5J-05

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決定とジョブバッチスケジューリング手法の提案

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　　†† 富士通株式会社　　††† 株式会社富士通研究所

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, 多様化している. 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の待ち時間が, 運用の立場では充填率が重視され, これらの両立が大きな課題となっている. 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が, 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, 経験則に基づいて設定されている. 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, ジョブ必要ノード数毎に実行領域を分割し評価を行う.

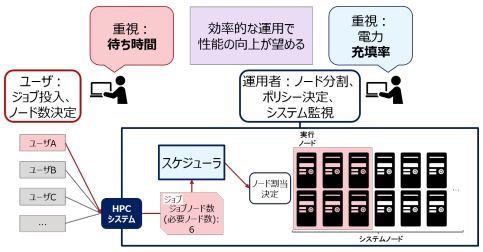

図 1: HPC システムの想定図

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. 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]. この検証ではマウイスケジューラを用いた大規模な 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調査するために実験を行う.

3　実験

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.

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 以上のジョブは投入割合が著しく小さいため, 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 とし, 必要ノード数は 1, 2, 4, 8, 16 のいずれかとする. 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. ジョブの投入間隔は等間隔とする. 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.

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

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.

Proposal of System Node Partitioning Based on Submitted Job Information and Job Batch Scheduling Method
†Sayaka Takayama　††Ryuichi Sekizawa　†††Shigeto Suzuki　†††Takuji Yamamoto　†Masato Oguchi
†Ochanomizu University
††FUJITSU LTD.
†††FUJITSU LABORATORIES LTD.

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

4 実験結果

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 2 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 2: 充填率と待ち時間

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．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 時間を超えており, 他の結果も考慮すると 32 + 64 がパーティション分割として適していると考えられる．図 2 にシステムノード 16 + 80 の充填率の時系列データを示す．この結果を見ると, 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．そのため, 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．

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．

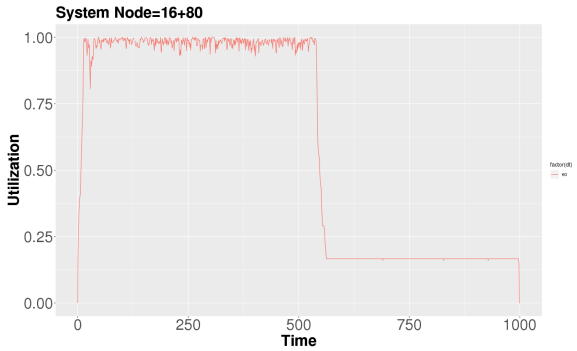

図 2: システムノード 16 + 80 の充填率

5 まとめと今後の予定

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, ジョブ必要ノード数毎に実行領域を分割し評価を行った．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, ジョブノード数分布を変更し同様の評価を行うことで, 最適なパーティション分割の導出を目指す．また, 上記を踏まえたパーティションアルゴリズムの考案と評価を行なう．

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．

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.