2004－QAI－13 (5)
2004／10／28

アプリケーション応答時間を最小化する並列データ転送アルゴリズム

山本 寛† 鶴 正人†† 尾家祐二††

† †† 九州工業大学 情報工学部 電子情報工学科
〒820–8502 福岡県飯塚市川津 680–4
Phone:(0948)-29-7687, Fax:(0948)-29-7651
E-mail: †yamamoto@infonet.cse.kyutech.ac.jp, ††{tsuru,oie}@cse.kyutech.ac.jp

あらまし ネットワーク分散計算環境でのマスタ/ワーカ型の並列計算において，アプリケーションの実行に要するデータの送信，およびその計算を複数のラウンドに分割し，データ送信/計算を重ね合わせることにより良好な応答時間を達成するスケジューリング手法が提案されている．既存の手法では，マスタおよびワーカが接続されているそれぞれのネットワークの通信性能が等しい限定された環境を想定している．しかし，実際のネットワークにおいてはそれぞれの通信性能が異なる可能性があり，そのような環境では既存の方式は応答時間を最小化できない．そこで本研究では，マスタからの複数のワーカに対する並列データ送信を考慮して通信のタイミングを決定するアルゴリズムを提案する．このアルゴリズムを用いることによりデータの送信時間がアプリケーション応答時間に与える影響を極力抑え，下限値に極めて近い性能を達成できる事を示す．

キーワード マスタ/ワーカ型，Divisible Workload，UMR，並列データ送信

Parallel Data Transfer Algorithm Enabling Minimum Application Turnaround Time

Hiroshi YAMAMOTO†, Masato TSURU††, and Yuji OIE††

† ††Dept. of Computer Science and Electronics, Kyushu Institute of Technology
Kawazu 680–4, Iizuka, 820–8502 Japan
Phone:(0948)-29-7687, Fax:(0948)-29-7651
E-mail: †yamamoto@infonet.cse.kyutech.ac.jp, ††{tsuru,oie}@cse.kyutech.ac.jp

Abstract In parallel computing based on the master/worker model in grid computing environment, in order to minimize the application turnaround time, some multiple-round scheduling algorithms have been proposed, in which the master dispatches its work to workers in a multiple round manner to overlap computation and communication. Those existing methods assume symmetrical environments where both the network attached by the master and that attached by workers have the same transmission capacity. However, in actual environments which have heterogeneous capacities, the existing methods cannot minimize the turnaround time. Therefore, in this study, we propose a new scheduling algorithm which determines how to divide the application data to be processed, and when to send them to multiple workers by considering that the master can transfer them to multiple workers in parallel. It is analytically shown that this proposed algorithm can extremely reduce the adverse effect of the data transmission time between the master and workers on the application turnaround time, and can achieve the turnaround time close to the lower bound.

Key words Master/Worker Model, Divisible Workload, UMR, Parallel Data Transfer

1. はじめに

近年，計算機/ネットワークの性能向上に伴い，インターネット上に遍在する計算資源を接続することにより，仮想的な高性能計算環境を構成することが可能となった．このような仮想的な計算環境をグリッドと呼び，産業，医療，研究の分野で幅広く用いられている [1] [2]．このような多数の計算機で構成されているグリッド環境をモデル化するには，アプリケーションの

処理を要求する計算機であるマスタが，実際に処理を行う計算機であるワーカ群に仕事を割り当てるマスタ/ワーカモデルが適している．このマスタ/ワーカモデルで処理されるアプリケーションとして，Divisible Workload アプリケーションが存在する [3] [4]．このアプリケーションの処理に要するデータは，任意の大きさの複数のチャンクに分割された場合にも同様の過程で処理可能である．マスタはデータを任意の大きさのチャンクへ分割し，複数のワーカに割り当てる．そして，ワーカ群が並列に計算を行うことにより，高速な並列処理を達成する．このようなアプリケーションの例として，画像の中から特定のパターンを検出する特徴検出 (Feature Extraction) などが挙げられる．近年，このようなアプリケーションで処理されるデータ量は増加の一途を辿り，マスタ/ワーカ間でのデータ送信に要する時間が，そのデータの計算時間と比較して無視できないほど大きくなったため，データの送信時間と計算時間を統合的に考慮可能なスケジューリング方式が必要となった [5]．この目的を達成するために，アプリケーションの実行に要するデータの送信とその計算を複数のラウンドに分割し，データ送信/計算に必要とされる時間を重ね合わせ，良好なアプリケーションの応答時間を達成する方式 (マルチラウンドスケジューリング) が提案されている [3] [6] [7] [8]．

これまでに提案されているマルチラウンドスケジューリングは，マスタおよびワーカが接続されているそれぞれのネットワークの通信性能が等しい限定された環境を想定し，マスタが同時にデータを送信するワーカ数を 1 台に限定する通信モデル (シングルポートモデル) を適用することにより，各ワーカに対する高速なデータ送信を達成していた [9] [10]．しかし，実際のネットワークではそれぞれの通信性能が異なる可能性があり，そのような場合においては，シングルポートモデルではデータ送信時に通信資源の性能を最大限発揮できず，データ送信時間がアプリケーションに与える影響を最小化できない．そこで本研究では，文献 [6] [7] で提案されている UMR(Uniform Multi-Round) を改良し，マスタが複数のワーカに対して並列にデータ送信することを考慮して，データの分割方法およびデータの送信タイミングを決定することにより，効果的なデータ送信/計算時間の重ね合わせを達成するアルゴリズム (PTUMR: Parallel Transferable UMR) を提案する．このアルゴリズムを用いることにより，現実的な環境においてもアプリケーションの実行に必要となるデータの送信時間がアプリケーションの応答時間に与える影響の最小化を目指す．

本稿では，まず 2 章においてマルチラウンドスケジューリングの概念を説明し，既存のアルゴリズムである UMR を紹介する．3 章では，現実的な環境において良好な性能を達成するアルゴリズムを提案する．4 章では，各アルゴリズムの性能を評価する環境と性能指標を定義し，提案したアルゴリズムの性能評価を 5 章で行う．最後に，6 章で本研究をまとめる．

2. 既存のスケジューリングアルゴリズム

本章では，マルチラウンドスケジューリングの概念と，マルチラウンドスケジューリングを適用したスケジューリングアルゴリズムの一例である UMR について簡潔に説明する．

2.1 マルチラウンドスケジューリング

近年，図 1 のように，データ送信とその計算を複数のラウンドに分割し，送信/計算に必要とされる時間を重ね合わせることにより良好なアプリケーションの応答時間を達成するスケジューリング手法が提案されている [3]．この図では，1 台のマスタが 1 台のワーカに対してアプリケーションの処理を依頼する例を示している．

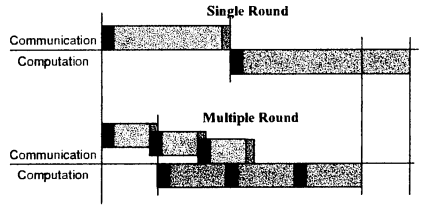

図 1 マルチラウンドスケジューリング

単一ラウンドのスケジューリングでは，マスタはアプリケーションの処理に必要となる全データ $W_{total}[units]$ を一度にワーカへ送信する．それに対してマルチラウンドスケジューリングでは，データ W_{total} は任意の大きさの複数のチャンクへ分割され，M ラウンドで処理される．ここで，ラウンド $j(=1,\cdots,M-1)$ においてマスタはワーカに対してサイズ $chunk_j[units]$ のチャンクを送信する．この際，マスタはワーカに対してデータ送信速度 $B[units/sec]$ でチャンクを送信し，ワーカは割り当てられたチャンクを計算速度 $S[units/sec]$ で処理する．さらに，マスタ/ワーカ間のデータ送信開始時にはコネクション確立のためにオーバヘッドが生じる．また，マスタがチャンクの最終ビットを送信し，ワーカが受け取るまでの遅延もオーバヘッドとなる．これらの送信開始/終了時のオーバヘッドをそれぞれ $nLat[sec]$，$tLat[sec]$ と定義する．また，マスタはワーカに対してチャンクの最終ビットを送信し終わった段階で，次のチャンクの送信が可能となる．そのため，図 1 に示すように，$tLat$ は次のラウンドのデータ送信と重ね合わせることが可能である．さらに，本研究では，ワーカによる計算時間の中で，チャンクサイズによらず固定である部分の計算時間を $cLat[sec]$ と定義する．この 1 つの例となるのが，遠隔計算機上で計算を開始する際に要する時間である．これらの定義より，ラウンド j におけるマスタ/ワーカ間のデータ送信時間は次の式で示される．

$$T_{comm_j} = nLat + \frac{chunk_j}{B} + tLat. \quad (1)$$

さらに，ラウンド j におけるワーカの計算時間は次の式で示される．

$$T_{comp_j} = cLat + \frac{chunk_j}{S}. \quad (2)$$

図 1 におけるシングルラウンドの例では，マスタがワーカにすべてのデータを送信し終わった後にワーカがその計算を行うため，全データ送信時間がそのまま計算時間に付加され，アプリケーション応答時間に占める送信時間の割合は非常に大きい．それに対して，M ラウンドでアプリケーションの処理を完了するマルチラウンドの例では，ラウンド 0 のチャンクの受信が完了した段階でワーカは計算を開始できるため，アプリケーション応答時間に占めるデータ送信時間の割合を減少できる．しかし，必要以上に多数のラウンドで処理を行うとオーバヘッドの総量が増加し，アプリケーション応答時間の悪化を招く．そのため，アプリケーション応答時間が最小となるラウンド数の導出が，マルチラウンドスケジューリングにおける重要な問題となる．

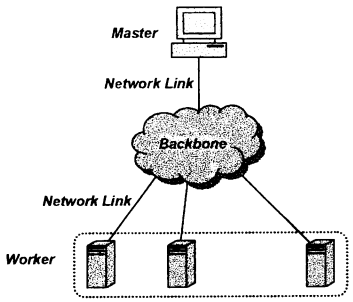

図 2　分散コンピューティングモデル.

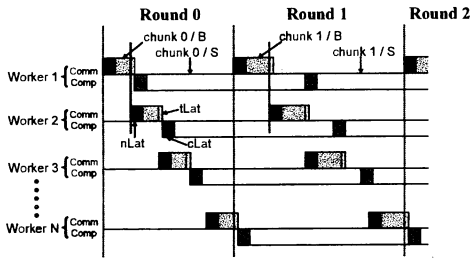

図 3　UMR によるアプリケーションの処理.

2.2　既存のアルゴリズム (UMR)

マルチラウンドスケジューリングアルゴリズムとして，UMR(Uniform Multi-Round) が提案されている [6] [7]. UMR はマスタが複数のワーカと接続している図 2 のモデルを対象としており，解析を容易とするために各ラウンドにおいて同サイズのチャンクを全ワーカに対して割り当てる．また，マスタが同時にチャンクを送信するワーカ数を 1 台とするシングルポートモデルを採用しており，アプリケーションの処理の流れは図 3 のようになる．このアルゴリズムは，前節で紹介した分割数の最適化を達成するために，初期ラウンドのチャンクサイズを小さく設定し，ワーカによる計算中にマスタは休みなくワーカに対してデータを送信することにより，ラウンドを経る毎に指数的にチャンクサイズを増加させる．以上の原理に基づき，このアルゴリズムはアプリケーションの応答時間が最小となる最適なラウンド数 M^* と，ラウンド 0 のチャンクサイズ $chunk_0^*$ を解析的に導出し，アプリケーション応答時間の最小化を達成する．詳細な最適パラメータの導出課程は参考文献 [7] に示されている．

しかし，UMR の採用しているシングルポートモデルが高速なデータ送信を達成するのは，マスタとワーカが接続されるそれぞれのネットワークの通信性能が等しい限定された環境であり，通信性能が異なる実際の環境では通信資源を最大限活用することができないため，アプリケーションの最適なスケジューリングが達成可能とは言い難い．

3.　提案するアルゴリズム

本研究では，マスタによる複数のワーカに対する並列データ送信を考慮して通信のタイミングを決定することにより，ネットワークの通信性能がマスタ側とワーカ側で異なる非対称な環境においてもデータ送信時間の影響が最小となるアルゴリズム (PTUMR: Parallel Transferable UMR) を提案する．このアルゴリズムも UMR と同様に図 2 に示すネットワークモデル

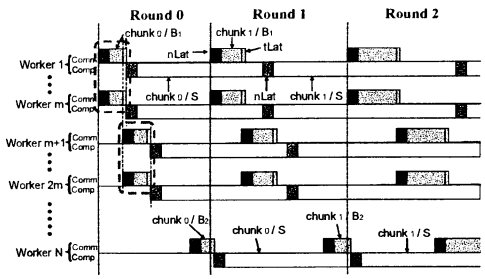

図 4　PTUMR によるアプリケーションの処理.

```
Int(M)
{
  T_real_floor = T_real(floor(M), chunk_0(floor(M)));
  T_real_ceil = T_real(ceil(M), chunk_0(ceil(M)));

  if(T_real_floor < T_real_ceil)
    return floor(M);
  else
    return ceil(M);
}
```

図 5　ラウンド数を整数化する関数

を対象とする．このモデルでは，マスタとワーカの接続するそれぞれのネットワークのデータ送信速度が異なる現実的な環境を想定している．また，バックボーンネットワークはエッジのネットワークと比較して非常に高速であり，マスタが各ワーカに対してチャンクを送信する際に，バックボーンネットワークがボトルネックとならない．提案アルゴリズムは，マスタが同時にチャンクを送信するワーカ数を並列度 m と定義し，与えられた並列度 m について，アプリケーションの応答時間が最小となる初期ラウンドのチャンクサイズ $chunk_0^*$ とラウンド数 M^* を解析的に導出する．さらに，m を変化させることで，最適な並列度 m^* を求める．

3.1　PTUMR による最適パラメータの導出

PTUMR は，1 台のマスタがバックボーンを介して N_{all} 台のワーカと接続している図 2 の環境において，データ送信時間がアプリケーションの処理に与える影響の最小化，すなわち総データ量 W_{total} を伴うアプリケーションの処理に要する応答時間 T_{real} の最小化を目的とする．ここで，実際にアプリケーションの処理を依頼するワーカ数を $N(=1,\cdots,N_{all})$ と定義する．PTUMR は，アプリケーション応答時間の最小化を達成する最適なパラメータを導出する．しかし，アプリケーションの処理中にワーカに休止時間が発生する場合には，最小化問題はきわめて複雑となる．そこで本研究では，初期ラウンドを除いてワーカの利用率が常に 1 となる場合のアプリケーション応答時間 T_{obj} を最小化問題の目的関数として扱う．

本研究では，ワーカ側の資源性能が均質である環境を想定し，マスタと各ワーカ間のオーバヘッドをそれぞれ $cLat$, $nLat$, $tLat$, 各ワーカの計算性能を S とする．さらに，図 2 におけるマスタ/バックボーン間のデータ送信速度を B_{master}, バックボーン/各ワーカ間のデータ送信速度を B_{worker} と定義する．

PTUMR も UMR と同様に，マスタは各ラウンドにおいて同サイズのチャックを全ワーカに対して割り当てる．ここで，マスタが並列度 m で N 台のワーカに対してチャンクを送信する際には，全ワーカに対して常に並列度 m でチャンクを送信することができず，余り ($N \bmod m$) が生じる可能性がある．こ

のとき，マスタはその余りのワーカに対して並列度 $N \bmod m$ でチャンクを送信する．並列度 m でチャンクを送信した際のマスタ/ワーカ間のデータ送信速度 B_1，並列度 $N \bmod m$ でチャンクを送信した際のデータ送信速度 B_2 はそれぞれ以下のように定義される．

$$B_1 = \min\left\{B_{worker}, \frac{B_{master}}{m}\right\}, \tag{3}$$

$$B_2 = \min\left\{B_{worker}, \frac{B_{master}}{N \bmod m}\right\}. \tag{4}$$

また，図4に示すように，ワーカ N におけるラウンド j の計算時間内に，マスタは休みなく全ワーカに対してラウンド $j+1$ のチャンクを送信する．これは以下のように定式化される．

$$\lfloor \frac{N}{m} \rfloor \left(nLat + \frac{chunk_{j+1}}{B_1}\right) + e\left(nLat + \frac{chunk_{j+1}}{B_2}\right) + tLat = tLat + cLat + \frac{chunk_j}{S}, \tag{5}$$

$$e = \begin{cases} 0 & \text{if } N \bmod m = 0, \\ 1 & \text{otherwise.} \end{cases}$$

左辺はラウンド $j+1$ のチャンクを全ワーカへ送信する際に要する時間を表し，右辺はワーカ N におけるラウンド j の計算時間を表す．この式より，各ラウンドのチャンクサイズ $chunk_j$ は次の式のように求まる．

$$chunk_j = \beta^j(chunk_0 - \alpha) + \alpha, \tag{6}$$

$$\alpha = \frac{B_1B_2S}{B_1B_2 - (eB_1 + \lfloor N/m \rfloor B_2)S}(\lceil N/m \rceil nLat - cLat),$$

$$\beta = \frac{B_1B_2}{(eB_1 + \lfloor N/m \rfloor B_2)S}.$$

この式が示すように，ラウンドを経る毎にチャンクサイズは指数的に増加する．ここで，マスタがワーカに対して送信するチャンクの合計は，総データ量 W_{total} と等しい．そのため，以下の式が (5) 式の仮定の下で M，$chunk_0$ が満たすべき制約条件となる．

$$G(M, chunk_0) = N \times \sum_{j=0}^{M-1} chunk_j = W_{total},$$

$$N \times \left\{\frac{1-\beta^M}{1-\beta}(chunk_0 - \alpha) + M\alpha\right\} = W_{total}. \tag{7}$$

前述したように，PTUMR はワーカの持つ計算資源の利用率が常に 1 である場合の応答時間 T_{obj} を目的関数とする．ここで，T_{obj} は次の式で表される．

$$T_{obj}(M, chunk_0) = \frac{W_{total}}{NS} + M \times cLat + \gamma \times chunk_0 + \delta \times nLat + tLat, \tag{8}$$

$$\gamma = \frac{1}{2eB_1B_2N}\{2B_1(N \bmod m) + B_2\lfloor N/m \rfloor (N + m + N \bmod m)\},$$

$$\delta = \lceil N/m \rceil - \frac{m\lfloor N/m \rfloor}{2N}\{\lfloor N/m \rfloor 2e - 1\}.$$

各ワーカの計算性能は均質であるため，各ワーカでのチャンク計算時間は一定となる．そのため，図4に示すような，各ワーカの1ラウンド目 (ラウンド 0) の計算開始時刻のずれが最終ラウンドまで保たれ，最終ラウンド (ラウンド $M-1$) における各ワーカの計算終了時刻に差が生じる．そこで，式 (8) では，全ワーカにおける計算終了時刻が同時となるように，最終ラウンド (ラウンド $M-1$) においてマスタが各ワーカへ割り当てるチャンクのサイズに補正を加えている．

式 (8) を目的関数，式 (7) を制約条件とする制約付き最小化問題を解くことにより，アプリケーションの応答時間が最小となる初期チャンクサイズ $chunk_0^*$，ラウンド数 M^* を導出する．しかし，ラウンド数は整数値である必要があるため，実数値を許すラウンド数 M^* を基にデータ送信のタイミングは決定できない．そこで，実数 M^* 付近で T_{real} が良好となる整数値を図5の関数 $Int()$ より求める．ここで，$floor(x)$ は x 以下で最大の整数を求め，$ceil(x)$ は x 以上で最小の整数を求める関数であり，$chunk_0(M)$ はラウンド数 M を式 (7) へ代入することにより初期チャンクサイズを導出する関数である．以上の過程により得られた $Int(M^*)$，$chunk_0(Int(M^*))$ に基づいて，マスタはワーカに対してデータを送信するタイミングを決定する．

表 1　評価時のパラメータ

アプリケーションの持つ総データ量 W_{total}	可変 $[units]$
総ワーカ数 N_{all}	100
アプリケーションを処理するワーカ数 N	可変
ワーカの計算速度 S	$1[units/sec]$
マスタ/バックボーン間のデータ送信速度 B_{master}	可変 $[units/sec]$
バックボーン/ワーカ間のデータ送信速度 B_{worker}	$120[units/sec]$
データ送信開始時オーバヘッド $nLat$	可変 $[sec]$
データ送信終了時オーバヘッド $tLat$	$0[sec]$
計算開始時オーバヘッド $cLat$	$0.5[sec]$
同時にデータを送信するワーカ数 (並列度) m	可変

4. 調査環境と性能指標

本研究では，マスタがバックボーンネットワークを介して100台のワーカと接続しているモデルを対象とする．性能評価時の各パラメータを表1に示す．性能評価の際には，前節で求められたパラメータ M^* と $chunk_0^*$ に基づいて，ラウンド0から段階的にラウンド $Int(M^*)-1$ の計算終了時刻を導出することにより得られる応答時間 T_{real} を，性能指標として扱う．

また，提案したアルゴリズムの有効性を調査する際の比較対象として，応答時間の下限値 T_{limit} を次のように定義する．

$$T_{limit} = cLat + \frac{W_{total}}{NS}. \tag{9}$$

この下限値は，データ送信時間が無視できるほど通信性能の良好な環境内における応答時間に相当する．提案アルゴリズムの目的は，アプリケーションの処理に要するデータの送信時間が応答時間に与える影響を最小化し，アプリケーションの応答時間 T_{real} を T_{limit} に近づけることである．

5. 解析結果

本章では，前節で説明した性能指標を基に，提案したアルゴリズムの性能を評価する．まず，マスタが1度にデータを送信するワーカ数 (並列度 m) の変化が，アプリケーション応答時間に与える影響を調査する．さらに，様々な資源状態の環境で性能を評価することにより，提案アルゴリズムの有効性を明らかにする．

5.1 並列度の変化がアプリケーションの応答時間に与える影響

はじめに，提案したアルゴリズムに関して，並列度の変化が

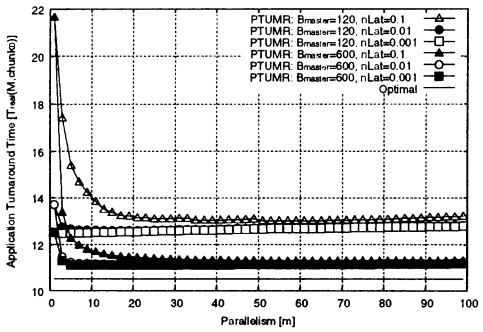

図 6　並列度の変化に伴うアプリケーション応答時間の変化.

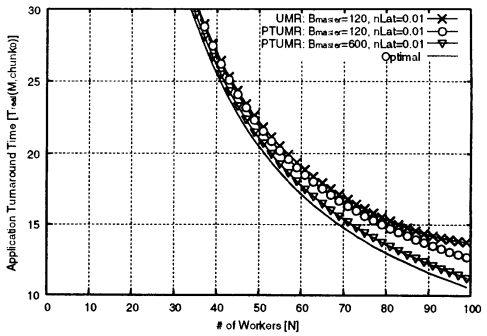

図 7　使用するワーカ数の変化に伴うアプリケーション応答時間の変化.

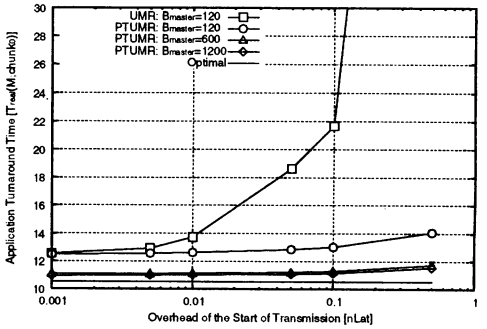

図 8　ネットワーク環境の変化に伴うアプリケーション応答時間の変化.

アプリケーションの応答時間に与える影響を調査する．ここでは，総データ量 W_{total} を 1000，アプリケーションを処理するワーカ数 N を 100 で固定とする．

図 6 に並列度 m とアプリケーション応答時間の関係を示す．マスタ/バックボーン間のデータ送信速度 B_{master} が 120 でありワーカ/バックボーン間のデータ送信速度と等しい場合には，UMR の適用するシングルポートモデルを用いることにより，マスタ/ワーカ間の通信性能を低下させることなく高速なチャンクの送信が可能となる．しかし，図 6 に示すように，マスタ/ワーカの接続するネットワークの性能が均質な環境においても，シングルポートモデルが良好に機能するのはデータ送信開始時のオーバヘッドが極めて小さい場合のみである．$nLat$ が大きい環境では，シングルポートモデルを用いて線形に $nLat$ を累積するのではなく，並列度 m を増加させ複数のワーカに対する $nLat$ を並列に重ね合わせることにより，データ送信時間がアプリケーション応答時間に与える影響を削減し，アプリケーション応答時間を下限値 T_{limit} に近づけることが可能となる．しかし，並列度 m の増加に伴い各マスタ/ワーカ間のデータ送信速度は減少し，チャンクの送信に要する時間が増加するため，m の増加に伴いアプリケーション応答時間は常に減少せず，応答時間が最小値となる最適な並列度 m^* が存在する．

また，マスタ/バックボーン間のデータ送信速度 B_{master} がバックボーン/ワーカ間の速度より大きい場合には，並列度 m が大きい場合にもマスタが各ワーカへデータを送信する際の通信性能 (B_1，B_2) は良好となる．そのため，図 6 に示すように，B_{master} の大きい環境では，並列度 m の増加に伴いアプリケーション応答時間は大きく改善され，応答時間が最小値となる並列度 m^* も B_{master} が小さい場合と比べて大きくなる．

以上より，データ送信開始時のオーバヘッドが大きく，マスタ/バックボーン間のデータ送信速度が大きい環境では，並列度の増加に伴いアプリケーションの応答時間は大きく改善される．さらに，最適な並列度を決定することにより，マスタからワーカへのデータ送信時間がアプリケーションの応答時間に与える影響を大きく削減できることが分かった．しかし，図 6 に示すように，並列度が最適値から多少変化してもアプリケーションの応答時間はほぼ一定の値となるため，スケジューラは最適な並列度を厳密に選択する必要はない．次節，PTUMR の性能を評価する場合，最適な並列度 m^* によって達成されたアプリケーション応答時間を用いる．

5.2　資源状況の変化が PTUMR に与える影響

次に，アプリケーションを実行する環境におけるネットワーク資源の状態が，アプリケーションの応答時間に与える性能を調査する．ここでは，総データ量 W_{total} を 1000 で固定とする．図 7 にアプリケーションの処理を行うワーカ数 N と，アプリケーション応答時間 T_{real} の関係を示す．この図には，PTUMR に加えて既存のアルゴリズムである UMR の応答時間と，応答時間の下限値 T_{limit} を併せて載せている．この図より，適用するアルゴリズムによらず，ワーカ数の増加に伴いアプリケーションの応答時間が減少し，特に PTUMR を用いた場合には，応答時間は下限値 T_{limit} に極めて近い値を示す．これは，使用するワーカ数を増加するにつれ，ワーカ 1 台当たりに割り当てられるデータ量が減少するためである．さらに，PTUMR はどのようなネットワーク状態においても，非常に良好な性能を発揮する．これ以降では，各アルゴリズムの性能指標として，応答時間が最小となるワーカ数 N^*，最適な並列度 m^* によって達成されたアプリケーションの応答時間 T_{real} を用い，資源状態が提案アルゴリズムの性能に与える影響を詳細に調査する．

次に図 8 に，データ送信開始時のオーバヘッド $nLat$ と，アプリケーションの応答時間 T_{real} の関係を示す．この図より，マスタ/バックボーン間の送信速度 B_{master} が増加するにつれて PTUMR のアプリケーション応答時間は減少し，既存のアルゴリズムである UMR と比較して応答時間の短縮が可能となることが分かる．さらに，データ送信開始時のオーバヘッド $nLat$ が増加しても，提案したアルゴリズムの性能は大きく悪化せず，下限値 T_{limit} に近い極めて良好な性能を示す．これは，提案したアルゴリズムではデータ送信開始時のオーバヘッドが大きい場合にも，並列度を増加させオーバヘッドを重ね合わせることにより，アプリケーション応答時間への影響が軽減可能となることが理由である．このように，提案したアルゴリズムは積極

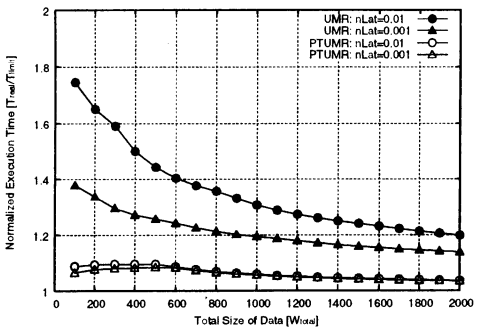

図 9 アプリケーションの総データ量と正規化応答時間の関係.

的に並列度を増加させ，複数のワーカに対するデータ送信時間の重ね合わせに努めるため，オーバヘッドの大きな環境でも下限値に近いアプリケーションの応答時間が達成可能となる．

5.3 スケジューリングアルゴリズムの適用範囲

本節では，アプリケーションの規模 (データ量) の変化が，既存のアルゴリズムである UMR，および PTUMR の性能に与える影響を調査することにより，各アルゴリズムの適用範囲を明らかにする．ここでは，マスタ/バックボーン間のデータ送信速度 B_{master} を 600 で固定とする．さらに，アプリケーションを割り当てるワーカ数は，最小の応答時間を達成する N^* に設定する．

図 9 に，総データ量 W_{total} と，応答時間 T_{real} を下限値 T_{limit} で正規化して得られた正規化応答時間の関係を示す．この図より，データ量 W_{total} が 2000 の場合には，UMR と PTUMR は共に正規化応答時間が 1.2 以下の良好な結果を示す．しかし，データ量が大きい場合にも，PTUMR は UMR と比べて正規化応答時間を減少でき，$nLat$ によらず良好な性能を発揮する．さらに，データ量 W_{total} が減少するにつれて UMR の性能は大きく悪化するが，PTUMR の正規化応答時間は常に 1 付近の極めて良好な結果を示す．UMR ではシングルポートモデルを適用しているためデータ送信開始時のオーバヘッド $nLat$ は線形に累積され，データ量が減少するにつれてアプリケーションの応答時間に占めるオーバヘッドの割合が大きく増加するため，正規化応答時間は悪化する．それに対して PTUMR は，データ量が小さい場合にも並列度 m を増加させてオーバヘッドの影響を積極的に軽減するため，アプリケーションの性能を良好な状態に保つことが可能となる．

以上より，アプリケーションの規模やネットワークの環境によらず，PTUMR は現実的なネットワーク環境において，マスタ/ワーカ間のデータ送信の影響を受けない極めて良好なアプリケーションの処理が達成できることが明らかとなった．

6. ま と め

多量のデータ送信を伴うアプリケーションにおいて，データ送信時間がアプリケーションの応答時間に与える影響の最小化を目的として，データの送信/計算時間を効果的に重ね合わせるマルチラウンドアルゴリズムが提案されている．しかし，既存のアルゴリズムは，マスタとワーカが接続されるそれぞれのネットワークの通信性能が等しい環境を対象としており，通信性能の異なる現実的な環境においては，大規模データ送信がアプリケーションの実行に与える影響を最小化できない．

そこで本研究では，マスタからの複数のワーカに対する同時データ送信を考慮して，マスタがワーカに対してデータを送信するタイミングを決定し，現実的な環境においてデータ送信時間の最小化を達成するアルゴリズム (PTUMR) を提案した．そして，様々な環境上で提案したアルゴリズムの有効性を調査した．

まず，PTUMR を用いた場合に，マスタが同時にデータを送信するワーカ数の変化がアプリケーションの応答時間に与える影響を調査した．これより，アプリケーションの応答時間が最小となる最適な並列度の存在を明らかにした．次に，最適な並列度により達成される応答時間を性能指標として様々なネットワーク環境上で性能を調査することにより，データ送信開始時のオーバヘッドが大きい環境においても，PTUMR は既存のアルゴリズムと比較してデータ送信時間がアプリケーションの応答時間に与える影響を大幅に低減し，下限値に極めて近い良好な性能を達成した．さらに，PTUMR は如何なる規模のアプリケーションにおいても，下限値近くの極めて良好な性能を維持できることを明らかにした．

謝辞 本研究の一部は，文部科学省における超高速コンピュータ網形成プロジェクト (NAREGI) および文部科学省における科学研究費補助金特定領域研究 (課題番号 16016271) の支援を受けている．ここに記して謝意を表す．

文 献

[1] I. Foster and C. Kesselman, *The GRID Blueprint for a New Computing Infrastructure,* Morgan Kaufmann Publishers, 1998.

[2] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid," *International Journal of Supercomputer Applications,* Vol. 15, Num. 3, 2001.

[3] V. Bharadwaj, D. Ghose, V. Mani, and T. G. Robertazzi, "Scheduling Divisible Loads in Parallel and Distributed Systems," *IEEE Computer Society Press,* 1996.

[4] T. G. Robertazzi, "Ten Reasons to Use Divisible Load Theory," *Jounal of IEEE Computer,* Vol. 36, No. 5, pp. 63–68, May 2003.

[5] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke, "The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets," *Journal of Network and Computer Applications,* 23:187–200, 2001.

[6] Y. Yang and H. Casanova, "UMR: A Multi-Round Algorithm for Scheduling Divisible Workloads," *Proc. of International Parallel and Distributed Processing Symposium (IPDPS'03),* Nice, France, April 2003.

[7] Y. Yang and H. Casanova, "A Multi-Round Algorithm for Scheduling Divisible Workload Applications: Analysis and Experimental Evaluation," *Technical Report of Dept. of Computer Science and Engineering, University of California CS20020721,* 2002.

[8] O. Beaumont, A. Legrand, and Y. Robert, "Optimal Algorithms for Scheduling Divisible Workloads on Heterogeneous Systems," *Proc. of International Parallel and Distributed Processing Symposium (IPDPS'03),* Nice, France, April 2003.

[9] C. Cyril, O. Beaumont, A. Legrand, and Y. Robert, "Scheduling Strategies for Master-Slave Tasking on Heterogeneous Processor Grids," *Technical Report 2002-12,* LIP, March 2002.

[10] A. L. Rosenberg, "Sharing Partitionable Workloads in Heterogeneous NOWs: Greeder Is Not Better," *Proc. of the 3rd IEEE International Conference on Cluster Computing (Cluster 2001),* pp. 124–131,2001.