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

2005－ARC－162 （30）
2005－HPC－101 （30）
2005／3／9

MPIETE2：MPI プログラム実行時間予測ツール MPIETE の通信予測誤差に関する改良

岩渕 寿寛† 杉田 秀‡ 山名 早人‡

概要

本稿では，MPI プログラム実行時間予測ツール MPI Execution Time Estimator（MPIETE）の，通信予測手法の改善案を提案する．MPIETE は，プログラムを計算ブロック，通信ブロックに分割し，各ブロックの実行時間情報からプログラム全体の実行時間を予測する．予測の際，各ブロックの実行時間情報の取得を高速に行うことで，実際の実行よりも速く実行時間予測が可能である．しかし，MPIETE は，通信コンテンション等が発生しない，理想的な環境での予測を前提としていたため，通信コンテンションの発生に伴う通信の待ち時間を予測することが不可能であり，通信時間予測誤差に問題がある．本稿では，MPIETE の通信予測手法を変更し，通信の待ち時間を考慮したブロック実行時間情報を導入する．本稿で示す提案手法は，通信の待ち時間による通信性能の低下を予測し，予測対象計算機上で最大の実行性能を示す Processing Unit(PU)数の特定を可能とするものである．また我々は，本手法を MPIETE に適用し，MPIETE2 を開発した．本稿では，MPIETE2 を用いて，NAS Parallel Benchmarks（NPB） ver2.4 の実行時間の予測を行った結果も示す．NPB2.4 の EP，CG，FT，MG の CLASS B の実行時間を予測したところ，2-128PU の実行で予測誤差は 14%以内であり，予測に必要な処理時間は，実際の実行と比較して約 1/4 の時間であった．特に，すべてのプログラムにおいて，通信性能の低下を予測でき，実際の実行よりも速く，最大の実行性能を示す PU 数を特定できた．

MPIETE2 : Improvement of the MPI Execution Time Estimator in Prediction Error of Communication Time

Toshihiro IWABUCHI†, Shu SUGITA‡, Hayato YAMANA‡

Abstract In this paper, we improve the MPI Execution Time Estimator (MPIETE) to reduce the prediction error of communication time. MPIETE we have proposed is the execution time estimation tool for MPI programs. MPIETE's scheme divides a MPI program into the computation blocks and the communication blocks, and then predicts the total execution time by summing the execution time of each block. Since estimating the block execution time is fast, MPIETE enables to predict the total execution time faster than executing MPI program actually. However, MPIETE assumes no network contension. This results in some errors to predict the delay-time with network contentions. In this paper, by proposing the new estimation scheme for communication block including the delay-time, we improve the MPIETE. The proposed scheme enables to predict the performance decrement and to find out the number of the Processing Unit (PU) where the target platform marks the best performance. We have evaluated MPIETE2 ,that improves MPIETE with the proposed scheme, using EP, CG, FT, MG from NAS Parallel Benchmmarks 2.4. As the results for 2-128PU, the prediction error ranges are less than 14% and the execution time of the prediction is 1/4 times smaller than the actual execution time. Moreover, MPIETE2 predicts exactly the number of PU where the target platform marks the best performance.

1. はじめに

本稿では，並列プログラムの実行時間予測手法について述べる．一般に並列プログラムは，Processing Unit（PU）数を変化させ実行させることができ，ユーザは実行 PU 数によって様々な結果を得ることができる．しかし，並列プログラムは，PU 数を増やして実行しても，期待する実行性能を得られない場合があり，ある実行 PU 数以降から実行性能は低下していくのが普通である．つまり，並列プログラムの性能向上には，プログラムの特徴を見極め，適切な PU 数で実行することが重要となる．

並列プログラムの実行時間予測は，プログラムの実行に先立ち実行時間予測し，最大の実行性能を示す PU 数での実行をすることで，並列プログラム自身の性能向上を実現する．また，予測により余剰 PU が存在した場合，それらを他の計算に利用し，結果的に並列計算機の稼働率を向上させる有用性もある．

従来，MPI プログラム[1]の実行時間の予測手法として，EXCIT&INSPIRE[2]や，LAPSE[3]，MPI-SIM[4]といったシミュレータを用いた手法が提案されている．[2]の手法では，EXCIT を用いて，アセンブラコードレベルのトレースを行い，計算時間の予測を行う．また INSPIRE を用いて，ネットワー

†早稲田大学大学院理工学研究科
Graduate School of Science and Engineering, Waseda University
‡早稲田大学理工学部
School of Science and Engineering, Waseda University

クシミュレータを生成し，通信時間の予測を行う．キャッシュヒット率の予測や，ネットワークのレイテンシ・バンド幅が変化した際の通信時間を予測することが可能である．また，LAPSE[3], MPI-SIM[4]のシミュレータを使用した予測手法では，プログラムの各ブロック実行時間情報からプログラム全体の実行時間を予測する．計算ブロック実行時間情報から，通信関数を呼ぶタイミングを予測することで，非同期な通信の実行時間を予測することが可能である．しかしいずれの予測手法も，実際の実行時間に対して，数倍から数十倍の処理時間を必要とし，実行に日単位を必要とする大規模プログラムの実行時間予測には適さない．

上記した問題に対して，我々は[8]で，実際の実行よりも速く実行時間予測を行う手法を提案し，手法を自動化する MPIETE を開発した．MPIETE では，MPI プログラムを計算ブロック，通信ブロックに分割し，予測対象計算機上で実行することで得る各ブロック実行時間情報から，プログラム全体の実行時間を予測する．予測の際，ブロック実行時間情報の取得を高速に行うことで，実際の実行時間よりも少ない処理時間で予測を実現するものである．NPB2.4 の実行時間予測を 2-128PU 上で行った結果，予測誤差は 16PU までは 14%以内であったが，32PU 以上では 30%以上あり，通信時間予測誤差が問題となった．

本稿では，[8]の通信予測時間誤差の改善に関する手法の提案を行う．本手法は，MPIETE では予測不可能であった，通信性能の低下を予測し，最大の実行性能を示す PU 数の予測を可能とするものである．本稿ではこれより，まず 2 節で提案手法を，3 節で提案手法を MPIETE に実装した MPIETE2 を示す．続いて，4 節で NPB2.4 を用いた提案手法の評価結果を示し，最後に 5 節でまとめを行う．

2. 提案手法

MPIETE[8]では，予測の際に通信コンテンション等，通信の待ち時間を発生させる要因のない，理想的な環境での予測が前提であり，NPB を用いた評価では通信予測誤差が問題となった．本稿では，MPIETE の通信予測方法を改良し，通信性能の低下が起こった場合でも，最大の実行性能を示す PU 数を予測できる手法を提案する．なお，本稿で提案する手法は通信予測の改良手法であるが，予測の全体像を示すため，本節では通信予測手法以外も併せて示す．

本手法は，並列プログラムを「計算ブロック」，「通信ブロック」に分割し，各ブロックの 1 回分の実行時間情報（以降，ブロック基礎データ）とブロック実行回数を求めることで実行時間を予測するものである．本節ではこれより，プログラムのブロック分割方法，計算ブロック・通信ブロック基礎データの取得方法，ブロック実行回数の取得方法を示す．

2.1 ブロック分割方法

本手法では，予測対象プログラムを，以下のように計算ブロック，通信ブロックに分割する．

- MPI 通信関数が宣言されている文を通信ブロックとする
- 通信ブロックや，プログラムの制御に関わる文[1]以外の，連続した実行文の集合を計算ブロックとする

図 2 に，図 1 のプログラムをブロック分割した例に示す．

[1] 繰り返し文（DO，DOWHILE，ENDDO 文），条件分岐文（IF，ELSE，ELSEIF，ENDIF 文），END 文，GOTO 文，STOP 文，RETURN 文

```
a =1
b =10
nloop1 = 100
nloop2 = 1000
nloop3 = 10000
size = 100
DO i = 1, nloop1
  DO j = 1, nloop2
    DO k = 1, nloop3
      x = x + 10
      y = y * 10
    END DO
  END DO
  MPI_BCAST(x, size, mpi_int, ･･･)
  a = a + b
  DO k = 1, nloop3
    z = z * a
  END DO
END DO
```

図 1 プログラム例

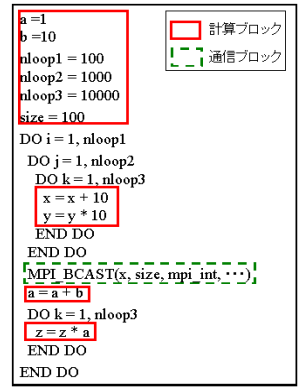

図 2 ブロック分割例

2.2 計算ブロック基礎データ取得方法

計算ブロック基礎データとは，計算ブロック 1 回分の実行時間であり，計算ブロック基礎データは，予測対象プログラムを予測対象計算機上で実際に実行し，予測の際に実行した，計算ブロックの総実行時間と総実行回数から算出する．ただし，コンパイラ最適化やキャッシュ効果を考慮した基礎データを得るため，計算ブロック基礎データは，当該計算ブロックのみをループボディとしてもつ，ループ全体の実行時間を計測した後に求める．

また，計算ブロック基礎データ取得を簡略化するため，予測対象プログラムに以下の処理を施す．

(1) 受信データが計算ブロックの計算順序を変化させない限り，通信ブロックをコメントアウト
(2) 計算ブロック基礎データ測定対象外のループ繰り返し回数を削減

(2)で述べた繰り返し回数の削減に関しては，キャッシュ効果を考慮すると，ある程度ループを回す必要がある．本稿では，繰り返し回数を 1/10 に削減した．

図 1 のプログラムの場合，上記した処理を施したプログラムは図 3 の様になる．実線部分が基礎データを取得するための測定範囲であり，(1)に該当する通信ブロックをコメントアウト，(2)に該当するループの実行回数を削減する．

そして得られたプログラムを予測対象計算機上で実行し，計算ブロック基礎データを高速に取得する．

2.3 通信ブロック基礎データ取得方法

本節で述べる通信時間の予測方法が MPIETE[8]との相違点であり，前述した問題の改善手法である．MPIETE では，通信ブロック基礎データは，予測対象計算機ネットワーク構成における，通信サイズに比例した通信時間であった．この基礎データと通信トレースファイルから通信時間を予測していくが，通信コンテンションの発生に伴う通信の待ち時間を予測できず，通信性能の低下を特定できない．そこで提案手法では，以下のように通信ブロック基礎データ定義を変更し，問題の解決を図る．

本稿での通信ブロック基礎データとは，通信ブロック 1 回分の実行時間であり，通信ブロック基礎データは，予測対象プログラムを予測対象計算機上で実際に実行し，予測の際に実行した，通信ブロック総実行時間と総実行回数から算出する．

ここで，計算ブロックと同様，基礎データを高速に取得するため，以下の処理を施す．

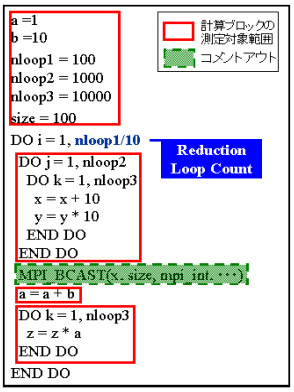

図 3 計算ブロック基礎データ取得プログラム

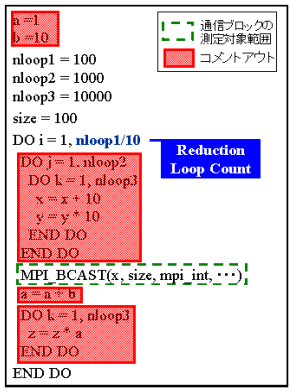

図 4 通信ブロック基礎データ取得プログラム

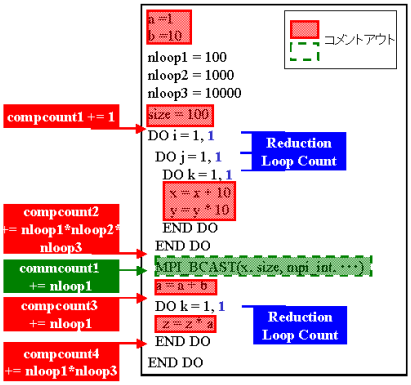

図 5 ブロック実行回数取得プログラム

(1) 通信ブロック基礎データを変化させない実行文のコメントアウト

(2) プログラム全体の通信順序を変化させない限り，ループ繰り返し回数を削減

(1)で述べた基礎データを変化させる実行文とは，

- 通信データサイズ，通信プロセス ID，メッセージ ID を決定する文
- IF 文等の，通信順序を決定する文

であり，これ以外をコメントアウトする．(2)で述べた繰り返し回数の削減に関しては，通信コンテンションの発生を考慮すると，ある程度ループを回す必要がある．本稿では，繰り返し回数を 1/10 に削減した．

図 1 のプログラムの場合，上記した処理を施したプログラムは図 4 の様になる．実線部分が基礎データを取得するための測定範囲であり，(1)に該当する実行文をコメントアウト，(2)に該当するループの実行回数を削減する．

そして得られたプログラムを予測対象計算機上で実行し，通信の待ち時間を考慮に入れた通信ブロック基礎データを高速に取得する．

2.4 ブロック実行回数取得方法

ブロック実行回数とは，実際に各ブロックが実行される回数である．ブロック実行回数も基礎データ同様，高速に取得するため，以下の処理を施す．

(1) ブロック実行回数，実行順序を変化させない実行文のコメントアウト

(2) ループボディにブロック実行回数，実行順序を決定する文がない限り，繰り返し回数の削減

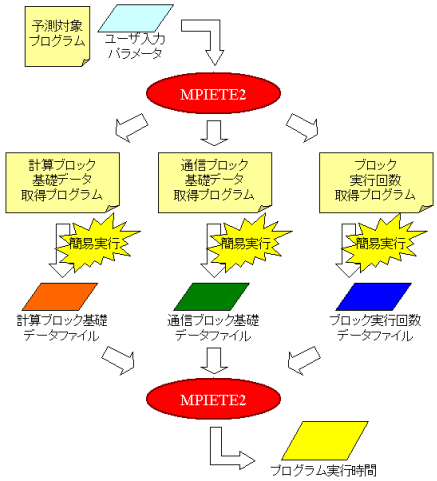

図 6 MPIETE2 の実行時間予測の流れ

(2)の繰り返し回数は，本稿では 1 回に削減した．

図 1 のプログラムの場合，上記した処理を施したプログラムは図 5 の様になる．

2.5 プログラム全体の実行時間

プログラム全体の実行時間は，上記した方法で得られた各ブロックの基礎データにブロック実行回数を掛け合わせ，総和を取ることで算出する．

3. MPIETE2

2 節で示した提案手法を MPIETE に組み込み，開発した MPIETE2 について述べる．

3.1 MPIETE2 の予測対象プログラム

MPIETE2 を用いて予測可能なプログラムは，以下の条件を満たすプログラムである．

- Fortran で記述されている
- MPI 関数で記述された通信のみを行う

3.2 MPIETE2 の機能

MPIETE2 は，以下の機能を持つ．

- MPI プログラムより，計算ブロック基礎データ取得プログラムを自動生成する
- MPI プログラムより，通信ブロック基礎データ取得プログラムを自動生成する
- MPI プログラムより，ブロック実行回数取得プログラムを自動生成する
- 予測対象計算機上で実行することで得られる，ブロック基礎データとブロック実行回数より，プログラム全体の実行時間を算出する

なお，上記で示した 3 つの基礎データ取得プログラムは，全て Fortran と MPI 関数を利用したプログラムである．

3.3 MPIETE2 による実行時間予測の流れ

図 6 に，MPIETE2 による実行時間予測の流れを示す．

MPIETE2 では，以下の手順で実行時間予測を行う．

(1) ユーザが，予測対象プログラムファイルとユーザ入力パラメータを入力する

ユーザ入力パラメータは以下のものである

- プログラムファイル名及びプログラムファイルパス
- 実行時間予測対象区間 (開始行，終了行)

(2) MPIETE2 が, 予測対象プログラムに対して字句解析, 構文解析及び, 基礎データに合わせた変数依存解析を行い, 以下3つの基礎データ取得プログラムを自動生成する
- 計算ブロック基礎データ取得プログラム
- 通信ブロック基礎データ取得プログラム
- ブロック実行回数取得プログラム

(3) ユーザが, MPIETE2 により生成された 3 つのプログラムファイルを予測対象計算機上で実行し, 各基礎データを取得する
基礎データは以下の3つからなる
- 計算ブロック基礎データ
- 通信ブロック基礎データ
- ブロック実行回数

(4) ユーザが, 取得した各基礎データを MPIETE2 に入力する

(5) MPIETE2 が, 入力された各基礎データからプログラム全体の予測実行時間を算出する

4. 提案手法の評価

3 節で示した, MPIETE2 を用いて, NAS Parallel Benchmark (NPB) ver2.4 の実行時間を予測することで, 提案手法の評価を行う.

本提案手法の検証には, MPIETE と NPB2.4 のベンチマークを用いて, 実際の実行時間と予測実行時間の比較, 実際の実行時間と予測に必要な処理時間の比較を行う. なお, 比較に利用したプログラムは, NPB2.4 の 8 つのプログラムのうち, カーネルベンチマークの EP, CG, FT, MG プログラムの4つである. それぞれのプログラムの実行時間予測対象は, それぞれのベンチマークが性能評価を求める対象となる区間とし, 問題サイズ (CLASS) B の, 2-128PU で実行時の実行時間を求め, 検証を行った.

本章では, 4.1 に NPB2.4 について, 4.2 に予測対象とする計算機構成について, 4.3 に NPB2.4 の実行時間の予測結果, 4.4 に予測に必要な処理時間について示す.

4.1 NAS Parallel Benchmarks

NAS Parallel Benchmarks (NPB)は, NASA Ames Research Center で開発された並列コンピュータのためのベンチマークである. NPB は, MPI を利用した Fortran または C で記述された, 5 つのカーネルベンチマークと, 3 つのアプリケーションベンチマークで構成されている. それぞれのベンチマークは, Make 時にプロセス数を 2^n, もしくは, n^2 と指定することで, プログラムを実行する PU 数を指定することができる. また, PU 数と同様に, Make 時に問題のクラスを W, A, B, C と指定することで, 計算の対象となる問題サイズを指定することができる.

4.2 予測対象計算機

予測対象となる計算機構成を表 1 に示す.

予測対象計算機のネットワーク構成は, ノード 16 台を 1 つのスイッチに接続し, ノード 16 台をまとめた 8 つのスイッチを, 上流で 1 つのスイッチにカスケードしている.

表 1 予測対象計算機構成

Node	128
CPU	Intel Pentium4 2.4GHz
Clock	2.4GHz
L2 Cache	512Kbyte
Memory	SDRAM 1Gbyte
Network	1000Base TX Ethernet
Network Seitch	NETGEAR Gigabit Switch
Compiler	PGI ver 5.1
Compiler Option	-fast
MPI Library	MPICH ver 1.2.5

4.3 実行時間予測結果

MPIETE2 を用いて, NPB2.4 の EP, CG, FT, MG の実行時間を予測し, 予測実行時間と実際の実行時間を比較する. なお, 予測する対象とする区間は, それぞれのプログラムが性能評価の対象としている区間とする.

図 7 に各プログラムの実際の実行時間と予測実行時間の比較を示し, 表 2 に各プログラムの予測誤差を示す. なお, 予測誤差は, 以下の式より算出した. 単純に, 実際の実行時間に対する予測時間の割合としないのは, 計算部分, 通信部分の予測誤差が互いに相殺しない予測誤差を求めるためである.

$$\text{予測誤差}[\%] = \frac{|\text{実際の計算時間} - \text{予測計算時間}| + |\text{実際の通信時間} - \text{予測通信時間}|}{\text{実際の実行時間}} \times 100$$

表 2 に示す通り, 予測誤差は EP が 7%, CG が 14%, FT が 12%, MG が 14%以内である. 特に, 図 7 に示す通り, MPIETE[8]では不可能であった, 全てのプログラムで通信性能の低下を予測でき, 最大の実行性能を示す PU 数が特定できた. MPIETE と比較して, MPIETE2 の有効性を示す結果であると言える.

4.4 予測に必要な時間と実際の実行時間

予測に必要な処理時間とは, 以下に示す 3 つの処理時間からなる.

(1) MPIETE2 が行う, 基礎データ取得プログラムの生成時間

(2) 予測対象計算機上で行う, 基礎データ取得プログラムの実行時間

(3) MPIETE2 が行う, 基礎データからの実行時間算出時間

上記3つの処理時間のうち, (1), (3)に関しては, 表 3 に示す計算機上での実行で, それぞれ 1 秒未満の処理時間であったため, 本稿では割愛する.

図 8 に, (2)に関する予測に必要な処理時間と, 実際の実行時間の比較を, 表 4 に予測に必要な処理時間の内訳を示す.

図 8 より, 実際の実行と比較して, EP で約 1/10, CG で約 1/4, FT で約 1/5, MG で約 1/7 の処理時間で, 最大の実行性能を示す PU 数の特定が可能であった.

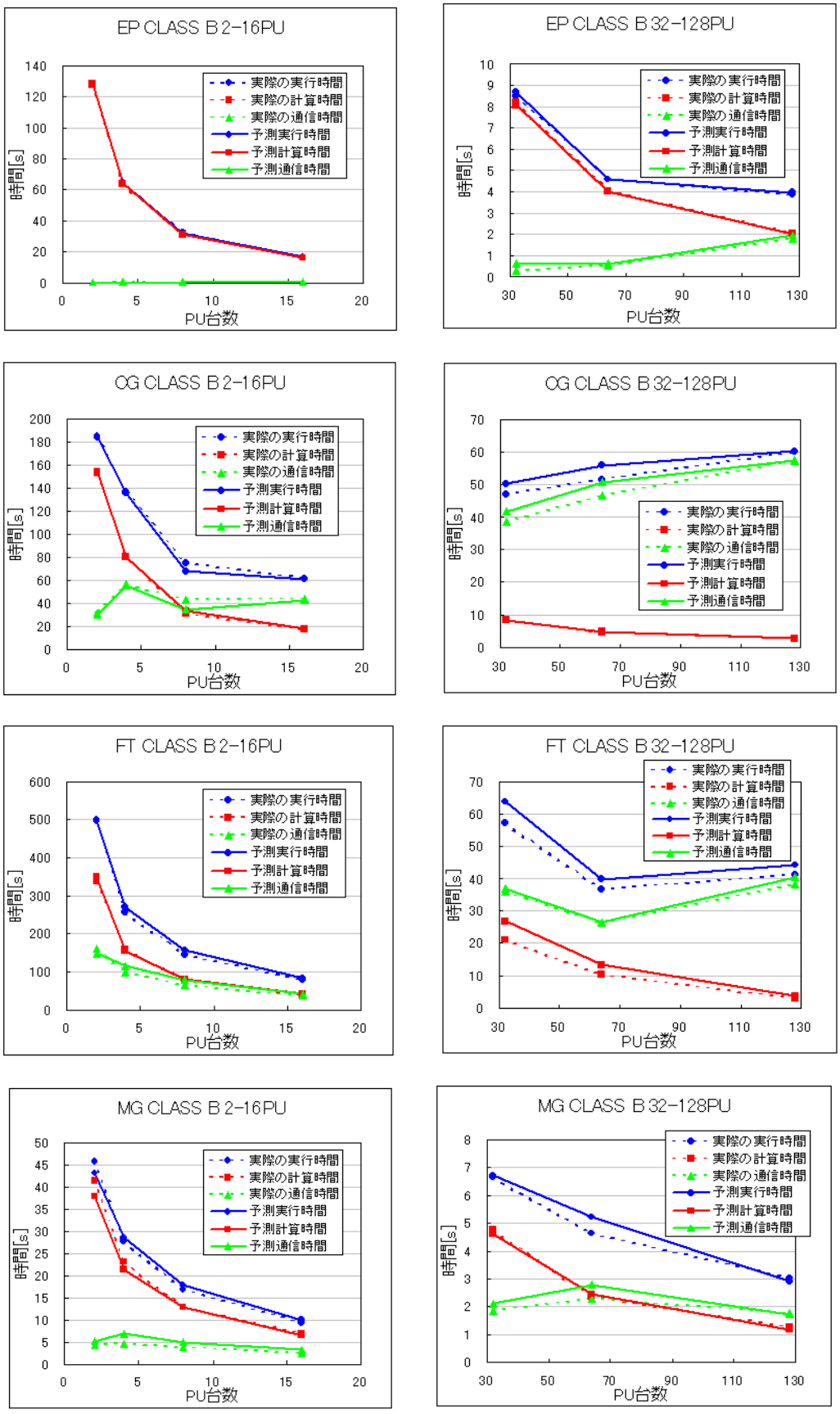

図 7 NPB2.4 各プログラムの実際の実行時間と予測実行時間の比較

表 2 NPB2.4 各プログラムの予測誤差

PU	EP	CG	FT	MG
2	0.33	1.35	5.39	9.41
4	0.46	0.84	8.45	13.87
8	3.37	13.57	8.53	6.34
16	3.35	3.15	6.99	13.56
32	5.54	7.05	11.40	5.98
64	2.49	8.39	8.60	12.31
128	6.31	0.46	7.16	4.14

単位[%]

表 3 MPIETE2 を実行する計算機構成

CPU	Pentium M 1GHz
L2 Cache	1Mbyte
Memory	SDRAM 256Mbyte
Compiler	Visual C++ .NET 2003
Compiler Option	/O2 /Ot /G7

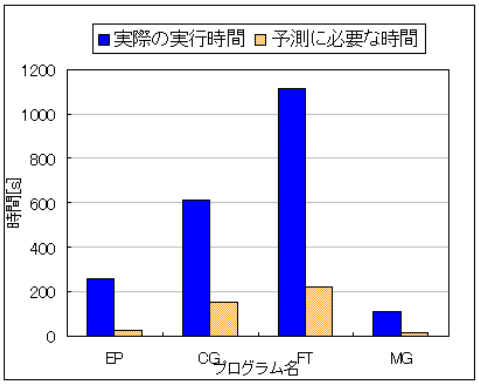

図 8 予測に必要な処理時間と実際の実行時間の比較

表 4 予測に必要な処理時間の内訳

Prog.	基礎データ取得プログラム実行時間		
	計算ブロック 基礎データ	通信ブロック 基礎データ	ブロック 実行回数
EP	23.49	4.79	0.17
CG	108.81	13.12	31.85
FT	90.94	120.11	10.48
MG	0.79	6.68	7.52

単位[s]

5. おわりに

本稿では, MPIETE[8]の通信時間予測手法に改良を加え, 通信コンテンションに伴う通信の待ち時間を予測できる手法を提案した. 本手法により, PU 数の増加に伴う通信性能を予測し, 最大の実行性能を示す PU 数の特定が可能となる.

本手法を MPIETE に組み込み, 開発した MPIETE2 を用いて, NPB2.4 EP, CG, FT, MG の実行時間を, 2-128PU 上で予測した結果, 以下のことが分かった.

- 全てのプログラムで実行時間予測誤差は 14%以内である
- 全てのプログラムで予測に必要な処理時間は, 実際の実行と比較して, 約 1/4 程度の時間である
- 全てのプログラムで最大の実行性能を示す PU 数の特定が可能である

最後に今後の課題を以下に挙げる.

- 提案手法における, ループ回数削減方法の確立
- MPIETE2 の C 言語への対応

前者に関しては, 本稿では 1/10 の回数に静的に決めていた. この方法では, 予測手法を適用できないプログラムが存在する可能性がある. 今後はより多くのプログラムに本手法を適用し, 検証を重ねていく.

後者に関しては, MPIETE2 は FORTRAN/MPICH プログラムにのみ対応している. MPICH は C 言語からも利用可能なライブラリであるため, MPIETE2 を C 言語に対応させ, 汎用性の向上を図る.

参考文献

[1] Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker, Jack Dongarra:“MPI The Complete Reference, The MPI Core second edition ”, The MIT Press,1998.

[2] Kazuto Kubota, Ken'ichi Itakura, Mitsuhisa Sato, Taisuke Boku:“Practical Simulation of Larege-Scale Parallel Programs and Its Performance Analysis of the NAS Parallel Benchmarks”, Proc. of Euro-Par, pp.244-254, 1998..

[3] Dickens P., Heidelberger P., and D. Nicol: “Parallelized Direct Execution Simulation of Message-Passing Parallel Programs”, IEEE Trans. on Parallel and Dist. Systems, Vol.7, No.10, pp.1090-1105, 1996.

[4] Sundeep Prakash, Ewa Deelman, Rajive Bagrodia:“Asynchronous Parallel Simulation of Parallel Programs”, IEEE Transactions on Software Engineering, 2000, vol. 26, pp.385-400, 2000.

[5] Maurice Yarrow and Rob Va der Wijngaart: “Communication Improvement for the LU NAS Parallel Benchmark:A Model for E cient Parallel Relaxation Schemes”, NAS Technical Report NAS-97-032, 1997.

[6] Thomas Fahringer, Hans P. Zima:“A Static Parameter based Performance Prediction Toolfor Parallel Programs”, Proc. of 1993 ACMInt. Conf. on Supercomputing, pp.207-219, July, 1993.

[7] Edward Rothberg,Jaswinder Pal Singh,and Anoop Guputa:“Working Sets, Cache Sizes, and Node Granularity Issues for Large-Scale Multiprocessors”,Proc.of ISCA ’93, pp.14-25, 1993.

[8] 堀井洋, 岩渕寿寛, 山名早人:“MPI プログラム実行時間予測ツール MPIETE の評価”, 情報処理学会研究報告（HPC）, Vol.2004, No.20, pp.55-60, 2004