社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－HPC－110 (1)
2007／6／8

Multi-Master Divisible Load スケジューリングの最適化と漸近性能

須田 礼仁†

Multi-master divisible load は著者らが提案してきたタスク再分散のためのモデルである．本稿の第 1 のテーマは，これまでに提案してきた手法の性能解析である．問題は 3 つのクラスに分けられ，それぞれタスク量 T に対して最適解との性能比が $1+O(\sqrt{T})$, $1+O(\log T/T)$, $1+O(1/T)$ となることが示された．第 2 のテーマはこれまでに提案してきた手法で得られたスケジュールの改良である．通信時間の定数項や必須のアイドル時間を考慮し，各プロセッサがほぼ同時に計算を終了するようにスケジュールを改良した．その結果，近似解と最適解との差を 1/2 から 1/3 にすることができた．

Optimization and Asymptotic Performance of Multi-Master Divisible Load Scheduling

REIJI SUDA†

Multi-master divisible load is a model for task redistribution. This paper first discusses a performance analysis of our scheme. The problems are classified into three classes, and the relative performance against the optimum solution is $1+O(\sqrt{T})$, $1+O(\log T/T)$, and $1+O(1/T)$, respectively, where T is the total task size. Second the schedules are improved, where the constant terms of the communication times and inevitable idle times are considered, and the completion times of the processors becomes nearly the same. The difference of the approximation and the optimum solutions is reduced into 1/2 or 1/3.

1. はじめに

高速ネットワークの普及と急速なマルチコア化に代表されるように，並列・分散処理は広く一般に普及するに至っている．しかし，グリッドやメタコンピューティングなど，従来型の均一な要素処理装置からなる並列処理環境からヘテロな並列計算環境へとアーキテクチャがシフトしており，新たな並列処理技術の開拓が必要となっている．GPGPU やアクセラレータを用いた高性能計算が模索されているが，これも一種のヘテロな並列処理である．ヘテロな並列計算環境にはタスク並列のパラダイム方がデータ並列よりも親和性が高い．拙著[1] はこのような考えに基づき，ヘテロ並列環境に対象を絞ったタスクスケジューリング理論のサーベイである．

しかしタスクスケジュールのアルゴリズムは多くの場合計算量が非常に大きい．我々はヘテロ並列計算環境においても現実的なコストでスケジューリングができる divisible load model（後述）を用い，データ再分散の効率的なアルゴリズムの研究を展開してきた[2)~6)]．これらでは均一なネットワークを想定しているが，論文 7) ではマルチクラスタ（クラスタ内では均一な通信性能だが，クラスタ間通信の性能は異なる）を扱っているが，通信性能が非一様だと問題が難しいことが分かる．

本稿では均一なネットワークの問題を扱う．テーマは 2 つである．第一に，タスク量が大きい極限における漸近性能について再評価を行った．これまでは Beaumont ら[8)] の評価にならい，近似解のスケジュール長 T_{app} と最適解のスケジュール長 T_{opt} の比を $T_{app}/T_{opt}=1+O(1/\sqrt{T_{opt}})$ と評価してきたが，multi-master divisible load モデルでは場合によって $1+O(1/T)$ あるいは $1+O(\log T/T)$ というよりよい漸近挙動を示すことがわかった．

第二に，これまでオーバーヘッドとして扱ってきたいくつかの効果を考慮してスケジュールを改良する手法を提案する．これまではサイズ x のタスクの通信にかかる時間を $\alpha+\beta x$ とモデル化しておきながら，スケジューリングにおいては α の効果を無視してきた．また必然的にアイドル時間が発生する場合もあるが，これもオーバーヘッドとして加算するだけで，対処はしてこなかった．今回はこれらの効果を考慮してスケジュールを改善するアルゴリズムを提案する．

2. Multi-Master Divisible Load Model

まず，multi-master divisible load のモデルを簡単に再掲する．タスクは任意の正の実数のサイズに分割することができる．タスクは均一であり，タスクの転送や処理（実行）にかかる時間は，タスクのサイズのみで決まる．**サイズ** x のタスクを任意のプロセッサ間で通信する際の**通信時間**は $\alpha+\beta x$, プロセッサ i で処理する際の**計算時間**は $\gamma_i x$ とする．各プロセッサはシングルポートと

† 東京大学 情報理工学系研究科 コンピュータ科学専攻
Dpt. of Computer Science, Grad. Schl. of Information Science and Technology, the University of Tokyo

するが，ネットワークはクロスバーであって通信は衝突しないとする．本稿では通信と計算は同時に実行できないものとする．タスクは独立で，相互に通信を行うことなく並列に処理することができるとする．本稿では結果の集計は考慮しない．

初期状態で第 i プロセッサに割り当てられている**初期タスク量**を x_i とする．古典的な divisible load model では，唯一つのプロセッサを除いて $x_i = 0$ となる**マスタワーカモデル**であった．MMDL モデルではこの制約が取り除かれ，任意のプロセッサが任意の量のタスクを初期状態で持つことができる．これにより動的負荷分散のような問題にも適用できるようになる．

解きたい問題は，プロセッサにタスクの処理や転送を行わせ，処理や転送の開始からすべてのタスクの処理が終了するまでの時間 (**スケジュール長**) を最小にするスケジュールを求めることである．

3. 漸近性能解析

著者ら[2)~6)] が提案してきたスケジューリングアルゴリズムでは，**ラウンド**と呼ばれる一定の通信・計算パターンを反復するスケジューリングを仮定する．最初と最後を除くラウンドは定常的な反復となるので，この**定常ラウンド**に対して最適スケジューリングを構築する[2)]．ここまでは Beaumont ら[8)] による手法と同じアイデアである．さらに我々は，ラウンドのサイズの可変にすることよりスケジュール長を短縮する手法を提案した[5),6)]．これは Yang ら[9)] にやや類似しているが，かなり拡張され，また計算量も格段に少ないアルゴリズムが構築できる．この節では，この提案手法の漸近性能を評価する．式の導出についてはこれらの論文を参照していただくこととし，ここでは必要な結果を参照する．

3.1 スケジューリングアルゴリズムとスケジュール長

スケジュールの第 1 段階では，最初（プロローグ）と最後（エピローグ）以外は同一のスケジュールをもつ定常ラウンドを反復するものと仮定して，通信時間の定数項 α を無視したモデル上における最適な定常ラウンドを決定する（図 1 上）．これはプロセッサ数 p に対して $O(p \log p)$ の計算量で求めることができる．そして第 2 段階では，こうして得られた定常ラウンドスケジュールを一定の割合 r_j で縮小したスケジュールを第 j ラウンドとするようなスケジュールを構築する（図 1 下）．

第 2 段階においてラウンドサイズ r_j は次のように決める．まずアイドル時間と呼ぶパラメタ ζ が与えられているとする（値の決め方は後述）．ワーカ i に対し，x_i を初期状態で持っているタスク量（ダミータスクを含む），w_i をサイズ 1 のラウンド（**基準ラウンド**とよぶ）で消費するタスク量，w_i' をサイズ 1 のラウンドの前に持っていなければならないタスク量とする．すると，第 j ラウンドの最適なサイズ r_j は $z_i = \zeta/\gamma_i$ として

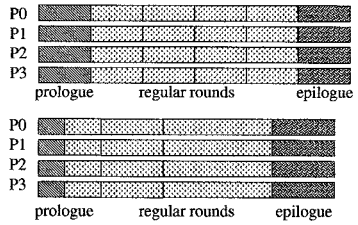

図 1 スケジューリングの方針；上：均一ラウンド（第 1 段階），下：ラウンドサイズ最適化（第 2 段階）

$$r_j = \min_i \left\{ \frac{x_i + z_i}{w_i'} - R_j \frac{w_i}{w_i'} \right\} \tag{1}$$

で与えられる[5)]．但し $R_j = \sum_{k=0}^{j-1} r_k$ で $R_0 = 0$ である．

上式で順次 $r_0, r_1, \cdots$ を決めてゆき，$R_k = 1$ に達したところですべてのタスクの処理が終わる．これによりラウンド数 k が決まる．k はアイドル時間 ζ に依存するので，$k(\zeta)$ と書くことにする．

基準ラウンドのスケジュール長（定数項 α を無視したもの）を T，定数項 α を含めたものを $T + A$ とする．すると，全体のスケジュール長 $T_{app}(\zeta)$ は

$$T_{app}(\zeta) = T + Ak(\zeta) + \zeta \tag{2}$$

となる[5)]．これを最小とするような ζ を求めればよい，というのが従来提案していた手法である．

3.2 漸近解析

Beaumont らは，図 1 上のような均一サイズラウンドの漸近性能を次のような設定で評価した．すなわち，プロセッサやネットワークは固定して，各プロセッサの初期タスク量をある定数 c 倍して $x_i^{(c)} = cx_i$ とする．この問題に対する最適スケジュール長 $T_{opt}^{(c)}$ と近似解法によるスケジュール長 $T_{app}^{(c)}$ とに対し，c が大きくなるとき $T_{app}^{(c)}/T_{opt}^{(c)} = 1 + O(1/\sqrt{c})$ となるというのが彼らの評価である．

我々も我々自身の提案手法について，この意味での漸近性能を評価する．なお，比例係数 c は括弧に入れて上付き添え字で表すが，必要な場合以外は省略する．また，基準ラウンドは単に c 倍されるだけで，そのスケジュール長 T は c に比例する．よって $O(c) = O(T)$ となることに注意する．

3.2.1 ワーカにも初期タスクがある場合

まず，すべてのワーカ i に対して $x_i > 0$ かつ $x_i - w_i > 0$ とする．このとき $\zeta = 0$ すなわち $z_i = 0$ でも常に $r_j > 0$ となり，スケジュールすることができる．すなわち $k(0)$ は有限の定数である．このとき

$$T_{app}(0) = T + Ak(0)$$

であるので，

$$\frac{T_{app}(\zeta_{opt})}{T_{opt}} \leq \frac{T_{app}(0)}{T} = 1 + \frac{Ak(0)}{T}$$

となり，最適スケジュールとの漸近性能比は $1 + O(1/T)$ となることがわかる．これは Beaumont ら[8)] の $1 + O(1/\sqrt{T})$ よりもよい漸近性能である．

3.2.2 ワーカに初期タスクがない場合の考察

これ以外の場合について考察するには，式 (1) をより詳しく調べる必要がある．そのため，式 (1) について，$\zeta(=z_i\gamma_i)$ を固定し，r_j を R_j の関数として考察する．係数 x_i, z_i, w_i, w_i' は固定されているので，各 i について括弧内は R_j の一次関数である．r_j はその最小値をとったものだから，一次関数の下側包絡線となる（図 2）．r_j はラウンドサイズなので，$0 \le R_j \le 1$ で正の値を取る．

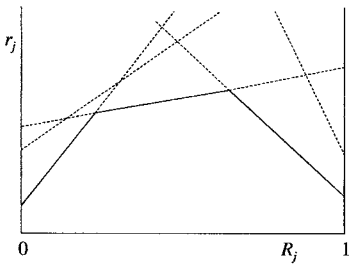

図 2 R_j と r_j の関係

さて，式 (1) は

$$r_j = \min_i \left\{ \frac{x_i}{w_i'} + \frac{\zeta}{T}\frac{1}{\gamma_i w_i'} - R_j \frac{w_i}{w_i'} \right\} \qquad (3)$$

と書き改められる．ここでスケールして変化するのは ζ と T だけであるので，ラウンドサイズは ζ/T の関数であり，ラウンド数 k も ζ/T で決まることがわかる．

3.2.3 常に通信しているマスタがある場合

さて，$x_i = w_i = 0$ となるワーカ i が存在する場合を考える．このとき

$$r_j \le \frac{\zeta}{T}\frac{1}{\gamma_i w_i'}$$

となる．よって

$$k \ge \frac{\gamma_i w_i'}{\zeta/T}$$

となることがわかる．そこで式 (2) の k を上式の右辺で置き換えたものを

$$\tilde{T}_{app} = T + A\frac{\gamma_i w_i'}{\zeta/T} + \zeta$$

とすると，$\tilde{T}_{app}$ は $\zeta/T = \sqrt{A\gamma_i w_i'/T}$ で最小値 $T + 2\sqrt{A\gamma_i w_i' T}$ を取る．よって

$$\frac{T_{app}}{T} \ge \frac{\tilde{T}_{app}}{T} = 1 + \Omega(1/\sqrt{T})$$

となることがわかる．この場合の漸近性能は Beaumont らの評価よりもオーダーとしてはよくはならない．

仮定となっている $x_i = 0$ という仮定は，ワーカ i が初期状態でタスクを持っていない完全なワーカであること，$x_i - w_i = 0$ は（マスタを含む）いずれかのプロセッサが常に通信をしていることを意味している．初期状態でタスクを持っていないワーカが多数ある場合にはこのような状況となる．

3.2.4 その他の場合

最後に，$x_i = 0$ または $x_i - w_i = 0$ となるワーカ i があるが，どのワーカについても $x_i = w_i = 0$ ではないと仮定する．まず $x_i = 0$ なるワーカ i と $x_{i'} - w_{i'} = 0$ となるワーカ i' が両方存在するとしよう．このとき，式 (3) の右辺は適当な正の実数 a, b_0, b_1, b_2 を選ぶと

$$\min_i \left\{ \frac{x_i}{w_i'} + \frac{\epsilon}{\gamma_i w_i'} - R\frac{w_i}{w_i'} \right\} \ge a\epsilon + \min\{b_0 R, b_1(1-R), b_2\}$$

($\epsilon = \zeta/T$ とした) のように下からおさえることができる（図 3 参照）．これを用いて

$$r_j = a\epsilon + \min\{b_0 R_j, b_1(1-R_j), b_2\}$$

でラウンドサイズ r_j を決めると，実際の最適ラウンドサイズよりも小さくなり，ラウンド数が増えるので，これでラウンド数を上からおさえることができる．

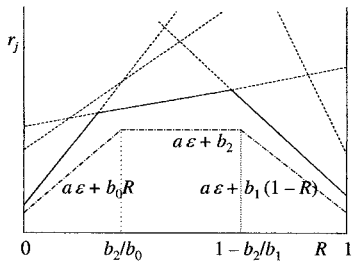

図 3 r_j を下からおさえる

まず $R_j \le b_2/b_0$ の範囲を考える．ここでは min の最初の項が選択され，$r_j = a\epsilon + b_0 R_j$ となる．よって $R_{j+1} = a\epsilon + (1+b_0)R_j$ となり，漸化式を解いて

$$R_j = \frac{(1+b)^j - 1}{b} a\epsilon$$

が得られる．$R_j \le b_2/b_0$ となる最後の j を k_0 とすると

$$k_0 = \left\lfloor \log_{1+b_0}\left(1 + \frac{b_2}{a\epsilon}\right) \right\rfloor$$

となる．ここで $\epsilon = \zeta/T \to 0$ とすると

$$k_0 = O(\log(T/\zeta))$$

であることがわかる．

次に $b_2/b_0 < R_j \le 1 - b_2/b_1$ の範囲を考える．ここでは $r_j = a\epsilon + b_2$ なので $\epsilon \to 0$ ではラウンドサイズは定数 b_2 に漸近する．すなわちこの範囲にあるラウンドの数を k_2 とすると $k_2 = O(1)$ である．

最後に $1 - b_2/b_1 < R_j$ の範囲であるが，これは最初の場合の R を $1-R$ に変えた形になっている．よってこの範囲にあるラウンドの数を k_1 とすると

$$k_1 = O(\log(T/\zeta))$$

である．

すべてあわせて，ラウンド数の合計は

$$k = k_0 + k_1 + k_2 = O(\log(T/\zeta))$$

であることがわかる．これより

$$T_{app} \leq T + A' \log(T/\zeta) + \zeta$$

となるが，この右辺は $\zeta = A'$ で最小値を取り，

$$\frac{T_{app}}{T_{opt}} \leq 1 + A' \frac{\log T - \log A' + 1}{T} = 1 + O\left(\frac{\log T}{T}\right)$$

となる．これは Beaumont ら[8] の結果よりもよい．

$x_i = 0$ のワーカがない場合には上記の解析で $k_0 = 0$，$x_i - w_i = 0$ のワーカがない場合には同様に $k_1 = 0$ とすればよく，漸近的な性能は同じとなる．

Yang ら[9] の UMR は我々の手法と同様にラウンドサイズを変更する．彼らはワーカに遊びの時間が生じないことを仮定しているが，これは通信が十分な速度を持っていることを意味しており，$x_i - w_i > 0$ にほぼ相当する．

3.2.5 実　　験

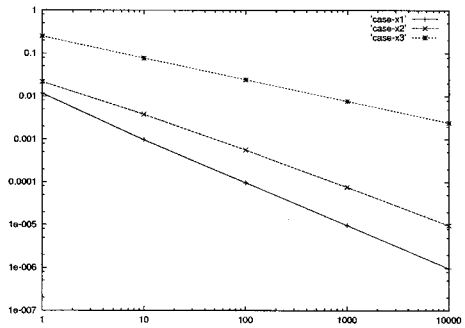

図 4　漸近解析の実験的確認

図 4 は，上記の漸近解析を実験的に確認したものである．上記の解析で漸近性能比が $1 + O(1/T)$, $1 + O(\log T/T)$, $1 + O(1/\sqrt{T})$ となる 3 つの例を構成した．そして，横軸に c, 縦軸にはスケジュール長 T_{app} とその下限 T に対して $T_{app}/T - 1$ を取ってプロットした．図を見てわかるとおり，解析通りの結果が得られている．(漸近解析はタスク量が大きいときの挙動を解析しているので，図の右端のほうが解析により合っている．)

4. スケジュールの改良

前節の解析も含め，我々のこれまでの研究では通信時間 $\alpha + \beta x$ の定数項 α を無視してスケジュールしていたため，プロセッサごとに通信回数が異なる場合にはスケジュールにずれが生じていた．またアイドル時間 ζ はプロセッサによらず一様に与えていたため，アイドル時間が必要でないプロセッサにとっては無駄な時間となっていた．本節ではこれらを考慮してスケジュールを改良し，スケジュール長のさらなる短縮を実現する．

図 5 は，従来我々が提案してきた手法によるスケジュールの例を示している．この例では，プロセッサ 0 のみが最初にタスクを持っており，他のプロセッサは手持ちのタスクがない，いわゆるマスタ・ワーカの状況である．この場合，最初に受信するワーカであるプロセッサ 1 以外は，最初やるべき仕事がないため必然的にアイドル状態となる．その結果，プロセッサ 2 は他よりも終了が遅くなってしまっている．プロセッサ 1 のスケジュールにはラウンドごとに隙間があるが，これはスケジューリングの際に通信時間の定数項が考慮されていないためである．この問題の場合，スケジュール長の下限は 6.00e-3 であるが，実際のスケジュール長は 7.49e-3 になっている．

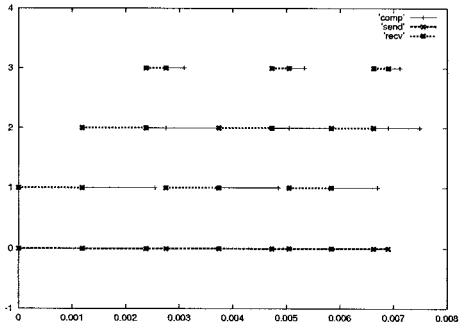

図 5　我々の従来手法によるスケジュール

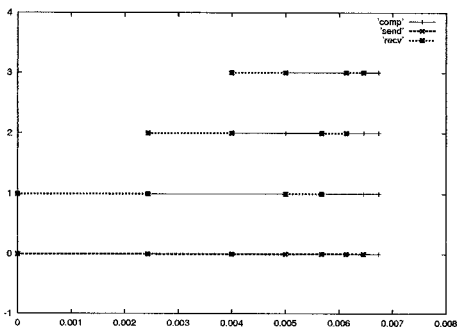

図 6　提案する改良手法によるスケジュール

これに対し，今回提案する改良手法でスケジュールすると図 6 のようになる．プロセッサ 2 と 3 には最初アイドル時間が発生することは避けられないものの，スケジュールに隙間がなくなり，処理の終了時刻もそろっている．マスタであるプロセッサ 0 自身が一部の計算を行うのも大きな違いである．ラウンド数も 3 だったものが 2 に減っており，スケジュール長も 7.49e-3 から 6.74e-3 に改善している．

4.1 改良手法のアルゴリズム

上述のようなスケジュールを構成するために加えた改良は主に 3 点である．それは (1) 通信時間の定数項を考慮したラウンドサイズ最適化，(2) ダミータスクの導入，(3) 減速による安定化で，以下これらを順に説明する．

4.1.1 ラウンドサイズ決定

従来のラウンドサイズの決定手法[5),6)] は，式 (1) に基づいている．通信時間の定数項を導入する形にこれを修

正することで，スケジュールに隙間ができないように改良する．

もともと式 (1) は，第 j ラウンドの前に残っているタスク量 $x_i + z_i - R_j w_i$ が，必要とされるタスク量 $r_j w'_i$ 以上あるという式

$$r_j w'_i \leq x_i + z_i - R_j w_i \qquad (4)$$

から導かれている．そこで通信時間の定数項を含めて式 (4) を定式化しなおす．

定数項が入ったことで，ラウンドサイズ r_j の意味を再定義する必要がある．今回の提案では次のとおりである．(1) ラウンド中の通信量が r_j 倍になるものとする．計算はできるかぎり行うので，計算量は必ずしも r_j に比例しない．(2) ラウンドは全プロセッサ同時にスタートするものとし，基準ラウンドと同じ順序で，できるだけ早く通信をスケジュールする．計算のスケジュールは通信のスケジュールに従属して決める．(3) ラウンド中の通信が終了した時点で次のラウンドに進む．

ラウンドの所要時間はラウンド中の最後の通信が終了する時刻であるが，これは r_j に関する区分線形関数になる．単純に 1 次関数になる場合もあるが，一般にはそう言えない．通信量は r_j に比例するが，定数項はプロセッサの通信回数によって異なるためである．r_j が小さいと定数項が大きい通信がラウンド時間を決め，r_j が大きいと通信量が多い通信がラウンド時間を決めるのである．

これと同じ理由で，式 (4) の右辺にあたる式は r_j に関する区分線形関数になる．但し 5) で述べているように，ラウンド中の計算は 2 箇所に集中しているため，ラウンド前に必要となるタスク量 W'_{ij} は

$$W'_{ij} = \max\{r_j w'_i + v'_i, r_j w''_i + v''_i\}$$

のように，2 つの線形関数の組み合わせになる．

式 (4) の左辺にあたる式は R_j だけの関数という形では書けない．まず，ラウンド $j-1$ の終了時刻を T_j とすると，そのうち通信に消費した時間が $c_i j\alpha + y_i \beta R_j$ となる (c_i はワーカ i の 1 ラウンド中の通信回数，y_i は受信する総タスク量である)．これ以外の時間は計算に使われているから，計算されたタスクの量は $(T_j - c_i j\alpha - y_i \beta R_j)/\gamma_i$ となる．受信したタスクが $y_i R_j$ なので，残っているタスクの量 W_{ij} は

$$W_{ij} = x_i + z_i + y_i R_j - (T_j - c_i j\alpha - y_i \beta R_j)/\gamma_i$$

である．

結局，新しい式は

$$r_j w'_i + v'_i \leq W_{ij}, \qquad r_j w''_i + v''_i \leq W_{ij} \qquad (5)$$

となる．たとえば w'_i が負の場合には

$$r_j \geq (W_{ij} - v'_i)/w'_i$$

のような形となり，r_j を下からおさえる条件となる．各ワーカにつき上や下からおさえる条件がかかるので，全体としては

$$r_{\min} \leq r_j \leq r_{\max}$$

という形になる．ここで $r_{\min} > r_{\max}$ なら解はなく，スケジュール不可能である．$r_{\min} \leq r_{\max}$ であれば $r_j = r_{\max}$ を選択する．

あとは従来と同様で，$R_k = \sum_{j=0}^{k} r_j$ が 1 に到達したらスケジュール終了である．アイドル時間 ζ を変化させてスケジュール長が最短になる値を探すのも同じである．

4.1.2 ダミータスクの導入

次に基準ラウンドのスケジュールの修正を考える．従来手法での基準ラウンドは「タスクの計算と通信との順序関係は無視してよい」という仮定でスケジュールされていた．これは基準ラウンドとして定常ラウンドを想定していて，処理するタスクを受信するのは前のラウンドであると仮定[2] されていたからである．

もちろん実際には処理すべきタスクが未受信のためアイドルにならざるをえない場合もある．基準スケジュールとラウンドサイズが与えられていれば，式 (5) を用いてアイドルタスク z_i の下限 z'_i を求めることができ，この必須アイドルタスクを「処理」する時間 $z'_i \gamma_i$ がワーカ i のアイドル時間に一致する．

言い換えると，もし初期状態で z'_i だけのタスクがワーカ i に追加されれば，ワーカ i のアイドル時間は 0 になり，基準ラウンドが仮定したとおりのスケジュールが実現されることになる．

このことから，一旦スケジュールした結果得られた z'_i を，プロセッサ i の初期タスク量にダミータスクとして追加して，もう一度スケジューリングをやりなおすことにする．こうすることにより，アイドル時間のためにワーカが引き受けられるタスク量が減少するという状況を基準ラウンドのスケジューリングに反映させることができると期待される．

しかし，与えたダミータスクの量が多すぎる場合もありうる．そこで，アイドル時間 ζ に負の値もゆるし，負の ζ では与えたダミータスク量を一定の比で圧縮することとした．

修正はさらに 2 点ある．ひとつは通信時間の定数項である．一旦スケジュールした結果かかった通信時間の定数項 $c_i k\alpha$ を通信や計算ができる時間から差し引く．

もうひとつは終了時刻のばらつきの修正である．一旦スケジュールした結果，終了時刻が他より遅かったワーカにはダミータスクを多くして送られてくるタスク量を減らし，他より早かったワーカからはダミータスクを減らして送られてくるタスク量を増やすようにした．これにより，各マシンがほぼ同時に終了するように修正されると期待される．

ただし，最適解では各マシンが同時に終了するとは限らない．また，終了時刻を無理に合わせようとしなくても，実はほぼ同時に終了するスケジュールに収束する．さらに，場合によっては終了時刻を合わせる工夫をしない方が動作が安定する．しかし多くの場合，終了時刻を合わせる工夫を入れた方が収束が速く，結果もわずかによい．このように速い収束と動作の安定性が対立するのは，非線形問題の常である．

4.1.3 減　　速

以上の修正は，一旦スケジュールした結果を用いて，次のスケジュールの条件を設定している．もし先のスケジュールと後のスケジュールが似たようなものであれば，それなりにうまくゆきそうだが，修正量が多いとまったく異なるスケジュールとなってしまい，修正の結果かえってスケジュール長が延びてしまうことがある．

この問題に対して，今回は数値計算の Newton 法でよく用いられる「減速」で対処した．すなわち，条件設定を修正して結果が悪くなった場合は，結果がよくなるまで，修正量を 1/2 倍することを繰り返す．修正の「方向」さえ正しければ，修正量を十分に小さくすれば，修正前よりも結果はよくなるはずである．

4.2 改良の結果

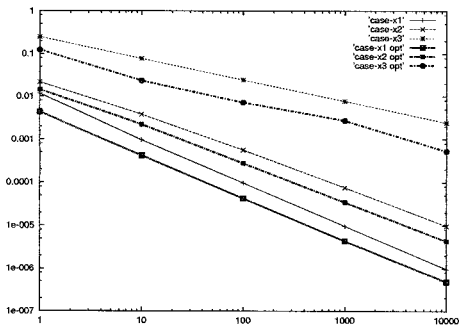

図 7　改良前と改良後の性能比

図 7 は，図 4 で示した例のそれぞれについて，改良後の性能を追加して示したものである．いずれの場合もおよそ似たような効果が得られ，スケジュール長下限との差が 1/2 から 1/3 に縮まっている．

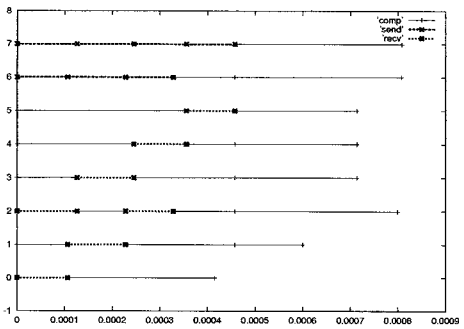

図 8　改良スケジュールが不十分な場合

図 4 の例ではスケジュール長が通信時間の定数項の少なくとも数十倍ある．しかし，スケジュール長が通信時間の定数項（ここでは 1e-4）に近い場合には，提案する改良手法は必ずしも十分よい結果を出すとはいえない．図 8 はそのような例のひとつである．このような場合には，通信の回数を減らすような工夫をしたほうがよく，根本的な解決には組み合わせ的な手法が必要である．また，4.1.2 の最後に述べた終了時刻をあわせる工夫はしないほうがよい結果を与える．

5. おわりに

本稿では我々が研究してきている multi-master divisible load によるタスク再分散手法について，漸近性能の新しい解析結果と，得られたスケジュールの改良手法の提案を行った．漸近性能は問題設定により 3 種類に分類され，それぞれ T_{app}/T_{opt} が $1+O(\sqrt{T})$, $1+O(\log T/T)$, $1+O(1/T)$ となる．また，改良手法により，近似解と最適解との差がさらに 1/2 から 1/3 に縮小された．

今後は通信と計算が重ねられる場合や，計算の結果を収集する場合についても考察する必要がある．また，今回提案した改良手法の計算量の軽減や，さらなる改良についても考えるべきと考えている．

謝辞

本研究の一部は文部科学省科学研究費による支援を受けています．

参 考 文 献

1) 須田礼仁: ヘテロ並列計算環境のためのタスクスケジューリング手法のサーベイ, 情報処理学会論文誌 : コンピューティングシステム, Vol. 47, No. SIG 18 (ACS 16), pp. 92–114 (2006).
2) 須田礼仁: Multi-master divisible load model における漸近最適スケジューリング情報処理学会研究報告 2004-ARC-157/2004-HPC-97, pp. 97–102, (2004).
3) 富さやか, 須田礼仁: Multi-master divisible load モデルに対する漸近最適スケジューリングの評価, 情報処理学会研究報告 2005-HPC-102, pp. 51–56 (2005).
4) 富さやか, 須田礼仁: Multi-master divisible load の漸近最適スケジューリングの実機への実装, 情報処理学会研究報告 2005-HPC-103, pp. 37–42 (2005).
5) 須田礼仁, 富さやか: 有限サイズの multi-master divisible load 問題に対する再分散スケジューリングアルゴリズム, 情報処理学会研究報告 2006-ARC-167/2006-HPC-105, pp. 109–114 (2006).
6) Suda, R. and Tomi, S.: Task redistribution scheduling using multi-master divisible load model, *Proc. Int'l Conf. Par. Distr. Comp. Sys.*, pp. 160–165 (2006).
7) 須田礼仁: マルチクラスタ環境での MMDL 漸近最適スケジューリング, 情報処理学会研究報告 2004-HPC-99, pp. 103–108 (2004).
8) Beaumont, O., Legrand, A. and Robert, Y.: Scheduling divisible workloads on heterogeneous platforms, *Par. Comp.*, Vol. 29, No. 9, pp. 1121–1152 (2003).
9) Yang, Y., van der Raadt, K. and Casanova, H.: Multiround Algorithms for Scheduling Divisible Loads, *IEEE Trans. Par. Dist. Sys.*, Vol.16, No.11, pp. 1092–1102 (2005).