Vol.2018-HPC-166 No.2
2018/9/27

正確な実行時間指定を促すジョブスケジューリング

滝澤 真一朗[1,a)]　高野 了成[1]

概要：スーパーコンピュータのような大規模並列計算機システムは，複数の利用者からの計算要求を処理するために，バッチジョブスケジューラによるジョブ管理を行っており，広く Backfill が用いられている．Backfill を効率よく動作させるためには，利用者によるジョブ実行時間の見積が正しくある必要があるが，運用されている多くのシステムで正しくないことが観測されている．我々は，利用者がより正確なジョブ実行時間見積を指定するインセンティブをジョブスケジューリングアルゴリズムに導入する．利用者のジョブ実行時間見積の正確性を示す指標を導入し，その指標に応じた優先度で Backfill する手法を提案する．実システムのワークロードを用いた評価の結果，一般的な Backfill に対して提案手法は最大で 6%の処理性能が向上したこと，より正確な実行時間見積を指定した利用者は平均待ち時間が短縮される等の優遇を受けることを確認した．

1. はじめに

スーパーコンピュータに代表される，複数の利用者に同時利用される共有並列計算機システムでは，従来よりバッチジョブスケジューリングシステム（以降，ジョブスケジューラ）により，利用者の計算はジョブという単位で実行管理されている．ジョブスケジューラにより計算を管理することにより，並列計算機資源を適切に空間・時間分割し，重複なくジョブに資源を配分すること，資源の有効活用が可能となる．従来の共有並列計算機システムは主に MPI を用いた科学技術計算が中心であったが，GPU の HPC での利用や深層学習での利用普及に伴い，GPU を用いた深層学習，分散深層学習のワークロードも実行されつつある．それに伴い，ジョブスケジューラが管理するジョブの特徴も変わりつつあり，利便性やシステム利用率向上を目的に，ジョブ特徴を考慮した資源管理・ジョブスケジューリング方針が定められている [1].

ジョブの特徴としてシステム利用効率に大きく影響を与えるものが並列度と実行時間である．一般的なジョブスケジューラでは，利用者が投入したジョブはキューに格納されて管理される．FCFS (First Come, First Serve) や，特定の優先度順にジョブを実行するスケジューリング方針では，様々な並列度と実行時間を持つジョブが複数混在すると，一部の計算機資源が使用されない時間帯（以降，空きスロット）が生じ，資源利用率が低下する．その問題を解決するために Backfill が提案され，近年のジョブスケジューラで実際に運用されている [2], [3]．Backfill とは，キューの後方のジョブであっても，キュー前方ジョブの実行開始時刻を遅らせない限り，それらを飛び越えて，空きスロットで実行する仕組みである．様々な方式の Backfill が提案されており，Backfill を採用することで資源利用効率，ジョブの平均待ち時間が短縮されることが知られている．

しかしながら，Backfill を十分に機能させるには，利用者によるジョブの実行時間の見積もりが正しくある必要がある．Backfill するジョブを決める際にはジョブの実行時間を評価する必要があるが，システムが実行時間を事前に正確に予測することは不可能であるため，利用者が指定する「実行時間見積」が用いられているためである．実行時間見積が実際の実行時間よりも短い場合には，ジョブは強制終了され，結果を得ることができない．そのため，利用者は実行時間見積を実実行時間よりも大きく見積もることが多い．しかし今度は，本来ならば Backfill できた空きスロットに Backfill されない，Backfill されたジョブも短時間で終了し依然空きスロットが残ったまま，となり，資源利用率が低下する問題がある．従来の科学技術計算ワークロードであれば，最大実行時間もたかだか 1 日から数日のジョブが大部分であったが [4], [5]，近年の深層学習ワークロードは 1 週間を超えるジョブもあり，実行時間の見積と実際に大きな乖離があった場合の影響が大きい．Backfill されることによりジョブの待ち時間が減るので利用者へのメリットは大きいものの，2 章に示す通り，Backfill が最初に提案されてから 20 年以上経つ今でも，実行時間の見積と実際には依然乖離があり，問題として残っている．一方

[1] 産総研・東工大 実社会ビッグデータ活用 オープンイノベーションラボラトリ, RWBC-OIL
[a)] shinichiro.takizawa@aist.go.jp

で，深層学習ワークロードはイテレーション処理であること，チェックポイントリスタートが比較的容易に行える点より，利用者への教育・動機付けが行えれば，見積の改善の余地はあると考えている．

本研究では，ジョブの実行時間見積と実実行時間の乖離を減らし，Backfill がより効果的に働くよう，利用者に，より正確な実行時間見積を指定するインセンティブを与える Backfill 手法を提案する．利用者が指定する実行時間見積の正確性を示す指標を導入し，その指標が高い利用者のジョブを優先的に Backfill する．本指標は，利用者が継続的に正確な見積値を指定するよう，ジョブ完了ごとに更新される．実システムのワークロードに即したワークロードを用いてシミュレーションによる評価を行った結果，提案手法を用いることで，既存手法より最大 6%処理性能が改善することを確認した．また，より正確な実行時間見積を指定することで，待ち時間が短縮される等の優遇を受けること，逆に不正確な見積値を指定することで，待ち時間が増加する等の罰則を受けることを確認した．

2. ワークロード分析

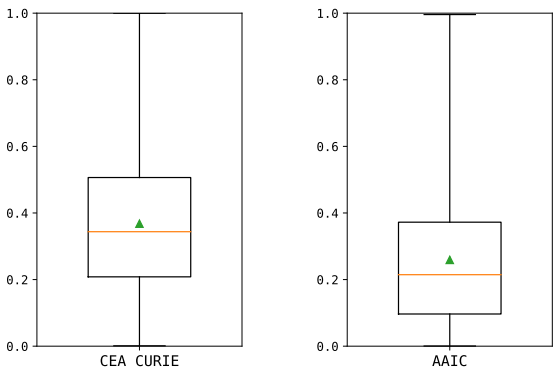
本章では，従来から PC クラスタやスーパーコンピュータで実行されていた科学技術計算のワークロードと，近年の GPU を搭載した並列システムで実行されている深層学習のワークロードを，実行時間見積の正確性の点で分析を行い，課題を明らかにする．科学技術計算のワークロードとしては，Parallel Workloads Archive（以降 PWA） [6] にて公開されている RICC，CEA CURIE，UniLu Gaia の 3 種類を用い，深層学習のワークロードとしては産総研が有する AI クラウド AIST Artificial Intelligence Cloud（以降，AAIC）から取得した情報を用いた．それぞれのワークロードの特徴を表 1 に示す．実実行時間が 90 秒以下のジョブはテストジョブとみなし除外した．表中の WRA は *Walltime Request Accuracy* の略であり，ジョブごとに定義される WRA の平均値である．各ジョブの WRA は実行時間見積と実実行時間より次の通りに計算される値であり，1 に近づくほど正確な実行時間見積を指定していることを意味する．

$$WRA = \frac{WalltimeEstimate}{ActualWalltime} \quad (1)$$

2.1 実行時間見積の正確性

表 1 より，各システムにおける WRA は低く，最低の AAIC では 0.1 以下である．これは，平均して利用者は実実行時間の 10 倍の実行時間を見積もっていることを意味する．最高の CEA CURIE の場合でも WRA は 0.3 程度であり，約 3 倍の実行時間を見積もっていることがわかる．

CEA CURIE と AAIC について，利用者ごとの WRA の平均値（以降，WRSA，*Walltime Request Specification Accuracy* と呼ぶ）を箱ひげ図として図示したものを図 1 に示す．オレンジのバーは中央値であり，緑の三角は平均値である．WRA の良い，CEA CURIE の方が平均的に良い WRSA となっているが，どちらの結果もばらつきが大きく，利用者ごとの実行時間見積の正しさにばらつきがあることが分かる．



(a) CEA CURIE の WRSA　　(b) AAIC での WRSA

図 1: CEA CURIE と AAIC での WRSA

次に，CEA CURIE と AAIC について，実行時間見積に対する WRA の分布を調査した結果を図 2 に示す．横軸は実行時間見積であり，図上部は実行時間見積に対する全ジョブの WRA の分布を表し，図下部は 1 時間毎の WRA の平均を表す．図より，以下が観測される．

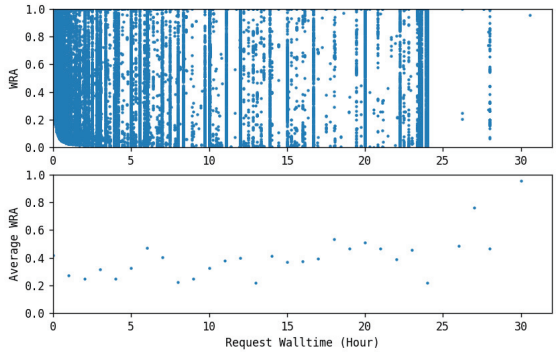
- 実行時間見積の分布には大きな偏りがある．デフォルト実行時間，最大実行時間を指定するジョブ，1 時間・24 時間などを単位に大雑把に実行時間を指定するジョブが多いと思われる．
- 実行時間が短いジョブの方が実行時間の見積もりは比較的容易と考えられるが，実行時間見積の大きさと WRA には関係がないよう見える．

2.2 解決すべき問題

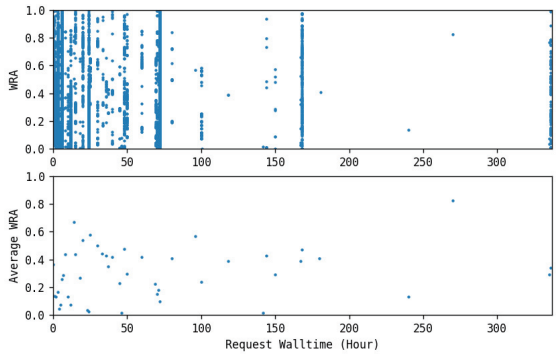
2.1 章の解析より，WRA は低いこと，長時間実行を要求するジョブの出現が確認された．長時間実行ジョブが増えているものの，WRA が低い場合，実行時間見積が外れた場合のペナルティが大きい．例を図 3 に示す．8 個のジョブが存在し，ジョブ A,B,D,E が実行中で，ジョブ C がスケジュール済み実行待ち状態，残り 3 ジョブがスケジュール待ち状態であることを表している．ジョブを示す白枠全体が利用者が指定した実行時間見積であり，オレンジ枠が実実行時間を表す．ジョブ H の実実行時間を考慮すると，ジョブ E 実行終了後の資源にて実行可能ではあるが，実行時間見積が長いため Backfill されない．これにより，利用者には待ち時間が増加するペナルティが，システム運用者には資源利用率が低下するペナルティが発生する．また，

表 1: ワークロード特性

System	Worload Duration	#Users	#Jobs	Max Walltime (hour)	WRA
RICC	May 2010 to Sep 2010	146	337272	72	0.143
CEA CURIE	Feb 2011 to Oct 2012	526	156318	31	0.295
UniLu Gaia	May to Aug 2014	76	37159	900	0.191
AAIC	Jul 2017 to Jun 2018	78	201258	336	0.090



(a) CEA CURIE の WRA の分布



(b) AAIC での WRA の分布

図 2: CEA CURIE と AAIC での WRA の分布

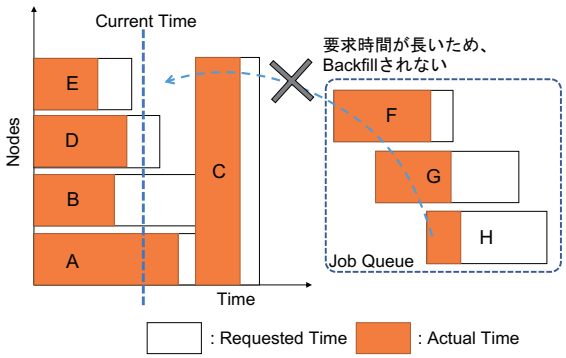


図 3: Backfill が機能しないシナリオ

ジョブ C の実行をブロックしているジョブ A，B，D，E において，WRA に大きなばらつきがあった場合，もっとも WRA が大きいジョブによりジョブ C の実行はブロックされ続けたままとなる．この場合，WRA が小さいジョブ B，D，E 終了後の資源に適切な Backfill が行えないかぎり，資源利用率は向上しない．

Backfill を採用するシステムにおいて実行時間見積の改善は資源利用率に影響するため，システム運用者から利用者への正確な値の指定についての教育は行われているはずである．しかしながら，文献 [5] では，4 年間のスーパーコンピュータの運用を通じた WRA の解析を行なった結果，WRA は運用期間とともに改善することはなかった，と報告している．このため，利用者がより正確な実行時間見積を指定するためには，そうすることへのインセンティブをジョブスケジューラの運用に組み込むことが手段の 1 つとして考えられる．

3. 提案

3.1 スケジューリングの目標

我々は，利用者の実行時間見積をより正確にするために，ジョブスケジューリングポリシにインセンティブを実装するアプローチを取る．インセンティブとして課金を優遇するアプローチも提案されており効果が示されているが [7]，スケジューリングポリシに対してのインセンティブ実装の効果は明らかにされていない．

ここでのスケジューリング目標は，実行時間見積を正しく指定した利用者のジョブは優先して実行され，時間経過とともに利用者の実行時間見積がより正確になることである．従来の Backfill でも正確な実行時間見積を指定したジョブが Backfill される確率が上がり待ち時間が短縮されるが，利用者の区別は考慮されておらず，実行時間見積の改善を促すものでもない．インセンティブをスケジューリングポリシに実装するにあたっての課題は，(1) 十分なインセンティブを提供できるか，(2) 資源利用率向上との両立，にある．利用者にとって十分なメリットがない限り，実行時間見積の改善は見込まれず，また，実行時間見積が改善しても資源利用率が向上しない限り，運用者にとってメリットがないためである．

一方で公平性の担保も重要である．具体的には，実行時間見積が正確な利用者のジョブのみ優先的に実行され，正確でない利用者のジョブが実行されない状況になってはならない．これらの制約を同時に満たす，スケジューリングポリシの設計が目標である．

3.2 設計

本研究では，利用者毎の WRSA を考慮した Backfill を提案する．従来の Backfill 同様，ベースとするスケジューリングポリシにてスケジュールされたジョブとジョブとの隙間である空きスロットを，提案 Backfill を用いてキューの後方ジョブで可能な限り埋める．ベースとするスケジューリングポリシはどれでも構わないが，4 章での評価を含め，本研究ではジョブ投入時刻に対して公平である FCFS をベースとする．

提案 Backfill の処理の概要を以下に示す．

(1) キュー内で実行待ち状態であるジョブ全てを，ジョブ投入者の WRSA の大きい順に Backfill する
(2) ジョブ完了後，該当ジョブの投入者の WRSA を更新する

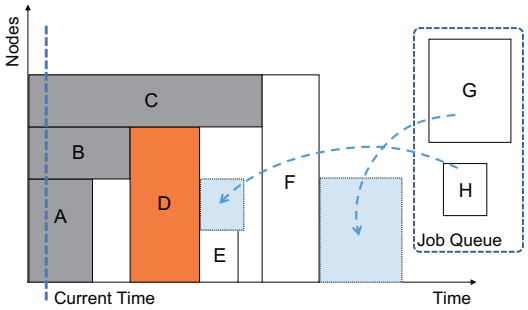
Backfill 処理の実行タイミングはスケジューラの実装依存であるが（ジョブ投入時，ジョブ終了時，定期的，など），上記の (1) は Backfill 処理ごと，(2) はジョブ終了ごとに実施される．

(1) の Backfill 処理において，既存のスケジューリング判断の尊重度合に応じて，2 種類の Backfill 手法を提案する．ここで既存のスケジューリング判断とは，あるジョブの Backfill 処理実施前までに Backfill 処理された全ジョブの，実行場所と実行開始時刻を意味する．

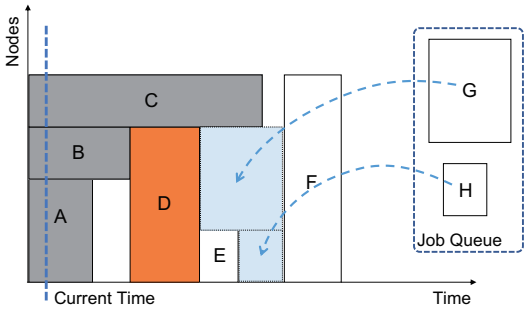
1 つ目は，既存の全てのスケジューリング判断を尊重し，空きスロットの時間以下のジョブのみを Backfill する手法である．この手法を WRSABF-AR（WRSA-aware BackFill All Reserved の略）と呼ぶ．図 4a に WRSABF-AR のスケジューリング例を示す．A, B, C は実行中ジョブであり，D〜H はキューで実行待ちジョブである．実行待ちジョブの内，D はキューの先頭ジョブ，E, F は Backfill 処理完了済みジョブ，G, H はこれから Backfill 処理を行うジョブを意味する．空きスロットの時間以下のジョブを Backfill するので，H が G に先行してスケジュールされる．ジョブの WRA が正確であるならば，空きスロットを時間方向に埋めることができる．

2 つ目は，キューの先頭ジョブのスケジューリング判断のみを尊重し，先頭ジョブの開始時刻を遅延させない制約の下，実行時間が空きスロットの時間以上（$\alpha\times$ 空きスロットの時間 ($\alpha > 1.0$)）となるジョブも Backfill する手法である．この手法を WRSABF-OR（WRSA-aware BackFill One Reserved の略）と呼ぶ．図 4b は WRSABF-OR のスケジューリング例である．F の実行開始時刻を遅延させて，G, H をスケジュールすることにより，WRSABF-AR よりも短い時間で全ジョブ完了する．WRSABF-OR を採用することにより，空きスロットを時間方向のみならず，並列度方向に埋める機会が増える．

WRSABF-AR，WRSABF-OR 共にキューの先頭のジョブは必ず優先してスケジュールされるため，ジョブ実行は保証される．WRSA を考慮する点を除くと，前者は従来 Backfill における保守的 Backfill，後者は EASY-Backfill に相当する [2].



(a) WRSABF-AR のスケジューリング結果



(b) WRSABF-OR のスケジューリング結果

図 4: All Reserved と One Reserved のスケジューリング結果の比較

(2) の WRSA 更新について，利用者が継続して正確な実行時間を指定し続けるよう，ジョブ実行完了のたびに以下の通りに更新する．

$$WRSA_i = \begin{cases} 1.0 & (\text{if } i = 0), \\ \beta \times WRSA_{i-1} + (1-\beta) \times WRA_i & (\text{if } i > 0). \end{cases} \quad (2)$$

WRA_i は利用者が投入したジョブの内 i 番目に完了したジョブの WRA，$WRSA_i$ は i 番目に完了したジョブまでの WRA を考慮して算出した WRSA である．過去のジョブから算出した WRSA よりも直近のジョブの WRA に加重を置くため，$\beta < 0.5$ とする．

4. 評価

提案 2 手法と EASY Backfill（以降 EASYBF），単純に投入者の WRSA の大きい順にジョブ実行する手法（以降 LWJF: Largest WRSA Job First）の比較評価を，次の 3 指標にて行う．

Makespan 最初のジョブが投入されてから，最後のジョブが終了するまでに要した時間である．ジョブスケジューラのスループットを評価する指標となる．

Waittime 一般にジョブ投入後，実行されるまでの待ち時

間が短い方が利用者にとってメリットが大きい．ジョブの待ち時間は並列度に依存するため，1 並列単位のジョブの待ち時間を評価する．具体的には，ジョブの待ち時間をジョブの要求ノード数で割った値を，ジョブの Waittime とする．利用者ごとのジョブの Waittime の平均を，該当利用者の Waittime とする．

Delay EASY Backfill や提案 2 手法では，他ジョブの早期終了の影響のみでなく，Backfill アルゴリズムの設計上，実行開始が予定された時刻から遅延される場合がある．ジョブスケジューラには実行開始予定時刻を通知する機能を有するものもあり，この遅延が小さい方が正確な時刻を通知できる点で，利用者にとってのメリットが大きい．

ジョブが最初に実行が予約された時刻を T_0，実際に実行開始された時刻を T_1 とした時に，各ジョブの遅延時間を以下の通りに定義し，利用者ごとのジョブの遅延時間の平均を，該当利用者の Delay とする．

$$Job\_Delay = \begin{cases} 0 & (\text{if } T_0 >= T_1), \\ T_1 - T_0 & (\text{if } T_0 < T_1). \end{cases} \quad (3)$$

Slowdown 各ジョブの Slowdown は $(Wait\_time + Run\_time)/Run\_time$ と定義される，ジョブスケジューラにより受けた遅延割合を表す指標である．小さい値が遅延の影響が少なく，利用者にとってのメリットが大きい．利用者ごとのジョブの Slowdown の平均を，概要利用者の Slowdown とする．

4.1 評価設定

評価はシミュレーションにて，実システムのワークロードを元に生成したワークロードを用いて行った．シミュレーションはジョブスケジューラの振る舞いを模倣する離散イベントシミュレータを実装し，次に示す設定で実行した．

- シミュレーションの 1 ステップは 60 秒とした．
- ジョブキューは 1 つとした．
- ジョブへの資源割り当て単位はノード単位とした．
- 要求並列数を満たすノード群が最も早く利用可能になる時刻にジョブ実行を予約する．
- ノードには通し番号を振り，同時刻に要求並列数以上のノードが利用可能な場合，番号の小さい順にジョブに割り当てる．
- ジョブ投入・完了イベント発生時にスケジューリング判断をする．
- キュー内のジョブのうち，最大 100 ジョブを 1 度のスケジューリング対象とした．[*1]
- 3 章のパラメータはそれぞれ $\alpha = 1.5$，$\beta = 0.3$ とした．

[*1] Slurm での default_queue_depth パラメータのデフォルト値と合わせた．

ワークロードは，AAIC の GPU を用いた深層学習ワークロード（以降 AAIC ワークロード），PWA で公開されている RICC ワークロードを元に 3 種類生成した．

Original 取得した，あるいは公開されているワークロードから，資源要求が正しく，実行に成功しているジョブを抽出して生成したワークロード．

100% WRA Oritinal にて，全てのジョブの WRA を 100%とし，実行時間見積を実実行時間と等しく設定したワークロード．

20% WRA Oritinal にて，全てのジョブの WRA が 20%とし，実行時間見積を「実実行時間 ÷ WRA」と設定したワークロード．

各ワークロード 10000 ジョブから生成される．一部の結果では，混雑状況をエミュレートするために，ジョブ到着間隔（IAT: Inter Arrival Time）をそれぞれ 1/2，1/4，1/8 にしたワークロードも用いている．

AAIC ワークロード個別の設定として，90 秒未満の短時間ジョブは除外した．AAIC 実環境相当の 32 ノードの設定としたが，簡易化のため，1GPU を 1 ノードと見立ててシミュレーションを行った．各ノードは 8GPU を搭載するため，256 ノードの環境をシミュレートしたことになる．RICC ワークロード個別の設定として，PWA で公開されている並列数の単位が CPU コア数であるため，ジョブの要求 CPU コア数を，1 ノード搭載コア数である 8 で割った値（切上げ）を，ジョブの要求ノード数とした．総ノード数は公開情報を元に 1024 ノードとした．

4.2 評価結果

4.2.1 Makespan

Makespan の評価結果を表 2 と 3 に示す．各ワークロードにて，IAT を調整した結果も示している．

Oritinal において，AAIC，RICC 共に WRSA-OR は EASYBF に対して 2〜6%Makespan が短縮しているが，IAT を短縮した場合には増加している．特に，EASYBF の場合は IAT を 1/8 に短縮しても Makespan の増加は 14%程度だが，提案 2 手法では最大で 190%程増加している．Makespan の増加は資源利用率の低下を意味する．この理由は，IAT を短縮することにより，キュー内にてスケジュール待ちジョブが増えるものの，先頭 100 ジョブまで（EASYBF の場合先着順，提案 2 手法の場合 WRSA の大きい順）をスケジュール対象としているため，最適なスケジューリングが行えていないことにあると考えている．EASYBF は厳密な先着順で Backfill を行うため，100 ジョブ中には様々な利用者の様々な並列度のジョブが含まれると想定される．一方で投入者の WRSA 順に選ばれた 100 ジョブには，投入者の偏りが大きくなり，ジョブ特性も似てくる可能性が高い．特性の似た一部の並列ジョブにて，必要ノード数を確保できずにスケジュールできず，未使用

となったノード群が生じたと考えている．

スケジューリングのオーバーヘッドを減らすために，最大 100 ジョブまでを対象とする条件で評価を行ったが，同様な制約は実環境でも設定されうる．スケジュール対象のジョブ数に制約を設けたときの，資源利用率向上が課題となる．

そのほかの特徴とし，100% WRA では提案 2 手法と EASYBF の結果がほぼ等しくなることがわかる．全利用者の WRSA が等しくなるため，特に WRSA-OR は EASYBF と完全に一致する．20% WRA では，提案 2 手法は EASYBF 同様に，全般的に 100% WRA よりも大きな Makespan となっており，資源利用率が低下していることを示している．提案 2 手法においては，平均的には WRSA-OR の方が優れた結果となった．3 章で述べたとおり，WRSA-OR の方が時間軸方向に加え並列軸方向にも資源を埋める機会が増えたためと考えている．

4.2.2 Waittime, Delay, Slowdown

図 5 と 6 に，各 Original ワークロードでの Waittime，Delay，Slowdown の結果を散布図にて示す．利用者ごとに WRSA が異なる Original を対象とし，同様な傾向を示した IAT を短縮したワークロードの結果は省略する．図中の点は個々の利用者を表し，縦軸が利用者の WRSA，横軸が 0-1 範囲に正規化された各評価値である．投入ジョブ数が 10 以上の利用者のみを対象とした．結果，AAIC ワークロードでは 22 名，RICC ワークロードでは 43 名を図示している．図中破線は $y = -x + 1$ を表す．WRSA の増加に伴い，Waittime，Delay，Slowdown が小さくなればジョブ実行が優遇されることになる．この破線に平行した分布を示すことができれば，インセンティブ設計として完璧であると言える．

図 5a，5c，6a，6b では，提案 2 手法にて，高い WRSA を持つ利用者の評価値が，EASYBF での評価値よりも優れた値（より小さい値）となっていることがわかる．高い WRSA を持つ利用者のジョブが優遇されていることを意味する．また，図 5b，6b において，WRSA が小さい範囲において，提案 2 手法の方が EASYBF よりも分布が広がっていることが確認できる．低い WRSA を持つ利用者がペナルティを受ける機会が増えたことを意味する．

WRSA の違いに対する各種評価値の差別化を，既存の EASYBF よりも厳密に行えている点で，ジョブの実行時間見積を正確にすることへのインセンティブを高めているが，以下に示す改善点も残されている．以下の課題を解決するよう，スケジューリングアルゴリズムの改良を行っていく．

- WRSA が小さいものの，優れた評価値をもつ利用者もいる．
- 多くの点が $y = -x + 1$ の下側にあるにかかわらず，数点は上側にある．これは一部の利用者に過剰なペナルティを与えていることを意味する．提案手法を用いることで，そのような点は減っているが，図 5b の AAIC の Delay では高い WRSA を持つ利用者が高いペナルティを受ける結果となっている．

5. 関連研究

様々な並列度・実行時間のジョブが混在する環境での資源利用率向上のために Backfill は提案された [2], [8], [9]．Backfill では，実行時間見積を超えたジョブは打ち切られるため，2 章で示した通り，利用者は実行時間を大きく見積もる傾向にある．その場合の Backfill の振る舞いについて多くの評価がされている [10], [11], [12]．これらではワークロードや Backfill 方針に依存するが，実行時間見積が正確な方が Makespan は向上するものの，一方で，不正確な方が待ち時間や Slowdown などの利便性が改善する結果を示すものもある．本研究では，システム運用の視点から Makespan 向上を第 1 目的とし，選択的に利用者の利便性を改善することを目指している．

保守的 Backfill，EASY Backfill が広く採用されているが，それらを改善する研究もある．Wong らは，後続ジョブが開始されるまで空き時間がある場合には，実行時間見積を超過しても実行を許可する Backfill を提案し，ジョブ実行完了率を向上させている [13]．Srinivasan らは，キュー内で一定時間待ったジョブを Backfill 対象とする Selective Backfilling Strategy を提案している [14]．十分なサイズの空きスロットがない限り Backfill されない保守的 Backfill と，高並列ジョブはキューの先頭になるまで待ち続けなければならない EASY Backfill の両方の欠点を補う提案である．Niu らは，チェックポイント・リスタートによるジョブプリエンプションを導入することで，資源利用率向上，待ち時間削減を実証している [15]．Shah らは，Backfill を実行時間見積とジョブ並列度を考慮した 2 次元ビンパッキング問題とみなし，それを効率よく解くヒューリスティックを提案している [16]．これらの提案を採用し，我々の提案を改善することを検討している．また，Chiang らは短時間ジョブを優先して Backfill するなど，特定の評価値による優先度順 Backfill を提案している [11]．我々の提案は彼らの提案に近い．

本研究では利用者による実行時間見積の改善を目的としているが，一方で，システムがジョブ実行時間を予測し，予測値ベースで Backfill を行うことも可能である．Galleguillos らは，実行時間予測手法として，利用者毎のジョブ履歴を蓄積し，新規ジョブが投入された時には履歴よりもっとも特性が似たジョブを探し出す手法を提案している [17]．Tsafrir らは，ジョブ投入利用者の直近 2 ジョブの実行時間の平均を実行時間予測として用いること，及び，利用者による実行時間見積はジョブ終了の判断のみに用い，システムによる予測値をスケジューリング判断に用いる使い分

表 2: AAIC ワークロードによる Makespan の比較．単位は秒．括弧内は EASYBF に対する相対 Makespan．

	IAT	WRSA-AR	WRSA-OR	LWJF	EASYBF
Original		8776740 (0.96)	8624640 (0.94)	9509640 (1.04)	9136440 (1.00)
	1/2	6918720 (1.02)	7022640 (1.04)	14352540 (2.13)	6754080 (1.00)
	1/4	11467260 (1.32)	12855840 (1.48)	18382800 (2.11)	8694240 (1.00)
	1/8	16699560 (1.60)	16624500 (1.60)	18782340 (1.80)	10412580 (1.00)
100% WRA		8169360 (1.00)	8169360 (1.00)	8433660 (1.03)	8169360 (1.00)
	1/2	5188140 (1.13)	4600740 (1.00)	6052980 (1.32)	4600740 (1.00)
	1/4	4057680 (1.00)	4051140 (1.00)	5535900 (1.37)	4051140 (1.00)
	1/8	3844920 (1.00)	3837060 (1.00)	5323080 (1.39)	3837060 (1.00)
20% WRA		11928720 (1.45)	10643160 (1.29)	12145620 (1.48)	8223360 (1.00)
	1/2	8492580 (0.97)	7635240 (0.87)	14285040 (1.63)	8784540 (1.00)
	1/4	9724320 (1.34)	7355280 (1.02)	15089880 (2.08)	7238580 (1.00)
	1/8	11554800 (1.16)	9107700 (0.92)	15157380 (1.53)	9928800 (1.00)

表 3: RICC ワークロードによる Makespan の比較．単位は秒．括弧内は EASYBF に対する相対 Makespan．

	IAT	WRSA-AR	WRSA-OR	LWJF	EASYBF
Original		1674180 (1.03)	1601640 (0.98)	2737440 (1.68)	1632480 (1.00)
	1/2	1740840 (0.89)	3031980 (1.56)	2446980 (1.26)	1948680 (1.00)
	1/4	2784300 (1.77)	3481080 (2.22)	1744620 (1.11)	1569960 (1.00)
	1/8	1575360 (0.94)	3192600 (1.90)	1828200 (1.09)	1682400 (1.00)
100% WRA		1449120 (1.00)	1451220 (1.00)	1461180 (1.01)	1451220 (1.00)
	1/2	1329000 (1.00)	1332240 (1.00)	1346760 (1.01)	1332240 (1.00)
	1/4	1302720 (1.00)	1305360 (1.00)	1319460 (1.01)	1305360 (1.00)
	1/8	1294980 (1.00)	1295940 (1.00)	1310520 (1.01)	1295940 (1.00)
20% WRA		1938600 (0.82)	1832760 (0.78)	2582400 (1.10)	2353200 (1.00)
	1/2	3277500 (1.32)	3656580 (1.47)	2148960 (0.86)	2489760 (1.00)
	1/4	3745740 (0.70)	4207740 (0.79)	4592820 (0.86)	5349540 (1.00)
	1/8	6025440 (1.15)	5905200 (1.13)	3500400 (0.67)	5243640 (1.00)

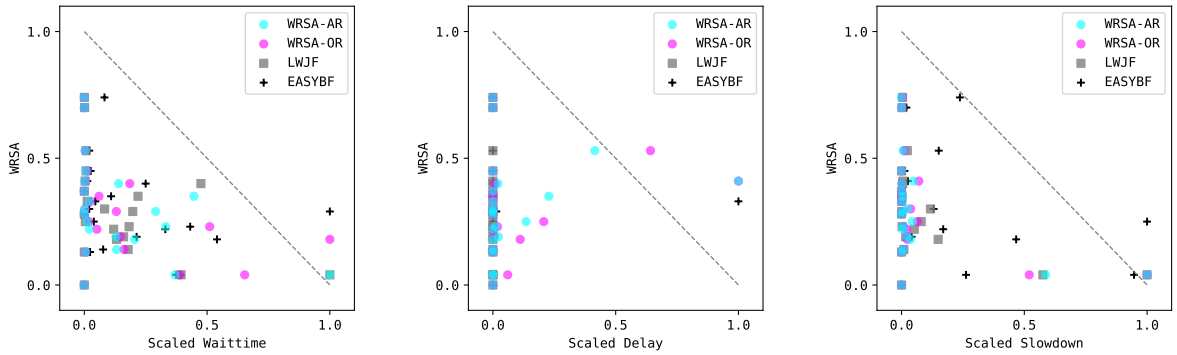
けを提案している [18]．Gaussier らは，機械学習を用いてジョブ実行時間を予測する手法を提案している [19]．これらはヒューリスティックであり，完璧な予測はできないため，利用者の見積を正確にする方が有用であると考えている．一方で，我々の提案に実行時間予測を組み合わせることで，利用者の実行時間見積の検証を行い，さらなる向上へと繋がる可能性もあると考えている．

システム運用者が期待する行動を利用者がするよう，行動にインセンティブを設定する運用が提案されている．野村らは，実行時間見積を改善することを目的に，正しい見積もりを行うことで課金を優遇する方法を提案し，TSUBAME での運用を通じて実証を行った [7]．西川らは，並列プログラムの普及を目的に，高並列なジョブの課金を優遇する方法を提案し，FOCUS スパコンでの運用を通じて実証を行った [20]．課金優遇することで利用者に対してメリットが明確化されるため，行動を期待できるが，収入が減る問題がある．Georgiou らは，システムの省エネ化を目的に，電力効率の良い利用者のジョブを優先して実行するスケジューリングポリシを提案し，フェアシェアをベースに実現している [21]．スケジューリングポリシにインセンティブを実装する点では本研究と等しいが，目的と技術が異なる．

6. まとめ

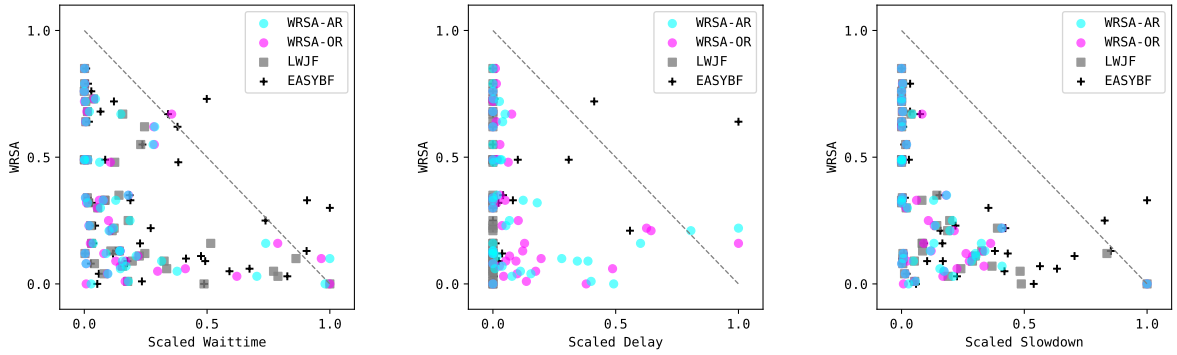
ジョブスケジューリングアルゴリズムに Backfill を採用する共有並列計算機システムにおいて，ジョブ実行時間見積の正確な指定は，資源利用効率向上，待ち時間削減につながるため，システム運用者・利用者の双方に利する．しかしながら，Backfill が提案され利用され始めて 20 年以上経つが，いまだに利用者の実行時間見積は正確ではなく，資源の浪費，ユーザビリティの低下が潜在的に起きている．この問題を解決するため，本研究では，より正確なジョブ実行時間見積を指定するインセンティブを提供する Backfill ジョブスケジューリングアルゴリズムを提案した．実システムのワークロードに即した擬似ワークロードを用いた評価の結果，EASY Backfill に対して，最大で 6%ほど Makespan が短縮され，資源利用効率が向上したことを確認した．また，ユーザビリティ観点では，実行待ち時間，実行開始遅延，Slowdown の 3 点において，実行時間見積が正確なほど優遇されることが確認できた．

今後の課題として，アルゴリズムのさらなる改良が挙げられる．具体的には，混雑時の資源利用率改善，正確な実



(a) Waittime (b) Delay (c) Slowdown

図 5: AAIC ワークロードによる Waittime, Delay, Slowdown の比較



(a) Waittime (b) Delay (c) Slowdown

図 6: RICC ワークロードによる Waittime, Delay, Slowdown の比較

行時間見積のインセンティブを高めるため，より明確な差別化，の実現を目指す．また，より大きなワークロードでの評価を行う．

参考文献

[1] Feitelson, D. G.: Workload Modeling for Performance Evaluation, *Performance Evaluation of Complex Systems: Techniques and Tools*, Vol. 2459, pp. 114–141 (2002).

[2] Mu'alem, A. W. and Feitelson, D. G.: Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 12, No. 6, pp. 529–543 (2001).

[3] Yoo, A. B., Jette, M. A. and Grondona, M.: SLURM: Simple Linux Utility for Resource Management, *Job Scheduling Strategies for Parallel Processing*, pp. 44–60 (2003).

[4] You, H. and Zhang, H.: Comprehensive Workload Analysis and Modeling of a Petascale Supercomputer, *Job Scheduling Strategies for Parallel Processing*, pp. 253–271 (2013).

[5] Rodrigo, G. P., Östberg, P.-O., Elmroth, E., Antypas, K., Gerber, R. and Ramakrishnan, L.: Towards understanding HPC users and systems: A NERSC case study, *Journal of Parallel and Distributed Computing*, Vol. 111, pp. 206–221 (2018).

[6] Feitelson, D.: Parallel Workloads Archive, `http://www.cs.huji.ac.il/labs/parallel/workload/index.html`.

[7] 野村哲弘，佐々木淳，三浦信一，遠藤敏夫，松岡聡：TSUBAME2 におけるスケジュール効率化への取り組みとユーザ動向の見える化，第 150 回ハイパフォーマンスコンピューティング研究会 (2015).

[8] Lifka, D. A.: The ANL/IBM SP Scheduling System, *Job Scheduling Strategies for Parallel Processing*, pp. 295–303 (1995).

[9] Feitelson, D. G. and Weil, A. M.: Utilization and Predictability in Scheduling the IBM SP2 with Backfilling, *Proceedings of the First Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing*, pp. 542–546 (1998).

[10] Wong, A. K. L. and Goscinski, A. M.: Evaluating the EASY-backfill job scheduling of static workloads on clusters, *2007 IEEE International Conference on Cluster Computing*, pp. 64–73 (2007).

[11] Chiang, S.-H., Arpaci-Dusseau, A. and Vernon, M. K.: The Impact of More Accurate Requested Runtimes on Production Job Scheduling Performance, *Job Scheduling Strategies for Parallel Processing*, pp. 103–127 (2002).

[12] Zotkin, D. and Keleher, P. J.: Job-length estimation and performance in backfilling schedulers, *Proceedings. The Eighth International Symposium on High Performance Distributed Computing*, pp. 236–243 (1999).

[13] Wong, A. K. and Goscinski, A. M.: The Impact of Under-Estimated Length of Jobs on EASY-Backfill Scheduling, *16th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP*

2008), pp. 343–350 (2008).

[14] Srinivasan, S., Kettimuthu, R., Subramani, V. and Sadayappan, P.: Selective Reservation Strategies for Backfill Job Scheduling, *Job Scheduling Strategies for Parallel Processing*, pp. 55–71 (2002).

[15] Niu, S., Zhai, J., Ma, X., Liu, M., Zhai, Y., Chen, W. and Zheng, W.: Employing Checkpoint to Improve Job Scheduling in Large-Scale Systems, *Job Scheduling Strategies for Parallel Processing*, pp. 36–55 (2013).

[16] Hussain Shah, S. M., Qureshi, K. and Rasheed, H.: Optimal job packing, a backfill scheduling optimization for a cluster of workstations, *The Journal of Supercomputing*, Vol. 54, No. 3, pp. 381–399 (2010).

[17] Galleguillos, C., Sîrbu, A., Kiziltan, Z., Babaoglu, O., Borghesi, A. and Bridi, T.: Data-Driven Job Dispatching in HPC Systems, *The Third International Conference on Machine Learning, Optimization, and Big Data*, pp. 449–461 (2018).

[18] Tsafrir, D., Etsion, Y. and Feitelson, D. G.: Backfilling Using System-Generated Predictions Rather than User Runtime Estimates, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 18, No. 6, pp. 789–803 (2007).

[19] Gaussier, E., Glesser, D., Reis, V. and Trystram, D.: Improving backfilling by using machine learning to predict running times, *SC15: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–10 (2015).

[20] 西川武志：FOCUS スーパーコンピュータシステムにおける並列課金インセンティブの効果 III，第 161 回ハイパフォーマンスコンピューティング研究会 (2017).

[21] Georgiou, Y., Glesser, D., Rzadca, K. and Trystram, D.: A Scheduler-Level Incentive Mechanism for Energy Efficiency in HPC, *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pp. 617–626 (2015).