Vol.2019-HPC-172 No.12
2019/12/18

高充填率と低待時間を両立するジョブバッチスケジューリング手法の提案

高山　沙也加[†1]　関澤　龍一[†2]　鈴木　成人[†3]　山本　拓司[†3]　小口　正人[†1]

概要：近年の HPC システムでは利用者の増加と利用者層の拡大に伴い, 投入されるジョブ数は増加しており, また, 多様化している. そのためシステム規模は増加傾向にある. システムの運用において, ユーザの立場ではジョブ投入から実行されるまでの待ち時間が, 運用の立場では利用可能な計算機資源のうち実際に利用された割合である充填率が重視され, これらの両立が大きな課題となっている. 待ち時間はシステム側で十分なノード数を用意することで, 充填率は必要ノード数に応じてジョブ受け入れに制限を設けることで改善可能だが, 一方を優先させることで他方が悪化する可能性がある. 本研究では高い充填率と低い待ち時間を両立するシステム運用の実現に向けて, ジョブの必要ノード数分布毎に適したシステムノード数を検討する.

キーワード：ジョブスケジューリング, HPC システム, リソース管理

1. はじめに

近年の HPC システムでは利用者の増加と利用者層の拡大に伴い, 投入されるジョブ数は増加しており, また, ジョブがシステムに要求する条件も多様化している. そのため HPC システムの規模は増加傾向にある. システムの運用においては, 図 1 のようにユーザの立場ではジョブを投入してから実行されるまでの待ち時間が, 運用の立場では利用可能な計算機資源のうち実際に利用された割合を指す充填率が重視され, これらの両立がシステム運用にあたって大きな課題となる. 待ち時間はシステム側で受け入れるジョブに対して十分なノード数を用意することで改善されるが, 投入されるジョブ分布によっては充填率が著しく下がる可能性がある. 充填率は必要ノード数に応じてジョブ受け入れに制限を設けることで改善可能だが, 受け入れなかったジョブをどのように処理するのか考える必要がある. 投入ジョブの必要ノード数に応じてシステムとキューにパーティションを設けることで, 充填率と待ち時間の改善を図るという手法は実環境の運用で既に導入されている. しかし, 現行の HPC システムの運用ではパーティションは動的に変更されるものではなく, 管理者が経験則に基づいて設定しており, 運用時の投入ジョブ情報は考慮されていない.

そこで, ジョブ情報を参考にしてパーティション分けとジョブ割り当てを決定するシステム構築を検討したい. 本研究では高い充填率と低い待ち時間を両立するシステム運用の実現に向けて, ジョブの必要ノード数分布毎に適したシステムノード数を実験を通して検討する.

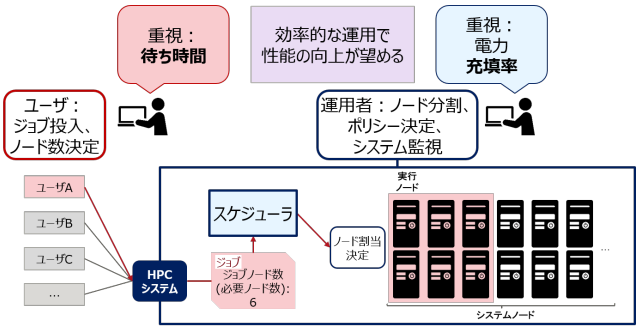

図 1　HPC システムの想定図

2. 先行研究

システムの効率的な運用を目的としたジョブスケジューリング手法は既にいくつか提案されている.

例えば, CPU の製造過程で生じる性能や消費電力のばらつきを考慮し，特定のアプリケーションに対する電力の可変性の影響を踏まえたスケジューリングが提案されている [1]. この研究ではソケットの違いやアプリケーション

†1　現在，お茶の水女子大学
Presently with Ochanomizu University
†2　現在，富士通株式会社
Presently with FUJITSU LTD.
†3　現在，株式会社富士通研究所
Presently with FUJITSU LABORATORIES LTD.

の違いによる消費電力の差を考慮した電力比可変性予測とベンチマークの訓練結果を利用する変動性訓練予測をスケジューリングポリシーとし, 実稼働クラスタで使用されているスケジューリングポリシーと比較して最大 31%のジョブターンアラウンドタイムの短縮と，最大で 5.5%の供給電力の削減が実現されている.

また, 多様なジョブ受け入れを前提とした, 不均一なマシンで構成された HPC システムの利用率とジョブの待ち時間の最適化を目的として，履歴ワークロードを用いた Portable Batch System（PBS）ベースのクラスタのジョブシミュレーションを実行する方法が提案されている [2]. この検証ではジョブスケジューリングのシミュレーション機能を持たない PBS リソースマネージャの代わりに, マウイスケジューラを用いて大規模な PBS ベースのクラスタのジョブシミュレーションを行っている.

本研究ではシステム全体の高い充填率と低い待ち時間の実現に向けて, ジョブの必要ノード数の分布に対する最適なシステム規模に注目して実験を行う.

3. 実験

異なるシステム規模, ジョブ分布条件でジョブスケジューリングを行い, その際の充填率と待ち時間を調査する.

ジョブスケジューリングのシミュレーションには Slurm Simulator[3] を用いる. ジョブの必要ノード数は Alibaba が公開しているクラスタデータ [4] のジョブ分布を参考に決定する. このデータでは必要ノード数 17 以上のジョブは投入割合が著しく小さく, 本実験での投入されるジョブの必要ノード数は最大で 16 とする. 比較に用いるジョブ分布は Alibaba の公開データの分布を参考にした Alibaba 分布, Alibaba 分布を反転させた反転分布, どの必要ノード数のジョブも同じ割合で投入される均等分布の 3 つとする. これらの分布は図 2, 図 3, 図 4 のようになっている. ジョブの実行時間は最小で 0 秒, 最大で 1800 秒とし, 短い時間に偏らせつつランダムに決定する. 図 5, 図 6, 図 7 にシステムノード数 20 の際のシミュレーションにおける必要ノード数別ジョブの実行時間の分布を記す. ジョブの投入間隔は等間隔とする. システムノード数 20 の際のシミュレーションで用いたジョブの投入間隔を図 8, 図 9, 図 10 に記す. 表 1 にジョブスケジューリングの条件を記す. スケジューリングアルゴリズムは FCFS, Backfill はありとする.

総投入時間は投入率が 1 となるように定める. 投入率は次式で求めた値とする.

$$Input_Rate = \frac{\sum_{i=0}^{n} NodeJob_i \times Duration_i}{TotalSubmit \times SystemNode} \times 100$$

この時, NodeJob はジョブが必要とするノード数, Duration はジョブの実行時間, TotalSubmit は総投入時間, SystemNode はシステム全体のノード数を表す. ジョブ数はシステムノード数に応じて複製して増やす. 増やしたジョブは複製元のジョブの直後に投入されるものとする. 本研究で試みたシステムノード数とジョブ数の組み合わせを表 2 に記す.

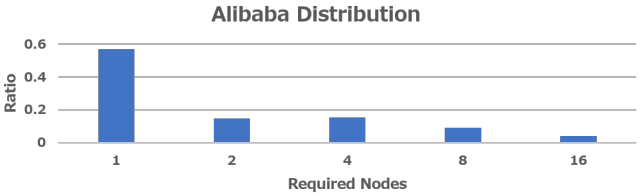

図 2 Alibaba 分布における必要ノード数別ジョブ分布

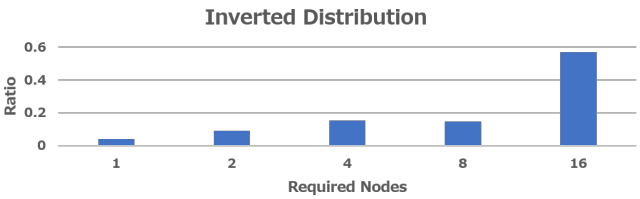

図 3 反転分布における必要ノード数別ジョブ分布

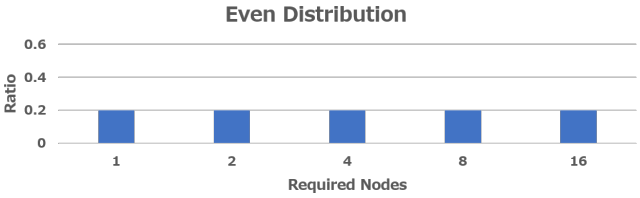

図 4 均等分布における必要ノード数別ジョブ分布

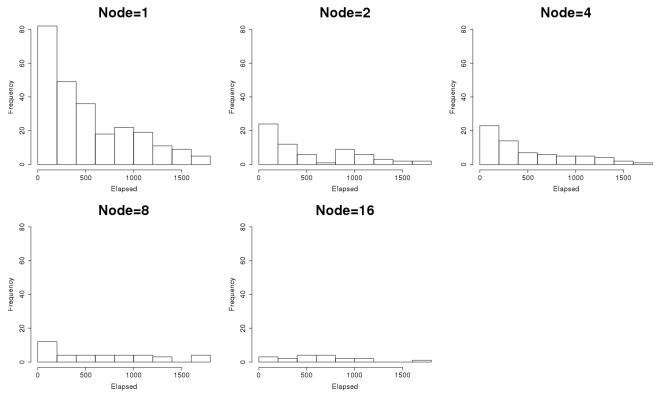

図 5 Alibaba 分布におけるジョブ実行時間の分布

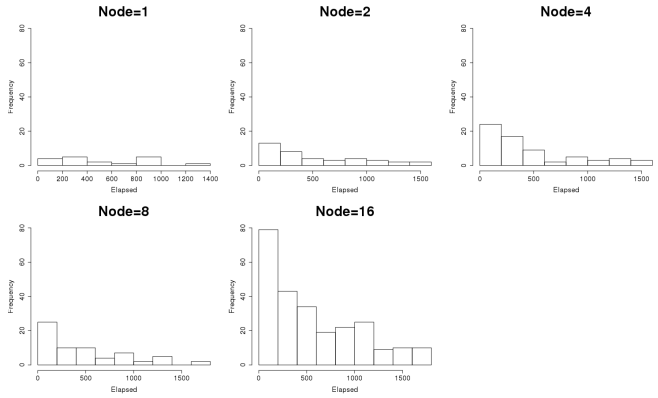

図 6　反転分布におけるジョブ実行時間の分布

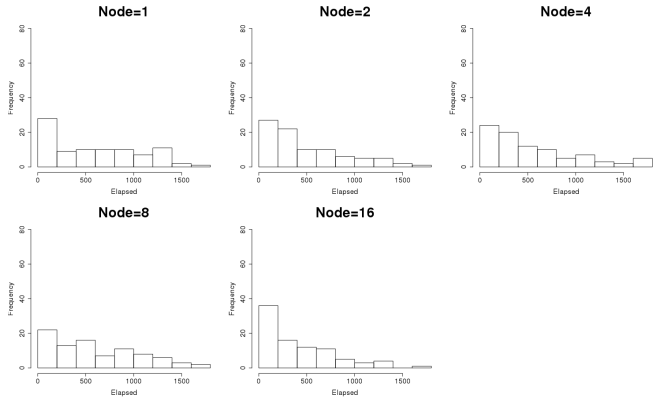

図 7　均等分布におけるジョブ実行時間の分布

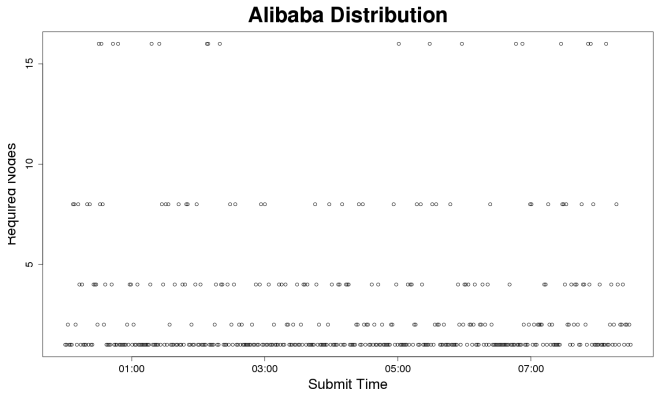

図 8　Alibaba 分布における必要ノード数別ジョブ投入間隔

3.1　スケジューリングアルゴリズム

本研究での実験でスケジューリングアルゴリズムとして利用した First-Come-First-Served (FCFS) では, ジョブは割り当て優先順位が到着順に定められ, 処理が行われる. FCFS の利点としてはアルゴリズムがシンプルであること, ジョブ処理が公平であることが挙げられるが, ジョブの投入タイミングによっては充填率が著しく悪くなる可能性がある.

そのため, Backfill を有効にすることで, 割り当て優先順が低い場合でも優先度の高いジョブを遅延させずに, かつ前方に実行可能領域を確保できる場合は優先度の高いジョブを追い越して割り当てできるようにする. Backfill の導入の有無によるスケジューリングの違いを図 11 に記す.

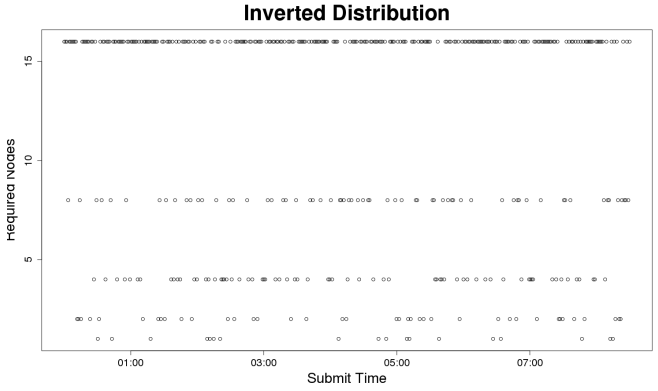

図 9　反転分布における必要ノード数別ジョブ投入間隔

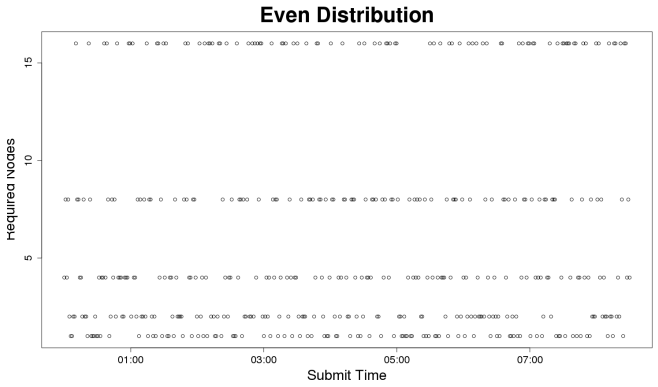

図 10　均等分布における必要ノード数別ジョブ投入間隔

表 1　スケジューリング条件

ジョブ数	440 (システムノード数=20 の時)
総投入時間	5.8 時間
ユーザ指定実行時間	指定なし
ジョブ実行時間	0 - 1800 秒
トポロジー	tree
Backfill	あり
アルゴリズム	FCFS
離散割当	無効
利用者指定のノード割当	無効
Fair Share	なし

3.2　Slurm Workload Manager

Slurm Workload Manager[5] はあらゆる規模の HPC システムで利用されているジョブスケジューリングシステムである. 一般的なリソース管理プログラムと同様にジョブのスループット, システムの利用率, およびジョブの待ち時間に大きく影響する可能性のある多くのパラメータ設定が可能である. ただし, 実験条件やポリシーの変更が実稼働 HPC システムにおけるスケジューラのパフォーマンスに与える影響を確認するには数日から数週間かかる場合があ

表 2　システムノード数とジョブ数

システムノード数	ジョブ数
16	353
20	440
30	660
32	706
40	880
50	1098
60	1225
64	1345
70	1311
80	1433
90	1380
96	1516
100	1444

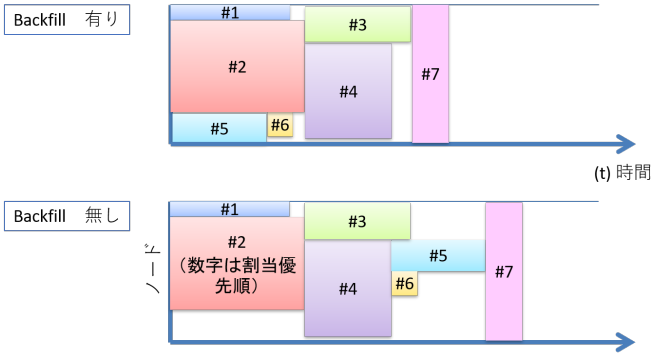

図 11　Backfill の有無によるスケジューリングの違い

るため, パラメータを変更することでシステムパフォーマンスを調整したり, 新しいポリシー選択を実装したりすることは実用的ではない.

そこで, 本研究では Slurm Simulator を用いる. このシミュレータでは小規模なクラスタ構成であれば, ジョブの構成とパラメータ条件によっては 1 時間で 17 日分のスケジューリングシミュレーションが可能である.

4. 実験結果

4.1 充填率と待ち時間

各分布における平均充填率と待ち時間の結果を図 12, 図 13, 図 14 に記す.

Alibaba 分布では, 他のジョブ分布と比べると平均待ち時間は著しく短い. 逆に, 反転分布では平均待ち時間は他の二つと比べると非常に長いが, システムノード数の増加に合わせて平均待ち時間は短くなっている. 均等分布では, 充填率に関しては反転分布と同じような変化が見られた. しかしその変化は反転分布ほど大きくない. Alibaba 分布と均等分布は, 反転分布とは異なり必要ノード数に関わらず投入されるジョブの割合が均等であるため, 必要ノード数の多いジョブの影響が小さい. これらの結果から, 必要ノード数の多いジョブの割合が大きいほどシステムノード数の増加による待ち時間改善の恩恵は大きいと考えられる.

全体の傾向として, システムノード数が 16 の整数倍の際に充填率と待ち時間が改善している. 本実験では投入されるジョブの必要ノード数は 1, 2, 4, 8, 16 のいずれかであり, 16 の約数であることが関係していると考えられる.

また, いずれのジョブ分布でもシステムノード数 20 または 30 で充填率と待ち時間が悪化していることが確認された. 待ち時間だけではなく充填率まで悪化しており, どちらも小さいシステムノード数での実験結果よりも悪化していることから, 単にジョブの必要ノード数と投入されるジョブの量に対してシステム規模が小さすぎるためではなく, 投入されたジョブの必要ノード数の分布も充填率と待ち時間に関係していると考えられる. 特に反転分布ではシステムノード数 16 から, システムノード数 20, 30 にかけて充填率と待ち時間の悪化が著しい. システムノード数 16 の時は, 必要ノード数 16 のジョブを 1 つ処理している間は充填率は 100%になるが, その間には他のジョブは処理できない. 必要ノード数 20, 30 の際は, 同様に必要ノード数 16 のジョブを 1 つ処理できるが, その間に他のジョブの処理は可能である. しかし, 反転分布では必要ノード数の小さいジョブは投入される割合が小さくなるので, 結果としてシステムノード数 16 と比べて充填率と待ち時間が悪化してしまったと考えられる. そのため, 必要ノード数の大きいジョブの割合が大きいほどシステムのリソースが小さいことによる影響が大きくなりやすいと考えられる.

これらの結果から, 一定以上の高充填率と低待ち時間を両立させるには, 投入ジョブの必要ノード数が 16 の約数の場合最低でも 32 ノード確保するべきであると考えられる.

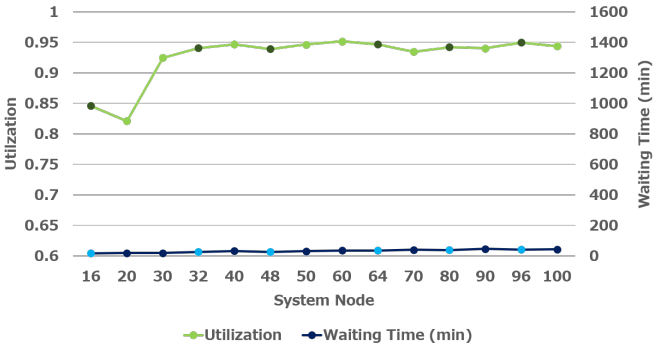

図 12　Alibaba 分布の充填率と待ち時間

4.2 必要ノード数別待ち時間

図 15, 図 16, 図 17 は各ジョブ分布における必要ノード数別待ち時間である.

Alibaba 分布では必要ノード数 16 のジョブのみ他の必要ノード数のジョブと比べて平均待ち時間の差が大きく, システムノード数 20 から 30 の間で平均待ち時間が大きく減

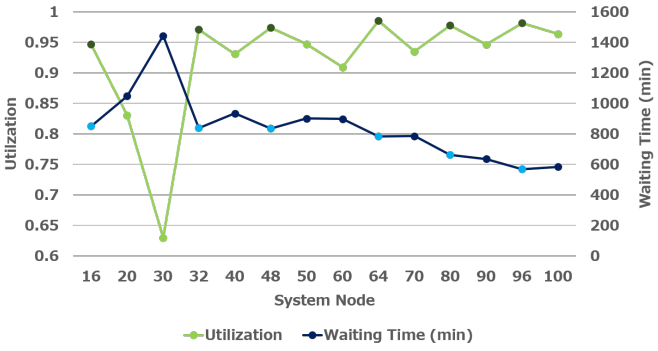

図 13　反転分布の充填率と待ち時間

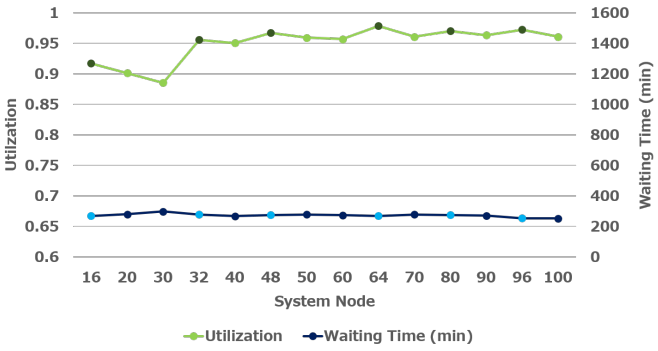

図 14　均等分布の充填率と待ち時間

少している．しかし, 必要ノード数 16 のジョブ以外は待ち時間が微増している．Alibaba 分布と比べると均等分布では必要ノード数 16 のジョブとそれ以外のジョブとで待ち時間の差は小さいが, 同様にシステムノード数 20 から 30 の間で必要ノード数 16 のジョブの平均待ち時間が減少している．

また, Alibaba 分布と均等分布ではシステムノード数を増やすと必要ノード数別待ち時間の差は小さくなる傾向にある．これはシステムノード数の増加によってシステムノード数とジョブの必要ノード数の比が縮まり, 必要ノード数の違いによるジョブの入りやすさの違いが小さくなり, 待ち時間が一定値に収束するようになるためだと考えられる．

反転分布では, 必要ノード数毎の平均待ち時間の差は小さいが, システムノード数が 30 の時にどれも悪化している．また, システムノード数 30 から 32 にかけてどの必要ノード数ジョブも平均待ち時間は改善されている．そのため, 必要ノード数が大きいジョブの投入割合が大きいことが他のジョブの待ち時間の増加にも関係していることがわかる．

これらの結果から, 総システムノード数を増やせない場合, 必要ノード数の大きいジョブと小さいジョブとをパーティション分けしたシステムに分けて投入することで必要ノード数の大きいジョブによる影響を軽減できると考えられる．

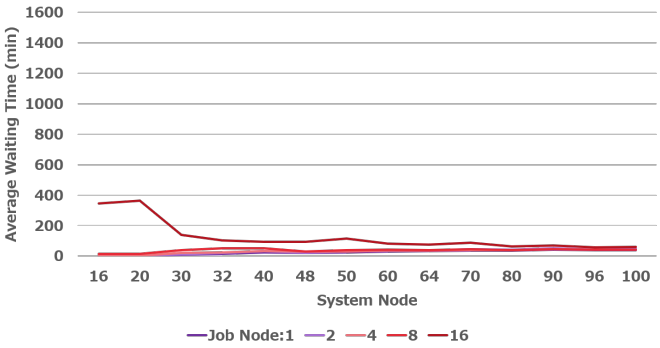

図 15　Alibaba 分布における必要ノード数別待ち時間

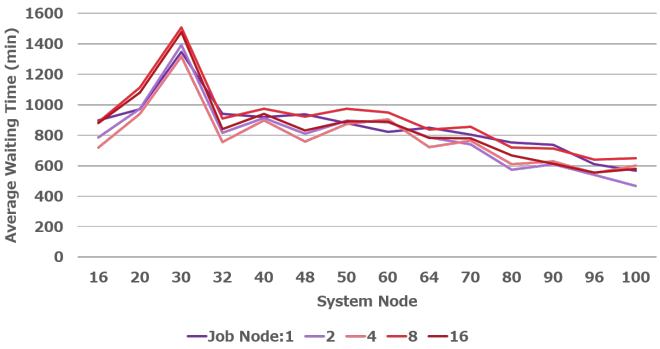

図 16　反転分布における必要ノード数別待ち時間

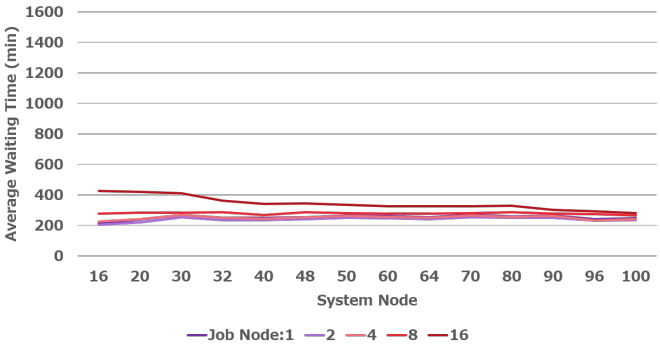

図 17　均等分布における必要ノード数別待ち時間

5.　まとめと今後の予定

HPC システムにおけるジョブの高い充填率と低い待ち時間の両立を目的として, 投入ジョブの情報を利用してパーティション分けとジョブ割り当てを行うシステム構築を検討したい．本研究ではシステム構築に向けて, あらゆるジョブ投入条件, システム規模条件でのジョブスケジューリングのシミュレーション結果からジョブの必要ノード数分布毎に適したシステムノード数を分析した．

シミュレーションの結果, 共通の特徴としてシステムノード数が 16 の整数倍である時に充填率と待ち時間の改善の傾向が見られた．今回の実験で投入されたジョブの必要ノード数が 1, 2, 4, 8, 16 のいずれかであり, システムノード数が投入ジョブの必要ノード数の公倍数の整数倍になっ

ていることが関係していると考えられる. 改善の程度は必要ノード数の多いジョブの割合が大きい反転分布が著しく, 投入ジョブの必要ノード数とその分布に依存していると考えられる. また, システムノード数が 20 または 30 の時に充填率と待ち時間の両方が悪化していた. そのため, システムノード数が 16 の整数倍で改善したことも踏まえると, 確保出来るノードを全てジョブスケジューリングに利用するよりも, システム規模として投入ジョブの必要ノード数の公倍数の整数倍のノード数にした方がより効率良く運用できると考えられる.

また, Alibaba 分布では必要ノード数 16 のジョブのみ他の必要ノード数のジョブと比べて平均待ち時間の差が大きく, システムノード数 20 から 30 の間で平均待ち時間が大きく減少していた. また, ジョブ分布毎に必要ノード数別平均待ち時間の差は異なっており, 必要ノード数の多いジョブの割合が小さい Alibaba 分布では必要ノード数 16 のジョブの平均待ち時間が他のジョブと比べて長かった. 逆に必要ノード数の多いジョブの割合が大きい反転分布では, 必要ノード数の違いによる待ち時間の差はあまり見られず, 必要ノード数の大きいジョブが必要ノード数の小さいジョブの待ち時間に悪影響を与えていると考えられる. Alibaba 分布と均等分布ではシステムノード数を増やすと必要ノード数別待ち時間の差は小さくなっていた. これは, システムノード数とジョブの必要ノード数の比が小さくなり, 結果としてジョブの投入しやすさの違いも小さくなったためだと考えられる. この結果から, 一定のシステムノード数条件下では, 必要ノード数の大きいジョブと小さいジョブとをパーティション分割したシステムに分けて投入することでこ必要ノード数の大きいジョブによる影響を軽減できると考えられる.

今後は, 他クラウドベンダが公開しているジョブ分布ではどのようなジョブスケジューリングになるのか調査を行いたい. 今回の実験では, 特定の必要ノード数のジョブが他のジョブの待ち時間に影響を与えていたことから, 投入ジョブの必要ノード数やジョブ分布を参考にしたパーティション決定アルゴリズムの考案も行いたい. 想定するシステムのイメージを図 18 に示す. また, 今回の実験では必要ノード数に対して, システムノード数を公倍数の整数倍にすることで充填率と待ち時間が改善されていた. そこで, ジョブのグルーピングによってあらゆるジョブ分布, 投入ジョブの必要ノード数でもスケジューリング改善を図れる手法を考案したい.

謝辞

本研究の一部はお茶の水女子大学と富士通研究所との共同研究契約に基づくものである.

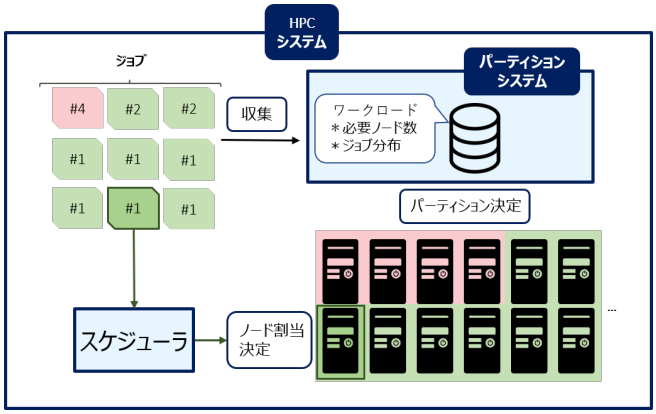

図 18　想定システム図

参考文献

[1] Dimitrios Chasapis, Miquel Moretó, Martin Schulz, Barry Rountree, Mateo Valero, and Marc Casas. Power efficient job scheduling by predicting the impact of processor manufacturing variability. In *Proceedings of the ACM International Conference on Supercomputing*, pp. 296–307. ACM, 2019.

[2] Georg Zitzlsberger, Branislav Jansík, and Jan Martinovič. Job simulation for large-scale pbs based clusters with the maui scheduler. *BIG DATA ANALYTICS, DATA MINING AND COMPUTATIONAL INTELLIGENCE 2018 THEORY AND PRACTICE IN MODERN COMPUTING 2018*, p. 137.

[3] Nikolay A Simakov, Martins D Innus, Matthew D Jones, Robert L DeLeon, Joseph P White, Steven M Gallo, Abani K Patra, and Thomas R Furlani. A slurm simulator: Implementation and parametric analysis. In *International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems*, pp. 197–217. Springer, 2017.

[4] Alibaba/clusterdata. *https://github.com/alibaba/clusterdata*, Accessed: November 2019.

[5] Slurm Workload Manager. *https://slurm.schedmd.com/*, Accessed: November 2019.