

# 利用状況に応じたジョブ投入権限制御の検討と実装

當山 達也<sup>1)</sup>, 戸田 庸介<sup>1)</sup>, 島袋 友里<sup>1)</sup>, 疋田 淳一<sup>1)</sup>

1) 京都大学 情報部

tohyama@kudpc.kyoto-u.ac.jp

## Study and implementation of job submission authority control according to usage status

Tatsuya Tohyama<sup>1)</sup>, Yosuke Toda<sup>1)</sup>, Yuri Shimabukuro<sup>1)</sup>, Junichi Hikita<sup>1)</sup>

1) Dept. of Information Management, Kyoto Univ.

### 概要

京都大学学術情報メディアセンターでは、システムの利用負担金の徴収方式を定額課金方式とし、契約期間と利用する資源量を事前に申請することで、ジョブ数やジョブの実行時間などに関係なく、承認された計算機資源の範囲内で自由に利用することが可能である。しかしながら、複数人が所属するグループにおいては、利用者ごとの利用可能時間を設定したいといった要望や、特定の利用者による過度な利用を制限したいといった要望が存在する。そこで、本稿では利用状況に応じたジョブ投入権限制御について検討した結果を報告し、実装した内容を紹介する。

## 1 はじめに

京都大学学術情報メディアセンター (以下、本センター) では、全国共同利用設備として、スーパーコンピュータシステムを運用し、学内外の研究者に対して計算機資源を提供している。

本センターの特徴的な点として、利用負担金徴収方式が挙げられる。多くのセンターでは使用時間に応じて課金する時間課金方式を採用しているが、本センターでは契約期間と契約ノード数に応じた定額課金方式を採用している。この方式によって、予算の見通しが立てやすく、研究ニーズに応じた自由度の高い利用が可能となっている。

しかし、複数の研究室から構成される大規模な研究グループや、学際大規模情報基盤共同利用・共同研究拠点 (JHPCN)、革新的ハイパフォーマンス・コンピューティング・インフラ (HPCI) などの共同利用拠点向けのバッチキューにおいては、利用者間の公平性の確保が重要な課題となっている。これらの大規模プロジェクトや共同研究では、多数の研究者が同一のバッチキューを共有するため、一部の利用者による過度な資源利用によって他の研究者の計算機会が制限されるリスクがある。そのため、利用者間の公平性を確保するために、利用者が1年間に利用可能なCPU時間を設定したいといった要望や、特定の利用者による過度な利用を制限したいといった要望がある。

そこで、これらの要望を利用状況に応じたジョブ投入権限制御機能として定義し、運用に向けた検討と実装を行った。

本稿では、第2章で本センターのスーパーコンピュータシステムを紹介し、第3章で利用状況に応じたジョブ投入権限制御機能の導入に関する検討結果を報告した上で、第4章でその実装について紹介する。

## 2 スーパーコンピュータシステム概要

本センターのスーパーコンピュータシステムは、図1に示す通り Camphor3 (System A), Laurel3 (System B), Cinnamon3 (System C), Gardenia (System G), クラウドシステムからなる5種の演算システムと、大容量ストレージ、高速ストレージからなる2種のストレージシステムによって構成された、総演算性能 11PFLOPS、総メモリ容量 370TiB、総ストレージ容量 44 PB の高性能かつ大規模なシステムである [1]。

本センターでは、契約期間と契約ノード数に応じた定額課金方式を採用しており、契約ノード数を利用可能な専用のバッチキューの提供、契約ノード数に応じた大容量ストレージを提供している。専用のバッチキュー及び大容量ストレージに対するアクセス制御には、当該グループの利用者群をまとめた専用のグループを定義することで、利用権限を付与する方式をと

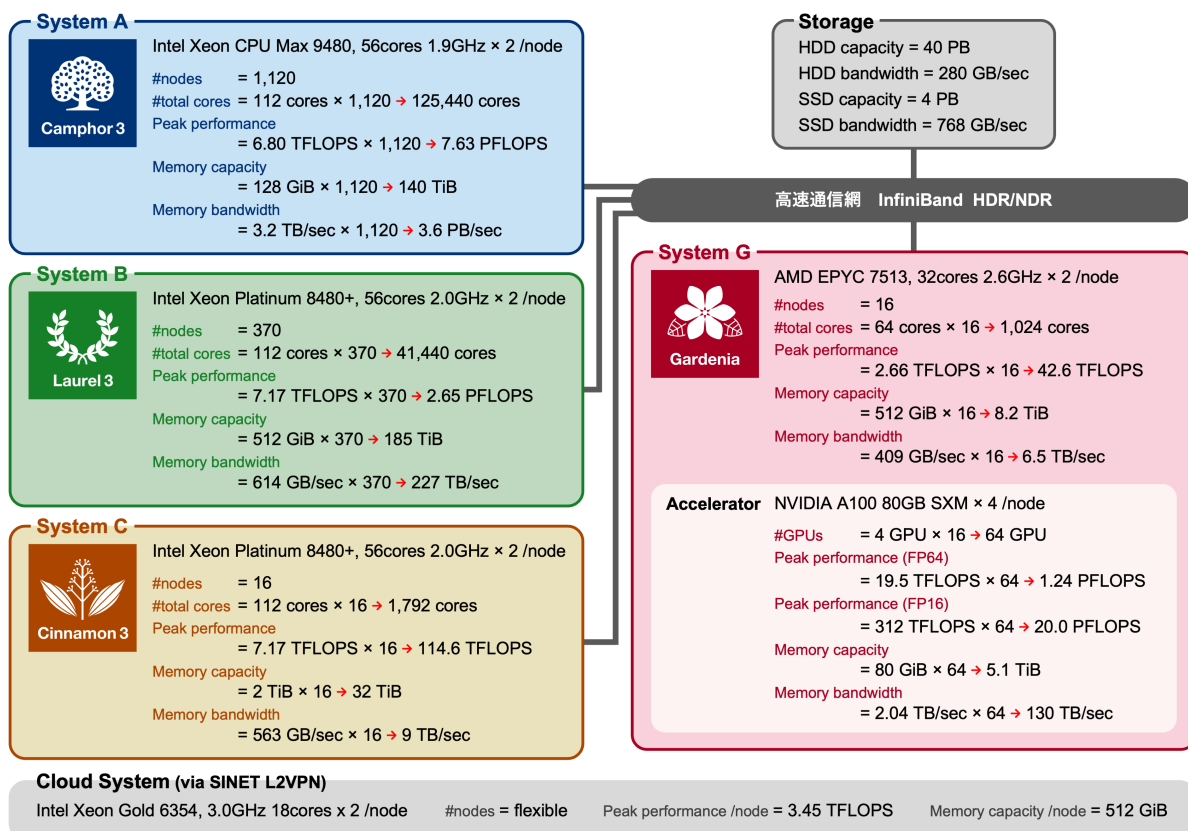


図 1 システム構成図

ている。また、グループの申請者および、申請者に指名された利用者をグループ管理者として定義する。グループ管理者は、管理者として指定されたグループに紐づくバッチキューの利用状況の確認や、特定ジョブの強制終了といった機能を利用することができる。

### 3 利用状況による制御方法の検討

本システムでは、本センター用に一部機能をカスタマイズした Slurm Workload Manager(Slurm)[2] というジョブスケジューラをシステムごとに立ち上げており、利用者間の公平性を担保するためにフェアシェアによるスケジューリングを行なっている。しかし、利用者が利用可能な CPU 時間を設定したいといった要望や、特定の利用者による過度な利用を制限したいといった要望については、Slurm の標準機能によって、ある一定の範囲では実現することが可能な面もあるが、柔軟な制御という観点を踏まえると、現時点では難しい状況である。

そこで、利用者からの要望に応えるべく「利用者が利用可能な CPU 時間の設定」および「特定の利用者による過度な利用の制限」について、実現可能性の検討を行った。

#### 3.1 利用可能な CPU 時間による制御

まずはじめに、利用者が利用可能な CPU 時間の残時間に基づき、ジョブの投入権限を制御する方法について検討した結果を報告する。

- バッチキューを構成するグループによって契約ノード数や、参加者の人数、参加形態などが異なる。このため、利用可能な CPU 時間については、システムで一律に設定するものではなく、バッチキューごとに柔軟に設定することが可能な設計とする。
- 利用可能な CPU 時間については、グループ管理者が簡易に指定できること。
- 利用可能な CPU 時間を設定したバッチキューについては、当該バッチキューの利用者に対して、設定した CPU 時間を初期値として付与する。ただし、グループの中で利用時間を多く設定しなければならない利用者がある場合に備えて、利用者ごとに利用可能な CPU 時間を増量する仕組みについても実装に含める必要がある。
- 利用可能な CPU 時間を超過した利用者については、当該のバッチキューに対してジョブ投入を拒

否する設定を行う。ただし、グループ管理者が当該利用者の利用可能な CPU 時間を増量することで、超過状態が解消した場合は、ジョブ投入を再度許可する。

以上の内容を実現することで、グループ管理者がバッチキューを共有する利用者の利用可能な CPU 時間を設定し、超過した利用者のジョブ投入を拒否することができると考えられる。これにより、グループに割り当てられた計算機資源を、各利用者の必要性に応じて適切に分配することで、特定の利用者が長期間にわたって計算機資源を過度に占有することを防ぐことが可能となる。

### 3.2 同時投入可能な資源の制限による制御

次に、特定の利用者による過度な利用を制限する方法について検討する。

3.1 節で検討した「利用可能な CPU 時間による制御」を実現することによって、CPU 時間の消費という観点では過度な利用の制限を実現することが可能になったと考えられる。しかし、利用可能な CPU 時間に抵触しない限り、特定の利用者が大規模なジョブを複数投入することが可能であり、特定のバッチキューで利用可能な資源を 1 人の利用者に占有される恐れがある。

そこで、CPU コアやノード数に応じて、ジョブの最大実行時間 (経過時間) や、投入数などを制限する方法について検討した結果を以下に報告する。

- 利用可能な CPU 時間による制御と同様に、システムやバッチキューごとに契約ノード数が異なるため、バッチキューごとに柔軟に管理可能な設計とする。
- 利用可能な CPU 時間による制御とは異なり、利用者単位での制御ではなく、バッチキューのポリシーとして設定する。
- バッチキューのポリシーとして、同時に投入可能なジョブの最大数やジョブの最大実行時間を、ノード数や CPU コア数に応じて設定できること。例えば、64 ノードの計算機資源を利用する契約をしたバッチキューのポリシーとして、4 ノードまでのジョブの最大実行可能時間は 48 時間、同時に投入できるジョブ数は 2 件まで。8 ノードまでのジョブについては、実行可能時間は 24 時間、同時に投入できるジョブ数は 1 件まで。それ以上の資源を要求するジョブの投入は認めない。といった設定を実現する。

- ジョブの投入可否については、ジョブ投入時点における実行中または実行待ちのジョブ数の合計と投入可能なジョブ数を比較して判定する必要がある。このため、リアルタイム性が強く求められる。

以上の内容を実現することで、グループ管理者は、ジョブの傾向に応じて、特定のユーザが同時に占有可能な資源を柔軟に制限することが可能となる。

## 4 利用状況による制御方法の実装

### 4.1 利用可能な CPU 時間による制御

利用可能な CPU 時間による制限を行うためには、すでに利用した CPU 時間を適切に把握する必要がある。利用者は複数のグループに所属していることも多く、利用可能な CPU 時間とすでに利用した CPU 時間はバッチキュー単位で比較する必要がある。

Slurm にはジョブの実行に使用した資源量や利用した CPU 時間に関する統計情報が記録されており、本センターでは 1 年分保持する設定としている。これにより、以下のようなコマンドを発行することで、指定した利用者が、集計開始日時から集計終了日時までの間に、指定したバッチキューを利用した時間を取得することが可能である。

統計情報の取得に使用するコマンドの例

```
$ sacct -r {キュー名} -A {利用者アカウント名} -S {集計開始日時} -E {集計終了日時} -s CD -o "cputimeraw" -X
```

上記コマンドを実行することで、利用者がジョブを投入した際に、当該利用者がすでに利用した CPU 時間の総量について検索し、その結果をもとにジョブ投入可否の判定を行うことは可能である。しかし、過去に投入したジョブ数の多寡や、検索する期間によってシステムへの負荷が懸念されるため、あまり好ましい実装とは言えない。また、利用した CPU 時間については、リアルタイム性が強く求められる要素ではなく、ある一定の閾値を超えた利用者のジョブ投入権限を剥奪できれば良い。そこで、制御対象のバッチキューを利用する利用者が利用した CPU 時間については、1 日に 1 度自動的に集計し、利用者ごとに利用可能な CPU 時間から利用した CPU 時間を減算することで、利用可能な CPU 時間の残時間を算出する実装とすることにした。

次に、利用可能な CPU 時間を超過した利用者に対してバッチキューの利用を制限する方法について検

討する。バッチキューの設定については Slurm の設定ファイルに直接記載されており、その中に利用可能な利用者アカウント名またはグループ名を指定する必要がある。Slurm の設定ファイルを変更した場合は、ジョブスケジューラだけではなく、ログインノードや計算ノードに対しても修正した設定ファイルを配布した上で、再構成を実施する必要がある。そこで、本センターでは、バッチキューを利用可能な権限設定については、グループ名を指定して制御する運用としている。

これにより、利用可能な CPU 時間を超過した利用者については、グループから外すことで、簡易にジョブ実行権限を剥奪することが可能となる。しかし、本センターでは、グループごとに大容量ストレージを作成し、グループに対してアクセス権を付与する運用としているため、利用可能な CPU 時間を超過した利用者をグループから外すと、ストレージへのアクセスも制限されることになる。このため、通常提供しているグループとは別に、バッチキューへのジョブ投入権限を管理するグループを新たに作成し、利用可能な CPU 時間に余剰がある利用者のみを登録することで、バッチキューの管理を行う運用とした。

最後に、グループ管理者向けの機能として、利用可能な CPU 時間の設定コマンドや、特定の利用者に対して利用可能な CPU 時間を追加するコマンド、グループに参加している利用者の利用状況を確認するコマンドをシェルスクリプトで作成した。いずれも、利用可能な CPU 時間による制御を有効なバッチキューが所属するグループの管理者はログインノードで実行することが可能である。

- `set_user_time`
  - ー 利用可能な CPU 時間を設定するコマンド
  - ー 利用可能な CPU 時間を引数に指定することで、設定することができる
- `add_user_time`
  - ー 特定の利用者に対して、利用可能な CPU 時間を追加するためのコマンド
  - ー 利用可能な CPU 時間に設定した時間を 1 単位として、追加する単位を指定することができる。例えば、利用可能な CPU 時間が 1000 時間として設定したグループがあり、`z59999` という利用者に対して、2 単位 (2000 時間) 追加した場合は、`z59999` の利用者のみ利用可能な CPU 時間は 3000 時間となる。

- `show_user_time`
  - ー グループに所属する利用者の利用可能な CPU 時間、すでに利用した CPU 時間を表示するためのコマンド
  - ー グループ管理者は、グループに属するすべての利用者の利用状況を一覧表示することができる

これにより、グループ管理者が利用可能な CPU 時間を任意の時間に設定し、利用可能な CPU 時間を上回った利用者については、自動的にバッチキューの利用権限が剥奪される仕組みを実現した。また、グループ管理者は、特定の利用者の利用可能な CPU 時間を増量することが可能であり、真に必要な利用者に対しては、簡易にその制限を緩和することが可能である。

## 4.2 同時投入可能な資源の制限による制御

3.2 節で述べた通り、同時投入可能な資源の制限による制御を実現するためには、利用者がジョブを投入した時点で実行されているジョブの情報を取得し、同時投入可能な資源に余剰があり、かつ最大実行時間内に収まるジョブであることを確認した上でジョブ投入を受け付ける必要がある。このため、利用可能な CPU 時間による制御とは異なり、ジョブの実行状況に応じて制限する必要があるが、このような機能は Slurm に搭載されていないため、Slurm のプラグイン機能を活用することにした。

Slurm のプラグイン機能は、ジョブスケジューラの核となる部分に変更を加えることなく、独自の制御ロジックを組み込むことが可能となる。これにより、システムの安定性を保ちつつ、柔軟な資源制限を実現することができる。

Slurm のプラグインは C 言語で記述し、共有ライブラリとしてコンパイルする必要があるが、Slurm のデータベースなどの内部情報にもアクセスが可能となり、ジョブスケジューラに登録されているジョブ情報を一括で取得し、逐次的に処理を行うことが可能である。これらの理由から、同時投入可能な資源の制限機能については、Slurm プラグインとして実装することが最適であると考えた。

今回実装したプラグインでは、利用者がジョブを投入するタイミングでジョブ情報を確認し、同時投入可能な資源の制限による制御が有効なバッチキューへのジョブ投入である場合は、以下の手順で処理を行う。

1. バッチキュー名ならびに利用者アカウントの情報をキーとして、現在の実行中のジョブならびに待

ち状態のジョブについて検索を行い、特定の利用者が特定のバッチキューで実行中または待機中のジョブの総数を検索する。

2. 検索した結果、実行中のジョブや待ち状態のジョブが存在する場合は、それぞれのジョブについて要求している資源量ならびに当該ジョブの実行可能最大時間の情報を取得し、投入可能ジョブ数から実行中のジョブまたは待ち状態のジョブ数を減算し、現時点で投入可能なジョブ数を算出する。
3. 実行中のジョブや待ち状態のジョブが存在しない場合や、2. で算出した投入可能ジョブ数に余剰がある状況であればジョブ投入を許可し、2. で算出した投入可能ジョブ数に余剰がない場合はジョブの投入自体を拒否する。

同時投入可能な資源の制限による制御を有効にしたいバッチキューについては、プラグインが参照する定義ファイルにおいて、要求コア数に応じた最大実行時間と投入可能ジョブ数を指定することで、バッチキューごとに投入可能なジョブに関する詳細を設定することが可能となる。

定義ファイルの例

[バッチキュー名]

#最大コア数	最大実行時間	投入可能ジョブ
1792	20160	5
7168	4320	2
14336	1440	1
99999	1	0

この定義では、1792 コア (16 ノード) までのジョブであれば、最大 20160 分 (約 14 日) の実行時間が許可され、同時に 5 つまでのジョブを投入できる設定となっている。その一方、7168 コア (65 ノード) ~ 14336 コア (128 ノード) の範囲のジョブの場合は、実行時間は最大 1440 分 (1 日) までに制限され、同時に実行可能なジョブについても 1 つまでとなっている。このように、小規模なジョブの投入可能数や最大実行時間を多めに設定し、大規模なジョブの投入数や最大実行時間を少なく設定することで、特定の利用者がバッチキューで利用可能な資源を占有する状況を防ぎつつ、利用実態を見ながら柔軟に制御することが可能である。

## 5 まとめ

本稿では、本センターのスーパーコンピュータシステムにおける、利用状況に応じたジョブ投入権限制御の検討と実装状況について報告した。具体的には、利用可能時間による制御と同時投入可能な資源の制限による制御の 2 つの手法を実装し、一部のバッチキューで試験運用を実施している。これらの制御方法を適用することにより、計算機資源の効率的な利用を促進しつつ、利用者間の公平性を確保することが可能となった。

本稿で報告した実装結果を踏まえ、今後の課題として以下の点が挙げられる。まず、個人利用向けのバッチキューや、JHPCN 課題用のバッチキュー、HPCI 課題用のバッチキューなど、複数のグループが共有しているバッチキューへの適用拡大が考えられる。これらのバッチキューに今回実装した制御方式を適用することで、さらに公平な運用が実現できると期待される。

また、同時投入可能な資源の制限による制御については、現在のバッチキュー単位での制御から、将来的には利用者単位での制御へと発展させることを検討している。これにより、よりきめ細やかな資源管理が可能となり、システム全体の利用効率の向上につながると考えられる。

## 参考文献

- [1] 深沢圭一郎, 新スーパーコンピュータシステムのご紹介, 京都大学情報環境機構広報誌「Info!」, Vol. 28, p10-12, 2023.
- [2] A. B. Yoo, M. A. Jette, M. Grondona: "SLURM: Simple Linux Utility for Resource Management", Lecture Notes in Computer Science, Vol.2862, pp.44-60, 2003.