投入ジョブ情報を踏まえたシステムノードのパーティション分割の考察

高山沙也加† 関澤 龍一†† 鈴木 成人††† 山本 拓司††† 小口 正人†

† お茶の水女子大学 〒 112–8610 東京都文京区大塚 2-1-1
†† 富士通株式会社 〒 105–7123 東京都港区東新橋 1-5-2 汐留シティセンター
††† 株式会社富士通研究所 〒 211–8588 神奈川県川崎市中原区上小田中 4-1-1
E-mail: †{sayaka-t,oguchi}@ogl.is.ocha.ac.jp, ††r_sekizawa@fujitsu.com,
†††{shigeto.suzuki,takuji}@fujitsu.com

あらまし 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, 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. 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いる. 本研究ではジョブバッチシステムにおける高充填率と短待ち時間の両立を実現するスケジューリングアルゴリズムの開発の基礎検討として, 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し, また, ジョブ必要ノード数毎に実行領域を分割した際のスケジューリングの評価を行う.

キーワード ジョブスケジューリング，HPC システム，パーティション

Consideration of System Node Partitioning Based on Input Job Information

Sayaka TAKAYAMA†, Ryuichi SEKIZAWA††, Shigeto SUZUKI†††, Takuji YAMAMOTO†††, and Masato OGUCHI†

† Ochanomizu University 2–1–1 Otsuka, Bunkyou-ku, Tokyo 112–8610 Japan
†† Fujitsu Ltd 1–5–2 Shiodome City Center, Higashishinbashi, Minato-ku, Tokyo 105–7123, Japan
††† Fujitsu Laboratories Ltd 4–1–1 Odanaka, Nakahara-ku, Kawasaki-shi, Kanagawa 221–8588, Japan
E-mail: †{sayaka-t,oguchi}@ogl.is.ocha.ac.jp, ††r_sekizawa@fujitsu.com,
†††{shigeto.suzuki,takuji}@fujitsu.com

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, 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. 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

て設定している．

そこで，投入される際に得られるジョブ情報を基に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．本研究ではジョブバッチシステムにおける高充填率と短待ち時間の両立を実現するスケジューリングアルゴリズムの開発の基礎検討として，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し，また，ジョブ必要ノード数毎に実行領域を分割した際のスケジューリングの評価を行う．

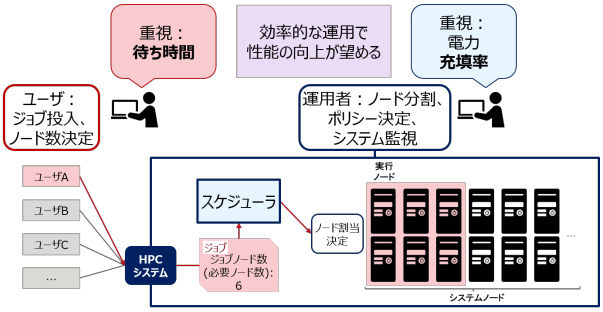

図 1: HPC システムの想定図

2. 関 連 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．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によって，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 [1]．

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た研究として，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．

本研究ではジョブの必要ノード数の分布とシステム規模に注目し，投入ジョブに対して適したパーティションサイズを決定する実験を行う．

3. 実　　験

3.1 異なるシステム規模，ジョブ条件でのスケジューリング

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．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する．このデータでは必要ノード数 17 以上のジョブは著しく投入割合が小さく，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 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 3 にシステムノード数 20 のシミュレーションにおける各ジョブ分布の実行時間を記す．ジョブの投入間隔は等間隔とする．今回実験で用いたジョブの投入間隔を図 4，図 5，図 6 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 First-Come-First-Served (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．

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

この時，NodeJob はその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増やす．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．

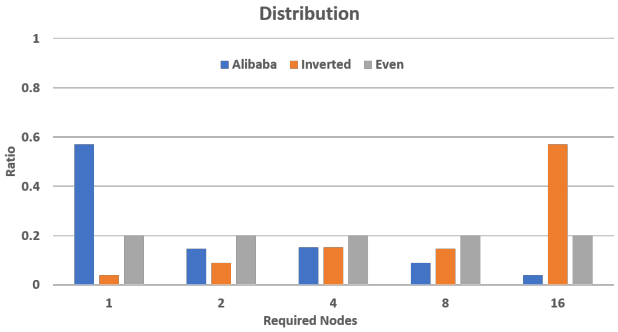

図 2: 各分布における必要ノード数別ジョブ分布

表 1: スケジューリング条件

ジョブ数	1516
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 パーティション分割した場合のジョブスケジューリング

複数のシステム分割パターン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．

総システムノード数は 96 とし，ジョブデータは Alibaba 分布のシステムノード数 96 の際のデータを用いる．

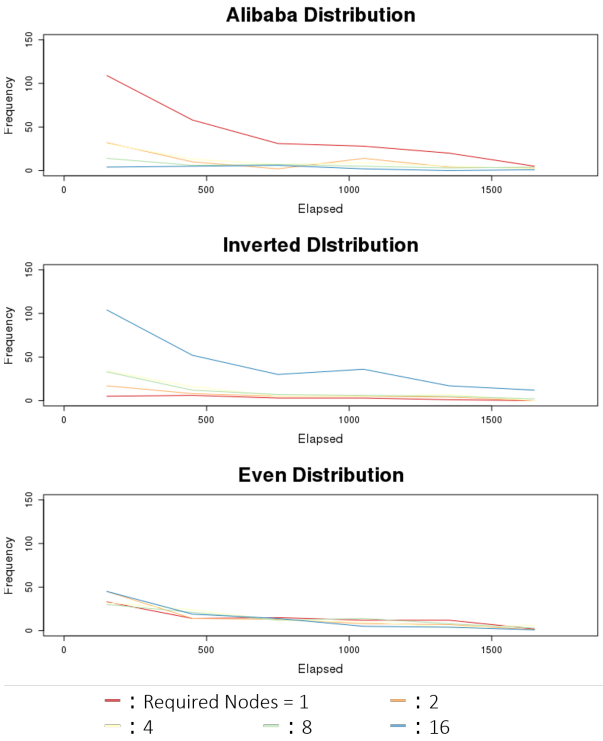

図 3: 各分布におけるジョブ実行時間の分布

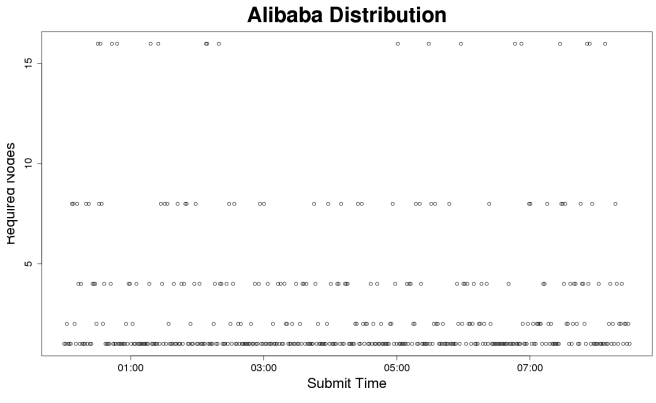

図 4: Alibaba 分布における必要ノード数別ジョブ投入間隔

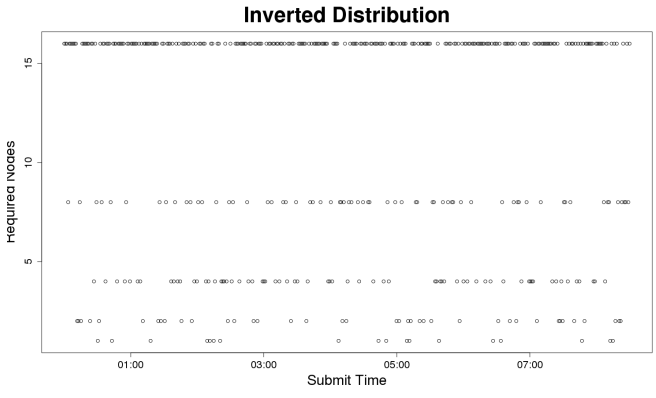

図 5: 反転分布における必要ノード数別ジョブ投入間隔

3.3 スケジューリングアルゴリズム

本研究でスケジューリングアルゴリズムとして利用した 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前方に実行可能領域があれば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 7 に記す.

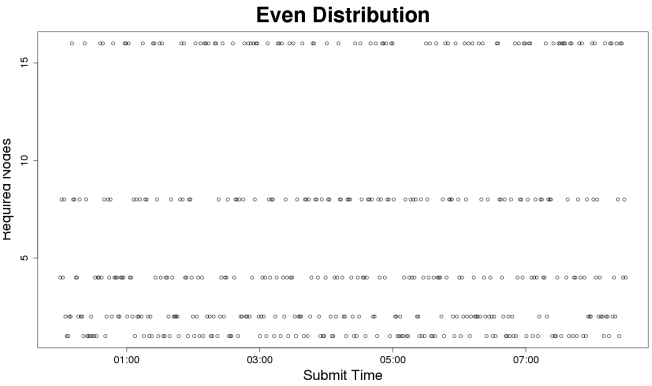

図 6: 均等分布における必要ノード数別ジョブ投入間隔

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

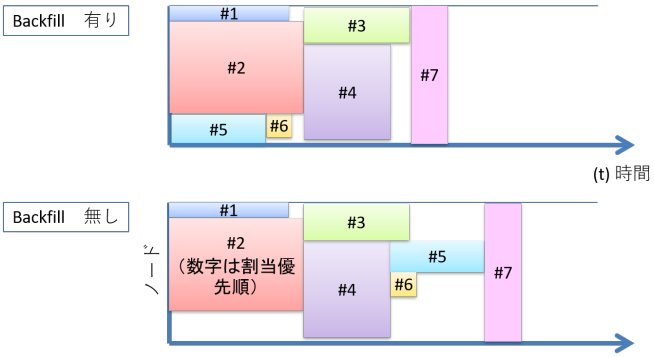

図 7: Backfill の有無によるスケジューリングの違い

3.4 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.

あらゆる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 8, 図 9, 図 10 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が, システムノード数を増やす毎に平均待ち時間が少しずつ長くなっている. これは, システムノード数が増えたことによるスケジューリングの改善よりもジョブの増加によるキューへのジョブの溜まりやすさの影響の方がやや大きいためと考えられる. 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. そのため, システムノード数の増加による待ち時間改善の恩恵は, 必要ノード数の大きいジョブが多い程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. 投入されるジョブの必要ノード数が 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というだけでなく, 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程システムのリソース不足による悪影響が大きくなると考えられる.

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システムノード数として 32 確保するべきであると考えられる.

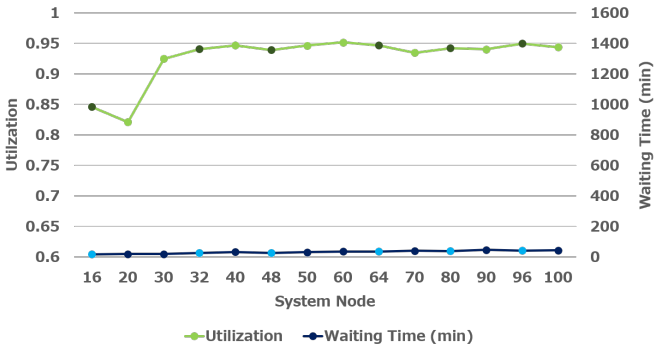

図 8: Alibaba 分布の充填率と待ち時間

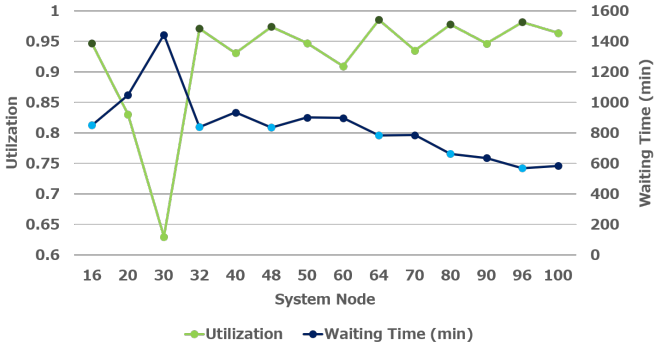

図 9: 反転分布の充填率と待ち時間

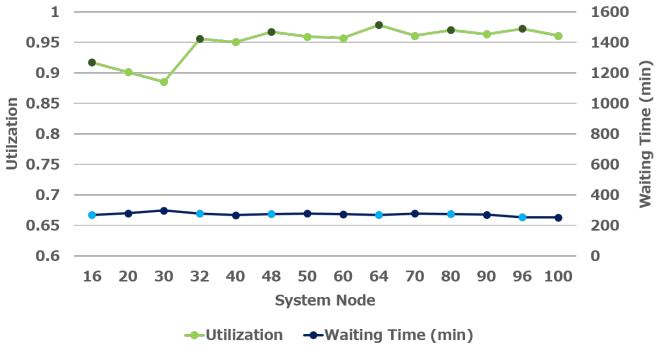

図 10: 均等分布の充填率と待ち時間

4.2 必要ノード数別待ち時間

図 11, 図 12, 図 13 は各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. そのため, システムノード数 20 から 30 の間では必要ノード数の大きいジョブの待ち時間解決の代わりに小さいジョブの

待ち時間が長くなり, 結果的に全体の待ち時間がわずかに悪化してしまっている.

反転分布では, 必要ノード数毎に平均待ち時間の差は小さい. 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投入の割合が大きいことが他のジョブの待ち時間の増加にも繋がっていることがわかる.

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とで待ち時間の差は小さい. 反転分布では必要ジョブ数毎の平均待ち時間に大きな違いは見られないが, システムノード数が 30 の時にいずれも悪化している.

これらの結果から, 必要ノード数の大きいジョブと小さいジョブとで投入するシステムをパーティション分けすることでこの問題による影響を軽減することができると考えられる.

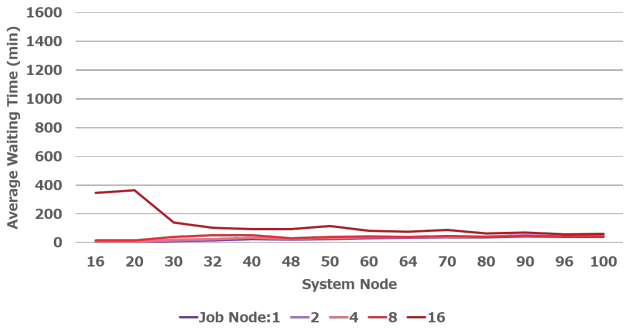

図 11: Alibaba 分布における必要ノード数別待ち時間

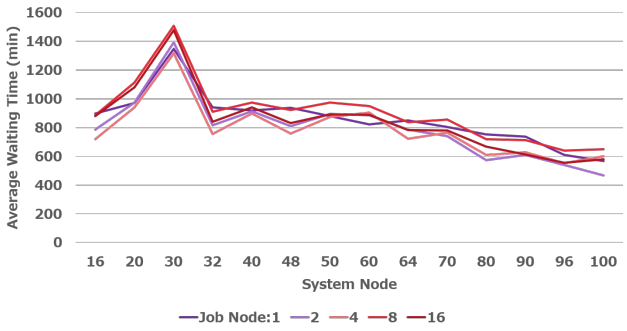

図 12: 反転分布における必要ノード数別待ち時間

図 13: 均等分布における必要ノード数別待ち時間

4.3 パーティション分割

各パーティション分割でのジョブシミュレーション結果を表 3 に示す. 96 ノードで構成されたジョブバッチシステムを 2 つのパーティションに分割した. 分割領域 a, b のノード数を変更し, a に必要ノード数 16 のジョブ, b にそれ以下の実行ノード数のジョブを投入した. 評価指標は a, b それぞれにおける (1) 平均充填率, (2) ジョブ処理が全く行われていない時間隔を除いた平均充填率, (3) 平均待ち時間, (4) 最長待ち時間とした.

表 3: 充填率と待ち時間 (パーティション分割)

システムノード	(1) 平均充填率	(2) 平均充填率 (0 除外)	(3) 平均待ち時間 (h)	(4) 最長待ち時間 (h)
16 + 80	0.605	0.606	0.418	8.069
30 + 66	0.600	0.602	1.123	8.009
32 + 64	0.874	0.878	1.091	2.468
48 + 48	0.662	0.665	2.768	6.095

シミュレーション結果を見ると, 各項目で最良となるパーティションは異なっている. 例えば, 平均充填率のみに注目すると 32 + 64 が, 平均待ち時間のみに注目すると 16 + 80 が最良となる. しかし, 16 と 80 にパーティション分割した際のシミュレーションではジョブの最長待ち時間は 8 時間を超えている. 図 14 にシステムノード 16 + 80 の, 図 15 にシステムノード 32 + 64 の充填率の時系列データを示す. 図 14 の結果を見ると, 途中で充填率が著しく低下し, ほぼ一定となっている様子が確認できる. これは一方のシステムノードでの処理がほぼ終了し, 他方のシステムノードのみがジョブ処理を行っている状態になっているためだと考えられる. それと比べると, 図 15 では充填率の低下の幅は小さく, 期間も短い. そのため, ジョブの投入タイミングも意識してパーティション決定する必要があると考えられる.

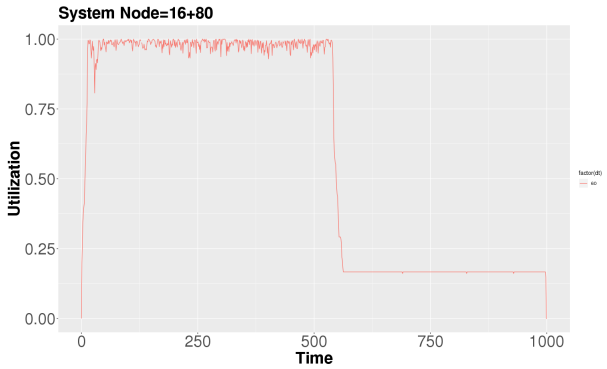

図 14: システムノード 16 + 80 の充填率

本研究では充填率と待ち時間の両立を可能とするシステム構築を目的としており, 充填率と待ち時間の両方を踏まえると今回の実験条件ではシステムノードを 32 と 64 に分割した際のジョブスケジューリングが最適であったと言える.

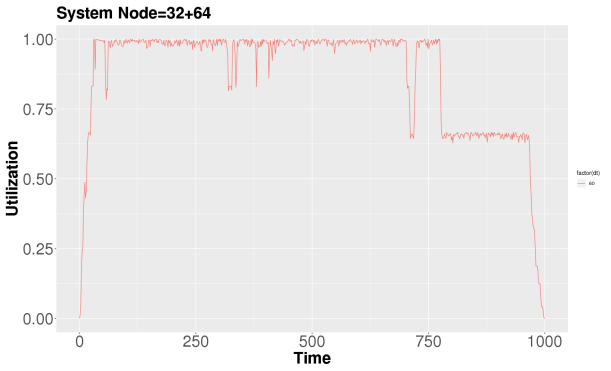

図 15: システムノード 32 + 64 の充填率

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短い待ち時間の両立を目的として, 投入される際に得られるジョブ情報を基に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. 本研究ではジョブバッチシステムにおける高充填率と短待ち時間の両立を実現するスケジューリングアルゴリズムの開発の基礎検討として, 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の分析とジョブ必要ノード数毎に実行領域を分割した際のスケジューリングの評価を行った.

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, 16 の約数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投入ジョブが本実験のような条件の際は, システム規模は 16 の整数倍, 最少でも 32 のシステムノード数を設けた方が効率が良いと考えられる.

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少してい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, Alibaba 分布とのスケールの違いはあるが, 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た. この結果から, 必要ノード数の大きいジョブと小さいジョブとで投入するシステムをパーティション分けすることでこの問題による影響を軽減することができると考えられる.

パーティション分割した際のスケジューリング実験では, 評価項目によって最良となるパーティションは異なっていること, 各パーティションのキューイングジョブ数を考慮したパーティション決定が必要であることを確認した.

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になるかの調査を行いたい. また, 今回の実験では必要ノード数は 16 の約数としており, システムノード数が 16 の整数倍にすることで充填率と待ち時間が改善された. そこで, ジョブをうまくグルーピングすることで異なるジョブ分布でもスケジューリング改善を図れる手法を考案したい. 今回の実験で, 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, この情報を基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.

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.

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.