休資源の解消に努めたが， 2) のノード分割を行うノード数は固定であり，ノード分割ジョブとノード非分割ジョブの間の負荷の不均衡の解消は困難であった．また，カレンダー予約については毎朝 9 時から 10 時の間に PBS におけるキューを再構成する必要があり，基本的には自動化がなされていたものの，ノード故障などで自動確保したキュー構成が予約者の要求ノード数を満たせなかった場合に手動で割り当てのやり直しが必要となるなど，予約時間枠をフレキシブルに設定することが困難な状況にあった．

3. TSUBAME3.0 におけるリソース分割戦略

TSUBAME3.0 [3] の 540 台の計算ノードは，2 台の Intel Xeon E5-2680 V4(Broadwell, 物理 14 コア) と 4 台の NVIDIA Tesla P100 SMX2 をを持ち，256GiB のメインメモリを備えたノードである (**図 2**)．TSUBAME2 におけるリソース分割の制約事項を解消するため，リソース分割機構を見直した．TSUBAME2 の Thin 計算ノード 1408 台に対して，TSUBAME3.0 の計算ノードは 540 台と，3 分の 1 強のノード数しかないため，TSUBAME2 と同レベルのリソース分割では，TSUBAME ユーザの計算需要を捌ききれないことが明らかであり，より積極的なリソース分割が必要となる状況であった．

3.1 Linux cgroup によるノード分割

TSUBAME3.0 におけるリソース分割の設計時には以下のような要件を念頭に検討を行った．

- GPU や相互結合網がリソース分割機構のせいで遅くならない
- リソース分割しても今までできていたことができる
  - 計算ノードへの直接ログインによる対話的操作
- 計算ノードを共有する各ジョブが互いに相手を見ることができない
- 計算ノードの分割方法を動的に変更できる
- ジョブに応じて最適なユーザランドを選択できる
  セキュリティパッチを容易に当てることができる

上述の要件のうち，最後の点についてはリソース分割とは直接は関係しないが，TSUBAME2 におけるリソース分

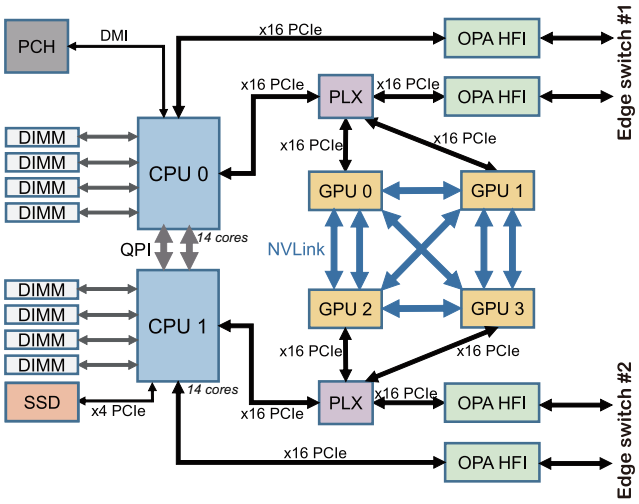

図 2 TSUBAME3.0 計算ノードのブロックダイアグラム

割の実装が VM であったことから，リソース分割と実行環境仮想化の両方の概念を「仮想化」の語のもとに混同しており，あわせて要件として加えられたものである．

これらの要件と，導入当時の技術トレンドから，TSUBAME3.0 においては Docker を活用したリソース分割を行うこととした．しかしながら，Docker は上記要件の最後の点を除いたリソース分割の観点では本質的には Linux cgroup でしかなく，スケジューラである Univa(現 Altair) Grid Engine(以下，Altair のものも含めて AGE) [4] への組み込みの際には，cgroup による資源分割と Docker による実行環境仮想化が独立して実装された．

なお，Docker による実行環境仮想化に関しては，Docker における root ユーザがホスト環境における root ユーザと同レベルの権限を持つ構造であるため，一般のスパコンユーザに (Docker 内の root を容易に取得できる) 任意の実行イメージを持ち込ませることを禁止せざるをえず，本センターによるイメージ管理を行うにもベアメタル環境に加えて複数の Docker コンテナの管理を行う人的リソースがなく，実質的に利用されることなく終わり，実行環境仮想化の役割に関しては TSUBAME3.0 運用期間中に追加で整備した Singularity(現 Apptainer) に譲られている．

Linux cgroup による資源分割をスケジューラと協調して行うことで，スケジューラの設定を変えることなく計算需要に応じて動的に計算ノードの分割を変更することができるようになった．TSUBAME3.0 の計算ノードの分割パターンの模式図を**図 3** に示す．実装には AGE の RSMAP 機能を用いて，CPU コア番号と GPU の対応付けを定義し，**表 1** に示す資源タイプに沿った形で計算ノードが切り出されるように設計した．

資源タイプは大別して 3 種類に分類でき，それぞれ 1) CPU・GPU をともに利用するグループ， 2) CPU のみを利用するグループ，3) 主に GPU で計算を行うグループ と

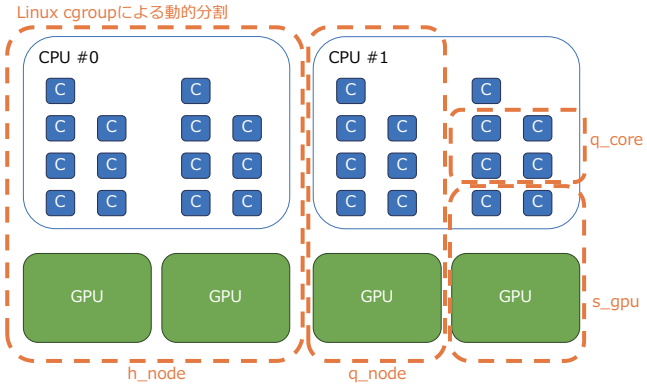

図 3 TSUBAME3.0 計算ノードの資源分割イメージ

表 1 TSUBAME3.0 における資源タイプ一覧

資源タイプ	CPU コア数	GPU 台数	メモリ [GiB]
f_node	28	4	235
h_node	14	2	120
q_node	7	1	60
s_core	1	0	7.5
q_core	4	0	30
s_gpu	2	1	15

特徴づけられる．資源の形状としては 1) が TSUBAME2 におけるノード非分割， 2) が VM による仮想ノード， 3) が VM を実行しているホスト環境 にそれぞれ対応する．このように資源タイプをお仕着せで定義せずに，ユーザに必要な CPU コア数，GPU 台数，メモリ容量をそれぞれ宣言させて cgroup を作る方式も検討したが，たとえばメモリの要求量が CPU・GPU に対して過剰となり，ジョブを割り当てられない CPU・GPU が出てくるなどの資源配分のインバランスが起こることが懸念されたため，それぞれの資源を比例で配分し，小さい資源タイプが大きい資源タイプに完全に包含されるよう資源タイプを定義して，ユーザには自分のジョブの要求に合った資源タイプを選択してもらう形を取った．また，資源タイプを事前定義することにより，確保した GPU と近い側の CPU ソケット内のコアが確保されることを保証し，CPU・GPU 間通信の性能が確保できることとなった．

AGE における制約事項として，1 つのジョブが複数の仮想ノードを要求する MPI ジョブであるときに，ジョブを構成する仮想ノードを同一の物理ノードから複数とることができない (パッキングした配置ができない) という問題があった．これは，ジョブ内で各仮想ノードを指定する際に物理ノードと同様にホスト名で弁別していることに起因する．配置のパッキングを行うことで，MPI プロセス間の通信がローカル通信になることや，ジョブが詰め込まれるためにスケジューリング効率が向上するなどの利点を予期していたが，実運用上はこれらの点が問題になることはなかった．また，同様の理由により，ジョブを実行しているノードに SSH で接続し X11 転送を利用したデバッガ GUI

の利用などの作業を行うという TSUBAME2 まででは制約なくできていた操作が TSUBAME3.0 では f_node 以外の資源ではできなくなった．こちらは AGE の qrsh コマンドにてジョブの実行開始と同時にシェルアクセスを得るという限定された条件下ではあるが代替することができた．

一方で，ノードの分割に VM ではなく cgroups を使うこととなり，複数のジョブが OS の名前空間を共有することとなった．このため，ユーザプログラムが原因でノードがダウンする事態が発生した場合に，同一ノードを共有している他のユーザのジョブが巻き込まれやすくなるとともに，OS 名前空間を共有することで，あるユーザのジョブが別のユーザのジョブに干渉しやすくなってしまった．計算資源に関しては，cgroups で分割することで他のジョブのためのプロセッサ・コアの利用を制限しており，性能上の問題はほぼ起こらないと考えている．他のユーザのジョブが見える点については，/proc ファイルシステムのマウントオプション hidepid を用いることで，他のユーザのプロセス情報を遮断することができている．しかしながら，通常の Linux 環境ではあまり用いられていない設定のためか，この設定に起因するバグを誘発するなどの欠点も存在する [5, 6]．

ノードのカレンダー予約については，AGE の Advance Reservation(AR) 機能を用いて実装した．ノード全体を単一のキューに入れ，キュー内に予約枠を作る形となるため，任意の時刻に開始する予約を作成することができるようになった．TSUBAME3.0 では予約用のユーザインタフェースの作成や，課金規則との兼ね合いを考慮して毎時 0 分に予約を切り替える 1 時間単位の予約として実装した．予約はノード単位であるが，予約後のキューには q_node などのノード分割を伴う資源タイプのジョブを投入することができる．

ノードを動的に分割することとしたため，TSUBAME2 のようにスケジューラインスタンスを分けて計算ノードを固定割り付けする必要がなくなった．このことはフレキシビリティの面では有利ではあったが，1 台のスケジューラインスタンスで全てのジョブを捌くことを意味しており，スケジューラ負荷の面では不利な変更となった．

3.2 インタラクティブジョブ専用キューの分離

スケジューラインスタンスが 1 台しかないことにより，定常的にスケジューラが高負荷となり，応答性能が悪化した．通常のバッチジョブの場合，ジョブの実行開始のタイミングをユーザが待ち構えていることはなく，実行開始オーバヘッドは体感できないものであるが，デバッグなどでリアルタイムに計算ノードにアクセスする際には，ノードが空いていてもジョブ投入から実行開始まで 1 分以上待たされるという状況は許容しがたいものであった．また，TSUBAME3.0 は運用期間全体を通して年度初めの期間以外は定常的に混雑しており，利用したいと思った時に空いている計算ノードがないという状況も定常的に発生していた．

この問題に対処するために，TSUBAME3.0 では 2019 年 11 月よりインタラクティブジョブ専用キューを導入した．[7] TSUBAME3.0 を構成する 540 台の計算ノードのうち 4 台を通常のキューから外し，インタラクティブジョブの待ち受けに専従させるものである．インタラクティブジョブに求められた主な要件は [7] にあるとおり，以下のようなものであった．

- インタラクティブジョブの実行要求は即時に実現されるか，失敗する
- インタラクティブジョブの実行要求には可能な限り応える
- 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
- 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
- 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てよい (オーバーコミット状態)

スケジューラ上ではインタラクティブジョブ専用キューのジョブを 1CPU コアのみ利用する極小ジョブと定義し，cgroup としては q_node 相当の制限をかけることで 7CPU コアおよび GPU 1 台への最大 7 ユーザによる共有アクセスを実現した．メモリはプロセッサと違い，複数ユーザで共有できるものではないため，計算ノード上の 2TB のローカル SSD のうち 1.5TB をスワップ領域として利用し，提供可能なメモリ領域を確保した．

上記から，計算ノード 4 台で，合計 112 名まで同時に利用可能となった．これは TSUBAME3.0 のインタラクティブジョブ専用キューを利用した実習を行った際に，受講者全員が問題なく接続できる容量であるとともに，システムの 1%未満という本目的のために減らすことが許容されると思われるノード数とのバランスをとることができたと考えている．

実装当初は，通常のバッチジョブと同じスケジューラインスタンスでインタラクティブジョブ専用キューのジョブも処理していたが，ジョブスケジューリングの遅延が 1 分以上と大きく，ユーザの快適性を損ねていたため，スケジューラインスタンスを分離し，インタラクティブジョブ専用キュー専用のスケジューラインスタンスを提供したところ，応答時間が数秒程度に減少し，快適に利用できるようになった．AGE の qrsh コマンドはスケジューラインスタンス間の振り分けを想定していないため，インタラクティブジョブ専用キューに関するコマンドは iqrsh のように接頭辞 i を用いて区別することとした．

3.3 システム利用率に関する考察

TSUBAME3.0 の運用期間中における資源タイプ別ノード時間積の推移を**図 4** に示す．各グラフの折れ線は各月の日数に単純に 24(時間/日)×540(ノード) をかけたもので，メンテナンス等を一切考慮しない提供可能なノード時間積の理論最大値を示すが，実際にはメンテナンス以外にも，リソース分割による売れ残り部分や，ノード予約の開始間際で新規のジョブが開始できない時間など，さまざまな要因からシステム利用率 100%となることはない．

図 4 左図は各資源タイプの大きさを考慮せずに単純にノード時間積を足したものであり，理論最大値を超えている期間が多数発生している．このことは TSUBAME3.0 においてリソース分割を行わず全てのジョブを物理ノード上で実行した場合，540 ノードのシステムではその計算需要を全く捌ききれなかったことを意味する．右図は各資源タイプのノード時間積に，資源タイプごとに割り当てられる CPU コア数に応じた係数 (例: `q_node`: 1/4) をかけたものであり，こちらが TSUBAME3.0 のシステム充填率に近い指標となる．この指標で一番重点率が高かった月は 2019 年 12 月で，86.7%となっている．他方，この月のノード使用率 (ノードの一部だけでも使われていた時間を 1 とカウントしたノード単位の充填率) は 96.5%であった．これらの値から，リソース分割を行っても 89.8%もの高効率でジョブを充填することができていたことが示されている．

4. TSUBAME4.0 におけるリソース分割戦略

TSUBAME4.0 [8] の 240 台の計算ノードは, 2 台の AMD EPYC 9654(Genoa, 物理 96 コア) と 4 台の NVIDIA H100 SXM5 HBM2e 94GB を持ち，768GiB のメインメモリを備えたノードである (**図 5**)．TSUBAME3.0 の 540 台に対して，TSUBAME4.0 の計算ノードは 240 台と，半分弱のノード数しかないため，TSUBAME3.0 以上に計算資源の細分割が求められる状況であった．

4.1 Linux cgroup および MIG によるノード分割

TSUBAME3.0 におけるコンセプトを継承しつつも, CPU のコア数および GPU1 台の能力が格段に増加したことをうけて，リソースの分割単位を再検討した．TSUBAME4.0 に搭載された H100 では TSUBAME3.0 の P100 と違い，NVIDIA Multi-Instance GPU(MIG) [9] と呼ばれる GPU の仮想分割機構が導入されたため，GPU を仮想分割して，別ユーザのジョブに割り当てることを検討した．

MIG では GPU を最大 7 分割することができるものの，分割しない場合と比較して 1 割程度の CUDA コアが利用できなくなるオーバヘッドがあることと，GPU の一部でプログラムを実行中に GPU の他の部分の MIG 構成を変更することができるかが定かではなかったことから，MIG を利用しない状態と，MIG で GPU を 2 等分した状態のみを利用することとした．

また，AMD EPYC 9654 のコアは Chiplet と呼ばれるそれぞれ 8 コアの 12 個の塊に分割されており，チップレットを跨ぐプロセス配置を行った場合に，キャッシュアクセスなどの効率が低下するため，できるだけチップレット境界を跨がないような分割とする必要があった．

加えて，TSUBAME3.0 では CPU のみを利用するユーザが利用できる最大の資源タイプが `q_core` でメモリが 30GiB しか割り当てられておらず，メモリ量を必要とするタスクを実行するために，`f_node` などの GPU を含む資源タイプ指定を行い，GPU を余らせる結果となっている事例が観測およびユーザへのヒアリングで判明していた．

これらを踏まえて，**図 6** に示すとおりのノード分割を行った．各資源タイプの資源割当量は**表 2** に示す通りである．

表 2 TSUBAME4.0 における資源タイプ一覧

資源タイプ	CPU コア数	GPU 台数	メモリ [GiB]
`node_f`	192	4	768
`node_h`	96	2	384
`node_q`	48	1	192
`node_o`	24	1/2	96
`gpu_1`	8	1	96
`gpu_h`	4	1/2	48
`cpu_160`	160	0	368
`cpu_80`	80	0	184
`cpu_40`	40	0	92
`cpu_16`	16	0	36.8
`cpu_8`	8	0	18.4
`cpu_4`	4	0	9.2

TSUBAME3.0 との大きな差異は, MIG による 1/2 GPU が割り当てられる資源タイプを追加した点および，CPU 資源タイプにおいて，`node_q` の境界を越えて 80 コアおよび 160 コアを利用する資源タイプが追加された点である．スケジューラは TSUBAME3.0 に引き続き Altair Grid Engine(AGE) を用い，RSMAP による実装も流用している．後者の変更により，TSUBAME3.0 と違い単純な階層的分割とはならず，リソース割り当ての複雑度が増加したが，現時点で特段の問題にはなっていない．

MIG については，計算ノードがアイドル状態のときには各 GPU の MIG を有効にして GPU を 2 分割した状態で待機する．この状態で `gpu_1` や `node_q` などの GPU を分割しないジョブが割り当てられたタイミングでジョブのプロローグ処理において MIG の無効化を，ジョブのエピローグ処理で MIG の再有効化を行う．これらの処理は cgroups によって対象の GPU のみにアクセスできるよう制限された状態で実行するため，ノード内の他の GPU に影響を与えることなく実行できる．

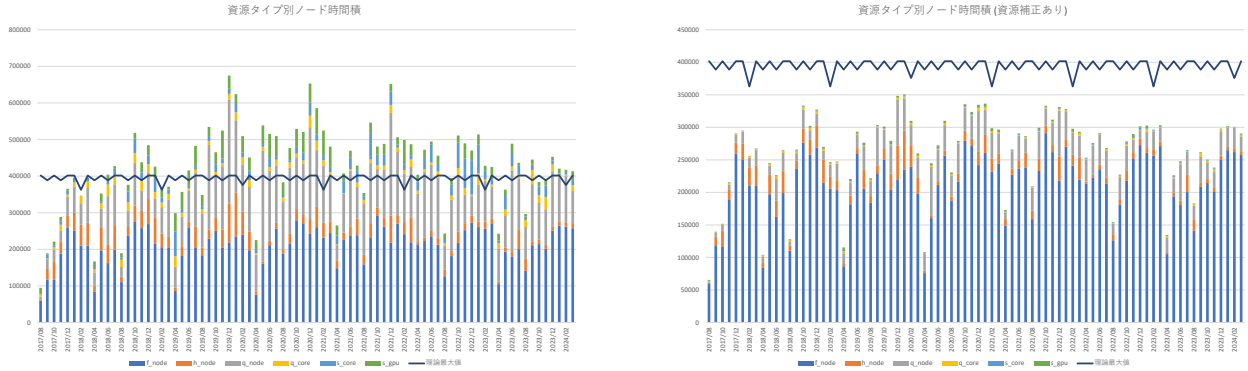

図 4 TSUBAME3.0 運用期間中の資源タイプ別ノード時間積の推移

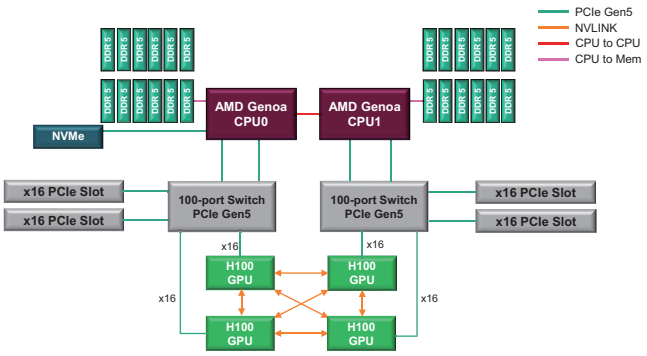

図 5 TSUBAME4.0 計算ノードのブロックダイアグラム

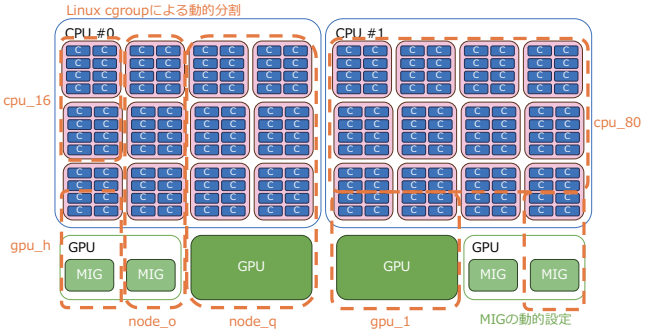

図 6 TSUBAME4.0 計算ノードの資源分割イメージ

4.2 定額制の実装

TSUBAME シリーズでは TSUBAME2 以降，原則としてジョブごとに TSUBAME ポイントと呼ばれるトークンを消費して実行する従量制の課金体系をとってきた．利用した分だけ負担し，ノードを事前確保，固定割り当てを行わない方式であるため，ジョブ充填効率の面では有利であるものの，ユーザ視点ではシステム混雑時に全てのジョブが等しく待たされることとなり，締め切りを抱えた計算の実施時に締め切りまでに計算が実施できる保証が得られず，不便であった．このような需要の一部はノード予約によっても解消できると考えられるが，システム繁忙期は予約枠も奪い合いとなり，予約したいタイミングで売り切れを起こしていたり，その一方で実験計画の変更などの理由で確保した予約ノードが十全に利用されないケースも観測された．

このような問題の緩和策として，TSUBAME4.0 では試行的に定額制による長期間のノード確保を実装した．定額制で確保されたノードは通常のキューから外れ，`prior` という名前のノードを確保したユーザのみが最高優先度でジョブを投入できるキューに所属する．このため，定額制でノードを確保したユーザは当該ノードを独占的に利用することができる．一方で，当該ユーザのジョブが実行されていない場合は，通常のキューにあるジョブのうち，実行時間が 1 時間を下回るジョブを低優先度で実行することを許可している．この仕組みによって長期間のノード独占によって必然的に生じる未利用時間帯を他のユーザに明け渡しつつ，ジョブ投入から 1 時間以内の実行を購入ユーザに保証することができるようになった．定額制キューについては，価格や最大確保可能ノード数などその在り方について検討の余地が残されており，試行中の利用状況を見極めてその継続如何を見極めたいと考えている．

4.3 インタラクティブジョブ専用キューの分離

TSUBAME4.0 においても引き続き，240 台中 2 台の計算ノードを通常のキューから外し，インタラクティブジョブ専用のキューを構成している．計算ノードの能力やコア数が TSUBAME3.0 と異なるため，cgroup の範囲を `node_o` 相当にするなどの調整を加えてはいるものの，基本的には TSUBAME3.0 と変わらない実装となっている．

5. おわりに

スーパーコンピュータを構成する部品の高度化と高価格化のトレンドは今後も続き，各センターにおいて同等の予算規模で調達可能な計算機のノード数は減少の一途を辿っている一方で，利用者の計算需要は高まり続けて

いる．そのような状況下において，東京工業大学ではその時々の技術に基づき，計算ノードを論理的に分割することで，同時に処理できる計算ジョブの数を増やし，もしくは維持してきた．TSUBAME2.0 では，VM 技術の活用により CPU のみを用いるジョブと GPU を主に用いるジョブを同一ノード上で同時に実行することができるようになった．TSUBAME3.0 では，Linux cgroups の活用により，動的にリソース分割方法を変更できるようになり，多数の資源タイプを定義することが現実的となった．そのため，各プロセッサの利用の有無だけではなく，仮想ノードあたりの GPU 台数など，プログラムの記述度およびユーザの資源要求により則った形で資源提供が可能になった．TSUBAME4.0 の資源分割は TSUBAME3.0 のものを基本的に踏襲しているが，NVIDIA MIG の利用による GPU の論理分割や，CPU のみかつサイズが大きい資源タイプの導入により，さらに柔軟に計算資源の要求に応えることができるようになった．

本センターでは，システムの利用状況や利用者からの問い合わせ，フィードバックをもとに，リソース分割方法に限らずシステムの運用に改善できる点がないか常に検討を続けている．大規模な設計変更はシステムの代替わりのタイミングでしか実行できないが，小規模な改善や機能追加の機会は常にうかがっており，そのためにも利用者から率直な意見をお寄せいただけると大変ありがたい．

謝辞　本研究は，東京工業大学のスーパーコンピュータ TSUBAME2.0，2.5，3.0 および 4.0 において実施した．

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.t4.gsic.titech.ac.jp/.
[2] 松岡　聡，遠藤敏夫，丸山直也，佐藤　仁，滝澤真一朗：TSUBAME2.0 の全貌，*TSUBAME e-Science Journal*, No. 1, pp. 2–4 (2010).
[3] 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).
[4] Altair: Altair Grid Engine, https://altair.com/grid-engine.
[5] GitHub singurality: fakeroot option does not work with hidepid procfs, https://github.com/apptainer/singularity/issues/4633.
[6] GitHub psutil: [Linux] ppid_map get PermissionError if /proc is mounted with hidepid=1, https://github.com/giampaolo/psutil/issues/2026.
[7] 野村哲弘，遠藤敏夫，三浦信一，朝倉博紀，越野俊充，草間俊博：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，情報処理学会研究報告，Vol. 2020-HPC-175, No. 23, pp. 1–6 (2020).
[8] 遠藤敏夫, 野村哲弘, 渡邊寿雄, 安良岡由規, 鶴見　慶：HPC-AI 時代に向けたもっとみんなのスパコン TSUBAME4.0，情報処理学会研究報告，Vol. 2024-HPC-195, No. 8 (2024).
[9] NVIDIA: NVIDIA Multi-Instance GPU User Guide, https://docs.nvidia.com/datacenter/tesla/mig-user-guide/index.html.