

スーパーコンピュータ Wisteria/BDEC-01 における利用状況を考慮した運用の再検討

中張 遼太郎¹⁾, 須貝 佳義¹⁾, 昆野 長典¹⁾, 前田 光教¹⁾, 佐藤 孝明¹⁾, 山崎 一哉²⁾, 三木 洋平²⁾, 胡 曜²⁾, 下川辺 隆史²⁾, 住元 真司²⁾, 塙 敏博²⁾, 中島 研吾²⁾

1) 東京大学 情報システム部情報基盤課

2) 東京大学 情報基盤センター

nakahari@cc.u-tokyo.ac.jp

Review of Wisteria/BDEC-01 Operations in Consideration of Usage Status

Ryotaro Nakahari¹⁾, Yoshinori Sugai¹⁾, Takenori Konno¹⁾, Mitsunori Maeda¹⁾, Takaaki Satoh¹⁾, Kazuya Yamazaki²⁾, Yohei Miki²⁾, Yao Hu²⁾, Takashi Shimokawabe²⁾, Shinji Sumimoto²⁾, Toshihiro Hanawa²⁾, Kengo Nakajima²⁾

1) Information Technology Group, Information Systems Department, The University of Tokyo

2) Information Technology Center, The University of Tokyo

概要

東京大学情報基盤センターでは 2021 年 5 月よりスーパーコンピュータ Wisteria/BDEC-01 を運用している。約 3,000 名の利用者で共有利用するために同時に実行可能であるジョブ本数等を制限しているが、実際の利用状況に適した制限値を設定することで利用率の向上やジョブ実行時の待ち時間削減が期待できる。本稿では、直近のシステム利用状況を元に制限値等の影響をシミュレーションし、システム運用に関して検討した内容を報告する。

1. はじめに

東京大学情報基盤センター（以下、本センターと記す）では、大規模超並列スーパーコンピュータシステム（Oakbridge-CX スーパーコンピュータシステム）[1], 「計算・データ・学習」融合スーパーコンピュータシステム（Wisteria/BDEC-01 スーパーコンピュータシステム）[2],[3] など、多数のシステムを運用してきた。一方で、本センターと筑波大学計算科学研究センターが共同で運営する最先端共同 HPC 基盤施設（以下、JCAHPC と記す）[4]では、メニーコア型スーパーコンピュータシステム（Oakforest-PACS スーパーコンピュータシステム）[5]の後継システムとして、最先端共同 HPC 基盤施設スーパーコンピュータシステム（Miyabi スーパーコンピュータシステム）[6]の導入も進めている。

2024 年 9 月現在運用している Wisteria/BDEC-01 は約 3,000 名の利用者を有しており、ジョブスケジューラを用いて利用者による計算ジョブを管理する。特定の利用グループ等により計算資源が専有されることを防ぐため、ジョブ 1 本あたりで利用可能なノード数・GPU 数、実行時間を制限[7]す

るとともに、グループ単位で実行可能なジョブ本数等も制限している。システムの運用開始時点では利用実績データは存在しないため、過去のシステムの利用実績や制限値の設定を参考に制限値を決定するが、運用開始後の利用状況に適した制限値になるとは必ずしも限らない。過剰な制限はシステム利用率の低下やジョブ実行時の待ち時間増加に繋がる恐れがあり、過剰な緩和は特定の利用グループ等による計算資源の専有に繋がる恐れがあるため、制限値の見直しは実際の利用状況を考慮したうえで、慎重に行う必要がある。

一方で、Wisteria/BDEC-01 のジョブスケジューリングにおいてはバックフィル機能が有効であるため、利用者自身が推定実行時間の見積精度を高めることができれば、前述の制限値が変わらずとも利用者自身の工夫によりジョブ実行時の待ち時間の短縮に繋がることが期待されるが、現在の Wisteria/BDEC-01 利用者に対してどの程度の効果が期待できるかは不明確である。

本稿では制限値の変更および利用者による指定実行時間の適正化による影響をシミュレーションした結果を報告する。

2. Wisteria/BDEC-01 システム概要

2.1 システム構成

Wisteria/BDEC-01 は、「FUJITSU Processor A64FX」を搭載した計算ノード 7,680 台を有するシミュレーションノード群 (Odyssey) と、インテル社製汎用 CPU 2 基 (Ice Lake) および NVIDIA A100 Tensor コア GPU 8 基を搭載した計算ノード 45 台を有するデータ・学習ノード群 (Aquarius) の 2 つの計算ノード群から構成される。公募制度等により、一方のノード群についてのみ採択された場合等を除いて、利用者は両方のノード群を利用することができる。ジョブスケジューラとして、FUJITSU Software Technical Computing Suite[8]を使用している。

2.2 リソースグループ構成

Odyssey, Aquarius で利用することができるバッチジョブ用のリソースグループを表 1、表 2 に示す。実行時間の上限はいずれも最大 48 時間である。長時間のジョブ実行が可能であり、多くの利用が想定される regular リソースグループとは独立して、デバッグ用途の debug リソースグループ、短時間ジョブを想定した short リソースグループを用意していることが特長である。システム全体の利用率が高い状況下においても、デバッグ等のジョブは待ち時間が短く実行されることが期待される。

Odyssey には、トークン消費量を他のリソースグループより高く設定した priority-o を用意している。利用率は低くなると想定されるため、トークンを多く消費する代わりにジョブの早期実行を見込むことができる。

Aquarius は 1 ノードあたり GPU 8 基を搭載しているため、資源量の指定がノード単位の場合のみは利用者が必要としている以上の GPU が割り当てられる懸念があるため、GPU 単位 (1, 2, 4GPU) で資源量を指定することができる share リソースグループを用意している。

2.3 システム利用状況

Wisteria/BDEC-01 の運用開始から 2024 年 9 月までの利用率推移を図 1 に示す。本稿における利用率は以下の数式で算出するものとする。Odyssey ではノード数、Aquarius では GPU 数から算出する。

$$\frac{\sum(\text{ジョブ実行時間} * \text{使用ノード数 or 使用 GPU 数})}{(\text{稼働時間} - \text{保守時間}) * \text{総ノード数 or 総 GPU 数}}$$

表 1 Odyssey バッチジョブ用リソースグループ

名称	1 ジョブあたり ノード数上限	実行時間上限
debug-o	144 ノード	30 分
short-o	72 ノード	8 時間
priority-o	288 ノード	48 時間
regular-o	1,152 ノード	48 時間
	2,304 ノード	24 時間

表 2 Aquarius バッチジョブ用リソースグループ

名称	1 ジョブあたり ノード/GPU 数上限	実行時間上限
debug-a	1 ノード	30 分
short-a	2 ノード	8 時間
regular-a	4 ノード	48 時間
	8 ノード	24 時間
share-debug	1, 2, 4 GPU	30 分
share-short	1, 2, 4 GPU	2 時間
share	1, 2 GPU	48 時間
	4 GPU	24 時間

Odyssey, Aquarius とともに、年度の始まりである 4 月は利用率が落ち込み、年度末の 1 月から 3 月にかけて利用率が上昇する傾向がある。Aquarius は運用開始以降から徐々に利用率が上昇しており、2024 年度は高い利用率を維持している。一方で、Odyssey の利用率は 2023 年度から停滞の傾向にある。

Odyssey, Aquarius におけるリソースグループ別の利用率推移を図 2、図 3 に示す。Odyssey, Aquarius とともに、regular リソースグループの利用率が最も高く、debug, short リソースグループの利用率は低調である。

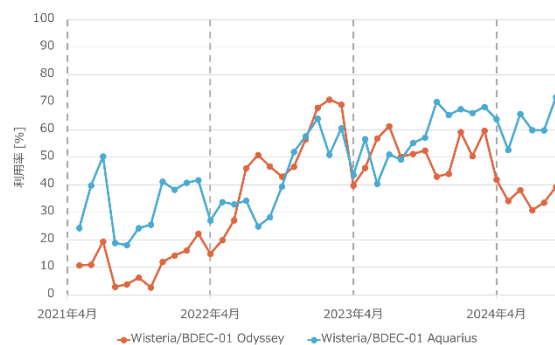


図 1 Wisteria/BDEC-01 利用率推移

Odyssey では short-o, priority-o が regular-o の利用率が高い時期に利用率が上昇する傾向にあることから、利用率の高い regular-o を避けて早期にジョブを実行するために使用されていると推測される。

Aquarius では share リソースグループの利用率が徐々に上昇しており、regular-a と同程度の高い利用率を維持している。

2.4 利用者傾向

2023 年度を対象に、Wisteria/BDEC-01 において実行されたジョブの傾向を分析した。インタラクティブジョブおよび実行時間が 60 秒以下のジョブは、通常のバッチジョブとは性質が異なる可能性があるため集計対象外とした。

1 ジョブあたりの利用ノード数、GPU 数別ジョブ本数分布を図 4、図 5 に示す。Odyssey, Aquarius とともに、最小の利用単位である 1 ノードもしくは 1GPU を指定するジョブが大半を占めている。Odyssey では 12 ノードのジョブが 1 ノードのジョブに次いで多いが、インターコネクトである Tofu インターコネクト D[9]における最小の構成単位が 12 ノードであることに起因すると推測される。Aquarius ではノード単位で指定する場合の最小値に相当するために 8GPU のジョブが多いと推測される。

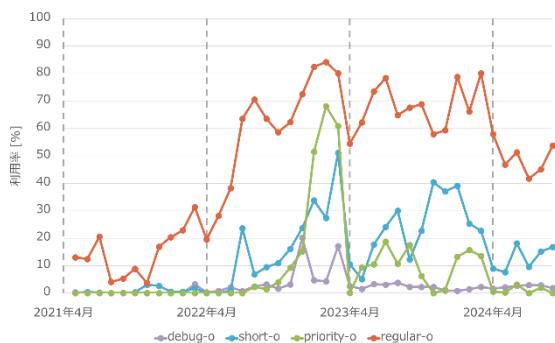


図 2 Odyssey リソースグループ別利用率推移

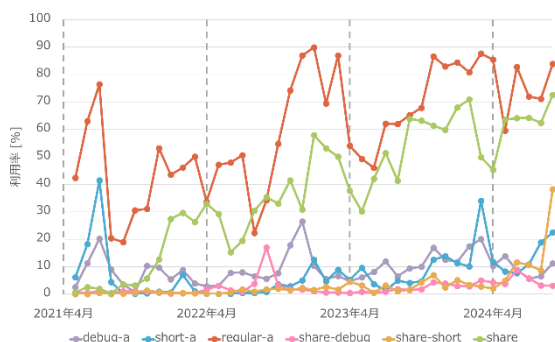


図 3 Aquarius リソースグループ別利用率推移

実行時間別ジョブ本数分布を図 6、図 7 に示す。

Odyssey, Aquarius とともに、1 時間以下のジョブが大半を占めている。regular リソースグループにおける実行時間の上限値が 48 時間もしくは 24 時間であるため、48 時間以下、24 時間以下のジョブ本数が若干増加していると推測される。

ジョブ投入時に指定された実行時間と実際の実行時間との比率分布を図 8、図 9 に示す。比率が高いほど、指定された実行時間と実際の実行時間との乖離が大きいことを示している。Odyssey, Aquarius とともに実際の実行時間の 10 倍程度を指定するジョブが多い。指定実行時間を超過したジョブは計算途中の場合でも強制終了され、結果を得ることができなくなるため、指定実行時間を長く見積もるジョブが多いと推定される。Odyssey では実際の実行時間の 100 から 500 倍程度と大きく乖離した値を指定するジョブも多い。

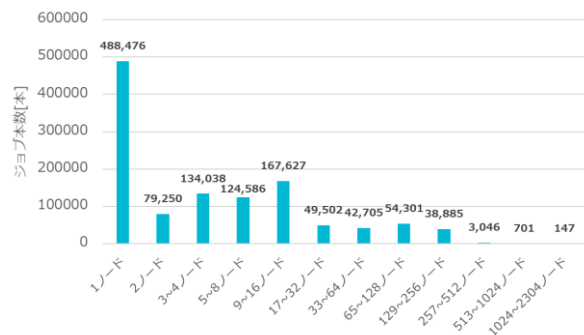


図 4 Odyssey 利用ノード数別ジョブ本数分布

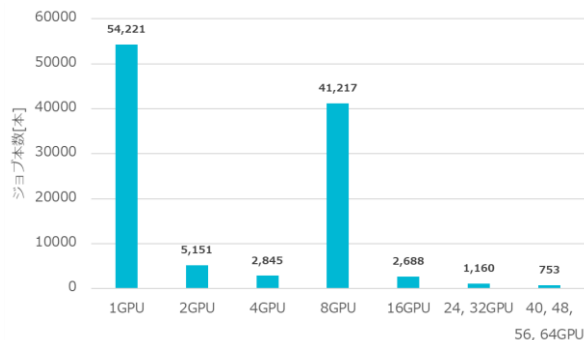


図 5 Aquarius 利用 GPU 数別ジョブ本数分布

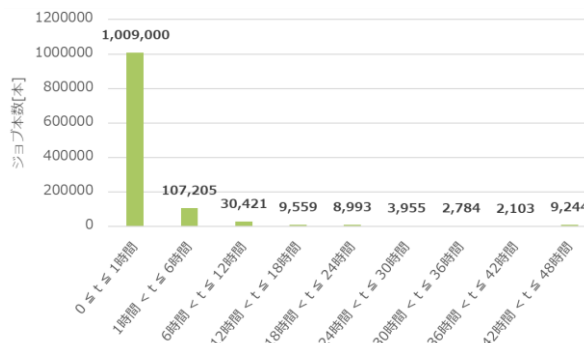


図 6 Odyssey 実行時間別ジョブ本数分布

3. シミュレーション概要

ジョブの実行本数の制限値の変更，リソースグループへの割当ノード数の変更，利用者による推定実行時間見積の適正化による影響をシミュレーションするプログラムの概要を示す．

3.1 シミュレーション条件

利用率が高い条件下で検討を行うために，シミュレーション対象の期間は 2024 年 3 月とし，2024 年 3 月 1 日 0:00 から 2024 年 3 月 29 日 9:00 までに投入され実行終了したジョブを対象とする．3 月 29 日 9:00 以降はサービス休止期間としてジョブの実行は不可とする．

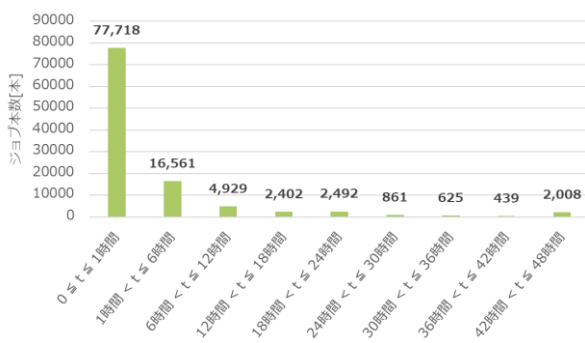


図 7 Aquarius 実行時間別ジョブ本数分布

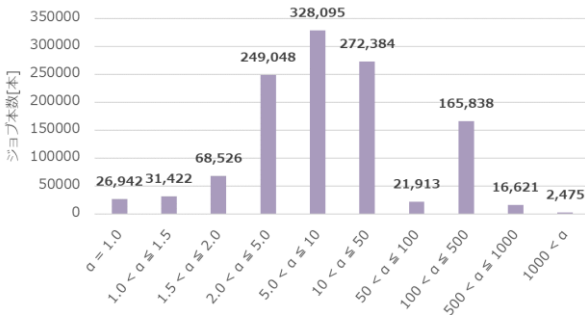


図 8 Odyssey 指定実行時間比率 α の分布
(α = 指定実行時間 / 実際の実行時間)

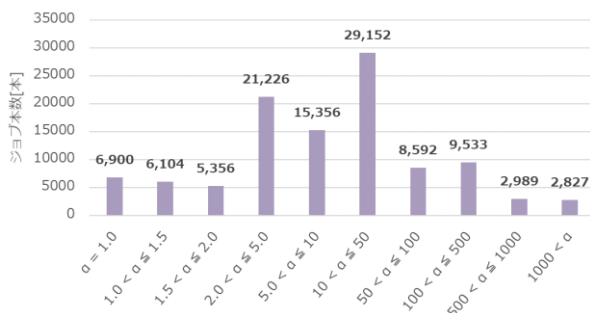


図 9 Aquarius 指定実行時間比率 α の分布
(α = 指定実行時間 / 実際の実行時間)

本シミュレーションに用いる入力情報は以下のとおりである．

- ジョブ ID
- 依存ジョブ ID (ステップジョブのみ)
- 投入先リソースグループ
- ジョブ投入日時 (UNIX 時間)
- 指定実行時間
- 実際の実行時間
- 指定ノード数 / GPU 数
- ユーザ ID
- ジョブ実行本数制限値
- 同時使用ノード数 / GPU 数制限値

ジョブスケジューリングの更新間隔は 10 秒とする．Odyssey, Aquarius どちらにも属さない，前処理後処理用のノード群およびノード固定制度等により特定グループのみで専有利用されているノードはシミュレーション対象外とする．

ジョブスケジューリングは，指定実行時間および指定ノード数/GPU 数に基づいて FIFO (First In First Out) 方式により行い，バックフィル機能を有効とする．バックフィル機能は，先に投入されたジョブの実行開始予定を遅延させることなく，空いている計算リソースにジョブ割当を行う機能である．指定実行時間を可能な範囲で短く設定することで，バックフィル機能により，ジョブ実行までの待ち時間を短縮できる可能性がある．後続ジョブの指定実行時間および指定ノード数/GPU 数に対して計算リソースの空きが不足しているためにバックフィルが行われない例を図 10 に示す．空いている計算リソースに割当可能なジョブが後続で投入され，バックフィルが行われた例を図 11 に示す．

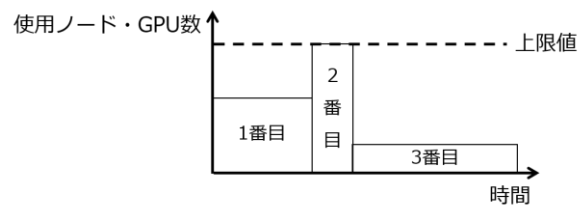


図 10 バックフィルが行われないジョブ投入例

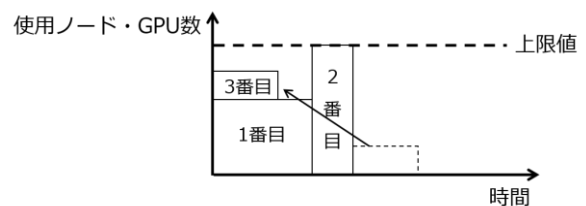


図 11 バックフィルが行われたジョブ投入例

ジョブの実行までに待ち時間が生じる場合は以下のとおり区別して集計を行う。以下の複数に該当する場合には、数字の最も若い区分の待ち時間として集計する。

1. 依存ジョブ終了待ち時間
依存ジョブが終了していない場合
2. グループ制限待ち時間
ジョブ実行本数もしくは同時使用ノード数/GPU 数制限値に達している場合
3. リソースグループ待ち時間
投入先リソースグループにおいて、
指定資源量の空きがない場合

3.2 擬似コード

本シミュレーションで用いたプログラムの擬似コードを図 12 に示す。

```
ジョブ情報読み込み;  
グループ情報読み込み;  
  
for (i = 開始日時; i < 終了日時; i += 10 秒) {  
    投入日時を経過したジョブをキューに投入;  
  
    各実行中のジョブについて  
    if (実際の実行時間を経過している) {  
        ジョブの状態を実行終了に変更;  
        スケジューリングから予約資源を削除;  
    }  
  
    各実行待ちジョブについて  
    if (依存ジョブが終了していない) {  
        依存ジョブ終了待ち時間 += 10 秒;  
        continue;  
    }  
    if (バックフィル可能) {  
        予定開始時刻を更新;  
    }  
    if (ジョブ即時実行可能 or 予定開始時刻超過) {  
        if (グループ制限値に達している) {  
            グループ制限待ち時間 += 10 秒;  
        } else {  
            ジョブの状態を実行中に変更;  
            スケジューリングの予約資源を更新;  
        }  
        continue;  
    } else {  
        予定開始時刻を更新;  
        スケジューリングの情報を更新;  
        リソースグループ待ち時間 += 10 秒;  
    }  
}  
  
シミュレーション結果出力;
```

図 12 スケジューリングプログラムの擬似コード

4. ジョブ制限値変更シミュレーション

4.1 シミュレーション条件

ジョブに対する制限値として、ジョブの実行本数を変更してシミュレーションを実施した。ジョブの実行本数変更は全グループに対して一律に適用した。区分ごとの待ち時間総和と利用者ごとの 1 ジョブあたり平均待ち時間の分布を比較する。シミュレーションパターンは以下のとおりである。

- 2 本削減
- 変更なし
- 2 本追加
- 4 本追加
- 6 本追加

4.2 シミュレーション結果

区分ごとの待ち時間総和の推移を図 13 に示す。制限値の緩和に伴いグループ制限待ち時間が減少したに加えて、リソースグループ待ち時間も減少している。グループ制限待ちのジョブのためにスケジューリングされる資源が減少したことによる影響と推測される。利用者ごとに 1 ジョブあたりの平均待ち時間を算出し、1 時間単位で集計した割合分布を図 14 に示す。大きな変化は確認できないが、制限の緩和に伴い平均待ち時間が 4 時間以下の利用者割合が増加傾向にある。グループ制限により、特定のグループのみで長く発生した待ち時間が、他ユーザに平均化された影響と推測される。

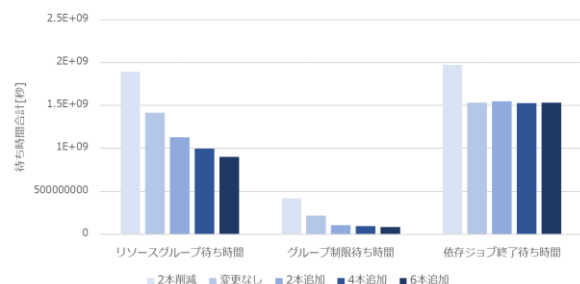


図 13 制限値変更による待ち時間総和の推移

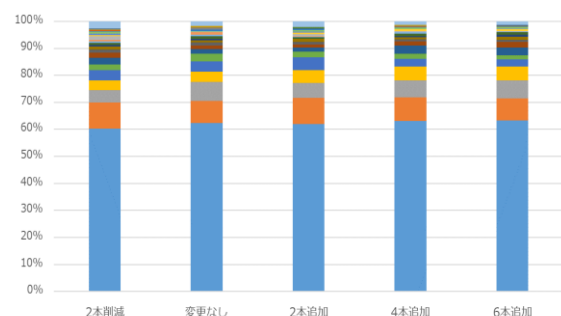


図 14 ユーザ別平均待ち時間の分布

5. 割当ノード数変更シミュレーション

5.1 シミュレーション条件

ジョブ投入時に指定するリソースグループにおける割当ノード数を変更してシミュレーションを実施した。シミュレーションパターンを検討するため、4章のシミュレーションパターン「変更なし」におけるリソースグループ別の待ち時間の分布を図15、図16に示す。regular-o, regular-a における待ち時間が多く、priority-o, short-o, debug-o, short-a, debug-a, share-short, share-debug の待ち時間が少ない。待ち時間の少ないリソースグループから待ち時間の多いリソースグループに割当ノード数を変更してシミュレーションを行う。Odyssey では Tofu インターコネクト D による構成を考慮して 96 ノードを、Aquarius は 1 ノードを割当変更するものとする。シミュレーションパターンは以下のとおりである。share-debug は割当ノード数が 1 ノードしかないため対象外とする。

[Odyssey regular-o への 96 ノード割当変更]

- 変更なし
- priority-o
- short-o
- debug-o

[Aquarius regular-a への 1 ノード割当変更]

- 変更なし
- short-a
- debug-a
- share-short

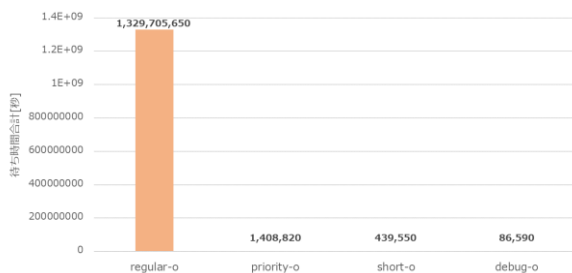


図 15 Odyssey リソースグループ別
待ち時間総和の分布

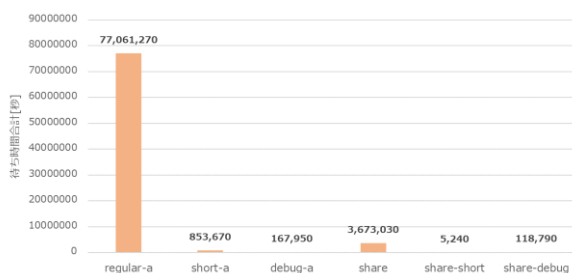


図 16 Aquarius リソースグループ別
待ち時間総和の分布

5.2 シミュレーション結果

区分ごとの待ち時間総和の推移を図 17, 図 18, 図 19 に示す。Odyssey では、いずれのリソースグループから割当ノード数を変更してもリソースグループ待ち時間が減少することが示唆されたが、Aquarius では debug-a から割当ノード数を変更した場合には全体での待ち時間が大きく増加し、short-a から割当ノード数を変更した場合も待ち時間が増加することが示唆された。debug-a は割当ノード数が 2 ノードしかなく、ジョブも一定数投入されているため、1 ノードの割当変更の影響が大きく現れたと推測される。short-a は図 3 に示すように 2024 年 3 月は利用率が比較的高いことが影響していると推測される。

6. 指定実行時間変更シミュレーション

6.1 シミュレーション条件

ジョブ投入時に指定する実行時間を実際の実行時間に近い値になるように変更してシミュレーションを実施した。シミュレーションパターンは以下のとおりである。既に指定している実行時間が変更後の実行時間よりも短い場合は指定実行時間の変更は行わない。

- 変更なし
- 【実際の実行時間】×100 倍を指定
- 【実際の実行時間】×10 倍を指定
- 【実際の実行時間】×6 倍を指定
- 【実際の実行時間】×2 倍を指定

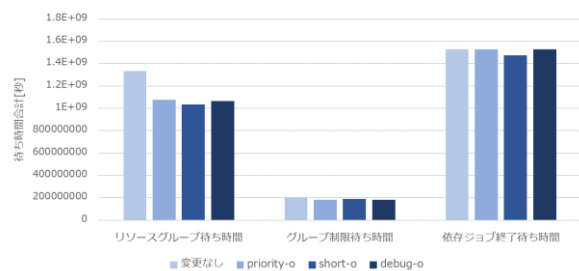


図 17 Odyssey 割当ノード数値変更による
待ち時間総和の推移

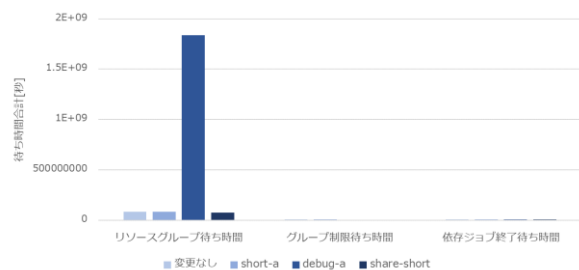


図 18 Aquarius 割当ノード数値変更による
待ち時間総和の推移

6.2 シミュレーション結果

区分ごとの待ち時間総和の推移を図 20 に示す。変更なしに対して、実際の実行時間の 100 倍以下を指定することで待ち時間が約 20%削減される可能性が示唆され、実際の実行時間の 10 倍以下を指定することで待ち時間が約 40%削減される可能性が示唆された。10 倍、6 倍、2 倍での結果に大きな差がない理由は、実行時間と指定実行時間の乖離の大きいジョブの多くが 10 倍の指定時点で待ち時間が改善されたためと推測される。100 倍以下の指定でも待ち時間改善が示唆されていることから、指定実行時間が大幅に実行時間と乖離しているジョブについて指定実行時間の見直しが行われるのみでも一定の効果が期待できる。

7. おわりに

本稿では、Wisteria/BDEC-01 におけるジョブ実行に関する制限値の変更および利用者による実行時間見積の適正化による影響をシミュレーションした。ジョブの実行本数制限を緩和することで待ち時間が減少する可能性が示唆されたが、待ち時間が長くなる利用者が増加する可能性も示唆された。また、リソースグループへの割当ノード数の変更により待ち時間が減少する可能性も示されたが、割当ノード数を変更するリソースグループにより待ち時間が増加する可能性も示された。得られた結果をふまえて運用設定の再検討を行う。

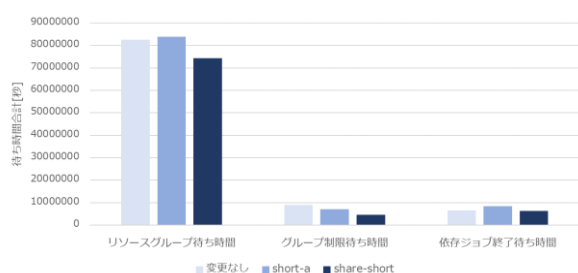


図 19 Aquarius 割当ノード数値変更による待ち時間総和の推移 (debug-a を除外)

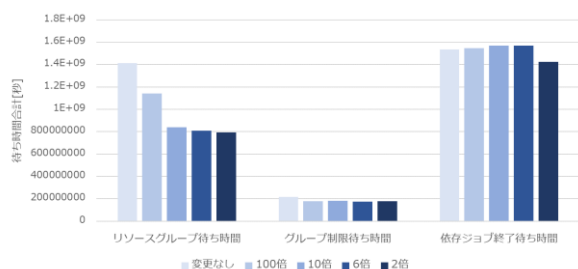


図 20 指定実行時間変更による待ち時間総和の推移

利用者側で実施する対応として、指定実行時間を実際の実行時間の 100 倍以下を指定することで約 20%、10 倍以下を指定することで約 40%待ち時間が削減される可能性が示唆された。実際の実行時間と大幅に乖離しているジョブを少しでも減少できるよう、定期的な周知を検討する。

今後も利用状況を定期的に分析し、システムの利用率の向上やジョブ実行時の待ち時間削減に繋がるよう運用の再検討を行い、よりシステムを効率よく使用いただけることを目指して尽力する。

謝辞 本論文の作成に際して、様々なご助言をいただいた情報基盤課、情報基盤センターの関係者の皆様に深く感謝いたします。この場をお借りして厚くお礼を申し上げます。

参考文献

- [1] Oakbridge-CX スーパーコンピュータシステム, <https://www.cc.u-tokyo.ac.jp/supercomputer/obcx/service/>
- [2] Wisteria/BDEC-01 スーパーコンピュータシステム, <https://www.cc.u-tokyo.ac.jp/supercomputer/wisteria/service/>
- [3] 中島研吾, 塙敏博, 下川辺隆史, 伊田明弘, 芝隼人, 三木洋平, 星野哲也, 有間英志, 河合直聡, 坂本龍一, 近藤正章, 岩下武史, 八代尚, 長尾大道, 松葉浩也, 荻田武史, 片桐孝洋, 古村孝志, 鶴岡弘, 市村強, 藤田航平, 「計算・データ・学習」融合スーパーコンピュータシステム「Wisteria/BDEC-01」の概要, 研究報告ハイパフォーマンクスコンピューティング (HPC), 2021-HPC-179, 1, pp.1-7, 2021.
- [4] 最先端共同 HPC 基盤施設 (JCAHPC), <https://jcahpc.jp/>
- [5] Oakforest-PACS スーパーコンピュータシステム, <https://www.cc.u-tokyo.ac.jp/supercomputer/ofp/service/>
- [6] Miyabi スーパーコンピュータシステム, <https://www.cc.u-tokyo.ac.jp/supercomputer/miyabi/service/>
- [7] Wisteria/BDEC-01 キュー, <https://www.cc.u-tokyo.ac.jp/supercomputer/wisteria/service/job.php>
- [8] FUJITSU Software Technical Computing Suite, <https://www.fujitsu.com/downloads/JP/jsuper/tcs-v4-datasheet.pdf>

[9] Y. Ajima, T. Kawashima, T. Okamoto, N. Shida,
K. Hirai, T. Shimizu, S. Hiramoto, Y. Ikeda,
T. Yoshikawa, K. Uchida, and T. Inoue, "The Tofu
Interconnect D", 2018 IEEE International Conference
on Cluster Computing, pp. 646-654, 2018