2H-01

待ち時間のばらつきの軽減を目的とした動的実行領域分割手法の一検討

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およびその必要ノード数は増加しており, また, 多様化している. 2019年に計算資源の共用を終了したスーパーコンピュータ「京」のジョブ規模別計算資源利用状況を見てみると, 必要ノード数が1000を超えるジョブのノード割当時間積が半数以上を占めている[1]. システムの運用において, ユーザの立場ではジョブの待ち時間が, 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る. あらゆる規模のジョブが投入される環境では, 小規模ジョブが大規模ジョブの割り当ての妨げとなり待ち時間が著しく長い大規模ジョブが生じてしまうことがあり, この軽減が大きな課題となる. 投入ジョブの必要ノード数に応じてシステムとキューを2分割することで, 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, 大規模環境では2分割の制御では十分な効果を得られない場合がある.

本研究ではジョブの必要ノード数の分布や待ち時間に注目し, 大規模ジョブの待ち時間の削減を目的として, 過去のジョブ分布やジョブ実行状況を分析し, 自動で実行領域の分割数と分割ノードサイズ, 実行ジョブのノード数範囲を決定するアルゴリズムの提案と効果検証を行う.

2　実行領域分割アルゴリズム

提案アルゴリズムでは, 過去の投入ジョブ分布とジョブの待ち時間, そして運用者が設定したジョブの最大最小待ち時間の差から, 実行領域の分割数, 各実行領域が受け入れるジョブの必要ノード数範囲, 各実行領域のサイズを決定する. アルゴリズムはジョブスケジューリングから独立して動作し, 一定間隔で適用する. 初めに, 各実行領域における最大最小待ち時間の差と運用者が設定した最大最小待ち時間の差を比較し, 実行領域の分割数を決定する. いずれの実行領域でも最大最小待ち時間の差が運用者の設定を下回っていれば分割数は-1, そうでなければ+1する. 次に, 各実行領域が受け入れるジョブの必要ノード数範囲を決定する. この時, 各実行領域のジョブ粒度が揃うよう分割する. 最後に, 決定した分割数とジョブの必要ノード数範囲で, 各実行領域の大きさを決める. ジョブの必要ノード数範囲別にノード時間積の総和を求め, その比に合わせてシステムノードを分割し, 各実行領域とする. この時, 最大ジョブノード数の2倍より大きくない実行領域があった場合, 該当する実行領域のノード数を最大ジョブノード数の2倍とし, 他の実行領域はシステムノードから該当実行領域のノード数を差し引き割合で分割した大きさとする. アルゴリズムの流れを図1に示す.

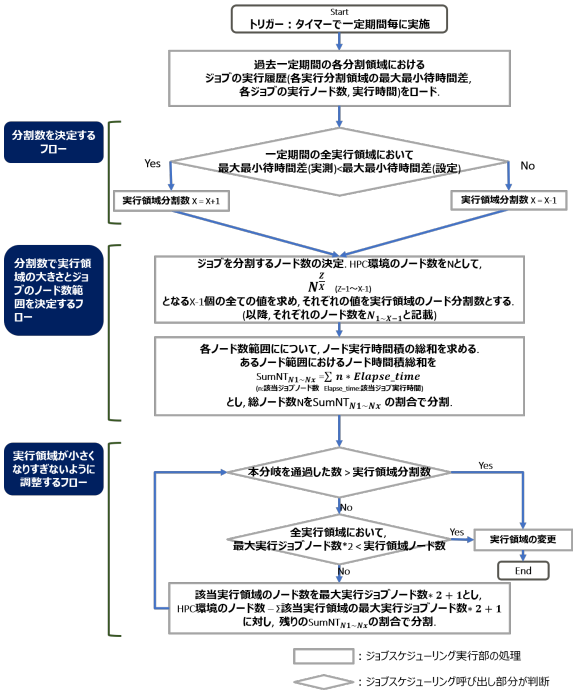

図1: 提案アルゴリズムのフローチャート

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.

3　実験

提案アルゴリズムの効果検証のために, あるジョブ条件でアルゴリズムを適用し, アルゴリズムに則って各設定を変えた際の充填率と待ち時間の評価を行う. ジョブスケジューリングには Slurm Simulator[2] を用いる. 本来のアルゴリズムでは各実行領域における最大最小待ち時間の差と 運用者が設定した最大最小待ち時間の差を比較し分割数を決定するが, 実験では効果確認のために最大最小待ち時間の差に関係なく分割数を増やし実験する. 検証には,「京」ジョブ統計情報から生成されたジョブミックスを利用する. 実験用に生成したジョブミックスの詳細を表 1 に示す.

表 1: ジョブデータの条件

ジョブの最大必要ノード数	1000
ジョブの取得期間	7 日間
ジョブ密度	95%

4　実験結果

各分割条件でジョブスケジューリングを実行した際の充填率を図 3 に示す. 分割アルゴリズム適用前後で

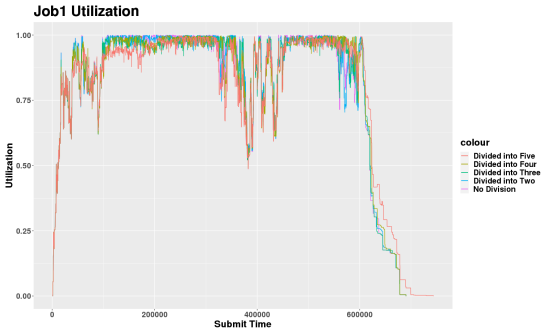

図 2: 各分割数でジョブスケジューリングした際の充填率

変化はほとんどないことがわかる. これらの結果から, 分割アルゴリズムによる充填率への悪影響はほぼなく, 分割数を増やしても同様であると考えられる.

次に, 各分割数での最大最小待ち時間の差を図**??**に示す. 分割した場合では, 最大最小待ち時間の差が最も大きい分割領域の結果を出力している. アルゴリズムを適用した場合では 2, 3, 4 分割では分割なしと比べて最大最小待ち時間の差は大幅に小さくなっている. この結果から, 特定のジョブのみ待ち時間が大幅に増加してしまう問題はアルゴリズムの適用によって軽減できていることが確認できる. ただし, 5 分割では他の

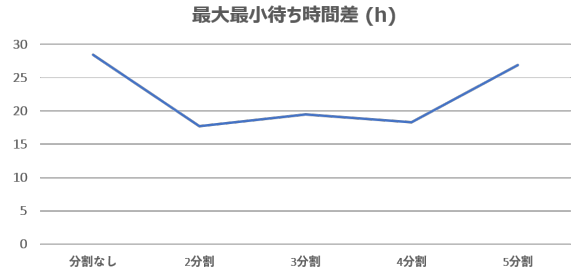

図 3: 各分割数での最大最小待ち時間の差

実験結果と比べて最大最小待ち時間の差が大幅に増えてしまっている. この結果を見ると, 分割数を増やせば必ずしも良くなるわけではないことがわかる.

5　まとめと今後の予定

過去のジョブ分布やジョブ実行状況を分析し, 自動で実行領域の分割数と分割ノードサイズ, 実行ジョブのノード数範囲を決定するアルゴリズムの提案と効果検証を行った. 提案アルゴリズムでは, 過去の投入ジョブ分布とジョブの待ち時間, そして運用者が設定したジョブの最大最小待ち時間の差から, 実行領域の分割数, 各実行領域が受け入れるジョブの必要ノード数範囲, 各実行領域のサイズを決定する. 実験結果から, アルゴリズムの適用によって充填率の悪化なく最小待ち時間の差が改善されることが示された. 今後の課題として, 投入されるジョブ分布や密度が同じ条件でもジョブの投入タイミングに偏りがあると待ち時間が長いジョブが発生し得ることが挙げられる. そのため, ある程度投入タイミングに偏りがあっても最小待ち時間の差を一定以下に抑えられるようアルゴリズムの改善を検討したい.

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.

参考文献

[1] 「京」の稼働状況. *https://www.r-ccs.riken.jp/jp/k/machine-status/*, Accessed: November 2020.

[2] 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.