Vol.2021-HPC-182 No.6
2021/12/6

深層強化学習を用いたジョブスケジューリングでの予測値の利用検証

滝澤 真一朗[1,a)]　野村 哲弘[2]　松葉 浩也[3]

概要：計算機システムの利用率最大化や待ち時間短縮のためにはより良いジョブスケジューリングが必要である．目的に応じた報酬設計を行うことで機械学習システムが方策を学ぶことから，深層強化学習を用いたジョブスケジューリングの研究が広まっている．しかしながら，既存研究ではジョブの要求実行時間のみを学習に用いているため，実際の実行時間との差異が学習されておらず，適切に学習できていない可能性がある．我々は，深層強化学習を用いたジョブスケジューリングにおいて，実行時に不確定なジョブ属性の予測値を考慮することの効果の検証を行う．具体的には，利用率最大化を目的として，学習時にはジョブ実行時間の要求値と実際値を入力にし，スケジューリング時にはジョブ実行時間の要求値と予測値を用いる．予測値を考慮しない場合との比較を行なった結果，考慮することは利用率の向上に貢献しない結果となったが，学習時に用いたワークロードと類似の傾向を持つワークロードをスケジュールする際に利用率が高くなる傾向にあることが確認できた．

1. はじめに

複数人で共有して使用する高性能計算機システムにおいて，ジョブスケジューリングは重要な役割を果たす．ジョブの実行順序を制御することにより，目的に応じた最適化を行えるからである．目的としては，システムの利用状況や設備環境の制約に応じた利用率，待ち時間，消費電力量の最適化や，利用者間の公平性実現などが採用されている．ジョブスケジューリングのためのアルゴリズムにはさまざまなものが提案されており，FCFS（*First Come First Serve*）や SJF（*Shortest Job First*）のような特定の指標に基づきジョブを並び替えたもの，利用者個人の過去の資源利用量を考慮したもの [1,2]，消費電力や実行時間の予測値を考慮したもの [3–6] など多数提案されている．目的に応じたシステムの最適化を行うためには，システム管理者は適切なアルゴリズムを選択し，運用状況をモニタリングしつつ，必要に応じて適切に設定を更新していく必要がある．

しかしながらこのようなジョブスケジューリングの運用は，卓越したシステム管理者であっても困難である．まず，システム管理者は対象システムを熟知している必要がある．これにはシステムの構成だけではなく，システムで実行されているワークロード[*1]の特性を把握していることが求められる．その知識をもとに適切なアルゴリズムやパラメータの選択をしなければならない．また，障害等でシステム構成が変更されたり，利用傾向が変わりワークロードに変化が生じることがある．時々刻々と変化するシステム状態に追従した最適化を継続的に行うことは人手では不可能である．さらには，将来，冷却や電源等の設備の状態も考慮したデータセンター全体の運用最適化が求められた場合，必要な知識とパラメータ量がさらに増えるため，より問題が複雑化する．このような背景から，ジョブスケジューリングに強化学習を用いることが注目されている．

強化学習とは機械学習の一種であり，対象とする環境と，その環境に対して行動を行うエージェントとから構成され，エージェントは環境から得られる報酬の累積を最大化するように取るべき行動を学習する．深層強化学習では，行動の判断を決める戦略をニューラルネットワークとして表現する [7]．ジョブスケジューリングへの適用では，システム状態を環境，次にスケジュールするジョブの選択を行動，利用率の最大化や待ち時間の最小化を報酬とした研究が提案されている [8–12]．報酬設計を行うだけでエージェントが適切にスケジューリング戦略を学習すること，また，システム状態変化に応じて再学習することで状態に適した戦略を学習できることから，卓越したシステム管理者でなくとも運用最適化のサイクルを実現する可能性を有

[1] 国立研究開発法人産業技術総合研究所
[2] 東京工業大学
[3] 東京大学
[a)] shinichiro.takizawa@aist.go.jp

[*1] 本論文では時系列に投入されたジョブ列からなるものとする

す．しかしながら，既存の深層強化学習を用いたジョブスケジューリングの研究では，ジョブ実行時間に関しては要求実行時間のみ学習しているため，実際の実行時間と要求実行時間の不一致によるスケジューリング判断の不正確性が排除できていない．一方で，既存の（深層強化学習を用いない）ジョブスケジューリングの研究では，実際の実行時間は利用できずとも，実行時間の予測値を用いることでスケジューリング性能指標が改善することを報告している [3–5]．深層強化学習を用いたジョブスケジューリングにおいても予測値を用いることで，性能指標の改善や，スケジューリング判断の正確性向上が期待できるが，効果について明らかになっていない．

本研究では，上記の疑問の解を出すことを目的に，深層強化学習を用いたジョブスケジューリングにおいて，ジョブ実行時間の予測値を用いることの効果を実験的に検証する．具体的には，学習時にはジョブ実行時間として要求値と実際値を入力とした学習を行い，スケジューリング時には要求値と予測値を用いて判断する手法を提案する．検証のための深層強化学習システムを PPO アルゴリズムを用いて開発し，人工的に生成した 6 種類のワークロードを用いて利用率を最大化するように学習を行なった．実行時間の予測値を含めて学習する場合，しない場合とで利用率を比較した結果，今回の評価では，予測値を考慮して学習しても利用率の向上には貢献しない結果となった．しかしながら，学習時とスケジュール時のワークロードの傾向が類似している場合，利用率が高くなる傾向にあることが確認できた．また，予測値を環境に含めることにより環境のデータ量が増えるため，それに伴いネットワークサイズを大きくすることにより，利用率が改善することが確認できた．

2. 関連研究

深層強化学習の発展に伴い，深層強化学習を計算機システムの資源管理に用いる研究が行われており，ワークフロースケジューリングの最適化 [13–15]，長期間サービスを提供するコンテナのスケジューリング最適化 [16]，ファイルシステムの性能チューニング [17] 等の報告がされている．本研究では，深層強化学習の資源管理への応用分野として，大規模並列システムにおけるジョブスケジューリングを対象とする．既存研究として，Mao らによる DeepRM [8] や Zhang らによる RLScheduler [9]，Bao らによる研究 [10]，Yi らによる研究 [11] が挙げられる．

我々の調べた限り，DeepRM [8] がジョブスケジューリングにおける深層強化学習の適用の効果を最初に検証した研究である．彼らの提案する強化学習手法は，環境を資源割当とジョブキューの状況からなる画像として表現し，それを全結合層からなるニューラルネットワークに入力として与え，スケジュールするジョブを出力する手法である．DeepRM を拡張し様々な技術を適用した研究もある．Chen らはネットワークに畳み込みニューラルネットワークを採用し [18]，Ye らはそれに加え，Imitation Learning を用いて SJF をベースに学習したポリシを初期値として使用する手法を提案している [19]．DeepRM は人工的な小規模なワークロードでの評価のみであったが，RLScheduler ではスーパーコンピュータで実行された大規模なワークロードを対象とし，それに適したネットワーク構成や学習データの削減手法を提案している [9]．Bao らは分散深層学習計算を効率よく実行するためのスケジューリングポリシを強化学習で学習し，実クラスタ環境での評価を行なっている [10]．Yi らはデータセンターの消費電力改善を目的に，深層強化学習を用いたジョブスケジューリングを行なっている [11]．彼らの提案は，シミュレータを用いずに，データセンターの環境も LSTM を用いた機械学習モデルで表現している点が特徴的である．これら研究ではジョブの実行時間として要求実行時間のみを学習に用いたり，無制限の実行時間を想定しており，実行時間の予測値を考慮した学習・スケジューリングは行なっていない．

ジョブの正確な実行時間と消費電力は，利用率向上や平均待ち時間改善，電力制約を満たすために重要な情報であるが，ジョブ実行完了後まで正確な値を取得することができない．そこで，これらの値の予測値を用いてスケジューリングする手法が提案されている．Tsafrir らは，利用者に指定されたジョブの要求実行時間をジョブの終了判断に用い，過去のジョブ実行履歴から予測したジョブ実行時間を Backfill 判断に用いる手法を提案している [3]．Tsafrir らの予想モデルは直近 2 ジョブの実行時間から予測する単純なモデルであるが，Tang らは属性でジョブを分類し，各分類ごとに簡単な統計処理をして予測する手法を提案 [5]，Gaussier らはジョブの様々な属性を考慮した回帰モデルを用いて予測する手法を提案している [4]．Wallance らは，システムの消費電力が一定値を超えないよう，ジョブの消費電力モデルを用いて予測した消費電力をスケジューリングに考慮する手法を提案している [6]．これら研究では，予測値を用いることでスケジューリング性能指標の改善や制約を満たすことを実現しているが，古典的なスケジューリング手法を基盤に用いている．

以上のように，深層強化学習をジョブスケジューリングポリシ学習に使用すること，実行時間や消費電力の予測値をジョブスケジューリングに使用することの効果は既存研究にて検証されているが，深層強化学習でジョブスケジューリングポリシを学習する際に予測値を考慮することの効果は検証されていない．本研究では，ジョブ実行時間の予測値を対象にその検証を行う点で既存研究と異なる．

3. 提案

関連研究に挙げた通り，既存の深層強化学習を用いたジョブスケジューリングでは，ジョブの要求実行時間しか

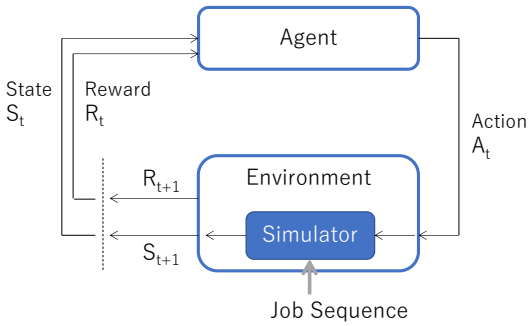

図 1 強化学習を用いたジョブスケジューリングの概念図

使用していない．そのため，実実行時間との差異が学習されておらず，学習の不正確性が排除できていない．本研究では，深層強化学習を用いたジョブスケジューリングにおいて，ジョブ実行時間の予測値を用いる手法を提案する．具体的には，学習時にはジョブ実行時間として要求値と実際値を入力とした学習を行い，スケジューリング時には要求値と予測値を用いて判断を行う．スケジューリング時に使用する予測値は，関連研究や既存研究 [20, 21] で提案されている手法等を用いて，本強化学習システムでのスケジューリングポリシ学習時と並行して予測モデルを学習し，本強化学習システムに提供されるものとする．実行時間の要求値と実際値の差異を考慮した学習を行うことで，学習時と類似したワークロードを実行する場合に，より正確なスケジューリング判断を行えるとともに，予測値を用いた既存のスケジューリング手法のように評価指標の改善が期待できる．期待通りの振る舞いが実証されれば，継続的にワークロードを収集し，スケジューリングモデルを更新し続ける運用の実現が期待できる．

4. 深層強化学習の設定

強化学習は，得られる累積報酬を最大化するように，エージェントが環境に対して行う行動を試行錯誤し，取るべき行動を学習する機械学習手法である．ジョブスケジューリングに強化学習を適用した場合の概念図を**図 1** に示す．ステップ t においてエージェントは環境を観測して状態 S_t を受け取り，行動 A_t を行う．行動 A_t を行なった結果，環境の状態は S_{t+1} に遷移し，エージェントは報酬 R_{t+1} を獲得する．この過程を繰り返す．環境に対する最初の行動から最後の行動までの期間をエピソードと呼び，エピソード全体で得られる累積報酬の最大化が学習の目的である．

以降，本研究におけるジョブスケジューリングを学習する強化学習の設定を説明する．

4.1 環境

強化学習の対象とする環境はジョブスケジューリングにより管理されている計算機システムと，そのシステムで実行されるワークロードから構成される．本研究で対象とするジョブスケジューリングでは，計算ノード単位で資源割り当てを行い，ジョブの中断・再開はないものとする．各ジョブは，到着時刻，並列度（ノード数），要求実行時間，実行時間の予測値（学習時は実際値）を属性として持つ．1 度のスケジューリング判断では，現在システムに存在する待ちジョブのうち，先頭 N_{jobs} 個のジョブを対象とする．この制限は強化学習の行動空間が有限であるために儲けているが，Slurm 等の実用ジョブスケジューラでも同様の制約が設けられている．

多数の強化学習の訓練計算を繰り返すために，環境はシミュレータにより提供される．シミュレータは，システム構成情報とワークロードを入力として受け取り，一定間隔（Sim_Step）で時刻を進めていく．以下のイベントが発生する場合に，ジョブスケジューリングを行う．

(1) ジョブ投入
(2) ジョブ終了
(3) 上以外で待ちジョブと未使用計算ノードが存在

(3) は，(1)(2) 発生時に N_{jobs} を超える待ちジョブ数が存在する場合，スケジューリング実施後も待ちジョブと未使用計算ノードが存在しうるために設けたイベントである．連続して (3) が発生する時は環境の状態がほとんど変化しないため，同じ行動が繰り返される可能性がある．学習を加速させるため，学習時に連続して (3) が発生する場合は時間間隔を倍にして時刻を進め，他のイベント発生を促す．

各イベント発生時には複数のジョブをスケジュールできる可能性がある．一方で，強化学習の各ステップでは 1 ジョブしか選択しないため，強化学習のステップとシミュレータの時刻は同期しない設定とした．つまり，連続する 2 強化学習ステップ間において，シミュレータ上の時刻は同一である場合や，大きく離れている場合がある．

4.2 状態の表現

環境から取得する状態は (1) 計算ノードの利用状況と (2) 待ちジョブ列からなるものとする．計算ノード利用状況は，N_{nodes} 個の各ノードへのジョブの割り当てと残り利用時間からなる．待ちジョブ列は，現在システムに存在する待ちジョブのうち先頭 N_{jobs} 個のジョブである．それぞれ，以下の通りに状態として表現する．

計算ノードの利用状況 計算ノード数長のベクトル 2 つから構成される．1 つ目の $SV_{node}^{remtime\_req}$ は，ジョブの要求実行時間に基づく，観測時点の各計算ノードの残り使用時間である．2 つ目の $SV_{node}^{remtime\_est}$ は，ジョブの実行時間予測値（学習時は実際値）に基づく，観測時点の各計算ノードの残り使用時間である．

待ちジョブ列 N_{jobs} 長のベクトル 3 つから構成される．それぞれ順に，SV_{job}^{nnode} はジョブの要求ノード数を，$SV_{job}^{time\_req}$ はジョブの要求実行時間を，$SV_{job}^{time\_est}$ はジョブ実行時間予測値（学習時は実際値）を，ジョブ

順に並べたベクトルである．システムに存在するジョブが N_{jobs} 以下の場合，ベクトル後方は 0 埋めする．エージェントにはこれらが結合され渡される．

4.3 行動

強化学習の各ステップでは，最大 N_{jobs} 個のジョブから，次にスケジュールするジョブを 1 つ選択することを行動とする．同一時刻に複数ジョブをスケジュールする場合を考慮し，行動の結果に応じてシミュレータの時刻を進める．

行動空間は $N_{jobs}+1$ とする．これは最大 N_{jobs} 個のジョブから 1 つ選択する行動と，その時刻におけるスケジューリングを終了し次の時刻に進める行動（以降，行動 X）からなる．待ちジョブ数が $N_{wait}(< N_{jobs})$ 個の時，N_{wait} 個からジョブを選択した場合，スケジュールの成否にかかわらず時刻は進めない．ただし，学習時間を短縮することを目的に，以下の場合は次のイベント発生まで時刻を進める．

- ジョブスケジュールに成功し，待ちジョブが 0 になった場合
- ジョブスケジュールに連続して 4 回失敗した場合

$N_{jobs} - N_{wait}$ 個からジョブを選択した場合と，行動 X を選択した場合は次のイベント発生まで時刻を進める．

4.4 報酬

本研究では，利用率の最大化を学習の目的とする．ここでの利用率は，スケジューリング開始時刻から最後のジョブの終了時刻までに実行された全ジョブが使用した資源量を，計算機システムが提供した総資源量で割った値である．この時，各ステップにおける行動の利用率への影響を定量化することが困難であるため，エピソード最終ステップを除き各ステップの報酬は 0 とした．最終ステップの報酬には，それまでに実行完了したジョブの利用率を含める．

エピソードの最終ステップとその報酬は以下の通りに定義する．4.3 章で示す行動では，無制限に学習ステップを消費することが可能である．それを防ぎ学習の効率化のために，エピソードあたりの最大ステップ数 $Max\_Episode\_Steps$ を導入し，最大ステップ数に達した場合は残りジョブがあっても学習を終了することとした．最終ステップは，全てのジョブスケジュールが完了した時と最大ステップに達した時のいずれかになる．最小ステップ数で全てのジョブを実行し，利用率を最大化することが目的なので，最終ステップにおける報酬は，(1) スケジュールしたジョブ数の割合，(2) 利用率，(3) 残りステップ数の割合の和として定義した．

4.5 強化学習アルゴリズム

強化学習アルゴリズムは大きく Value-Based と Policy-Based に分類される．Value-Based は価値関数をモデル化し価値を最大化する行動を選択し，Policy-Based では方策をモデル化し方策に従い確率的に行動を選択する手法である．複数ジョブ実行後の利用率最大化を目的とした場合，特に非混雑時においては，個々のジョブのスケジューリング順序が利用率に与える影響はほとんどないため，確率的な行動を選択できるよう，Policy-Based アルゴリズムの 1 つである PPO [22] を採用した．PPO の学習ループをアルゴリズム 1 に示す．PPO では方策をニューラルネットワークで表現し，そのパラメータ（θ）のバッチ学習を行う．バッチは，N_{envs} 個のエージェントが T ステップ分選択した行動から算出された Advantage（エージェントが選択した行動の価値）の集合である（2-5 行目）．その Advantage を入力として，エポック数を K，ミニバッチサイズを M とした確率的勾配降下法を用いて損失を最小化し，パラメータを更新する（6,7 行目）．以上を一定ステップ繰り返す（1,8 行目）．

Algorithm 1 PPO の学習ループ

1: **for** step = 1, 2, ... **do**
2: **for** actor = 1, 2, ... N_{env} **do**
3: Run policy $\pi_{\theta_{old}}$ in environment for T steps
4: Compute advantage estimates
5: **end for**
6: Optimize surrogate loss with K epochs and $M(\leq N_{envs} \times T)$ minibatch size by minibatch SGD
7: $\theta_{old} = \theta$
8: **end for**

5. 評価設定

ジョブの要求実行時間と実行時間の予測値（学習時は実際値を使用）の双方を考慮して学習した場合（以降，提案手法）と，要求実行時間のみを考慮して学習した場合（以降，従来手法）のスケジューリング結果の比較を行う．具体的には，ワークロード，シミュレーション，学習の設定を揃え，強化学習の環境をそれぞれ以下の通りに設定する．

提案手法 ベクトル $SV_{node}^{remtime\_req}$，$SV_{node}^{remtime\_est}$，SV_{job}^{nnode}，$SV_{job}^{time\_req}$，$SV_{job}^{time\_est}$ を結合

従来手法 ベクトル $SV_{node}^{remtime\_req}$，SV_{job}^{nnode}，$SV_{job}^{time\_req}$ を結合

この設定の下で学習したモデルを用いて，ワークロードをスケジューリングした時の利用率を比較する．

評価には人工的に生成したワークロードを用いる．ワークロードは Lublin モデル [23] を用いて `WL1` と `WL2` の 2 種類生成した．それぞれ 2000 ジョブから構成され，**表 1** に示す特徴を有す．ただし，Lublin モデルではジョブの要求実行時間しか生成しない．そこで，要求実行時間に対する実実行時間の割合（以降，WRA (Walltime Request Accuracy)）を定義し，要求実行時間と掛け合わせた値を実実行時間とした．WRA には以下の 3 種類を採用した．

100 全ジョブの WRA が 100%．全ジョブにて実実行時間と要求実行時間が等しい．

表 1　ワークロード特性

Parameter	WL1	WL2
#Jobs	2000	←
#Nodes	256	←
Ratio of single node jobs	80%	20%
Walltime	Max: 6 Hour Avg: 0.5 Hour	←
Inter Arrival Time	Max: 0.5 Hour Avg: 100 Sec	←

表 2　シミュレーションパラメータ

Parameter	Value
N_{nodes}	256
N_{jobs}	100
Sim_Step	60 Sec

050　全ジョブの WRA が 50%．全ジョブにて実実行時間は要求実行時間の半分．

RND　ジョブごとの WRA はランダムだが，全ジョブの平均 WRA は 50%．

以上より，次に示す合計 6 種類のワークロードを評価に用いた．`WL1_100`，`WL1_050`，`WL1_RND`，`WL2_100`，`WL2_050`，`WL2_RND`．学習には各ワークロードの前半 1000 ジョブを用い，生成したモデルを用いて後半 1000 ジョブをスケジュールした際の利用率を評価した．以降論文中では，それぞれ区別するために，前半には「`a`」，後半には「`b`」を付け，`WL1_100a`，`WL1_100b` のように表記する．

Python 3.6.12 を用いてプログラムを開発し，強化学習フレームワークには Stable Baseline3 v1.0 [24] を用いた．シミュレーションパラメータには**表 2** に示す値を採用した．PPO の実装には，Stable Baseline3 が提供する実装を用いている．PPO の学習パラメータは以降に示す一部を除き，標準の値を採用している．ニューラルネットワークは，3 層の隠れ層を持つ全結合ネットワークとした．各層のサイズは順に [1024, 512, 256] とした．この設定のネットワークを `NN1` と呼ぶ．提案手法の場合は，従来手法に比べ状態のサイズが 2 倍程度大きくなるため，[2048, 512, 256] 設定のネットワーク `NN2` も評価に用いる．学習パラメータには**表 3** に示す値を採用した．複数値を持つパラメータは，ワークロードごとにチューニングを行い，最良の値を選択した．具体的には，構築したモデルを用いて学習に用いたワークロードを 5 回実行し，平均利用率の最も高いモデルを生成したパラメータを選択している．1 度の学習における最大ステップ数は $N_{envs} \times Max\_Episode\_Steps \times 50$ とした．各環境にて，最小 50 エピソード以上学習することを意味する．

実験には ABCI [25] の計算ノード（V）を用いた．シミュレーションには CPU を用い，ニューラルネットワーク計算に GPU を用いた．計算ノード（V）は 4GPU 搭載しているが，1 度の学習で使用する GPU は 1 つとした．

表 3　学習パラメータ

Parameter	Value/Values
N_{envs}	8
T	[512, 1024, 2048, 4096, 8192]
M	[512, 1024, 2048, 4096, 8192]
K	[4, 8, 10]
$Max\_Episode\_Steps$	10000

表 4　シミュレーションによる各ワークロードの利用率

Workload \ Algorithm	SJF	SJF w/ BF	FCFS	FCFS w/ BF
WL1_100a	0.54	0.70	0.54	0.74
WL1_050a	0.44	0.47	0.49	0.51
WL1_RNDa	0.45	0.47	0.44	0.50
WL2_100a	0.74	0.88	0.68	0.92
WL2_050a	0.69	0.79	0.65	0.82
WL2_RNDa	0.67	0.78	0.61	0.80
WL1_100b	0.64	0.78	0.57	0.75
WL1_050b	0.46	0.49	0.55	0.53
WL1_RNDb	0.45	0.55	0.49	0.63
WL2_100b	0.78	0.90	0.69	0.94
WL2_050b	0.77	0.86	0.69	0.90
WL2_RNDb	0.69	0.86	0.62	0.90

表 5　強化学習による各ワークロードの利用率 - 学習時と同じワークロードを使用

Workload \ Method	Baseline	Proposal (NN1)	Proposal (NN2)
WL1_100a	0.67	0.64	0.67
WL1_050a	0.46	0.48	0.48
WL1_RNDa	0.45	0.42	0.46
WL2_100a	0.84	0.83	0.85
WL2_050a	0.76	0.76	0.75
WL2_RNDa	0.76	0.76	0.77

6. 評価結果

古典的なスケジューリングアルゴリズムを用いて各ワークロードをシミュレーション実行した際の利用率を**表 4** に示す．用いたアルゴリズムは `SJF` と，`FCFS`，それらに Backfill [26, 27] を適用した手法の 4 種類である．深層強化学習を用いたスケジューリングの結果として，学習時と同じワークロードを用いて実行した際の利用率を**表 5** に，学習時と異なるワークロードを実行した際の利用率を**表 6** に示す．凡例 `Baseline` が従来手法，`Proposal(NN1)` が提案手法にて NN1 ネットワークを用いた場合，`Proposal(NN2)` が提案手法にて NN2 ネットワークを用いた場合である．表 6 中，下線が引かれた数値は，学習に用いたワークロードと同じ傾向を持つワークロードの結果を意味する．強化学習を用いた結果では，5 回実行し平均値を示している．

6.1 深層強化学習を用いる効果

表 5 で示される結果は，モデルが最も最適化されている

表 6　強化学習による各ワークロードの利用率 - 学習時と異なるワークロードを使用

Test Workload	WL1_100b			WL1_050b			WL1_RNDb		
Method / Train WL	Baseline	Proposal (NN1)	Proposal (NN2)	Baseline	Proposal (NN1)	Proposal (NN2)	Baseline	Proposal (NN1)	Proposal (NN2)
WL1_100a	0.74	0.71	0.72	0.47	0.46	0.47	0.53	0.48	0.52
WL1_050a	0.72	0.70	0.72	0.49	0.49	0.48	0.54	0.55	0.55
WL1_RNDa	0.70	0.60	0.71	0.48	0.51	0.47	0.55	0.50	0.54
WL2_100a	0.68	0.69	0.70	0.44	0.45	0.47	0.50	0.52	0.50
WL2_050a	0.71	0.73	0.71	0.46	0.46	0.46	0.51	0.53	0.54
WL2_RNDa	0.73	0.72	0.73	0.46	0.45	0.46	0.51	0.52	0.54

Test Workload	WL2_100b			WL2_050b			WL2_RNDb		
Method / Train WL	Baseline	Proposal (NN1)	Proposal (NN2)	Baseline	Proposal (NN1)	Proposal (NN2)	Baseline	Proposal (NN1)	Proposal (NN2)
WL1_100a	0.86	0.85	0.85	0.83	0.82	0.83	0.83	0.81	0.81
WL1_050a	0.78	0.77	0.76	0.77	0.75	0.74	0.74	0.74	0.74
WL1_RNDa	0.78	0.70	0.86	0.69	0.64	0.82	0.71	0.63	0.83
WL2_100a	0.87	0.87	0.87	0.81	0.82	0.83	0.82	0.83	0.84
WL2_050a	0.87	0.86	0.87	0.83	0.82	0.83	0.84	0.83	0.83
WL2_RNDa	0.87	0.86	0.86	0.83	0.81	0.82	0.84	0.82	0.83

状況における結果である．この結果と表 4 上半分のシミュレーション結果を比較すると，強化学習を用いることで，多くの場合（14/18）にて SJF と FCFS より優れた結果を示すが，全ての場合にて Backfill を適用した手法ほどの結果は得られていない．また，表 6 と表 4 下半分を比較すると，学習時と異なるワークロードを用いた結果について，全ての結果が SJF と FCFS に劣る WL1_050b ワークロードの結果を除き，以下が観測される．

- 学習時と同じ傾向のワークロードを用いた場合（WL1_100a で学習し WL1_100b を実行，など）は SJF と FCFS より優れるが，Backfill を適用した場合より劣る結果であった．
- 学習時と異なる傾向のワークロードを用いた場合（WL1_100a で学習し WL1_050b を実行，など）では，SJF と FCFS 以下となる結果（10/75）もあった．

2 点目について，SJF と FCFS 以下となる件数は，既存手法は 1 件に対して，提案手法は 9 件あった．提案手法では，実実行時間を考慮した学習を行なっているため，既存手法より学習に用いたワークロードに最適化されたことにより，汎化性能に劣る可能性が考えられる．いずれにせよ，モデルにはまだチューニングの余地があると考える．

表 6 から観測できる特徴として，学習時と同じ傾向のワークロードを用いた場合が利用率が高い傾向にあることがわかる．グラフを列方向に読むことで確認することができ，そのような結果は 18 件中 7 件ある．これにより，システムで実行されるワークロードの傾向が大きく変わらない限り，事前に取得したワークロードで学習したモデルを用いて，以降のジョブスケジュールを行うことには効果があると言える．

6.2　実行時間の予測値を用いる効果

ジョブ実行時間の予測値を用いた場合の効果を検証するために，学習時と同じ傾向のワークロードを実行した場合の結果に着目する．既存手法と提案手法を比較すると，提案手法はどのワークロードにおいても利用率の向上には貢献していないことがわかる．理由は調査中であり，強化学習の環境を構成する各特徴量の行動への貢献度の調査を進めている．次に提案手法にてネットワークサイズを大きくした場合の効果について確認する．Proposal（NN1）と Proposal（NN2）を比較すると，多くの場合（30/36）にて利用率の改善が見られる．実実行時間を環境に含めることの情報量増加に伴い，ネットワークサイズを増加させることの効果が確認できる．

7.　まとめ

本研究では，深層強化学習を用いたジョブスケジューリングにおいて，ジョブ実行時間の予測値を用いる手法の提案を行い，予測値を考慮した学習の効果の実験的な検証を行なった．今回の評価設定では，予測値を考慮して利用率を最大化するよう学習しても，利用率の向上には貢献しない結果となった．一方で，学習時のワークロードと傾向が類似している場合に利用率が高くなることと，実行時間予測値を学習に用いる際に適切なネットワークサイズを設定することで利用率が改善することが確認できた．

現在，強化学習の行動の選択に対する，実行時間の予測値の影響を定量的に評価すべく，機械学習モデル説明技術を用いた分析を進めている．更なるパラメータチューニングを行いより高い利用率を達成することと，その分析結果を示すことを今後の課題とする．

謝辞 本研究は，JSPS 科研費 (JP19H04121) の助成を受けたものである．

参考文献

[1] Kay, J. and Lauder, P.: A Fair Share Scheduler, Commun. ACM, Vol. 31, No. 1, pp. 44–55 (1988).
[2] 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).
[3] 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).
[4] 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).
[5] Tang, W., Desai, N., Buettner, D. and Lan, Z.: Job Scheduling with Adjusted Runtime Estimates on Production Supercomputers, Journal of Parallel and Distributed Computing, Vol. 73, No. 7, pp. 926–938 (2013).
[6] Wallace, S., Yang, X., Vishwanath, V., Allcock, W. E., Coghlan, S., Papka, M. E. and Lan, Z.: A Data Driven Scheduling Approach for Power Management on HPC Systems, Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Piscataway, NJ, USA, pp. 1–11 (2016).
[7] Arulkumaran, K., Deisenroth, M. P., Brundage, M. and Bharath, A. A.: Deep Reinforcement Learning: A Brief Survey, IEEE Signal Processing Magazine, Vol. 34, No. 6, pp. 26–38 (2017).
[8] Mao, H., Alizadeh, M., Menache, I. and Kandula, S.: Resource Management with Deep Reinforcement Learning, Proceedings of the 15th ACM Workshop on Hot Topics in Networks, HotNets '16, pp. 50–56 (2016).
[9] Zhang, D., Dai, D., He, Y., Bao, F. S. and Xie, B.: RLScheduler: An Automated HPC Batch Job Scheduler Using Reinforcement Learning, SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–15 (2020).
[10] Bao, Y., Peng, Y. and Wu, C.: Deep Learning-based Job Placement in Distributed Machine Learning Clusters, IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, pp. 505–513 (2019).
[11] Yi, D., Zhou, X., Wen, Y. and Tan, R.: Toward Efficient Compute-Intensive Job Allocation for Green Data Centers: A Deep Reinforcement Learning Approach, 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), pp. 634–644 (2019).
[12] Li, F. and Hu, B.: DeepJS: Job Scheduling Based on Deep Reinforcement Learning in Cloud Data Center, Proceedings of the 2019 4th International Conference on Big Data and Computing, pp. 48–53 (2019).
[13] Mao, H., Schwarzkopf, M., Venkatakrishnan, S. B., Meng, Z. and Alizadeh, M.: Learning Scheduling Algorithms for Data Processing Clusters, Proceedings of the ACM Special Interest Group on Data Communication, pp. 270–288 (2019).
[14] Hu, Z., Tu, J. and Li, B.: Spear: Optimized Dependency-Aware Task Scheduling with Deep Reinforcement Learning, 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), pp. 2037–2046 (2019).
[15] Mirhoseini, A., Pham, H., Le, Q. V., Steiner, B., Larsen, R., Zhou, Y., Kumar, N., Norouzi, M., Bengio, S. and Dean, J.: Device Placement Optimization with Reinforcement Learning (2017).
[16] Wang, L., Weng, Q., Wang, W., Chen, C. and Li, B.: Metis: Learning to Schedule Long-Running Applications in Shared Container Clusters at Scale, SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–17 (2020).
[17] Li, Y., Chang, K., Bel, O., Miller, E. L. and Long, D. D. E.: CAPES: Unsupervised Storage Performance Tuning Using Neural Network-based Deep Reinforcement Learning, Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–14 (2017).
[18] Chen, W., Xu, Y. and Wu, X.: Deep Reinforcement Learning for Multi-Resource Multi-Machine Job Scheduling (2017).
[19] Ye, Y., Ren, X., Wang, J., Xu, L., Guo, W., Huang, W. and Tian, W.: A New Approach for Resource Scheduling with Deep Reinforcement Learning, CoRR (2018).
[20] Matsunaga, A. and Fortes, J.: On the Use of Machine Learning to Predict the Time and Resources Consumed by Applications, 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pp. 495–504 (2010).
[21] McKenna, R., Herbein, S., Moody, A., Gamblin, T. and Taufer, M.: Machine Learning Predictions of Runtime and IO Traffic on High-End Clusters, 2016 IEEE International Conference on Cluster Computing (CLUSTER), pp. 255–258 (2016).
[22] Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O.: Proximal Policy Optimization Algorithms (2017).
[23] Lublin, U. and Feitelson, D. G.: The workload on parallel supercomputers: modeling the characteristics of rigid jobs, Journal of Parallel and Distributed Computing, Vol. 63, No. 11, pp. 1105–1122 (2003).
[24] Raffin, A., Hill, A., Ernestus, M., Gleave, A., Kanervisto, A. and Dormann, N.: Stable Baselines3, `https://github.com/DLR-RM/stable-baselines3` (2019).
[25] Takizawa, S., Tanimura, Y., Nakada, H., Takano, R. and Ogawa, H.: ABCI 2.0: Advances in Open AI Computing Infrastructure at AIST, IPSJ SIG Technical Reports HPC-180 (2021).
[26] Lifka, D. A.: The ANL/IBM SP Scheduling System, Job Scheduling Strategies for Parallel Processing (Feitelson, D. G. and Rudolph, L., eds.), Berlin, Heidelberg, Springer Berlin Heidelberg, pp. 295–303 (1995).
[27] 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).