情報処理学会論文誌　コンピューティングシステム　Vol. 4　No. 4　203–213 (Oct. 2011)

マイグレーションによる複合型計算システム向けジョブスケジューリング

小山　賢太郎[†1,*1]　佐藤　功人[†1]　小松　一彦[†2]
村田　善智[†2,†3]　滝沢　寛之[†1,†3]　小林　広明[†2,†3]

消費電力が厳しく制約された条件下で演算性能を大幅に向上させることができるシステムアーキテクチャとして，汎用プロセッサに加えてアクセラレータを混載する複合型計算システムが注目されている．本論文では，大規模複合型計算システムにおけるターンアラウンドタイムの短縮を目的とし，マイグレーションとプリエンプティブバックフィルに基づくスケジューリング手法を提案する．また，ジョブ投入時にマイグレーションのコストを予測するため，その予測モデルも提案する．予測モデルの精度を評価した結果，ほぼすべてのアプリケーションにおいて，マイグレーションコストの最悪値をジョブの最大メモリ使用量から高精度で予測できることが明らかになった．また，提案スケジューリング手法はマイグレーションとプリエンプティブバックフィルの両方の長所を利用できるため，それらのいずれかが有効に機能する状況において，ターンアラウンドタイムを短縮可能であることが示された．

Job Scheduling with Migration for Heterogeneous Computing Systems

Kentaro Koyama,[†1,*1] Katsuto Sato,[†1]
Kazuhiko Komatsu,[†2] Yoshitomo Murata,[†2,†3]
Hiroyuki Takizawa[†1,†3] and Hiroaki Kobayashi[†2,†3]

A heterogeneous computing system of general-purpose processors and accelerators is a promising approach to improve the system performance under severe power consumption limitation. This paper proposes a job scheduling method that uses job migration and preemptive backfilling to reduce the turn around time of job execution in a large-scale heterogeneous computing system. A prediction model is also proposed to predict the migration cost of a job when the job is submitted. The evaluation results indicate that the prediction model can accurately estimate the worst-case migration costs of most applications from their maximum memory usage. It is also demonstrated that the proposed mechanism can reduce the turn around time of a job in the situations where either job migration or backfilling works well because it has the advantages of both of the two scheduling policies.

1. はじめに

近年，従来の汎用プロセッサ（Central Processing Unit，CPU）に加えて，描画処理プロセッサ（Graphics Processing Unit，GPU）に代表される**アクセラレータ**を搭載する大規模計算システムが普及しつつある．そのような異種複数のプロセッサを混載する**複合型計算システム**は，消費電力を抑えながら高い演算性能を実現する有望なシステムとして注目されている．

大規模な計算システムで複数のジョブを効率的に実行するためには，ジョブの実行順序と資源割当てを管理する**ジョブスケジューリング**がきわめて重要である．本論文では，ジョブが投入されてからその実行が完了するまでの時間である**ターンアラウンドタイム**を短縮することを考え，複合型計算システム向けのジョブスケジューリング手法を検討する．

各ジョブはそれぞれ異なる種類と量の計算資源を要求する一方で，システム中の各計算資源の割合は設置時から不変である．したがって，たとえばCPUとGPUを混載するシステムに，CPUのみを数多く要求するジョブが多数投入された場合，CPUが不足する一方でGPUが**遊休状態**になる．そのような遊休資源はジョブの実行に寄与しないばかりか，依然として電力を消費し続けるためにシステム全体としての電力あたりの性能を低下させる．よって，複合型計算システム全体としての利用効率を高めるために，すべての種類のプロセッサの利用率を高める必要がある．

同じアーキテクチャのプロセッサのみで構成される計算システムでターンアラウンドタイムを短縮する場合，ジョブの実行時間よりも待機時間の短縮が主な検討課題であった．一方，

†1 東北大学大学院情報科学研究科
Graduate School of Information Sciences, Tohoku University
†2 東北大学サイバーサイエンスセンター
Cyberscience Center, Tohoku University
†3 科学技術振興機構戦略的創造研究推進事業
Japan Science and Technology Agency, Core Research for Evolutional Science and Technology
*1 現在，東日本旅客鉄道株式会社
Presently with East Japan Railway Company

複合型計算システムでは各計算資源の得手不得手が異なるため，ジョブに割り当てられる計算資源の種類によってそのジョブの実行時間が大きく変化する．このため，待機時間に加えて，それぞれの計算資源上で動作させた場合のジョブの実行時間も考慮する必要がある．

本論文では，複合型計算システムにおけるターンアラウンドタイムの短縮を目的とし，ジョブ実行中に使用する計算資源を切り替える**マイグレーション**と，ジョブ実行順序の変更（Preemptive Backfilling．以下，**プリエンプティブバックフィル**とする）に基づくスケジューリング手法を提案する．計算資源を切り替える際には中断と再開のためのオーバヘッド（以下，**マイグレーションコスト**）が生じる．このため，ジョブ投入時点でマイグレーション時のコストを予測し，その予測結果に基づいてスケジューリングを行う．

本論文の構成は以下のとおりである．2 章では，関連研究を述べる．3 章では，マイグレーションを用いる複合型計算システムのためのスケジューリング手法を提案し，マイグレーションコストの予測方法についても議論する．4 章では，予測精度およびスケジューリングによって得られる性能向上を評価した結果を考察する．5 章で本論文の結論と今後の課題を述べる．

2. 関 連 研 究

2.1 OpenCL とマイグレーション

複合型計算システムには，異種複数の計算資源が混載されている．それらの計算資源を統一的に扱うためのプログラミングインタフェースとして，OpenCL が提案されている[1]．OpenCL では，制御用の汎用プロセッサを**ホスト**，演算用のアクセラレータを**デバイス**と呼ぶ．ホストとデバイスではコードが分かれており，デバイスコードはジョブの実行時にコンパイルされる．OpenCL では，デバイス上で実行される処理を**カーネル**として記述し，複数のスレッドで同じカーネルを並列実行する Single-Program Multiple-Data（SPMD）型のプログラミングモデルを採用している．現在，様々な計算資源向けに OpenCL の実行時環境が提供されており，各カーネルを実行するデバイスをプログラム実行時に選択することが可能である．

OpenCL で記述されたプログラムの実行を一時中断し，再開する機能を実現するために，我々は CheCL を提案している[2]．CheCL は，OpenCL プログラムのチェックポイント取得と，そのチェックポイントからのリスタートを実現する．このチェックポイント取得時に利用していた計算資源と，リスタート時の計算資源を変更することで，CheCL は OpenCL プログラムのマイグレーションを実現することが可能である．

本論文では，プログラムが OpenCL で記述されていると仮定し，CheCL を利用してジョブをマイグレーションできることを想定して，複合型計算システムのためのジョブスケジューリング手法を考える．

2.2 ジョブスケジューリング

Maheswaran らは，複合型計算システムで First-Come，First-Serve（FCFS）でジョブを実行する環境を対象として，Minimum Completion Time（MCT）というスケジューリング手法を提案している[3]．MCT では，計算資源 i に割り当てた場合のジョブ j の完了時間（Completion Time，C_i^j）を，式 (1) から算出する．

$$C_i^j = W_i + T_i^j \tag{1}$$

ここで W_i は計算資源 i にすでに割り当てられているすべてのジョブが完了するまでに要する時間，T_i^j はジョブ j を計算資源 i で実行した場合に要する時間である．MCT では，システム中のすべての計算資源においてジョブ j が投入された時点での C_i^j を算出し，C_i^j が最も小さくなる計算資源にジョブ j を割り当てる．MCT は各計算資源の完了時間に基づいてスケジューリングを行うため，実行時間 T_i^j のみを用いて，最も T_i^j が小さくなる計算資源へジョブを割り当てる Minimum Execution Time（MET）とは異なり，高速な計算資源のみにジョブが集中することを避け，負荷を分散することが可能である．

Feitelson らは，バックフィル（backfilling）というスケジューリングポリシを提案し，その有効性を示している[4]．バックフィルポリシは，先行ジョブの実行時間を遅らせない範囲で後続ジョブを先行実行するポリシである．必要計算資源数の違いから，先行ジョブの実行には計算資源数が足りないが後続ジョブの実行は可能な場合，後続ジョブを先行実行することで遊休計算資源の有効利用が可能である．したがって，このポリシを用いることで，各計算資源の利用率を高めることが可能である．しかし，後続ジョブの実行により先行ジョブの実行開始を遅らせてしまう恐れがある場合には，バックフィルポリシは適用されない．したがって，短期間だけ遊休状態になる計算資源が多い場合には，バックフィルによる利用率向上は期待できない．

これに対して Snell らは，バックフィルポリシをさらに拡張したプリエンプティブバックフィルを提案している[5]．Snell らの手法では，後続ジョブの実行が先行ジョブの実行開始時間までに終わらない場合でも，後続ジョブを先に実行する．後続ジョブの実行中に先行ジョブの実行開始時刻に達した場合には，後続ジョブの実行を一時中断し，先行ジョブの実行を開始する．先行ジョブの実行完了によって計算資源が遊休状態に戻ったときに，後続ジョブの実行が再開される．この手法は，バックフィルの柔軟性を向上させ，計算資源をよ

り効率的に利用可能となる．しかし，ジョブの中断と再開を同じ計算資源上で行うことを前提としているため，異なる計算資源にマイグレーションすることは考慮していない．

Vadhiyar らは，Globus Monitoring and Discovery Service（MDS）と Network Weather Service（NWS）を用いて，グリッド環境における負荷の監視とジョブのマイグレーションを統一的に扱うフレームワークを提案している[6]．Vadhiyar らの提案フレームワークは，ジョブ終了によって高性能な計算資源が遊休状態となるときに，別の計算資源で実行されていたジョブをその高性能な遊休計算資源にマイグレーションすることで，システム全体の実効性能を向上させる．しかし，複数のジョブをスケジューリングする条件下では計算能力の高い計算資源にジョブが集中し，負荷の不均衡が起こることが知られている．

本論文では，高性能計算資源へのジョブの集中を避けるために MCT を用い，ジョブのターンアラウンドタイムの短縮が期待できる場合のみ，より高い計算能力を持つ計算資源へのマイグレーションやプリエンプティブバックフィルを実行するスケジューリング手法を提案する．

3. 複合型計算システムのためのジョブスケジューリング手法

本章では，ホスト（汎用プロセッサ）と 1 種類以上のデバイス（アクセラレータ）を混載する複合型計算システムのためのジョブスケジューリング手法を提案する．本手法では，複合型計算システムにジョブが逐次投入され，投入時点でスケジューリングを行うオンラインスケジューリングを仮定する．各計算資源上でのジョブの実行時間は，スケジューリングの時点で十分な精度で予測できるものとする．また，OpenCL におけるデバイスコードのコンパイル時間やジョブの最大メモリ使用量などの情報も，ジョブ投入時にあらかじめ得られるものとする．

複合型計算システムに投入されるジョブは，それぞれ異なる種類と数の計算資源を必要とする．そのため，ジョブを投入された順番でスケジューリングする場合，先行して投入されたジョブが必要な計算資源を確保できない状況では，確保できるまでそのジョブの実行開始を遅らせなければならない．その遅延によって，後続のすべてのジョブの実行開始時間も遅らせられる．**図 1** (a) の例では，ジョブ A〜E がすでに計算資源に割り当てられている状態で，ジョブ F が新たに投入されたという状況を示している．横軸は時間経過を示し，縦軸は複合型計算システムに存在する計算資源数を表している．この場合，ジョブ F が要求する計算資源数分の遊休計算資源（図 1 における，破線部）が存在しているにもかかわらず，どの遊休領域にも収めることができない．そのため，ジョブ F はジョブ D の終了直後にスケジュールされる．このような状況では，プリエンプティブバックフィルやマイグレーションを単独で用いても，これ以上の効率化は困難である．

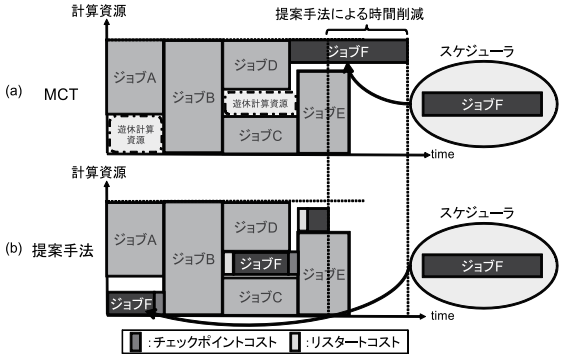


図 1　提案スケジューリング手法の概要
Fig. 1　Overview of the proposed scheduling method.

提案手法では，実行時間予測によりジョブ F のターンアラウンドタイムを短縮可能であると判断した場合にバックフィルポリシを適用し，先行ジョブの実行を阻害しない範囲でジョブ F の実行をスケジュールする．図 1 (b) のように，異なる遊休計算資源を利用することで必要な計算資源数を確保できる場合には，計算資源間のマイグレーションも組み合わせて用いる．このように提案手法では，マイグレーションを用い，断片化している遊休計算資源を組み合わせてプリエンプティブバックフィルポリシを適用する．このようなプリエンプティブバックフィルにより，従来では利用できなかった遊休計算資源を利用できるようになり，計算資源の利用効率の向上と，それによるターンアラウンドタイムの短縮を達成できる．

ただし，プリエンプティブバックフィルでは，マイグレーションを行った回数分の中断コストと再開コストが，マイグレーションコストとして要求される．マイグレーションコストを無視してスケジューリングを行った場合には，先行ジョブの実行を阻害したり，想定どおりにマイグレーションを行うことができなくなったりする．そのため，スケジューリング時にマイグレーションコストを正確に予測する必要がある．

3.1 節では，マイグレーションコストを予測してプリエンプティブバックフィルを行うスケジューリング手法を提案する．3.2 節では，マイグレーションコストのモデルを構築し，十分な精度でマイグレーションコストを予測する方法を考察する．

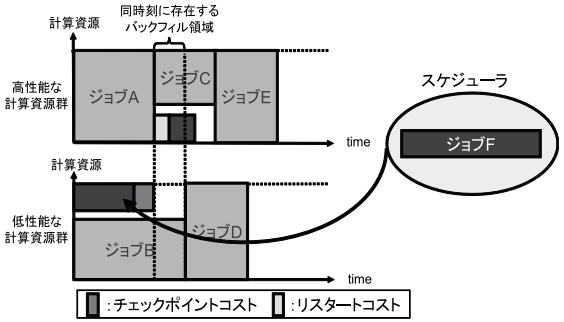


図 2　バックフィル対象領域の選択
Fig. 2　Selection of backfill regions.

3.1　マイグレーションとプリエンプティブバックフィルに基づくジョブスケジューリング手法

本提案手法は，ジョブのスケジューリングには原則として MCT を用いる．ただし，後述のプリエンプティブバックフィルの適用条件を確認し，適用することでターンアラウンドタイムの短縮が可能であればプリエンプティブバックフィルを行う．

複合型計算システムにおけるプリエンプティブバックフィルでは，同時刻に異種複数の計算資源が遊休状態にある可能性があり，それぞれの計算資源に期待される性能が異なる可能性がある．この場合，性能が高い遊休計算資源を優先的に利用した方が，ジョブ実行のターンアラウンドタイムを短縮可能である．本提案手法では，**図 2** に示すようにバックフィルの対象となる**バックフィル領域**が同時刻に複数ある場合，性能が高い領域を優先的に利用する．

以下，スケジューリングの手順を述べる．ジョブ $j_0, j_1, j_2, j_3, \ldots$ が，この順番で投入される状況を想定する．また，各ジョブの投入時点でジョブの最大メモリ使用量が既知であり，後述の予測モデルによってマイグレーションコストの最悪値が予測可能であることを前提とする．スケジューリング手順の説明で用いる変数を**表 1** に示す．手順をフローチャートにまとめたものを，**図 3** に示す．

(1)　未スケジューリングのジョブ j_i を選択し，手順 (2) へ進む．また，ジョブ j_i の総進捗割合 $P_{i,\mathrm{sum}}$ を 0 で初期化する．未スケジューリングのジョブがない場合は終了する．

(2)　未選択の遊休計算資源領域が存在する場合には，その中から最も開始時間の早い領域 $I_r(t_s, t_e)$ を選択する．$t_s \le t'_s \le t_e$ であり，かつ計算資源 r より高性能な別の遊休計算資源 r' の領域 $I_{r'}(t'_s, t'_e)$ がある場合には，$t_e = t'_s$ とする．未選択の遊休計算資源領域が存在しない場合には手順 (7) へ進む．

(3)　$R_i > N_r$ の場合，その領域ではジョブ j_i を実行できないため，手順 (2) へ戻る．

(4)　$t_e - t_s \le C_i(r) + R_i(r)$ の場合，その領域内ではジョブを実行する時間を確保できないため，手順 (2) へ戻る．

表 1　プリエンプティブバックフィルで考慮する変数
Table 1　Variables in preemptive backfilling.

ジョブ j_i の実行に必要な計算資源数	R_i
計算資源の集合	r
計算資源が遊休状態となる時間	t_s
計算資源の遊休状態が終了する時間	t_e
遊休計算資源領域	$I_r(t_s, t_e)$
領域 $I_r(t_s, t_e)$ で使用可能な計算資源数	N_r
領域 $I_r(t_s, t_e)$ でジョブ j_i が進捗する割合	$P_i(r, t)$
ジョブ j_i の総進捗割合	$P_{i,\mathrm{sum}}$
ジョブ j_i のチェックポイントコスト	$C_i(r)$
ジョブ j_i のリスタートコスト	$R_i(r)$
計算資源 r でジョブ j_i を実行した場合の実行時間	$E_i(r)$

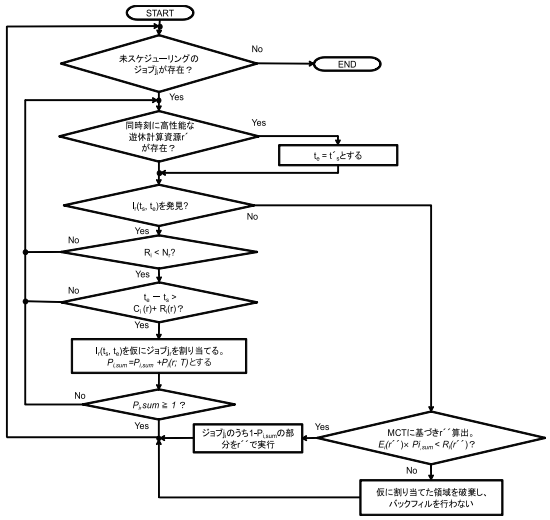


図 3　プリエンプティブバックフィルの適用フローチャート
Fig. 3　Flowchart of preemptive backfilling.

(5)　領域 $I_r(t_s, t_e)$ をジョブ j_i に仮に割り当てる．マイグレーションコストを考えると，計算資源をジョブの実行に利用可能な時間 T は $T = t_e - t_s - (C_i(r) + R_i(r))$ である．したがって，その領域を使うことでジョブは $P_i(r, T)$ だけ進捗することが分かる．このため，$P_{i,\mathrm{sum}} = P_{i,\mathrm{sum}} + P_i(r, T)$ とし，これまでに割り当てられた全領域を使うことで期待されるジョブの進捗状況を更新する．

(6)　$P_{i,\mathrm{sum}} \geq 1$ の場合，プリエンプティブバックフィルによりジョブの実行は完了する．そのため，手順 (5) で仮に割り当てられていた領域をジョブ j_i に割り当てることを確定し，手順 (1) へ戻る．$P_{i,\mathrm{sum}} < 1$ の場合，手順 (2) へ戻る．

(7)　MCT に基づいて，計算資源 r'' を決定する．$E_i(r'') \times P_{i,\mathrm{sum}} < R_i(r'')$ の場合，ジョブ全体を計算資源 r'' 上で実行する方がターンアラウンドタイムが短いため，これまで仮に割り当てられていた領域をすべて破棄し，プリエンプティブバックフィルは行われない．それ以外の場合には，プリエンプティブバックフィルによるターンアラウンドタイム短縮が期待できるため，ジョブ j_i のうち $1 - P_{i,\mathrm{sum}}$ の部分だけを計算資源 r'' で実行する．

3.2　マイグレーションコストのモデル化と予測手法

本節では，マイグレーションコストの予測手法を提案する．まず，マイグレーションコストが影響を受ける要因を明らかにするために，NVIDIA GPU Computing SDK 3.0，Parboil benchmark suite，SHOC benchmark suite に含まれるベンチマークプログラムを用いて，予備実験を行う．

本予備実験では，CheCL を用いて各ベンチマークのチェックポイント・リスタート時間を計測し，これらを合わせたコストの内訳を調査する．本予備実験で用いたシステムの構成を**表 2** に示す．

システムに搭載された CPU と GPU をそれぞれホストとデバイスとして用いる場合，チェックポイント時点でのメモリ使用量 M とチェックポイントコスト C およびリスタートコスト R とのそれぞれの関係を**図 4** と**図 5** に示す．これらの結果から，チェックポイントコスト C は，メモリ使用量 M と強い線形関係にあることが分かる．これは，コストの大半がチェックポイントファイルをハードディスクに書き込むために費やされており，書き込まれるファイルのサイズがメモリ使用量に比例する傾向にあるためである．このため，メモリ使用量を M，システム固有のチェックポイントパラメータを α_C，β_C と表記することで，式 (2) に示す線形予測モデルをたてることができる．

$$C = \alpha_C M + \beta_C \tag{2}$$

表 2　予備実験の複合型計算システム構成
Table 2　System configuration for preliminary evaluation.

CPU	Intel Core i7 920
GPU	NVIDIA Tesla C1060
File Write	72.5 MB/sec over NFS
File Read	21.2 MB/sec over NFS
PCI-express	Host to Device 5.35 GB/s
Bandwidth	Device to Host 4.87 GB/s
OS	Linux 2.6.18 CentOS 5.5
Video Driver Version	NVIDIA Video Driver 256.40

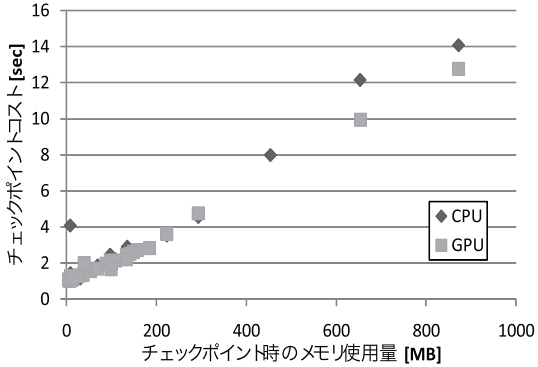


図 4　チェックポイントコストとメモリ使用量の関係
Fig. 4　Relationship between checkpoint cost and memory usage.

同様に，リスタートコスト R もメモリ使用量 M との間に線形性を見いだすことができる．リスタートコストの大半も，チェックポイントファイルをハードディスクから読み込む時間に費やされるため，図 5 に示す線形性が現れる．しかし，一部のベンチマークでは例外的にリスタートコストが大きくなることも，図 5 から分かる．これは，リスタート時にデバイスコードの再コンパイルを行うためである．一般的に，SHOC ベンチマークに含まれる S3D のように，多数のカーネルを含むベンチマークでは，リスタートコストの中で再コンパイル時間が無視できないほど大きい値となる．

プログラムで利用されるデバイスコードが実行中に動的に変化することはないため，コンパイルに要する時間を一定と仮定し，各プログラムに固有の定数として扱うことが可能で

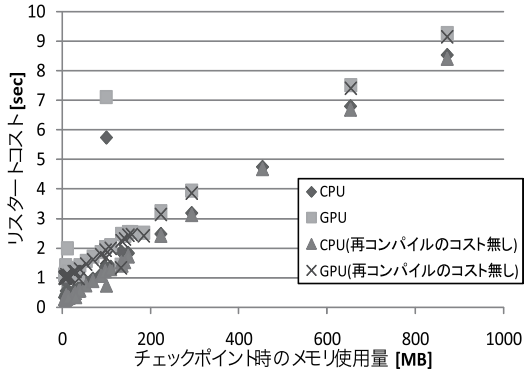


図 5　リスタートコストとメモリ使用量の関係
Fig. 5　Relationship between restart cost and memory usage.

ある．図 5 に示すように，リスタートコストからカーネルのコンパイル時間を除いたコスト R' とメモリ使用量 M の間には，例外なく強い線形性がある．この結果より，カーネルのコンパイル時間を K，システム固有のチェックポイントパラメータを α_R，β_R と表記する場合，式 (3) に示す線形予測モデルが得られる．

$$R = R' + K = \alpha_R M + \beta_R + K \tag{3}$$

式 (2) および式 (3) を用いてコストを予測するためには，チェックポイント取得時のメモリ使用量を正確に把握する必要がある．しかし，実行前であるジョブの投入時に，実行中に動的に変化するメモリ使用量を把握することは困難である．このため，各ジョブの最大メモリ使用量のみを既知と仮定し，その仮定の下でスケジューリング手法を検討する．

マイグレーションコストを過小に見積もることは，プリエンプティブバックフィルを適用した際に先行ジョブの開始時間を遅らせる原因となるため，極力避けるべきである．一方で，マイグレーションコストを過大に見積もることによってバックフィルの効果を減ずる恐れがあるが，この場合にもバックフィルされたジョブのターンアラウンドタイムを依然として短縮できる．このため本論文では，ジョブ投入時点で既知となるジョブの最大メモリ使用量 $M_{\max}$ を用いて，式 (4) および式 (5) からマイグレーションコストの最悪値を予測する．

$$C_{\max} = \alpha_C M_{\max} + \beta_C \tag{4}$$

$$R_{\max} = \alpha_R M_{\max} + K + \beta_R \tag{5}$$

なお，システム固有の係数（α_C，β_C，α_R，β_R）はベンチマークやメモリ使用量によらずに一定であると仮定できることが，予備実験結果から分かっている．このため本論文では，スケジューラがいくつかのベンチマークを実行し，その結果から最小二乗法に基づく線形近似によってシステム固有の係数を事前に求めてあるものと仮定する．

次に，提案手法で仮定されている以下の 4 点についての実現性の観点から考察を行う．

(a)　投入されるジョブが OpenCL で記述され，CheCL を利用してマイグレーション可能
(b)　ジョブの実行時間が投入時点で予測可能
(c)　ジョブのコンパイル時間とジョブの最大メモリ使用量が既知
(d)　マイグレーションコストで用いるシステム固有値が事前に算出済

本研究では，大規模複合型計算システムが対象として考えられている．現在，OpenCL は異種複数の計算資源を統一的に利用するための標準規格として整備されつつあり，将来の大規模複合型計算システムの有効利用を検討する場合，仮定 (a) は妥当であるといえる．

一般的に計算機センタで実行される大規模アプリケーションは，同一のアプリケーションが条件を変えて繰り返し実行される場合が多い．同一のアプリケーションの実行では，アプリケーションの実行時間，コンパイルに要する時間，最大メモリ使用量などが同様の傾向となることが期待できる．このため，アプリケーションが過去に実行されている場合には，過去の実行履歴からそれらの情報を予測できるため，(b) および (c) を仮定することは可能である．

システムの固有値はシステムが持つファイル I/O 性能に依存する．ファイル I/O 性能は各システムに対して一意に決定し，それらの値は事前にベンチマークを用いることにより算出可能である．このため，仮定 (d) も他の仮定と同様に妥当である．

以上のように，将来の計算センタにおける大規模複合型計算システムでの運用を考えると，それぞれの仮定は妥当であり，提案手法の効果を十分に発揮できると考えられる．

4. 性 能 評 価

4.1　マイグレーションコスト予測の評価

本節では，マイグレーションコストの予測モデルの予測精度を評価する．3.2 節で構築したチェックポイントコストモデルとリスタートコストモデルからマイグレーション全体にかかるコストを算出し，実測されたマイグレーションコストと比較する．評価実験には，表 2 と同一のシステムを用いる．システムの固有値は，NVIDIA GPU Computing SDK 3.0, Parboil benchmark suite，SHOC benchmark suite に含まれるベンチマークプログラムを用いて事前に求めており，それぞれの値を**表 3** に示す．

表 3　評価で用いたシステムの固有値
Table 3　System parameters used in the evaluation.

	α_C (sec/MB)	β_C (sec)	α_R (sec/MB)	β_R (sec)
CPU	1.5×10^{-2}	9.2×10^{-1}	9.7×10^{-3}	1.8×10^{-1}
GPU	1.4×10^{-2}	8.5×10^{-1}	9.6×10^{-3}	8.9×10^{-1}

マイグレーションコストの実測値と予測値とを**図 6** と**図 7** に示す．縦軸はマイグレーションコストの実測値と予測値，横軸はベンチマークを示す．予測値には，2 種類の場合を併記する．一方は最大メモリ使用量から算出される予測値であり，他方は実際にチェックポイントを取得する際のメモリ使用量から算出される予測値である．図 6 は，CPU でデバイスコードを実行中のジョブを GPU でデバイスコードを実行するようにマイグレーションする場合のコストである．一方，図 7 は，GPU でデバイスコードを実行中のジョブを CPU でデバイスコードを実行するようにマイグレーションする場合のコストである．

最大メモリ使用量は，つねにチェックポイント取得時のメモリ使用量以上である．このため，最大メモリ使用量に基づく予測値はチェックポイント取得時のメモリ使用量を用いる予測値よりも大きくなることが，図 6 および図 7 から分かる．この最大メモリ使用量から予測されたマイグレーションコストは，一部の例外を除き，実測値よりも大きい．したがって，最大メモリ使用量からマイグレーションコストの最悪値を予測することが可能であり，その最悪値を用いることで提案するスケジューリングを実現可能である．

ここで，マイグレーションコストの実測値を M_r，予測値を M_p とし，$\frac{M_p - M_r}{M_r}$ をエラー率と定義する．このとき，最大メモリ使用量に基づくマイグレーションコストのエラー率は，全体の 33%のベンチマークで 0.5 以下となり，全体の 95%のベンチマークで 1.0 以下となった．したがって，ジョブ投入時にジョブの最大メモリ使用量が既知であると仮定することにより，ほぼすべてのベンチマークにおいてエラー率 1.0 以下の精度でマイグレーションコストを予測できていることが分かる．エラー率 1.0 の場合，実測値と予測値との間には 2 倍の差があることを示している．チェックポイント取得時のメモリ使用量に基づいた予測では，1 つの例外を除くすべてのベンチマークにおいて 0.33 以内のエラー率で予測できていることから，予測誤差がチェックポイント取得時のメモリ使用量と最大メモリ使用量の差から生じていることは明らかである．この誤差がジョブスケジューリングに与える影響については，4.2 節で議論する．

Parboil benchmark suite の cp では，例外的にエラー率が大きくなり，実測値が最大メモリ使用量に基づく予測値よりも大きくなっている．このプログラムは，メモリ使用量が極

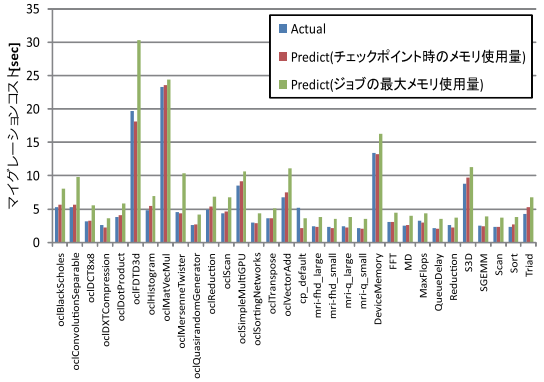


図 6　CPU から GPU へのマイグレーションのコストの実測値と予測値
Fig. 6　Predicated and actual values of CPU-to-GPU migration costs.

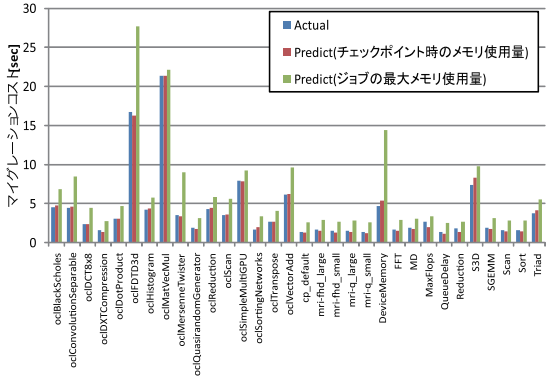


図 7　GPU から CPU へのマイグレーションのコストの実測値と予測値
Fig. 7　Predicated and actual values of GPU-to-CPU migration costs.

端に少なく実行時間の長いカーネルを短時間に多数起動する．本評価で用いた CheCL ではチェックポイント取得要求の時点でコマンドキューに入っているすべてのカーネルの実行が終了してから，チェックポイント取得処理が開始される．そのため，cp ではキューに入っているすべてのカーネルの実行が終わってからチェックポイントを取得するため，チェックポイントを取得するまでの待機時間のばらつきが大きくなり，予測誤差が大きくなってしま

う．これらのキューに入っているカーネルの待機時間が提案する予測モデルでは考慮されていないため，今後このようなアプリケーションのためにカーネルごとの実行時間予測もスケジューリング時に行うべきなのか，さらに検討する予定である．

4.2　スケジューリング手法の評価

本節では，提案手法によるターンアラウンドタイム短縮効果をシミュレーションにより評価する．以下，提案手法を MCTBM と表記する．提案手法との比較のため，MCT，バックフィル付き MCT（以下，MCTB），およびマイグレーション付き MCT（以下，MCTM）の 3 種類のスケジューリング手法の性能も評価する．

本評価では，アクセラレータを OpenCL のデバイスとして利用可能な**高性能計算資源**と，アクセラレータを搭載しないために汎用プロセッサを OpenCL のホスト兼デバイスとして利用する**低性能計算資源**から構成される複合型計算システムを仮定する．高性能計算資源と低性能計算資源との間でマイグレーションを行う場合には，カーネルを実行するデバイスを汎用プロセッサとアクセラレータの間で切り替える必要がある．したがって，ジョブスケジューリングのためには，図 6 や図 7 に示されるような異種プロセッサ間のマイグレーションのコスト予測が求められる．

ここで，低性能計算資源での実行時間と高性能計算資源での実行時間の比を加速度倍率と定義する．低性能計算資源での実行時間を t_{low}，加速度倍率を X とすると，高性能計算資源での実行時間 t_{high} は以下の式 (6) で表される．

$$t_{\mathrm{high}} = \frac{t_{\mathrm{low}}}{X} \tag{6}$$

アクセラレータ利用による高速化の効果はジョブによって大きく異なるため，加速度倍率も各ジョブに対して固有の値が設定される．

本論文では，上記の複合型計算システムのシミュレータを作成し，提案手法の有効性の評価に用いる．本評価に用いるシミュレーションパラメータを**表 4** に示す．本シミュレーションは，高性能計算資源 512 台および低性能計算資源 512 台から構成される計算システムに対して，ジョブが逐次投入される状況を想定する．ジョブは複数の計算資源を必要とする並列ジョブであり，必要な計算資源数は 2 のべき乗とする．各ジョブの加速度倍率は最大値を 10 とする一様分布に従うものとする．また，ジョブの投入はポアソン分布に従う．ここで，単位時間あたりのシステムの処理性能を P_{ave}，単位時間あたりにシステムに到着する仕事量を W_{ave}，定常状態でのシステム負荷を L とすると，ジョブの平均投入間隔 I_{ave} は式 (7) に従う[7]．

表 4　シミュレーションパラメータ
Table 4　Simulation parameters.

高性能計算資源数	512
低性能計算資源数	512
ジョブの最大必要計算資源数	512
低性能計算資源でのジョブの最大実行時間	1440 分
ジョブの最大加速率	10
計算資源あたりの最大メモリ使用量	4 GB
1 GB あたりのマイグレーションコスト	25 秒，50 秒，75 秒
ジョブの総数	100000
システムの負荷	0.9

$$I_{\mathrm{ave}} = \frac{P_{\mathrm{ave}}}{W_{\mathrm{ave}}} \times L \tag{7}$$

ジョブのマイグレーションコストは，ジョブのメモリ使用量に比例するものと仮定する．ジョブのメモリ使用量は，そのジョブの計算資源あたりのメモリ使用量と必要計算資源数の積とする．各ジョブごとに計算資源あたりのメモリ使用量は異なり，その分布は最大値を 4 GB とする一様分布に従うとする．また，図 4 および図 5 の予備実験結果より，1 GB あたりのマイグレーションコストを 25 秒とする．さらに，1 GB あたりのマイグレーションコストを 50 秒および 75 秒に過大評価した場合のスケジューリング結果も評価する．

本評価では，ジョブの必要計算資源数が異なる 2 パターンの条件下において，各スケジューリング手法のターンアラウンドタイムを比較する．**図 8** にシミュレーションの結果を示す．図 8 の縦軸では，各スケジューリング手法のターンアラウンドタイムが MCT のターンアラウンドタイムで正規化されている．横軸は各スケジューリング手法を示す．ターンアラウンドタイムの内訳は，平均実行時間（図中の実行時間），マイグレーションを行ったジョブの平均マイグレーションコスト（図中のマイグレーションコスト），遊休計算資源がシステム中にあるが，待機状態となるジョブの平均待機時間（図中の待機時間 A），必要な計算資源がなく，待機状態となるジョブの平均待機時間（図中の待機時間 B）である．

図 8 (a) は，必要計算資源数の少ないジョブがシステムに大量に投入される状況を想定している．各ジョブの必要計算資源数が少ないため，システムの計算資源は可能な限り利用される．このため，システム内にバックフィル領域が発生しにくく，MCTB は効果を発揮することができない．一方で，低性能計算資源と高性能計算資源の性能差が非常に大きいために，式 (1) に基づくスケジューリングを行う MCT では，高性能計算資源しか利用されず，低性能な計算資源が遊休状態となる．MCTM は，この利用されない低性能な計算資源上で

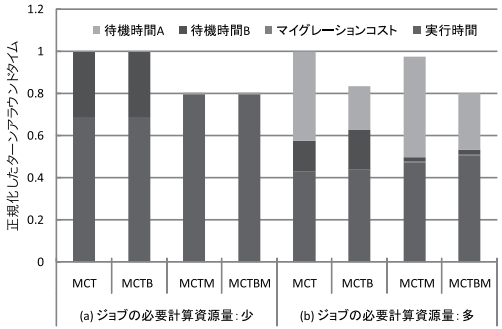


図 8　1 GB あたりのマイグレーションコストが 25 秒のときのターンアラウンドタイム
Fig. 8　Turn-around time if migration of 1 GB data takes 25 seconds.

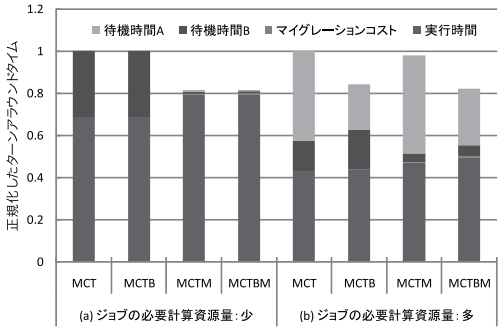


図 9　1 GB あたりのマイグレーションコストが 50 秒のときのターンアラウンドタイム
Fig. 9　Turn-around time if migration of 1 GB data takes 50 seconds.

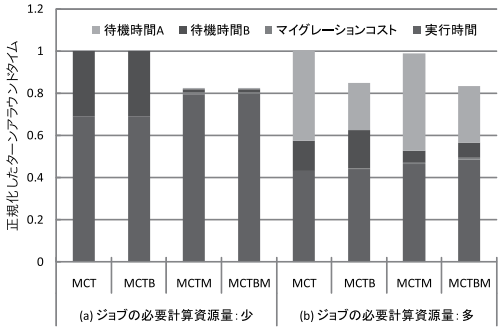


図 10　1 GB あたりのマイグレーションコストが 75 秒のときのターンアラウンドタイム
Fig. 10　Turn-around time if migration of 1 GB data takes 75 seconds.

待機状態のジョブを先に実行し，高性能な計算資源が利用可能となる時点でマイグレーションすることにより，ターンアラウンドタイムの短縮を実現する．そのため，このような状況では MCTM は十分な効果を発揮することができる．

図 8 (b) は，必要計算資源数の多いジョブが大量にシステムに投入される状況を想定している．各ジョブは多くの計算資源を必要するため，十分な数の計算資源を確保できずに待機状態となるジョブが増加する．このため，バックフィル領域も増加する．高性能計算資源を利用するまでの待機時間が長くなると，低性能計算資源を利用する方がターンアラウンドタイムを短くできる状況が増える．このため，MCT でも低性能な計算資源が積極的に利用され，遊休状態の低性能計算資源は少なくなる．その結果，MCTM はほとんど効果を発揮することができない．一方で，待機中の後続ジョブが少ない計算資源数しか必要としない場合，後続ジョブを遊休状態にある少数の計算資源を使って先に実行することにより，そのターンアラウンドタイムを短縮することができる．このため，バックフィル領域を有効利用できる MCTB は高い効果を発揮できる．

提案手法である MCTBM は，両方のスケジューリング手法の長所を取り入れるため，どちらの状況においても最もターンアラウンドタイムを短縮することが可能である．図 8 (a) から，マイグレーションの効果によりターンアラウンドタイムを約 19.7%短縮でき，図 8 (b) から，プリエンプティブバックフィルの効果によりターンアラウンドタイムを約 20%短縮できることが明らかになった．これらの結果より，他の手法よりも広いシミュレーション条件下において提案手法が有効であることが示された．

最後に，マイグレーションコストの予測精度がスケジューリングに与える影響について考察する．提案スケジューリング手法においてマイグレーションコストを予測する目的は，マイグレーションやバックフィルによるターンアラウンドタイムの短縮が期待できない場合に，それらを行わないためである．**図 9** と**図 10** から，マイグレーションコストを 2 倍や 3 倍に過大評価しても，待機時間 B がわずかに長くなるだけで，各スケジューリング手法の平均ターンアラウンドタイムに与える影響は十分に小さいことが分かる．コストを過大評価した場合に待機時間 B がわずかに長くなる原因は，マイグレーションコストとバックフィル領域の長さが僅差である場合に前者を過大評価するとそのバックフィル領域が使われないため，遊休計算資源が存在している場合でもジョブが待機状態になるためである．しかし，そのような，比較的短期間しか存在しないバックフィル領域を使ってジョブを実行した

としても，得られるターンアラウンドタイム短縮効果はわずかである．各手法の平均ターンアラウンドタイムに大きな変化がないことから，仮にコストを 3 倍に過大評価した場合でも，マイグレーションやバックフィルが効果的な状況ではそれらを適切に実施できていることが分かる．4.1 節で述べたとおり，提案予測モデルに基づくマイグレーションコスト予測のエラー率はほぼつねに 1.0 以下であり，マイグレーションコストを 2 倍以上に過大評価することは稀である．この結果から，適切なスケジューリングを行うという観点から見て，提案予測モデルは十分な精度でマイグレーションコストを予測できることが示された．

5. 結　　論

本論文では，複合型計算システムにおけるターンアラウンドタイムの短縮を目的とし，マイグレーションとプリエンプティブバックフィルに基づくスケジューリング手法を提案した．また，ジョブ投入時点でマイグレーション時のコストを予測する予測モデルを提案した．

予測精度の評価の結果，一部の例外を除いて，最大メモリ使用量からマイグレーションコストの最悪値を高精度で予測できることが明らかになった．また，マイグレーションとプリエンプティブバックフィルではそれぞれ効果的な状況が異なるため，両者の長所を兼ね備える提案スケジューリング手法がターンアラウンドタイムを最も効果的に短縮できることが示された．

今後の課題として，メモリ使用量が少なく，短期間に多くのカーネルを実行するアプリケーションに対して，マイグレーションコストの予測誤差が大きくなる問題の解決があげられる．

謝辞　本研究の一部は，科研費若手研究（B）（21700049），中山隼雄科学技術文化財団，および科学技術振興機構戦略的創造研究推進事業（JST CREST）「自己修復機能を有する 3 次元 VLSI システムの創製」の助成による．

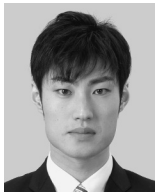
参 考 文 献

1) Khronos OpenCL Working Group: The OpenCL Specification version 1.0 (2010).
2) Takizawa, H., Koyama, K., Sato, K., Komatsu, K. and Kobayashi, H.: CheCL: Transparent Checkpointing and Process Migration of OpenCL Applications, *IEEE IPDPS* (2010).
3) Maheswaran, M., Ali, S., Siegel, H.J., Hensgen, D. and Freund, R.F.: Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems, *Journal of Parallel and Distributed Computing*, Vol.59, pp.107–131 (1999).
4) Feitelson, D.G. and Weil, A.M.: Utilization and predictability in scheduling the IBM SP2 with backfilling, *12th International Parallel Processing Symposium*, pp.542–546 (Apr. 1998).
5) Snell, Q., Clement, M.J. and Jackson, D.B.: Preemption Based Backfill, *Revised Papers from the 8th International Workshop on Job Scheduling Strategies for Parallel Processing, JSSPP '02*, pp.24–37 (2002).
6) Vadhiyar, S. and Dongarra, J.: A Performance Oriented Migration Framework For The Grid, *Proc. 3rd International Symposium on Cluster Computing and the Grid*, pp.130–137 (2003).
7) 藤原一毅，合田憲人：資源予約に基づく並列ジョブスケジューリング手法の評価，情報処理学会・電子情報通信学会並列処理シンポジウム JSPP2002，pp.159–160 (2002).

(平成 23 年 1 月 28 日受付)
(平成 23 年 5 月 26 日採録)

小山賢太郎
平成 23 年東北大学大学院情報科学研究科博士前期課程修了．修士（情報科学）．平成 23 年より東日本旅客鉄道株式会社に在籍．GPU クラスタによる大規模演算に関する研究に従事．

佐藤　功人（正会員）
平成 21 年より東北大学大学院情報科学研究科博士後期課程に在籍．GPU による汎用演算処理に関する研究に従事．IEEE CS 会員．



小松　一彦

平成 20 年東北大学大学院情報科学研究科博士後期課程修了．博士（情報科学）．平成 20 年東北大学サイバーサイエンスセンター産学連携研究員．並列計算，大規模計算，コンピュータグラフィックスとその応用に関する研究に従事．IEEE CS 会員．

村田　善智

平成 20 年東北大学大学院情報科学研究科博士後期課程修了．博士（情報科学）．平成 20 年東北大学サイバーサイエンスセンター産学連携研究員．Grid コンピューティング，ボランティアコンピューティング，P2P コンピューティングおよびベクトルスーパコンピュータの Grid 連携に関する研究に従事．IEEE CS 会員．

滝沢　寛之（正会員）

平成 11 年東北大学大学院情報科学研究科博士課程修了．博士（情報科学）．同年新潟大学総合情報処理センター助手，平成 15 年東北大学情報シナジーセンター助手，平成 16 年同大学大学院情報科学研究科講師を経て，平成 21 年より同研究科准教授．高性能計算システム，コンピュータアーキテクチャとその応用に関する研究に従事．平成 16 年 ISPA04 最優秀論文賞，平成 18 年船井情報科学奨励賞，平成 20 年情報処理学会東北支部野口研究奨励賞，平成 21 年石田記念財団研究奨励賞受賞．電子情報通信学会，IEEE CS 各会員．

小林　広明（正会員）

昭和 63 年東北大学大学院博士課程修了．同年東北大学助手．平成 3 年東北大学講師．平成 5 年東北大学助教授．平成 13 年東北大学教授（平成 20 年 4 月よりサイバーサイエンスセンター長兼任）．平成 18 年より NII 客員教授併任．コンピュータアーキテクチャ，並列処理システムとその応用に関する研究に従事．工学博士．IEEE Senior Member，ACM，電子情報通信学会各会員．