2004－DSM－32　(6)
2004／3／29

ログ情報に基づく Grid 上での MPI アプリケーションにおけるタスク割り当て手法の提案とその評価

羅 柳[†]　片桐 孝洋[†,††]　本多 弘樹[†]　弓場 敏嗣[†]
[†] 電気通信大学大学院情報システム学研究科
[††] 科学技術振興機構さきがけ

概要

本稿では、Grid 上で MPI アプリケーションの実行時間を最小にするタスク割り当て手法を提案する。提案手法では、ブロック通信方式を用いた MPI アプリケーションを対象とする。MPI アプリケーションの計算・通信特性と Grid 計算環境の性能を考慮し、対象の MPI アプリケーションの実行時間を最小にするタスク割り当てを決める。擬似 Grid 環境上で実験を行った結果、提案するタスク割り当て手法を用いることで最適なタスク割り当てを選ぶ誤差が 10%以内に収まることを確認した。

A Task Allocation Method for MPI Applications Based on Logged Information on Grid and Its Evaluation

Liu Luo[†], Takahiro Katagiri[†,††], Hiroki Honda[†], and Toshitsugu Yuba[†]
[†] Graduate School of Information Systems, The University of Electro-Communications
[††] Japan Science and Technology Agency, PRESTO

Abstract

In this paper, we propose a new allocation method for MPI programs, which can determine the best task allocation to nodes. With the method, the execution time for MPI programs can be shortened on Grid. The method is designed for the MPI programs with blocking communications. The method determines the best task allocation to the nodes after analyzing the specification of the MPI programs and the performance on Grid. To test the method, a pseudo Grid environment is built by using actual machines. As a result, the rate for the allocation to the nodes by using the proposed method were less than 10%.

1. はじめに

近年のネットワーク技術の発展により、世界中に分散した計算資源を広域ネットワークで接続し、１つの高性能な計算基盤として提供する計算環境が注目を集めている。この計算環境は Grid と呼ばれ、現在、Grid 計算環境に関する研究開発が盛んに行われている[1)~3)]。

Grid に関する研究の中でも、スケジューリングに関する研究は注目を集めており、特にアプリケーションスケジューリングが極めて重要な課題となる。アプリケーションスケジューリングは、単一並列アプリケーションの実行時間を短縮させることを目的として、アプリケーションにおけるデータ分割、通信パターンなどの特性と、Grid 計算システムの性能特性の両方を考慮してスケジューリングを行う[4)]。

大規模な科学技術計算アプリケーションを開発する並列プログラミング言語としては MPI がよく利用されている。MPI は各種の分散メモリ型並列計算機上で最大の性能を引き出すように設計・実装され、広く利用されている。

従来、均質計算環境（ホモジニアス）向きの均等データ分割を用いた MPI プログラムの性能評価が盛んに行われてきた。しかし、不均質（ヘテロジニアス）計算環境向きの、不均等なデータ分割を用いた MPI アプリケーション（プログラム）については、性能評価がまだ不十分である。そこで不均等なデータ分散を用いた MPI プログラムを本研究のターゲットとする。

本論文では、MPI プログラムで利用されているデータ分割、通信パターンなどのアプリケーションの特性、および Grid 計算環境での計算・通信性能を考慮し、MPI プログラムの実行時間を最小にするタスク割り当て手法である **LSMG (Log based Scheduling for MPI programs in Grid)** を提案する。この手法は Launch-time Scheduling のように、MPI プログラムの起動時に各ノードへの最適なタスク割り当てを

決める静的な手法である。

2. タスク割り当て問題

MPI プログラムの通信方式には大きく分けて、ブロック通信、ノンブロック通信という 2 種類の通信方式がある。ブロック通信方式では、各タスクにおいて同時に計算と通信を行うことができない。また複数の送受信イベントがある場合、送受信を順番に行う。ブロック通信方式をもつ MPI プログラムの各タスクの処理時間は算出しやすいため、本研究ではブロック通信方式を利用しており、通信インタフェースは MPI_Send と MPI_Recv だけを利用する MPI プログラムを対象とする。

MPI プログラムを実行する時、指定すべき重要な情報は machinefile というファイルに記述される。この machinefile は MPI 起動時において、生成されるタスクを、どのように計算資源（計算ノード）に割り当てるかの情報を記述する。ここで、MPI プログラムを起動した時に、各ノードに割り当てる処理対象を**タスク**と呼ぶ。また各タスクにおいて、計算や通信処理などの事象の発生から終了までの処理を**イベント**とよぶ。

従来の均質な計算環境向きの均等データ分割を用いた MPI プログラムでは、machinefile に記載されているタスクの割り当てを変えても、MPI プログラムの実行時間は変わらないと推定される。それに対して、不均質な計算環境向きの不均等データ分割方式を用いた MPI プログラムの実行時間は、タスクの割り当てにより実行時間が大きく変動すると推察される。したがって、どのようにタスクをノードに割り当てれば MPI プログラムの実行時間を最小にできるかが、問題となる。

3. 最適なタスク割り当て手法 LSMG

3.1. LSMG の処理手順

LSMG では、対象プログラムは入力データのみ変更されるが、プログラム自体は変更されずに Grid 上で複数回実行される状況を想定する。そこでまず、均質な計算環境上で MPI プログラムを実行し、通信パターンを記録するログファイルを生成する。次に Grid 計算環境情報（各ノードの計算性能と各通信線路の通信性能）を収集する。次に全探索ですべてのタスク割り当てを調べる。それぞれのタスク割り当て方法による MPI プログラムの実行時間を、収集された情報を基にして予測する。その中で実行時間を最小にするタスク割り当てを見つけ、最後にその結果を machinefile に書き込む。

2.2. LSMG の処理手順

図 1 に LSMG の処理手順を示す。

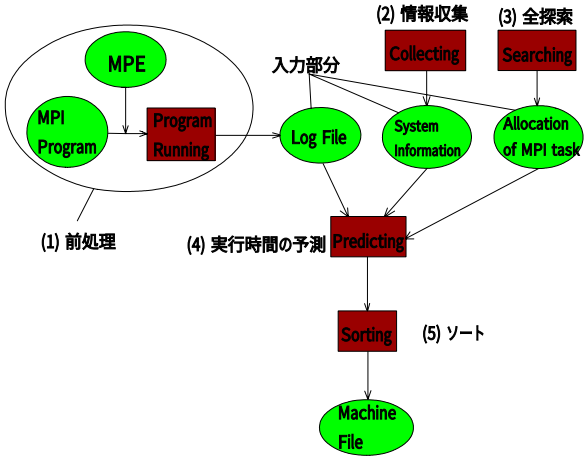

図 1　LSMG の処理手順

図 1 において、楕円は入力ファイル、リンクするライブラリ、および出力ファイルを示す。長方形は処理の内容を示す。詳細は以下の通りである。

(1) **前処理**: MPE(MPI Parallel Environment)[5] というライブラリを呼び出し、MPI プログラムのログファイルをとる。

(2) **情報収集**：非均質な計算環境情報を調べる。具体的には、計算環境の各ノードの計算性能とノード間の通信回線の通信性能を測定する。以降 3.2 節で詳しい測定方法を述べる。

(3) **全探索**：全てのタスク割り当てを探す。

(4) **実行時間予測**: (1)と(2)による情報に基づき、(3)の情報をもとに、それぞれのタスク割り当てによる MPI プログラムの実行時間を、イベント時間を操作することで予測する。

(5) **ソート**：(4)の結果に基づき、全ての実行時間をソートする。そのなかで最小のものを見つける

以下に手順 1、手順 4 の詳細について述べる。

3.2. 前処理

前処理では MPE ライブラリを呼び出し、MPI プログラムの計算、通信パターンを記録するログファイルを生成する。MPE によって生成されたログファイルの形式は 3 種類あり、それらは CLOG 形式、SLOG 形式、および ALOG 形式である。本研究では ALOG 形式のログファイルを利用する。ALOG ファイルはテキスト形式で送受信状況を記録するファイルである。ALOG ファイルは一行ずつ発生したイベントを数字で記録する。行と行の間の処理、すなわちイベントは、計算処理か、OS のオーバヘッドと考える。表 1 は、ある処理における 2 タスク間の送受信を記録する ALOG 形式のログファイルの各行の内容と意味を示す。

表1　ALOG 形式のログファイルの例

行番号	各行の内容（イベント）と意味
(1)	155 1 0 1 0 1189 1189μs に task1 が MPI_Recv 関数を呼び始める。
(2)	161 0 0 1 0 1335 1335μs に task0 が MPI_Send 関数を呼び始める。
(3)	-101 0 0 1 0 1 1395 0 7992 1395μs に task0 から task1 への送信が始まる．データサイズが 999byte である。
(4)	162 0 0 2 0 1 1575 1575μs に task0 で MPI_Send 関数の呼び出しが終わる。
(5)	-102 1 0 0 0 1 2158 0 7992 2158μs に task0 から task1 への通信が終わる。データサイズが 999byte である。
(6)	156 1 0 2 0 1 2198 2198μs に task1 で MPI_Recv 関数の呼び出しが終わる。

3.3.　実行時間の予測

実行時間の予測は重要な処理である。予測手順を以下に示す。

(1)　ALOG 形式のログファイルと Grid 計算環境情報を読み込む。

(2)　タスクの割り当てによるタスクと計算ノードの対応関係に関する情報を読み込む。

(3)　Grid の各ノードの計算性能に基づいて計算時間を予測する。

(3a)　(1a)のイベントをタスク番号によって分ける。

(3b)　各タスクの計算イベントを探す。そのタスクと対応する Grid のノードの計算性能に基づいて、計算時間を予測する。

(3c)　(1a)のイベントの順番に従って(3b)の算出した結果を集める。

(4)　Grid の各通信線路の通信性能に基づき、通信時間を予測する。

(4a)　(3c)で処理したイベントに基づいて MPI_Send()イベントを探す。見つけたら、MPI_Send() の発行時間 $time_start、送信側のタスク番号、受信側のタスク番号、送信データサイズなどの情報を取る。以上の情報に基づいて Grid の各通信線路の通信性能を利用して通信時間 T_comm を算出する。

(4b)　(3c)の処理したイベントに基づいて(4a)で見つけた MPI_Send()イベントとペアになる MPI_Recv()イベントを探す。見つけたら、MPI_Recv()の終わる時間$time_end の情報をとる。$time_end と(4a)の$time_start の差 t_comm を算出する。

(4c)　(4a)の T_comm と(4b)の t_comm の差 gap_time を算出する。

(4d)　(4b)で見つけた MPI_Recv()イベントに基づき、発生順番が後ろになり、かつその MPI_Recv()と同じタスク番号をもつイベントに(4c)の gap_time を加算する。

(4e)　(3c)で処理したイベントの中で、最後の MPI_Send()イベントまで(4a)から(4d)までの処理を繰り返して行う。

4.　評価環境

4.1.　擬似 Grid 計算環境の構成

本研究では、Grid Testbed の代用として擬似 Grid 計算環境を評価環境とした。この擬似 Grid 計算環境は性能仕様が同じである 4 つの PC（ノード）からなり、構成は以下の通りである。

(1)　各ノードに計算負荷プログラムをかけ、各ノードの計算性能を不均質にする。

(2)　ネットワーク制御技術（Linux QoS サポートカーネルの TBF queue を利用する方法）を用いて、ノード間の通信回線の通信性能を異なるようにする。

(3)　2 種類の擬似 Grid 計算環境を構築した。すなわち、スイッチングハブを用いた結合網の実験環境と NIC を直接接続した結合網の実験環境であり、それぞれを図2と図3に示す。

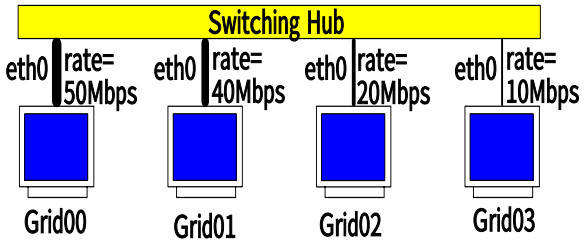

図2　スイッチングハブを用いた結合網の実験環境

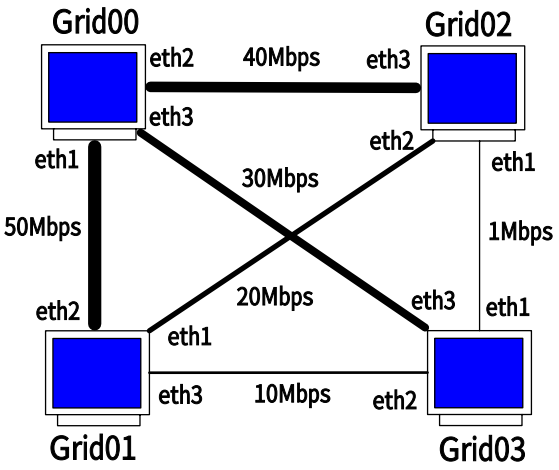

図3　NIC を直接接続した結合網の実験環境

4.2.　擬似 Grid 環境の計算・通信性能の設定

各ノードの計算性能の設定法を以下に示す。

(1) ノードに対して、計算負荷プログラムをかける前に 1 回、ベンチマークでそのベンチマークの実行時間、t1 を測定する。

(2) 計算負荷プログラムをかけた後もう 1 回同じベンチマークの実行時間 t2 を測定する。

(3) t1 の値と t2 の値の比を算出し、それを性能劣化比とする。

ノード間の通信性能の測定方法を以下に示す。

MPI ベンチマークで 2 ノード間の通信性能を測定する。送信データ長は 0～1000 までで、25 ずつ長さを増加させる。1 回の測定では 5000 回測定し、その平均値を送信時間とした。測定した複数のデータについて、以下の線形 1 次多項式に近似したパラメータ a0、a1 を最小 2 乗法で決定する。

$$T = a0 + a1 * N \qquad (1)$$

ここで、変数は以下の通りである：

T：送信時間、a0, a1：係数、N：送信データの長さ。

5.　評価実験

5.1.　評価基準

提案手法が妥当か否か評価するために、擬似 Grid 計算環境上で実測した実験データを正解と仮定し利用する。LSMG の有効性を評価するための評価基準は、以下の通りである。

$$a = |(t_n - t_m)| / t_m \qquad (2)$$

ここで、

a: 最適なタスクの割り当てを選ぶ誤差、

n: 予測した最小の実行時間を与えるタスク割り当ての番号、

m: 実際の最小の実行時間を与えるタスク割り当ての番号、

t_n: 番号 n の割り当てによる実験環境上での実測時間、

t_m　番号 m の割り当てによる実験環境上での実測時間、

である。すなわち誤差 a の値が十分小さければ、提案した手法、LSMG の有効性があると判定する。

5.2　LU 分解プログラムによる評価

5.2.1　LU 分解プログラムのデータ分割

本研究では、ガウス法外積形式の LU 分解ベンチマークを用いて、図 2 と図 3 で示した実験環境上で性能評価実験を行った。各通信回線の通信パラメータは図 2 と図 3 の通りである。また各ノードの負荷プログラム数は、Grid00 に 0 個、Grid01 に 1 個、Grid02 に 3 個、Grid03 に 3 個とした。なお、各ノードの性能劣化比も負荷プログラム数に比例していることを確認した。問題サイズは 1000 である。本実験では、LU 分解プログラムで利用するデータ分割方式について、3 種類の方式を用いる。それらはブロック・サイクリック分割（幅は 5）、ブロック分割、および不均等ブロック分割である。ここで不均等ブロック分割における task0, task1, task2, task3 が処理するデータ量の比は、4:2:1:1 とした。

5.2.2.　LU 分解プログラムの実験結果

それぞれのデータ分割による実験結果を、図 4～図 9 に示す。図 4～図 9 においては、縦軸は時間（秒）で、横軸はタスク割り当てに関する対応関係である。この対応関係とは、タスク番号と Grid 環境の計算ノード番号と割り当てを意味している。4 ノードを使う場合、この対応関係は 24 個存在する。

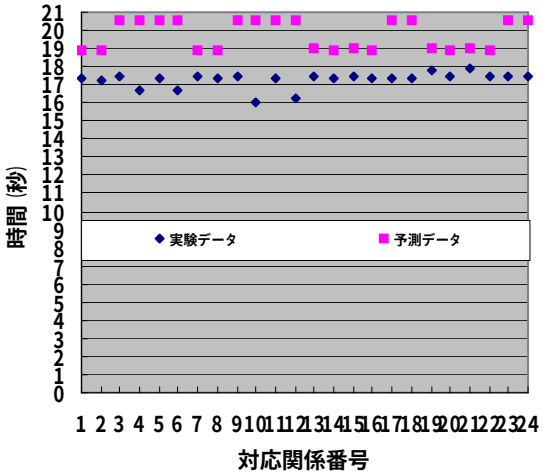

図4　スイッチングハブを用いた結合網実験環境上でのブロック・サイクリック分割方式での結果

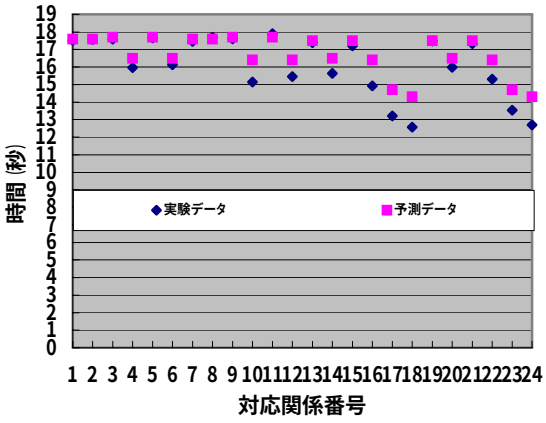

図 5　スイッチングハブを用いた結合網実験環境上での
ブロック分割方式の結果

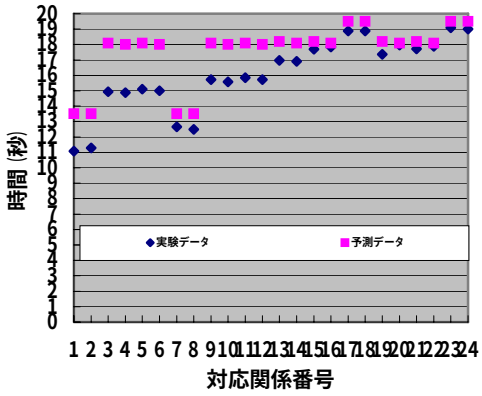

図 6　スイッチングハブを用いた結合網実験環境上での
不均等ブロック分割方式の結果

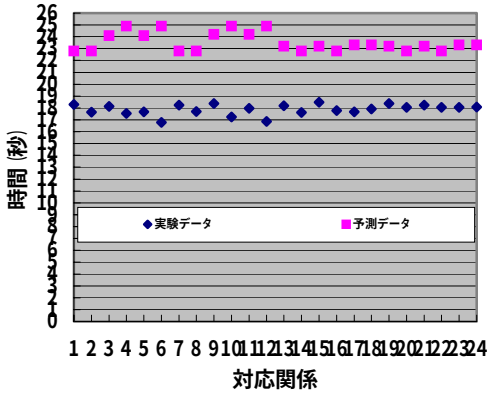

図 7　NIC を直接接続した結合網実験環境上での
ブロック・サイクリック分割方式の結果

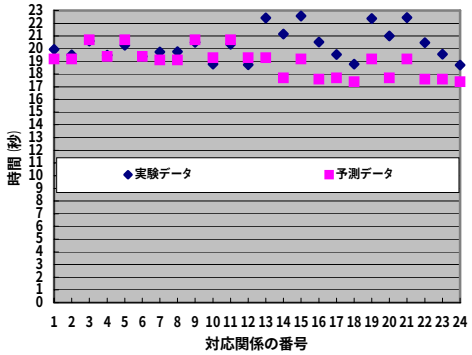

図 8　NIC を直接接続した結合網実験環境上で
のブロック分割方式の実験結果

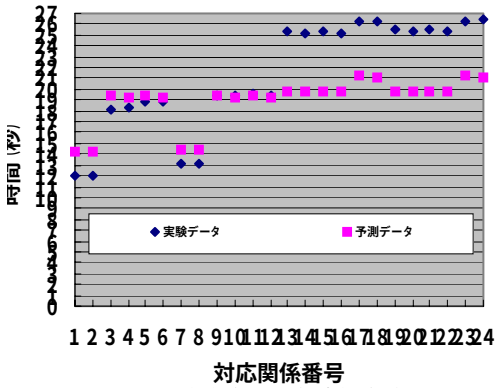

図 9　NIC を直接接続した結合網実験環境上での
不均等ブロック分割方式の結果

5.2.3.　結果分析

5.1 節の式（2）で述べた評価基準を用いた実験結果の誤差 a の値を、以下の表 2 に示す。

表 2　LU プログラムにおける予測誤差

データ分割方式	スイッチングハブを用いた結合網の実験環境	NIC を直接接続した結合網の実験環境
ブロック・サイクリック分割	誤差 a=9.1%	誤差 a=8.9%
ブロック分割	誤差 a=0.8%	誤差 a=0.5%
不均等ブロック分割	誤差 a=2.1%	誤差 a=0.8%

5.3　行列掛け算プログラムによる評価

ここでは、問題サイズが 1000 である行列掛け算プログラムを用いて評価実験を行った。この行列掛け算のデータ分割は不均等ブロック分割で、task0, task1, task2, task3 の各タスクが有するデータ量の比は、4:2:1:1 とした。またここでは、図 2 のスイッチングハブを用いた結合網の実験環境のみで評価実験を行った。各通信回線の通信パラメータは図 2 の通りである。各ノードの負荷数は、Grid00 に 0 個、Grid01 に 1 個、Grid02 に 3 個、および Grid03 に 3 個実行させた。実験結果を以下の図 10 に示す。

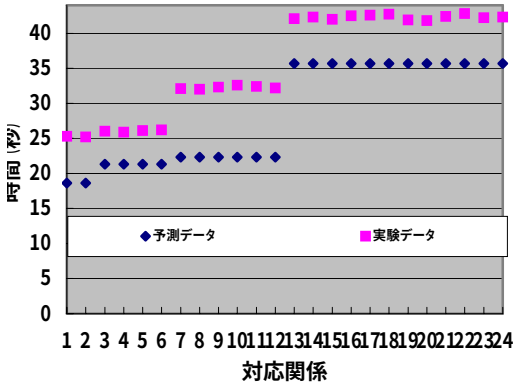

図 10　スイッチングハブを用いた結合網実験環境上での
行列掛け算の実験結果

式(2)の評価基準による誤差は以下の通りである。

評価基準による誤差 a＝0.4%

5.4 姫野ベンチマークによる評価

ここでは、姫野ベンチマーク[6]を用いて、提案手法の評価を行う。姫野ベンチマークのデータ分割は均等データ分割であり、各タスクが所有するデータの量は、ほぼ均等となる。ここでは図 2 のスイッチングハブを用いた結合網の実験環境で評価実験を行った。また各ノードの負荷プログラム数は、Grid00 に 0 個、Grid01 に 1 個、Grid02 に 3 個、Grid03 に 3 個とした。実験結果を以下の図 11 に示す。

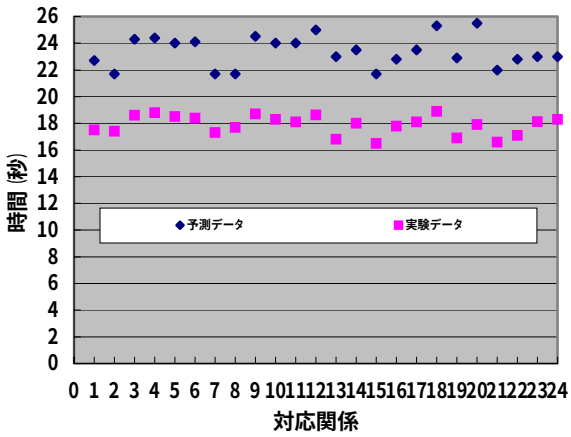

図 11　スイッチングハブを用いた結合網環境上での結果

式（2）の評価基準による誤差は以下の通りである。

評価基準による誤差 a=5.3%

5.5 考察

LSMG の有効性おける考察を、以下に列挙する。

(1) 全ての実験結果に対して、予測誤差 a の値は 10% 以内に収まる。ゆえに式（2）の誤差の観点では、本手法は妥当な手法であると言える。

(2) ブロック・サイクリック分割方式を用いた MPI プログラムでは、各タスクの処理量（計算量と通信量）はほぼ同じとなる。このような MPI プログラムに対しては、どのような方法でタスクの割り当てを変えても、プログラムの実行時間が変わらない。したがって、ロードバランスを取ったデータ分割方式の MPI プログラムに対しては、LSMG は有効でない。

(3) 各タスクの処理量が大きく異なる MPI プログラムでは、タスクの割り当てを変えると実行時間が大きく変動した。このような MPI プログラムに対しては、10%以内の誤差による実行時間は十分小さくなるため、本研究で提案した LSMG は有効となる。

6 おわりに

本論文では、Grid 計算環境の性能（各ノードの通信性能、およびノード間の通信性能）と MPI プログラムの特性（データ分割方式に依存する計算量の偏り、および通信パターン）を考慮し、最適なタスク割り当てを決める手法 LSMG を提案した。擬似 Grid 計算環境上での実験結果では、各タスクの処理量が大きく異なる MPI プログラムに対しては、式（2）による予測誤差 a が十分小さい（10%以内）ため、LSMG の有効性があるといえる。

今後の課題は以下の通りである。

(1) より複雑な通信パターンをもつ MPI プログラムに対する LSMG の評価

(2) ノンブロッキング通信方式を用いた MPI プログラムに対する LSMG の枠組みの再構築と評価

(3) 擬似 Grid 計算環境ではなく、Grid Testbed 上での LSMG の評価

(4) より現実の Grid と合うような擬似 Grid 計算環境上のパラメータ設定と、そのパラメータ用いた LSMG の評価

(5) ノード数が 8 台、16 台と増加した場合の実験環境の構築と、LSMG の評価

(6) 予測手法自体の高速化

本提案手法では全探索を行っているため、最適なタスク割り当て決定のための計算量が多く、その結果、最適化時間がオーバヘッドとなる。効率が良い最適化手法の開発、および最適化所要時間の高速化は、LSMG の実用に向けた重要な今後の課題となる。

参考文献

1) 中田秀基、高木浩光、松岡聡、長嶋雲兵、佐藤三久、関口智嗣: Ninf による広域分散並列計算、並列処理シンポジウム JSPP ’97 論文集, Vol.97, No.3, pp.281--288 (1997).

2) Globus Project: http://www.globus.org/.

3) Conder: http://www.cs.wisc.edu/condor/.

4) AppLeS: Application-Level Scheduling, http://apples.ucsd.edu/.

5) Performance Visualization for Parallel Programs: http://www-unix.mcs.anl/gov/perfvis/.

6) 姫野ベンチマーク: http://w3cic.riken.go.jp/HPC/HimenoBMT/index.html