社団法人 情報処理学会　研究報告　　2003－HPC－95　(28)
IPSJ SIG Technical Report　　2003／8／5

不均一クラスタ上での実行時間予測モデルとその評価

岸 本 芳 典† 市 川 周 一†

均一環境用に書かれた応用を不均一クラスタで実行すると，負荷不均衡により性能上の問題を生ずる．また，一部の PE には仕事を割り当てないほうが全体の実行時間が短縮できる場合がある．本研究では，高速な要素プロセッサ (PE) 上で複数のプロセスを起動することにより，全体の実行時間を短縮する方法を検討する．さらに，各 PE 上の実行時間を実測値からモデル化し，得られた予測モデルを用いて最適な PE 構成およびマルチプロセス数を予測することを試みる．HPL について N=400〜6400 の測定値からモデルを実装し，N=3200〜9600 に対して最適な実行方法をモデルから予測した．得られた構成の実行時間は，真に最適な実行時間から 0%〜3.6%の誤差であった．

The Execution Time Estimation Model for Heterogeneous Clusters and Its Evaluation

Yoshinori Kishimoto† and Shuichi Ichikawa†

A heterogeneous cluster can incur the performance degradation caused by the load unbalance in executing the application for homogeneous cluster. The total execution time can be sometimes improved by neglecting some of the PEs because communication time is reduced. This study examines to invoke multiple processes on fast processing elements (PEs) to avoid load unbalance. In this study, the execution time of each PE is firstly modeled from measurement results. Then, the derived model is used to estimate the optimal PE configuration and process configuration. We implemented the models for HPL (N = 400–6400), and estimated the optimal configuration for N = 3200–9600. The error of the estimated execution time was 0%–3.6% of the actual optimal execution time.

1. はじめに

不均一クラスタとは，演算性能・通信速度・メモリ容量など構成や性能が異なる要素プロセッサ (PE) で構成されたクラスタを言う．近年，手近な計算機をよせ集めて一時的にクラスタとしたり，既存のクラスタに最新のプロセッサを追加して増強したいなどの要求が高まっている．

しかし，既存の多くの高性能計算 (HPC) 応用では MPP など PE 性能が均一な環境を想定しており，プロセッサ性能によらず負荷を各分散プロセスに均一に割り当てるため，プロセス間の実行時間不均衡によりピーク性能を発揮できない．応用を不均一環境向けに設計しなおせば不均一クラスタを効率よく利用できるが，蓄積された過去の膨大な資源を捨て去るのは得策ではない．

本研究の目的は，高速な PE 上にプロセスを複数起動することで，既存の応用を修正せずに不均一環境上で適切な負荷分散を行うことである．各プロセッサ上で起動するプロセス数は，クラスタの実行時間を最小化する組合せ最適化問題を解いて求める．

組合せ最適化問題としてモデル化するには，ある構成において各プロセスの実行時間を見積もるモデルが必要である．そこで，アルゴリズムで決定される通信量・計算量のオーダと，実行時間の実測値から，予測モデルを作成する．さらに実際の不均一クラスタに関して，実測された実行時間と予測された実行時間を比較し，本手法により最適〜準最適な構成が得られることを示す．

2. 過去の研究

科学技術計算で多く用いられる行列行列積 (MMM) や LU 分解では，負荷分散手法としてブロックサイクリック分割が用いられることが多い．しかしこの方法は不均一環境では必ずしも最適な手法ではないため，いくつかの代替手法が提案されている．LU 分解では Kalinov ら[1] の Column-based 不均一ブロックサイクリック分割や，Beaumont ら[2] の ScaLAPACK 向け不均一 2 次元グリッド分割などがある．しかし，これ

† 豊橋技術科学大学 知識情報工学系
Department of Knowledge-based Information Engineering, Toyohashi University of Technology

らの不均一ブロックサイクリック分割ではソースの大規模な修正が必要となり，本研究の目的に合わない．また負荷分散の最適化はプロセッサのサイクルタイムに基づいており，真に最適な分散となるか疑問が残る．

笹生ら[3] は HPL (High Performance Linpack)[4] について，ブロックサイクリック分割はそのままに複数の分割ブロックを 1 度に処理する修正を加えた．この修正には通信処理の大幅な変更が必要となるが，プロセス内通信を効率化することでオーバヘッドを軽減することができる．しかし，処理ブロック数の最適化は各 PE (1 CPU) の性能比に基づいており，並列実行時の実行時間最適化は今後の課題としている．

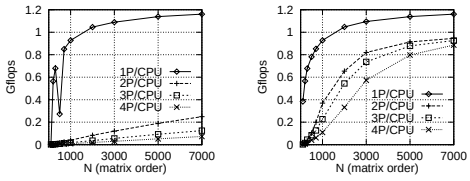
3. マルチプロセス法

不均一クラスタ上で負荷を均衡化する直観的手法として，高速 PE 上に性能に応じた数のプロセスを起動する手法 (以下，**マルチプロセス法**) が考えられる．マルチプロセス法はソースの修正が不要で実現も容易であり，様々な応用に適用可能である．ただし，マルチプロセス法では複数プロセスの起動による実行時のオーバヘッドが問題になる．笹生ら[3] はマルチプロセス法について検討し，性能が低いとして棄却した．しかし，著者ら[5] が HPL に関して測定した結果では，問題サイズが大きい場合のオーバヘッドは実行時間の 2〜3 割程度であった．

単独の Athlon (1.33GHz) 上に複数の MPI プロセスを起動した場合の，HPL の性能測定結果を図 1 に示す．図中，nP/CPU は，Athlon 上に n プロセスを起動して測定したことを示す．MPICH-1.2.1 を使うとピーク性能の 1〜2 割まで性能が低下するが，MPICH-1.2.2 ではピークの 7 割程度の性能が得られることがわかる．netpipe を用いて，同一 PE 上で MPICH-1.2.1 と 1.2.2 の通信性能を測定した結果が図 2 である．16KB ブロック以上の通信性能が大きく異なっていることがわかる．図 2 から，MPICH-1.2.1 での性能低下は同一 PE 上での通信不具合によるもので，この不具合は 1.2.2 では解決されていると考えられる．

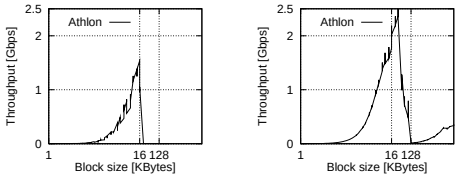
実際に Athon 1.33GHz × 1 ＋ Pentium II 400MHz × 4 (1000base-SX 接続) の不均一クラスタで，HPL の性能を測定した (図 3)．Athlon も Pentium II も各 1 プロセスで実行した場合は，Pentium II × 5 の均一クラスタと同じ性能になってしまう (図 3(a))．この性能は Athlon × 1 と同程度である．一方，Athlon 上に 2〜4 プロセスの負荷をかけると性能が向上し，4 プロセス時 ($N = 10000$) には，ピーク性能 (約 2Gflops) の 85%の性能を発揮した (図 3(b))．

以上の結果から，マルチプロセス法のオーバヘッドは許容可能と考える．以下, 本研究では，マルチプロセス法による最適化手法について，HPL を例として詳しく検討する．



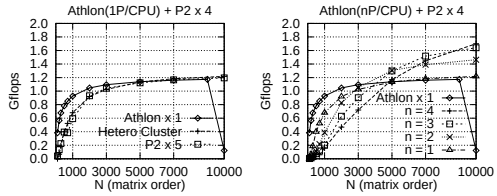
(a) MPICH-1.2.1　(b) MPICH-1.2.2

図 1 MPICH のバージョンによる演算性能の差



(a) MPICH-1.2.1　(b) MPICH-1.2.2

図 2 MPICH のバージョンによる通信性能の差



(a) 負荷不均衡時　(b) 起動プロセス数

図 3 ヘテロクラスタの各パラメータにおける性能

4. 実行時間予測モデルの構築

マルチプロセス法を不均一クラスタに適用するには，(1) 最適な PE 群を選択し，(2) 各 PE 上で起動する最適なプロセス数を決めなければならない．本研究では，この問題を組合せ最適化問題としてモデル化する．モデル化には，与えられた PE 群とプロセス数に対して，その構成の実行時間を予測する式が必要である．

本研究では，クラスタ上で小規模な HPL を実行し，その結果から実行時間の予測モデルを構築する．具体的には，アルゴリズムから計算時間と通信時間の近似式を求め，実測した性能データを最小二乗法で処理して近似式の定数項を求める．

実測値に基づくモデリング技術は多くの工学分野で利用されており，特に奇異なものではない．例えば回路シミュレーションのためのトランジスタモデル[6] では，デバイス物理に基づいた解析モデルに，測定結果に基づくパラメータを導入する場合が多い．本研究では，このようなモデル化を実行時間予測に用いて，組合せ最適化による実行時間最適化に利用する．

本研究で採用する技術 (マルチプロセス法とモデル化) は実装や応用に依存しないため，HPL 以外の幅広

い応用に適用可能である．実測値に基づいたモデルであるため，通信バッファの影響やキャッシュ効率など，システム内の様々な未知のオーバヘッドを内包したモデルを構築できる可能性がある．

4.1 モデルの概略

HPL の問題サイズを N，PE_i 上のプロセス数を M_i，不均一クラスタ内の全プロセス数を $P = \sum M_i$ とする．予測モデルは，N, P, M_i から PE_i 上の各プロセスの実行時間を予測する．本研究では，プロセス格子は横一列に限定して 1 次元ブロックサイクリック分割とした．☆

HPL のアルゴリズムとソースコードを解析して，問題サイズ N とプロセス数 P に対する計算量・通信量のオーダを解析する．ここでは 2 次式，3 次式などの関数の概形が解ればよい．次に，得られた近似式の係数をパラメータフィッティングにより決定する．フィッティングは計算時間 T_a と通信時間 T_c について別々に行い，最終的な実行時間 T は各プロセス i の見積実行時間 T_{ai}, T_{ci} を用いて $T = \max_i(T_{ai} + T_{ci})$ で計算する．

不均一クラスタ内のプロセスの実行時間は通信相手の速度によって影響される可能性があるが，本研究で構築する実行時間予測モデルでは通信相手や通信トポロジーを無視することで単純化を行っている．即ち，不均一クラスタ中の各 PE の計算時間や通信時間が，その PE と等価な PE で構成された均一クラスタ上での計算時間や通信時間に等しいという仮定を行う．また，不均一クラスタ内に複数の同性能 PE が含まれていた場合，同性能 PE 上のマルチプロセス数は同一とすることで，モデルを統一し組合せ数の減少を図る．

同性能 PE をまとめたグループを G_i と表すとしよう．各 G_i 内で使用するプロセッサ数を P_i，それらのマルチプロセス数を M_i で表すと，全プロセス数 P は $P = \sum P_i M_i$ と表される．本研究では，まず，各 G_i ごとに性能を実測して均一クラスタモデルを構築し，次に均一クラスタモデルを補正して不均一クラスタでの実測値に合わせこむ．このようなモデル構築の詳細に関しては，以下の 4.6 節と 4.7 節で述べる．このようなモデル化が妥当であるか否かは，最終的には実測値と比較して検証されなければならない．検証結果については 5 章で述べる．

4.2 HPL 測定結果と計算・通信時間

計算時間 T_a と通信時間 T_c の実測値を得るには，HPL の測定結果からこれらを算出する必要がある．

まず，HPL 測定結果に含まれる実行時間の内訳とその集計処理を図 4 に示す．Total Time はそのプロセスの総実行時間である．rfact はパネル列再帰 LU 分解フェーズの実行時間を表し，パネル LU 分解 pfact と軸交換通信 mxswp を含む．update は更新フェーズを表し，行方向ブロードキャスト通信 laswp を含む．uptrsv は後退代入処理フェーズ,bcast は LU 分解全体を通じて行なわれるブロードキャスト転送を表す．計算終了後，マスタープロセスは各プロセスの実行時間データから All Reduce 演算により各項目の最大値を求めベンチマーク結果として表示する．

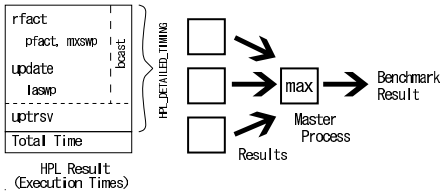


図 4 HPL 測定結果の内容と max による集計

これらの実行時間は通常のベンチマークでは計測されないが，HPL のコンパイル時に HPL_DETAILED_TIMING 定数を定義することにより計測処理が追加される．ただし bcast の測定処理はオリジナルには含まれていなかったため，ソースを修正することで独自に追加した．

以上の調査結果より，本研究では，以下の式で計算時間 T_a と通信時間 T_c を見積もることにする．

$$T_a = (rfact - mxswp) + (update - laswp) + uptrsv$$
$$T_c = mxswp + laswp + bcast$$

4.3 近 似 式

HPL の実行時，問題サイズ N, プロセス格子サイズ $1 \times P$ に対して，通信量と計算量のオーダは以下のようになる．

HPL の各計算処理の計算量は次式で表される[3]．

$$rfact = \frac{3}{2} \cdot N^2 + O(N)$$
$$update = \frac{2N^3}{3P} + \frac{P+1}{P} \cdot O(N^2) + O(N)$$
$$uptrsv = \frac{1}{P} \cdot O(N^2)$$

上記より計算量は問題サイズ N に対して $O(N^3)$ で抑えられるので，予測モデルは 3 次式とする．ただし実測によれば $update$ の実行時間が支配的で，$rfact$ と $uptrsv$ の実行時間は $update$ の 1/100 程度である ($N = 9600$)．そのため，計算量は $update$ の式に基づいて見積もることとする．

各プロセスの通信量は，HPL ソースの通信量に関するコメントから, 次のように概算した．

$$mxswp = O(1)$$
$$laswp = \frac{1}{P} \cdot O(N^2)$$
$$bcast = (P-1) \cdot O(N^2)$$

上記より通信量は問題サイズ N に対して $O(N^2)$ で抑

☆ もちろん他の分割法でもモデル構築は可能である．

えられるので，予測モデルは 2 次式とする．プロセス数 P に対しては P と $\frac{1}{P}$ が 2 次の項にかかっているため，予測モデルは両項を含んだ形とする．

4.4 フィッティング処理

4.3 節の結果から，問題サイズ N に対する PE_i の計算時間 $T_{ai}(N)$ と通信時間 $T_{ci}(N)$ を式 (1)〜(2) のように予測する．添字の P, M_i は特定の構成 $[P, M_i]$ に対するモデルであることを表す．

$$T_{ai}(N)|_{P,M_i} = k_0N^3 + k_1N^2 + k_2N + k_3 \quad (1)$$
$$T_{ci}(N)|_{P,M_i} = k_4N^2 + k_5N + k_6 \quad (2)$$

式に含まれる定数 (k_0〜k_6) は，実行時間の実測値から最小二乗法で決定する．以下，この予測式を N–T 予測モデルと呼ぶ．

このままでは P と M_i の値ごとに別々の N–T 予測モデルが必要になり，モデルの作成と利用が煩雑になる．そこで N–T モデルを統合して，プロセス数 P をパラメータに含むモデルを構築する．4.3 節の結果と N–T 予測モデルから，計算時間 $T_{ai}(N, P)$ と通信時間 $T_{ci}(N, P)$ は，以下の式 (3)〜(4) のように表現できる．先に構築した N–T モデルに色々な問題サイズ N を与えて，この予測式のパラメータ (k_7〜k_{11}) を決定すれば，P を含むモデルが得られる．この予測式を以下 P–T 予測モデルと呼ぶ．添え字の M_i は特定のマルチプロセス数に対するモデルであることを表す．

$$T_{ai}(N, P)|_{M_i} = k_7\frac{T_a(N)|_{P,M_i}}{P} + k_8 \quad (3)$$
$$T_{ci}(N, P)|_{M_i} = k_9P \cdot T_c(N)|_{P,M_i} + k_{10}\frac{1}{P} \cdot T_c(N)|_{P,M_i} + k_{11} \quad (4)$$

4.5 構成によるモデル切り替え

単独 PE_i での実行時 ($P = M_i$) とクラスタ実行時 ($P > M_i$) では，通信の有無など実行過程が大きく異なる．そのため P–T 予測モデルを $P = M_i$ まで含めて構築すると，フィッティング精度が低下してしまう．そこで，$P = M_i$ の時は N–T 予測モデルをそのまま使用し，P–T 予測モデルは $P > M_i$ の N–T 予測モデルから構築することにする (図 5)．

また，マルチプロセス数 M_i を含めた計算時間・通信時間の定式化は，関与する要素が多いため困難である．そこで，マルチプロセス数 M_i ごとに異なる P–T 予測モデルを利用する (図 5)．ちなみに図 5 の×の部分は，$P = \sum P_iM_i$ であるから元々存在しない．

このように，条件によって使用するモデルを切り替える手法は，回路シミュレーションのトランジスタモデリングでは “ビニング” として広く知られている[6]．トランジスタモデルのビニングは，支配的な物理過程が異なる領域に対して異なるモデルを適用するという思想で行われるが，本研究の実行時間モデルでは実行過程 (通信など) が異なる場合に，異なるモデルを適用する．

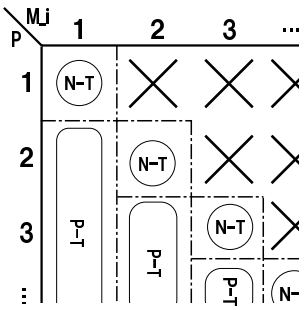


図 5 パラメータによるモデルの切り替え

4.6 モデル構築

モデルの構築は 2 段階に分けて行う．

Step.1 では不均一クラスタ内の同性能 PE をグループ化し，各グループ G_i で均一クラスタモデルを構築する (4.1 節参照)．測定パラメータには問題サイズ N，投入 PE 数 P_i，マルチプロセス数 M_i を適切な範囲で与える．

問題サイズ N を大きな範囲まで測定すればモデル精度は向上するが，その分モデル構築に要する時間が増加するので，要求される精度に応じて範囲を決定すべきである．また，マルチプロセス数 M_i の測定範囲は他種 PE との性能比で決定する．例えば，プロセッサ A が B より 4 倍高速であれば，A のマルチプロセス数 M_A は 1〜5 程度の範囲で測定する．

Step.2 では，Step.1 で作成した均一クラスタモデルと現実 (不均一クラスタ) の不一致を補正する．不一致は主として通信時間に発生するので，モデルの通信誤差を補正するために，不均一な実行環境で時間を実測し，Step.1 で構築したモデル群に適用する補正係数を求める．

実際には，不均一構成で可能な全ての組み合わせ (N, P, M_i) について補正係数を測定することは，組合せが多いため現実的でない．測定サンプル数の削減については，5.3 節であらためて検討する．

各プロセスの予測モデルの出力 (予測実行時間) の max が，クラスタ全体の予測実行時間となる．各モデルへの入力は，全体に共通な問題サイズ N，全プロセス数 P と，モデル別のマルチプロセス数 M_i に分かれる．各モデル内部では P, M_i に対し図 5 のようにフィッティングモデルの切替が行われる．

4.7 モデルの代用

等価な PE 数がモデル式の未知パラメータ数より少ない (例えば 1〜2 個しかない) 場合，データ不足でパラメータ抽出ができない．少数の高速な PE で既存クラスタの増強を図る場合など，実際に多く起こり得る状況である．

この場合，フィッティング可能な他種 PE の P–T 予測モデルで代用する方法が考えられる．最低 1 つの PE が存在すれば $P = 1$，$M = 1$ の N–T 予測モデ

表 1　HPL 実行環境

OS	RedHat Linux7.0J (kernel 2.4.2)
コンパイラ	gcc 2.96, -DHPL_DETAILED_TIMING -fomit-frame-pointer -O3 -funroll-loops -W -Wall
ライブラリ	MPICH–1.2.5, ATLAS 3.2.1

表 2　HPL 測定時のクラスタ構成パラメータ

モデル構築時	N=400〜6400, Athlon(P_1:1,M_1:1〜6), Pentium II (P_2:1〜8,M_2:1〜6)
評価用実測時	N=3200〜9600, Athlon(P_1:0〜1,M_1:1〜6), Pentium II (P_2:0〜8,M_2:1)

ルは作成可能であり，代用元の PE と対象 PE の間で N–T 予測モデルの出力値を比較し二乗誤差が最小となる比を求めることができる．この比をもとに代用元 PE の P–T 予測モデルを定数倍し，対象 PE の代用 P–T 予測モデルを作成することができる．

5. 実行時間予測モデルの評価

Athlon 1.33GHz (プロセッサ A とする) 1 個と Pentium II 400MHz (プロセッサ B とする) 8 個を 100base-TX で接続した不均一クラスタについて実行時間予測モデルを構築し，その出力から実行時間が最小となる最適構成の選択を試みた．HPL の実行環境を表 1，モデル構築時と評価時のクラスタの測定パラメータを表 2 に示す．以下，Athlon の PE 数とマルチプロセス数を M_1, P_1，Pentium II の PE 数とマルチプロセス数を M_2, P_2 とする．

今回の評価では高速なプロセッサ A 側のみマルチプロセス実行を行なう．ピーク性能時の実行時間比がおよそ 1:4 であるので，Athlon のマルチプロセス数 M_1 は 1〜6 の範囲で測定した．

5.1 モデル構築

モデル構築のための測定は，問題サイズ N =400〜6400 の範囲で 9 セット行なった．

プロセッサ A の測定は $P_1 = 1$，$M_1 = 1$〜6 で行い，単独 PE のマルチプロセス数 $M_1 = 1$〜6 に対応する N–T 予測モデルを構築する．クラスタ実行時 ($P_1 \geq 2$) のプロセッサ A の P–T 予測モデルを構築することは，プロセッサ A が 2 つ以上必要となるため不可能である．そこで 4.7 節で述べたスケーリング処理を行って，プロセッサ B の P–T 予測モデルで代用する．

プロセッサ B の測定は $P_2 = 1$〜8，$M_2 = 1$〜6 で行い，単独 PE 時のマルチプロセス数 $M_2 = 1$〜6 に対応する N–T 予測モデル，およびクラスタ実行時 ($P_2 \geq 2$) に対応する P–T 予測モデルを構築する．低速なプロセッサ B ではマルチプロセス実行を行なわないため，本来 $M_2 \geq 2$ の予測モデルは必要ないが，上の段落で述べたとおり，プロセッサ A の P–T モデルを構築するためにプロセッサ B でも P–T モデル構築を行なう．

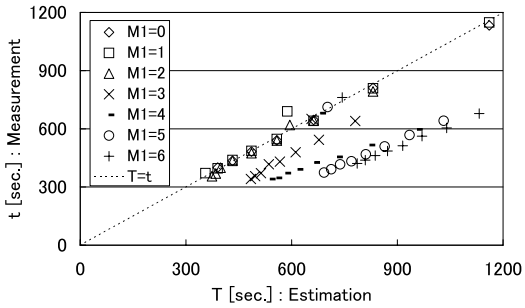


図 6　実測・予測値相関 (モデル修正前)

5.2 モデル評価

構築したモデルを評価するため，問題サイズ $N = 9600$ で，予測モデルの予測実行時間と実測実行時間の相関関係について調べた (図 6)．対角線に近いほど予測値が良く一致していることを表すが，図 6 より，マルチプロセス数 M_1 が 3 以上の場合に誤差が大きくなることがわかる．原因を調べたところ，プロセッサ A 上でマルチプロセス実行時の通信時間が過大に見積もられることが解った．

5.3 通信時間誤差の修正

通信時間に大きな誤差が発生する原因は不均一クラスタモデルの簡略化で通信相手やトポロジーを無視したためと考えられる．

この誤差を補正するため，本研究ではモデル予測値と実測値の逆比をかけることにする．図 6 では各マルチプロセス数 M_1 ごとに強い線形性が見られるため，線上の 1 構成の実測値のみを取得すれば M_1 上の予測値全体の補正が可能である．

今回の評価例では各 M_1 に対して $N = 6400, P_1 = 1, M_1 = 3$〜$6, P_2 = 8, M_2 = 1$ の不均一構成時の実測値をもとに補正を行なった．$M_1 \leq 2$ の範囲については予測値と実測値が良く一致しているため補正は行なわない．また $P_2 = 8$ で補正したのは，Pentium II を全て投入した場合の精度を良くするためであるが，妥当性については検討の余地がある．

上記手法による補正後の相関関係を図 7 に示す．補正前に比べてほとんどの構成が対角線上にあり精度が改善されている．

5.4 予測モデルの精度評価

前節で改良した予測モデルを評価するため，表 2 (下

表 3 N に対する予測値・実測値の最良値と誤差 (モデル修正後)

サイズ	予測による最良構成			実測による最良構成		誤差	
N	P_1, M_1, P_2, M_2	τ	$\hat{T}$	P_1, M_1, P_2, M_2	$\hat{t}$	$(\tau-\hat{t})/\hat{t}$	$(\hat{T}-\hat{t})/\hat{t}$
3200	1,1,0,0	20.0	20.4	1,1,0,0	20.4	-0.019	0.000
4800	1,1,8,1	65.2	64.0	1,1,8,1	64.0	0.019	0.000
6400	1,1,8,1	129.8	129.7	1,2,8,1	125.2	0.037	0.036
9600	1,3,8,1	338.9	341.1	1,4,8,1	340.9	-0.006	0.0006

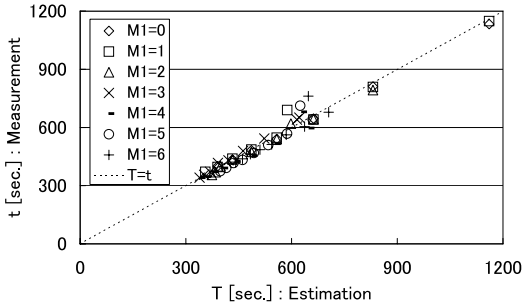


図 7 実測・予測値相関 (モデル修正後)

段) の範囲内で可能な全構成について，予測モデルで実行時間を予測し，さらに実際に実行時間を測定した．

予測モデルで最良とされた構成 (予測最良構成) の予測実行時間を τ，予測最良構成の実測実行時間を $\hat{T}$ とする．さらに，実際に実行時間が最小となる構成 (実測最良構成) の実行時間を $\hat{t}$ とする．

$N = 3200$ では最適構成 (Athlon1 台のみ使用し Pentium II は使わない) の予測に成功している．$N = 4800$ でも最適構成の予測に成功している．一方，$N = 6400$ では実測結果の最適構成より 3.6%遅い準最適構成，$N = 9600$ でも実測結果の最適構成より 0.06%遅い準最適構成であった．

$N = 6400$ では最適構成 $M_1 = 2$ に対して 4.5 秒遅い構成となっているが，これは $M_1 = 2$ の時の実行時間を過大に見積もり最適構成として選択しなかったためである．補正を $M_1 \geq 3$ についてのみ行ったためであろうが，同様の補正を $M_1 = 2$ にも適用するとかえって結果が悪化するため，誤差の原因や補正手法について別途検討する必要がある．$N = 9600$ では最適構成 $M_1 = 4$ に対して 0.2 秒しか違わないのでモデル精度から考えれば誤差の範囲内であろう．いずれにせよ，モデルの精度は実用上充分に高いといえる．

6. おわりに

本研究では，不均一クラスタ上でソース修正を伴わずに HPC 応用を負荷分散するため，マルチプロセス法について検討した．候補となる PE 群およびマルチプロセス数の最適性を判定するため，実測値から実行時間予測モデルを構築し，モデルを使って実際に実行時間最小もしくはそれに近い値となる構成を予測することに成功した．

今回の研究では，最適構成を探すために，不均一クラスタ上で可能な全構成について総当りで実行時間を予測した．しかし，このような素朴な手法では，プロセッサの種類が増加すれば組み合わせ数が爆発する．今後は分枝限定などの手法を取り入れて，探索空間を制限したい．また，準最適構成を求めるための近似解法 (ヒューリスティックなど) についても検討し，その求解時間や精度について評価を進めてゆきたい．

モデルの汎用化と高精度化の両立も，今後の課題である．現在はビニングによって多くのモデルを切り替えて使っているが，モデルの数が多いとモデル構築に時間がかかる．M_i をパラメータに含み，かつ精度の高いモデルが望まれる．

謝辞 本研究の一部は，堀情報科学振興財団・第 11 回研究助成「不均一な分散処理環境のための行列計算高速化手法」による．

参 考 文 献

1) Kalinov, A. and Lastovetsky, A.: Heterogeneous Distribution of Computations while Solving Linear Algebra Problems on Networks of Heterogeneous Computers, *Proc. HPCN Europe 1999*, LNCS 1593, Springer, pp. 191–200 (1999).
2) Beaumont, O., et al.: A Proposal for a Heterogeneous Cluster ScaLAPACK (Dense Linear Solvers), *IEEE Trans. Comput.*, Vol. 50, No. 10, pp. 1052–1070 (2001).
3) 笹生健, 松岡聡, 建部修見: ヘテロなクラスタ環境における並列 LINPACK の最適化, 情処研報 2001–HPC–86, pp. 49–54 (2001).
4) Petitet, A., et al.: HPL – A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers. http://www.netlib.org/benchmark/hpl/.
5) 岸本芳典, 市川周一: 不均一クラスタ上での並列 Linpack の性能に関する検討, 並列処理シンポジウム JSPP2002, pp. 177–178 (2002).
6) Cheng, Y. and Hu, C.: MOSFET のモデリングと BSIM3 ユーザーズガイド, 丸善 (2002).