2004－MPS－51　(8)
2004／9／13

グリッド環境における独立粗粒度タスク集合の動的スケジューリングアルゴリズムの性能評価

田 中 邦 明†　藤 本 典 幸†　萩 原 兼 一†

インターネットに接続された家庭やオフィスの計算機の余剰計算パワーを使用して大規模な計算を行うことをデスクトップグリッドコンピューティングと言う．各計算機の所有者の利用状況により提供される余剰計算パワーは動的に変化するため，デスクトップグリッドコンピューティングの余剰計算パワーも動的に変化する．また，各計算機のピーク性能も様々である．このように計算機性能がヘテロかつ動的に変動するデスクトップグリッド環境で余剰計算パワーを有効に使用するには，タスクのスケジューリングアルゴリズムが重要である．本研究では，計算機性能の変動の予測情報を用いない近似アルゴリズム RR と計算機性能の変動の予測情報を用いる発見的アルゴリズム DFPLTF とのシミュレーションによる実験的評価を行った．その結果，計算機のピーク性能のヘテロ性が高く，余剰計算パワーの正確な予測ができない場合，RR のほうが有効なアルゴリズムであることがわかった．

Peformance Evaluation of Grid Scheduling Algorithms for Independent Coarse-Grained Tasks

Kuniaki Tanaka† ,Noriyuki Fujimoto† and Kenichi Hagihara†

To solve a time-consuming problem using spare computing power of many PCs in homes and offices through the Internet is called desktop grid computing. The spare computing power dynamically changes. Also, peak performance of each PC is different. In this case, task scheduling algorithm is important. In this research, we experimentally evaluate approximation algorithm RR, which uses no prediction information on PCs' performance, and algorithm DFPLTF, which uses such prediction information. As a result, it turns out that RR is more effective than DFPLTF if heterogeneity of PCs' peak performance is high and we cannot well predict accurate spare computing power of PCs.

1. はじめに

スケジューリング問題で最も標準的に用いられている評価関数はメイクスパンである．しかし，デスクトップグリッド環境においてメイクスパンを評価関数とするスケジューリング問題には，近似アルゴリズムは一般に存在しない[2]．デスクトップグリッド環境における別の評価関数として TPCC(Total Processor Cycle Consumption)[2] がある．TPCC とは全タスクを終了するまでに消費されたグリッドの計算パワーの合計である．

メイクスパンを評価関数としたグリッドスケジューリングアルゴリズムの研究は多く行われている．その例として DFPLTF，Sufferage，Min-min，Max-min などが挙げられる[3)5)]．これらのアルゴリズムは計算パワーの変動の予測情報を用いた動的アルゴリズムであり，計算パワー変動の正確な予測ができれば有効なアルゴリズムである．しかし，一般的に計算パワー変動の正確な予測を行うことは容易ではないと考えられる．また，これらのアルゴリズムは発見的なアルゴリズムであり，性能保証はない．

性能保証のあるグリッドスケジューリングアルゴリズムとして計算パワーの予測情報を用いず，タスクの複製を行う RR(listscheduling with Round-robin order Replication) がある．RR は TPCC 最小化スケジューリング問題に対する近似アルゴリズムである[2]．

計算パワー変動の予測情報を用いない RR は計算パワー変動の正確な予測が行える場合，予測情報を用いるアルゴリズムに劣る可能性が高い．しかし，計算パワー変動の正確な予測が行えない場合は，どちらが優れているアルゴリズムかはわからない．

よって本論文では，RR において性能保証されている評価関数 TPCC を評価関数とし，RR と計算パワー

† 大阪大学 大学院情報科学研究科/Graduate School of Information Science and Technology, Osaka University

の予測情報を用いた発見的アルゴリズム DFPLTF との比較を計算パワーの予測情報に誤差を設定して行い，RR が有効なアルゴリズムであるか検討する．

2. グリッドスケジューリングモデル

2.1 グリッドの性能モデル

タスクサイズとはタスクに含まれる命令数であり，プロセッサ速度とは単位時間あたりにそのプロセッサが余剰計算パワーによって処理可能な命令数とする．

t を非負の整数とし，時刻 [t,t+1) 間のプロセッサ p のプロセッサ速度を $s_{p,t}$ とする．$s_{p,t}$ =0 は，計算機の所有者が非常に負荷の大きい処理を行っているか、計算機の電源が入っていないなどの理由により，プロセッサが使用不可能な状況を表す．

2.2 TPCC

TPCC とは，全タスクを終了するまでに消費されたグリッドの計算パワーの合計である．全タスクの終了時刻を M，プロセッサ台数を n，プロセッサ番号を p($1 \leq p \leq n$)，スケジューリング開始からの時間を t，プロセッサ p の時刻 [t,t+1) でのプロセッサ速度を $s_{p,t}$ とすると TPCC は式 (1) で表される．

$$\sum_{p=1}^{n} \sum_{t=0}^{\lfloor M \rfloor - 1} s_{p,t} + \sum_{p=1}^{n} (M - \lfloor M \rfloor) s_{p,\lfloor M \rfloor} \qquad (1)$$

2.3 TPCC 近似率

スケジュール S の TPCC 近似率とは，最適スケジュールの TPCC に対する S の TPCC の比率である．S の TPCC をタスクサイズの合計で割った値は，S の TPCC 近似率の上界となる．タスク数を n，タスク番号を t($1 \leq t \leq n$)，タスク番号 t のタスクサイズを L_t とすると TPCC 近似率は式 (2) で表される．本論文では，この TPCC 近似率を評価関数として採用する．

$$TPCC / \sum_{t=1}^{n} L_t \qquad (2)$$

2.4 スケジューリング問題

各プロセッサでタスクの処理が完了すればスケジューラにそのタスクが終了したという通知が届くものとする．この通知のことをタスク実行完了通知と呼ぶことにする．

スケジューリングを行うタイミングは，時刻 0 とタスク実行完了通知を受け取った場合である．時刻 0 のスケジューリングでは各プロセッサに 1 つ以上タスクが割り当てられるまでスケジューリングを行う．

以下に，本スケジューリング問題の入力と出力をまとめる．

- 入力
  独立なタスク集合 V，プロセッサ数，各時刻におけるプロセッサ速度の予測値
- 出力
  V の動的スケジュール

3. グリッドスケジューリングアルゴリズム

WQ(Workqueue) はプロセッサ速度の予測情報およびタスクサイズを用いない単純なスケジューリングアルゴリズムである．プロセッサが処理可能になると任意のタスクが割り当てられ，この手続きをタスクがなくなるまで繰り返す．

RR(listscheduling with Round-robin order Replication) はタスクの複製を行うスケジューリングアルゴリズムである．WQ と同じ方法で全てのタスクがプロセッサに割り当た後，必要に応じて RR はタスクの複製を行う．RR はプロセッサ速度の予測情報およびタスクサイズを必要としないスケジューリングアルゴリズムである．

DFPLTF(Dynamic FPLTF) はプロセッサ性能がヘテロでプロセッサ速度が動的に変化するようなデスクトップグリッド環境で適用できるように静的スケジューリングアルゴリズム FPLTF(Fastest Processor to Largest Task Fast)[4] を改良した動的スケジューリングアルゴリズムある．DFPLTF は一番大きいタスクから順に優先度を与え，優先度順にタスクを一番速く処理可能なプロセッサに割り当てる．この手続きをすべてのプロセッサにタスクを割り当てるまで繰り返す．1 つのタスクが終了する度に，各プロセッサで未実行のタスクはアンスケジュールされ，すべてのプロセッサにタスクを割り当てるまで優先度順に一番速く処理可能なプロセッサにタスクを割り当てる．すべてのタスクを割り当て，各プロセッサで未実行のタスクがなくなればスケジューリングを終了する．そのため，DFPLTF にはプロセッサ速度の予測情報とタスクサイズが必要である．

DFPLTF はタスクの完了通知がくると，未実行のタスクの各々について，プロセッサ速度の予測情報を用いてすべてのプロセッサの中から一番速く処理可能なプロセッサを見つけなければならない．このため DFPLTF のスケジューリングのオーバーヘッドは大きい．これに対して RR は，タスクの完了通知に対して，タスク複製を行わない場合は未実行のタスクの任意の一つを通知を送ってきたプロセッサに割り当てるだけでよく，タスク複製を行う場合でもスケジューリ

ングのオーバヘッドは比較的小さい [2].

4. シミュレーション結果と考察

本章では，まず 4.1 節でプロセッサ速度の予測情報を用いるスケジューリングアルゴリズムに対してその予測情報に誤差を導入するために用いたモデル QoI について説明し，4.2 節でシミュレーションを行ったシミュレーション環境について説明し，4.3 節でシミュレーション環境下でのシミュレーション結果について述べた後，4.4 節でシミュレーション結果からうかがえる考察について述べる．

4.1 QoI(Quality of Information)

プロセッサ速度の予測情報を用いるアルゴリズムに対してその予測情報に誤差を与えるため，QoI(Quality of Information) モデル [1] を本実験では採用する．QoI とは各タスクの 100%正確な処理時間に対して [-p,+p](p は 0% から 100%) の範囲のランダムな誤差をかけたものをそのタスクの予測処理時間とするモデルである．

4.2 シミュレーション環境

本実験は，プロセッサ速度の予測情報を用いずにタスクの複製のみを行うスケジューリングのオーバーヘッドが小さいアルゴリズム RR がプロセッサ速度変化の予測情報を用いるスケジューリングのオーバーヘッドが大きいアルゴリズム DFPLTF と比較して有効なスケジューリングアルゴリズムであるかを検討するための初期的な実験である．なお，本実験では RR も DFPLTF もスケジューリングのオーバーヘッドは 0 としてシミュレーションしている．

タスク数，タスクサイズ，プロセッサ台数は固定値として，QoI のパラメータとプロセッサ性能のヘテロ性を変動させて RR と DFPLTF との TPCC 近似率の比較を行う．

具体的なパラメータとして以下を与えた．なお，0 ストリングとはプロセッサ速度が一定期間 0 になり，その期間はプロセッサが使用不可能な状態のことである．0 ストリングは計算機の所有者が実行する計算負荷の重いジョブをモデル化したものである．

- タスク集合
  サイズ 3600000x64 個, サイズ 7200000x64 個, サイズ 10800000x64 個, サイズ 14400000x64 個 (計:256 個)
- プロセッサ集合
  プロセッサ性能のヘテロ性の異なる以下の 3 つの環境でシミュレーションを行った．
  (1) ヘテロ性:小
  ピーク性能 1000x4 台, ピーク性能 2000x4 台, ピーク性能 3000x4 台, ピーク性能 4000x4 台 (計:16 台)
  (2) ヘテロ性:中
  ピーク性能 1000x4 台, ピーク性能 2000x4 台, ピーク性能 4000x4 台, ピーク性能 8000x4 台 (計:16 台)
  (3) ヘテロ性:大
  ピーク性能 1000x4 台, ピーク性能 4000x4 台, ピーク性能 8000x4 台, ピーク性能 16000x4 台 (計:16 台)
- QoI : 0%,5%,10%,25%,50%,75%
- 単位時間:1 秒
- 0 ストリング発生確率 : 1/5400
- 0 ストリングの時間 : 60 分

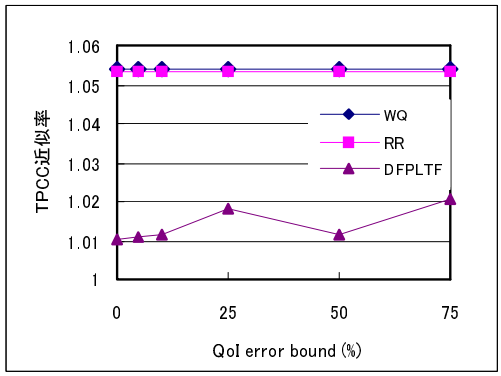

図 1 サーバのヘテロ性 : 小
Fig. 1 heterogeneity of servers : low

4.3 シミュレーション結果

4.2 節で示したシミュレーション環境下でシミュレーションを行った結果を**図 1** から**図 3** に示す．WQ と RR はプロセッサ速度の予測情報を用いないアルゴリズムなので，QoI の値に関わらず，その TPCC 近似率は一定となる．なお，DFPLTF に対して予測情報に誤差を与えた場合の TPCC 近似率の値はシミュレーションを 5 回行った平均値である．

4.4 考　　察

図 1 から図 3 より以下のことが実験的に得られた．

(1) プロセッサのピーク性能のヘテロ性が上がるにつれて，WQ と RR の TPCC 近似率の差が広がっている．このことからプロセッサのピーク性能のヘテロ性が上がるにつれてタスクの複製が効果的になることがわかった．

(2) プロセッサ速度の予測情報の誤差が大きくなるにつれて，DFPLTF の TPCC 近似率は悪くなり，またプロセッサのピーク性能のヘテロ性が

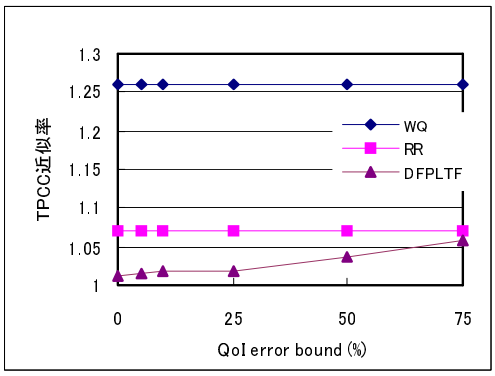

図 2 サーバのヘテロ性：中
Fig. 2 heterogeneity of servers : middle

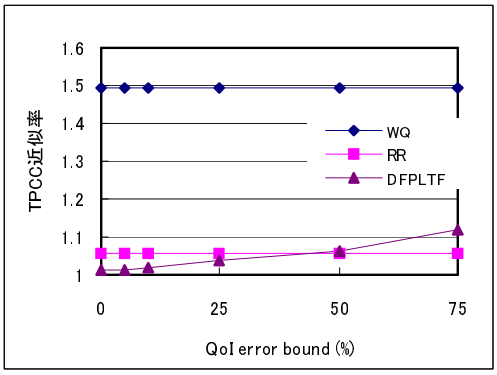

図 3 サーバのヘテロ性：大
Fig. 3 heterogeneity of servers : high

大きくなるにつれて TPCC 近似率が悪くなることがわかった．

(3) プロセッサのピーク性能のヘテロ性が大きく，かつ正確なプロセッサ速度の予測が行えない場合，TPCC 近似率において RR のほうが DFPLTF より有効なアルゴリズムであることがわかった．

(4) 正確なプロセッサ速度の予測が行えると仮定して，DFPLTF と RR の TPCC 近似率を比較すると，RR の方が TPCC 近似率は若干悪くなっている．しかし，TPCC 近似率の差は，プロセッサ速度のピーク性能のヘテロ性が大きくなるにつれて縮まっている．このことから，正確なプロセッサ速度の予測が行える状況でも，プロセッサのピーク性能のヘテロ性が大きい場合，RR の TPCC 近似率は DFPLTF に対して若干劣るが，スケジューリングのオーバーヘッドを考えると，RR は十分有効なアルゴリズムだと言える．

5. 結　　論

デスクトップグリッド環境で，プロセッサ速度の予測情報を用いずにタスクの複製を行うスケジューリングのオーバーヘッドが小さいアルゴリズム RR をプロセッサ速度の予測情報を用いるスケジューリングのオーバーヘッドが大きいアルゴリズム DFPLTF と比較して，RR が有効なスケジューリングアルゴリズムであるかを検討するための初期的な実験を行った．その結果，プロセッサ性能のヘテロ性が高くプロセッサ速度の正確な予測が行えない場合，プロセッサ速度の予測情報を用いないスケジューリングアルゴリズム RR の方が性能がよくなることを実験的に示した．また，プロセッサ速度の予測が正確に行える場合でも，予測情報を用いない RR は予測情報を用いる DFPLTF とそれほど性能が変わらないということもわかった．

今後の課題として，プロセッサ台数，タスク数，タスクサイズのパラメータを変更し，様々なデスクトップグリッド環境において，本論文で実験的に得られた結果と同じ結果が得られるか確かめることが挙げられる．

参 考 文 献

1) H.Casanova, A.Legrand, D.Zagorodnov and F.Berman. Heuristics for Scheduling Parameter Sweep Applications in Grid Environments. 9th Heterogeneous Computing Workshop, pp. 349-363, 2000

2) N.Fujimoto and K.Hagihara. Near-Optimal Dynamic Task Scheduling of Independet Corse-Grained Tasks onto a Computational Grid. 32nd Annual International Conference on Parallel Processing(ICPP-03), pp. 391-398, 2003

3) M.Maheswaran, S.Ali, H.J.Siegel, D.Hensgen and R.Freund. Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems. 8th IEEE Heterogeneous Computing Workshop (HCW'99), pp. 30-44, 1999

4) D.Menascé, D.Saha, and P.Porto et al. Static and Dynamic Processor Scheduling Disciplines in Heterogeneous Parallel Architectures. Journal of Parallel and Distributed Computing, pp. 1-18,1995

5) D.Paranhos, W.Cirne and F. V. Brasileiro. Trading Cycles for Information: Using Replication to Schedule Bag-of-Tasks Applications on Computational Grids. International Conference on Parallel and Distributed Computing (Euro-Par), pp. 169-180, 2003