日本オペレーションズ・リサーチ学会
2022年 春季研究発表会

スーパーコンピュータのジョブスケジューリング

	京都大学	*丸山怜	MARUYAMA Rei
02203940	東京理科大学	鮭川矩義	SUKEGAWA Noriyoshi
01012660	東京理科大学	池辺淑子	IKEBE Yoshiko
	東京理科大学	立川智章	TATSUKAWA Tomoaki
	理化学研究所	山本啓二	YAMAMOTO Keiji
	理化学研究所	庄司文由	SHOJI Fumiyoshi

1. はじめに

理化学研究所は，外部ユーザから計算依頼（ジョブ）を常時受け付け，スーパーコンピュータ（2019年8月までは「京」，2021年3月以降は「富岳」）上で計算を行なうサービスを提供している．同サービスの質は，各ジョブをスーパーコンピュータの計算ラック上のどこで・何時から計算を開始するかを決定する，ジョブスケジューリングアルゴリズムの性能に大きく左右される [2].

本研究では，同サービスの運用で特に重要視される，スーパーコンピュータの稼働率と外部ユーザの待ち時間の公平性に焦点を当て，解決すべき課題を記述するための単純なモデルを導入する．そして，同モデル上で「良い」ジョブスケジューリングアルゴリズムについて議論し，実際の運用に役立つ知見の発掘を図る．

2. 提案モデル

提案モデルは長方形詰込み問題 [1] と区間スケジューリング問題の要素を併せ持つオンライン型の最適化問題とみなすことができる．計算ラックを正方形，各ジョブを正方形に収まる長方形としてそれぞれ表現する（図1）．各ジョブ j はユーザが指定した以下の情報を持つ．

- **実行時間上限** u_j：計算開始後 u_j 時間が経過したら，計算を強制的に打ち切る
- **計算ノード数** n_j：ジョブをモデル化した長方形の面積を決定する

これらの情報はジョブの到着後にわかる．一方，各ジョブ j の（実際の）**実行時間** e_j は到着後もわからない．しかし，$e_j \leq u_j$ は保証される．

決定すべきことは，各ジョブ j の**計算開始予定時刻** s_j（および，対応する計算ラック上の計算開始予定位置）である．ジョブの終了を「トリガー」として頻繁にリスケジュールを行なうため，計算開始予定時刻は随時更新される．最終的に計算を開始した時刻を s_j^*，**到着時刻**を a_j とすれば，**待ち時間** w_j は $s_j^* - a_j$ で与えられる．

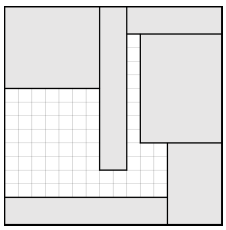

図 1: 計算ラック（正方形）へのジョブ（長方形）の配置例．各マスは計算ノードに対応（計算ノードは格子上に連結）

計算ラックの**稼働率**は計算ラックに占める実行中のジョブの割合（充填率），言い換えると，使用中の計算ノードの割合で定義する．また，待ち時間の公平性をつぎの仮定に基づいて定義する：

仮定. 計算ノード数が少ない（計算ラックに収まりやすい）ジョブのユーザは待ち時間が短いと期待する．逆に，計算ノード数が多い（計算ラックに収まりにくい）ジョブのユーザは待ち時間が長くなる可能性があることを理解している．

この仮定を認めれば，横軸を計算ノード数，縦軸を待ち時間とするグラフでジョブが「右肩上がり」に並ぶことが望ましい．そこで，待ち時間の**公平性**を，この右肩上がりの線形性で定義する．

3. アルゴリズムの考察

離散時刻 $T = \{0, 1, 2, \ldots\}$ を考える．ただし，0は常に現在の時刻を表す．ジョブスケジューリングアルゴリズムは，計算ラック（2次元）を離散時刻 T（1次元）に関して時間発展させた，3次元の空間上で動く．各ジョブに対し，到着後すぐ，計算

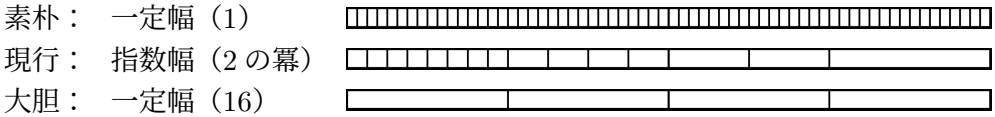

図 2: 各アルゴリズムで走査する離散時刻．左端が現在の時刻 0 に対応し，左から右に時間が発展

表 1: 実験結果のまとめ（評価指標の定義や数値の詳細を割愛し，傾向のみ示す）

シナリオ１	アルゴリズム		
評価指標	素朴	現行	大胆
稼働率（充填率）	◎	○	△
公平性（線形性）	△	◎	×
計算速度	×	○	◎

シナリオ２	アルゴリズム		
評価指標	素朴	現行	大胆
稼働率（充填率）	△	○	◎
公平性（線形性）	◎	◎	×
計算速度	×	○	◎

開始予定時刻を付与する（ユーザに通知する）こと以外には特に制約はない．

素朴なアルゴリズムは，(i) 到着したジョブを各時刻に計算開始可能か（配置可能性も含めて）判定し，計算開始可能な最早時刻を初期の計算開始予定時刻とし，(ii) あるジョブが終了ないし強制終了したら，未実行の各ジョブに対し，(i) と同じ手順で前倒しが可能か検討してリスケジュールを行なうというものである．配置可能性は各時刻に対して計算ラック（2 次元）上を Bottom-Left 法によって走査することで判定する．

上述の素朴なアルゴリズムは，T の全時刻を走査するため，処理が遅く，実用的でない．そこで現行のアルゴリズムでは，近い将来は細かく，遠い将来は粗く，T の一部の時刻のみを走査する．ここでは，図 2 の 2 段目に示したように，時刻 0 の次は時刻 2 を走査し，時刻 16 の次は時刻 20 を走査し，と 2 の冪の粒度で徐々に走査を粗くする方法を考え，これを**現行アルゴリズム**と呼ぶ．また，計算コストの観点からは，図 2 の 3 段目に示したように，最初から走査を粗くすることも考えられる．これを**大胆なアルゴリズム**と呼ぶ．

4. 予備実験結果

本発表では上述の 3 つのアルゴリズムに関する性能検証の結果を報告する．このために実データを模したランダムデータを作成した．

まず，実実行時間が精緻に予測でき，実行時間上限 = 実行時間上限となる状況を考える（**シナリオ１**）．このとき，結果は表 1 左のようになった．稼働率と計算速度に関しては予想通りの結果となった．計算速度は計算のボトルネックとなる Bottom-Left 法の呼び出し回数で評価した．この結果からは「現行」が最も望ましいと言える．

つぎに，実行時間上限が（平均すると）実行時間上限の約半分となる状況を考える（**シナリオ２**）．計算速度はシナリオ１と同様であるが，稼働率は「大胆」が最良となった．これは，頻繁に行なわれるリスケジューリングにおいて，現在時刻に近い時刻にジョブを配置しない「大胆」が最も柔軟に対応できる（言い換えると，粗く見ることがバッファの役割を果たす）ためであると推察する．

5. おわりに

上述のように，シナリオや評価指標（また，ここでは示していないが，ジョブの到着確率等）によって「最良」のアルゴリズムは異なる．これらのアルゴリズムを使い分ける方法や有機的に融合する方法を検討することが今後の課題である．また，アルゴリズムのパラメータ（たとえば「2 の冪」の 2）の決め方についても追加実験を通して理解を深めていきたい．

参考文献

[1] 今堀慎治, 梅谷俊治 (2005): 切出し・詰込み問題とその応用: (2) 長方形詰込み問題．オペレーションズ・リサーチ: 経営の科学. **50**, 335–340.

[2] 山本啓二, 宇野篤也, 関澤龍一, 若林大輔, 庄司文由 (2014): 区間スケジューリングを用いたジョブスケジューリングの性能評価. ハイパフォーマンスコンピューティング, **3**, 1–5.