# 大規模並列システムの性能評価を目的としたプログラムコード抽象化技法

松本, 幸
九州大学大学院システム情報科学府

本田, 宏明
九州大学情報基盤研究開発センター

稲富, 雄一
九州大学情報基盤研究開発センター

薄田, 竜太郎
福岡県産業・科学技術振興財団

他



出版情報：情報処理学会研究報告. HPC, [ハイパフォーマンスコンピューティング]. 2007 (80), pp.55-60, 2007-08. 情報処理学会
バージョン：accepted

# 大規模並列システムの性能評価を目的としたプログラムコード抽象化技法

松本 幸[†1,†2] 本田宏明[†3] 稲富雄一[†3]
薄田竜太郎[†4] 柴村英智[†2] 井上弘士[†5]
青柳睦[†3] 村上和彰[†5]

現在我々は，ペタフロップス級時代の次世代スーパーコンピュータを対象とした統合型システム性能評価環境 PSI-SIM を開発している．数万から数十万個のプロセッサを搭載した大規模並列計算機システムを対象とし，その実効性能を現実時間内で予測する．これを実現するためには，特に，1) プロセッサ間通信プロファイルの取得，ならびに，2) インターコネクト・シミュレーションによる性能予測，を高速かつ正確に実施する必要がある．本稿では，高速な通信プロファイルの取得を目的としたプログラム・コードの抽象化技法を提案する．また，2 つのアプリケーション・プログラムを用いた適用実験を行い，抽象化方式の違いが通信プロファイル取得速度と精度に与える影響を解析する．

# Program Abstrction for Performance Evaluation of Large-Scale Parallel Systems

Yuki MATSUMOTO,[†1,†2] Hiroaki HONDA,[†3]
Yuichi INADOMI,[†3] Ryutaro SUSUKITA,[†4]
Hidetomo SHIBAMURA,[†2] Koji INOUE,[†5] Mutsumi AOYAGI[†3]
and Kazuaki MURAKAMI[†5]

Our purpose is to build a performance evaluation environment for peta-scale computing systems. Particularly, we focus on interconnect designs. The performance of parallel machines with a very large number of computing nodes strongly depends on the capability of communication. So, it is essential to explore the design space of system interconnect in order to show the potential ability of massively parallel computing. To achieve this goal, there are at least two requirements: 1) fast, accurate correction of inter-node communication traces and 2) very fast interconnect simulation. In this paper, we present an abstraction methodology of large-scale parallel programs in order to answer the first requirement. Furthermore, we also evaluate the efficiency of the abstraction approach.

## 1. はじめに

大規模並列計算機 (スーパーコンピュータ) は，多数の計算ノードを相互接続して並列分散処理を行うことにより極めて高い計算性能を達成する．例えば，現在世界最高性能である IBM 社の BlueGene は 13 万個のプロセッサを搭載しており，その LINPAC 性能は 280TFlop/s と極めて高い．このような多数の計算ノードを有するスーパーコンピュータにおいては，計算ノードのみならず，それらを相互接続するネットワーク構成が極めて重要となる．

現在我々は，計算ノード間ネットワークに重きを置いた統合型システム性能評価環境である **PSI-SIM** の開発を行っている．PSI-SIM は，ペタフロップス級の大規模並列計算機システムの性能予測を目的としており，ネットワーク・シミュレータを核とする各種ツール郡で構成される[1)]．PSI-SIM を用いることで，所望の性能を満足する大規模並列計算機システム向けネットワーク・アーキテクチャを探索することが可能となる．

数万から数十万の計算ノードを有する大規模並列計算機システムの性能を予測するためには，1) 当該システムを前提としたノード間通信プロファイル (通信ログ) を生成し，2) それを入力としたネットワーク・シミュレーションを実施する必要がある．これらの処理は，実在する小規模計算機システムを用いて，現

†1 九州大学大学院システム情報科学府
Graduate School of Information Science and Electrical Engineering, Kyushu University
†2 (財) 九州システム情報技術研究所
Institute of Systems & Information Technologies/KYUSHU
†3 九州大学情報基盤研究開発センター
Research Institute for Information Technology, Kyushu University
†4 福岡県産業・科学技術振興財団
Fukuoka Industry, Science & Technology Foundation
†5 九州大学大学院システム情報科学研究院
Department of Informatics, Kyushu University

実的な時間内で完了できなければならない．つまり，PSI-SIM においては，通信プロファイルの取得とネットワーク・シミュレーションの高速化が極めて重要な技術課題となる．

そこで本稿では，上記 1) に着目し，通信プロファイル生成の高速化を可能にするプログラム抽象化技法について述べる (ネットワーク・シミュレーションの高速化に関しては文献[1] を参照)．また，小規模 PC クラスタを用いた評価実験を行い，本手法の有効性を議論する．本手法では，計算ノード間通信に影響を与えないプログラム・コード部分を抽出し，これを「実行時間」という極めて抽象度の高い表現に置換える．そして，当該コード部分を実際には実行することなく通信プロファイルを得る．基本ブロックなどの細粒度なコード部分のみならず，関数レベルの大規模な抽象化を積極的に行う．これにより，より高速な通信プロファイルの生成が可能となる．

以下，第 2 節では大規模並列計算機システムの性能評価において解決すべき課題を整理し，現在我々が開発している PSI-SIM を説明する．次に，第 3 節では高速な通信プロファイルの生成を可能にするプログラム・コードの抽象化技法を提案し，第 4 節では実アプリケーションへの適用例を示す．第 5 節では，通信プロファイル生成に関する正確性と高速性を評価する．最後に第 6 節で簡単にまとめる．

## 2. 統合型性能評価環境 PSI-SIM

システム開発時に設計空間を探索する代表的な方法として，開発対象システムのシミュレーションに基づく性能評価がある．精度よく実効性能を予測する事で，開発対象システムにおける適切な各種設計パラメータを決定できる．特に，スーパーコンピュータに代表される大規模並列計算機システムに関しては設計開発コストが極めて大きい．そのため，システム仕様検討時だけでなく，設計・開発段階においても継続的な性能評価の実施が必要となる．このような大規模並列計算機向け性能評価環境は，以下の要件を満足しなければならない．

- 高速性：一般に，実機実行と比較してシミュレーション実行は多くの時間を要する．この問題は，シミュレーション対象の大規模化に伴いより深刻になる．したがって，ペタフロップス級システムのシミュレーションを高速に実行可能でなければならない．
- 正確性：大規模並列計算機システムにおいては，計算ノード構成だけでなくネットワーク構成がシステム性能へ与える影響も大きい．したがって，ネットワークを含めた精度の高い性能予測を行わなければならない．

現在我々は，これらの要件を満足する統合型性能評価環境として PSI-SIM を開発している．PSI-SIM はペタフロップス級の大規模並列計算機システムの性能予測を主な目的としており，図 1 に示す構成を採る．ここで，性能予測に関しては以下 2 つのツールが重要となる．

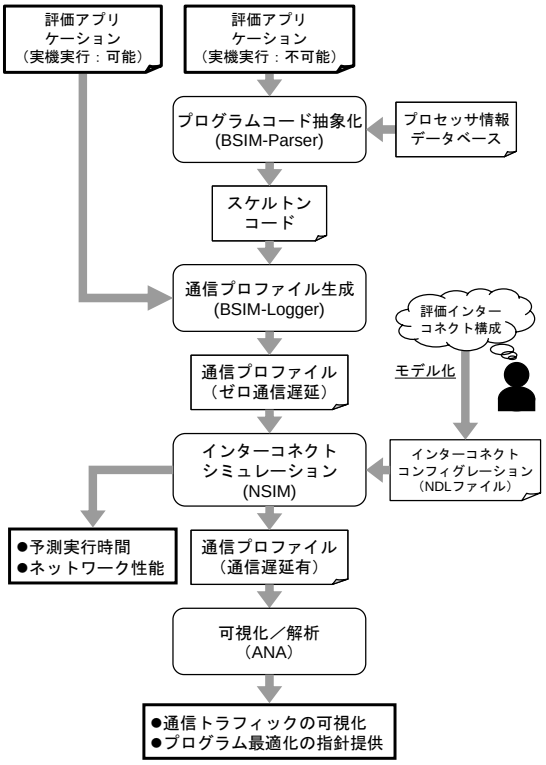


**図 1** システム性能評価環境 PSI-SIM のワークフロー

- BSIM-Logger：通信レイテンシ「ゼロ」の理想ネットワークを想定した通信プロファイル生成ツール．通信プロファイルには各プロセス間通信の時刻や送信/受信プロセスに関する情報などが含まれる．性能評価対象システムが有するプロセッサ数を考慮して MPI プログラムを実行する．
- NSIM：構成ファイルで定義したネットワークの振舞いを詳細に模擬するシミュレータ．BSIM-Logger が生成した通信プロファイルを入力とし，各通信の実レイテンシを求める．並列実行による高速シミュレーションを実現している．

## 3. プログラム抽象化による通信プロファイル生成の高速化

### 3.1 プログラムの抽象化

大規模並列計算機システムの性能を高速に予測するためには，ネットワーク・シミュレーションだけでなく，その入力となる通信プロファイルの生成も高速化する

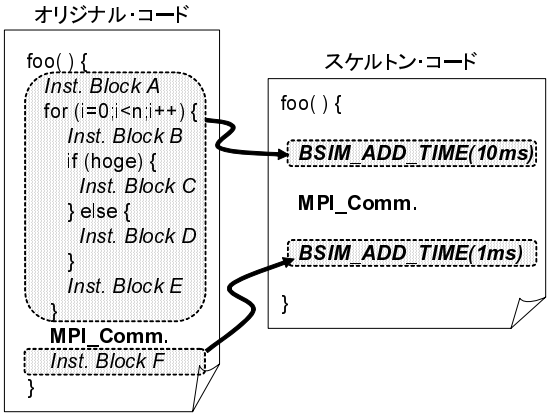


**図 2** プログラム・コードの抽象化

必要がある．しかしながら，MPE(Multi-Processing Environment) 等を用いた従来の通信プロファイル生成では，対象プログラムを実際に実行しなければならない．そのため，ペタフロップス級の大規模アプリケーションを実在する小規模システムで実行した場合には極めて多くの時間が必要となる．

この問題を解決するため，高速な通信プロファイルの生成を目的としたプログラム・コードの抽象化技法を提案する．第 2 節で説明した BSIM-Logger の目的は，性能評価対象システムでのプログラム実行を想定した通信プロファイルの生成である．したがって，プログラム実行結果を保証する必要はなく，プロセス間通信に関する情報 (発生時刻など) のみ正確に求めれば良い．これは，通信に影響を及ぼさないプログラム・コード部分に関しては，その**実行時間**を取得して通信プロファイルへ反映することが重要であり，**実行結果**そのものは不要であることを意味する．そこで，計算ノード間通信への影響が小さいプログラム・コード部分を抽出し，これを「実行時間」という極めて抽象度の高い表現に置換える．このような抽象化により作成されたプログラム・コードを**スケルトン・コード**と呼ぶ．

スケルトン・コードの例を図 2 に示す．ここでは，MPI 通信の前後に存在する 2 つのコード部分に対する抽象化の例を表している．これらのコード部分は，スケルトン・コードにおける 2 つの **BSIM_ADD_TIME()** 関数の呼び出しに置換されており，本関数の引数は対応するコード部分の見積り実行時間を表す．ここで，**BSIM_ADD_TIME()** は，第 2 節で説明した BSIM-Logger において定義される関数である．したがって，スケルトン・コードの作成は以下のような手順となる．

(1) 抽象化対象となる連続コード部分の選択
(2) 性能予測対象計算ノードにおける当該コード部分の実行時間見積り
(3) 当該コード部分を BSIM_ADD_TIME 関数で置換え (引数に上記 2 で求めた見積り実行時間を指定)

BSIM-Logger では，BSIM_ADD_TIME 関数で渡された引数の値 (つまり，見積り実行時間) に基づき通信プロファイルを生成するための内部時計を更新する．これにより，図 2 の場合，BSIM-Logger はオリジナル・コードにおける 2 つのコード部分を実行することなしに精度の高い通信ログを生成する．

文献[2] では，本手法と同様に連続した演算コード部分を抽出しその実行時間を計測することで MPI プログラム実行時間を見積る手法を提案した．また，文献[3] では，インスツルメンテーション技術に基づく通信トレースファイル生成環境を構築している．本稿で提案する抽象化もこれら関連研究と同様に各プログラムコード部分の実行時間を見積る．しかしながら，我々の手法では，基本ブロック等の細粒度なコード部分のみならず，関数レベルといったより大規模な抽象化を試みる．これにより，大規模なアプリケーションを対象とした性能評価の実現を目指す．

### 3.2 抽象化レベル

プログラム・コードの抽象化においては様々な選択肢が存在する．そこで，以下のように抽象化レベルを定義する．なお，添え字の数が大きいほど高レベル (上位レベル) とする．

- **レベル 1：**ストレート・ライン・コードの抽象化．ただし，IF 文や LOOP 文よる分岐，関数呼出し，ならびに，MPI 通信は含まない．
- **レベル 2：**IF 文を含む連続コード部分の抽象化．ただし，LOOP 文による後ろ方向への分岐や関数呼出し，ならびに，MPI 通信は含まない．
- **レベル 3：**LOOP 文を含む連続したコード部分の抽象化．ただし，関数呼出し，ならびに，MPI 通信は含まない．
- **レベル 4：**関数呼出しを含む連続コード部分の抽象化．ただし，MPI 通信は含まない．
- **レベル 5：**MPI 通信を含む連続コード部分の抽象化．

プログラム・コードは様々なレベルでの抽象化が混在する．例えば，図 2 において，「Inst. Block A-F」がストレート・ライン・コードの場合，MPI 通信前のコード領域はレベル 3，MPI 通信後のコード領域はレベル 1 の抽象化となる．

### 3.3 抽象化レベルが通信プロファイル生成に与える影響

第 3.2 節で示したように，プログラム・コードの抽象化レベルは様々である．本節では，抽象化レベルの違いが通信プロファイル生成における正確性ならびに高速性へ与える影響を考察する．ここで，「正確である」とは，オリジナル・コード実行により生成した通信プロファイルと，スケルトン・コードの実行により得られたそれとが一致していることを意味する．

```
for (ijcs=0; ijcs<ncs_pair; ijcs++) {
   カットオフのための準備(外側)
   for (klcs=0; klcs<=ijcs; klcs++) {
      カットオフのための準備(内側)
      if (カットオフ処理で生き残ったら) {
         ERI計算(抽象化)
         Fock行列への加算(抽象化)
      }
   }
}
```

**図 3** ERI 部分の抽象化

まず，正確性について議論する．低レベル抽象化では，当該コード部分が繰返し実行される場合，実行命令数の変動は小さくなる傾向にある．そのため，高レベル抽象化と比較して実行時間ばらつきによる影響は改善される．しかしながら，当該コード部分の実行命令数が極めて少ない場合，その実行時間見積り精度は難しくなる．より粒度の細かい実行時間計測が必要となり，測定誤差の影響が大きくなるためである．一方，高レベル抽象化を行った場合には，(ループ回転数などの) 実行振舞いの変化に伴う実行命令数の変動が大きく影響する．実行時間の見積りに関しては，比較的長い実行時間となるため，性能評価対象システムの計算ノードが既存の場合には時間計測用システムコールを用いた実機計測も可能となる．

次に，高速性について議論する．低レベル抽象化では，高レベル抽象化ほどの実行命令数削減は期待できない．そのため，より高いレベルでの抽象化により通信プロファイル生成時間を積極的に短縮できる．特に，抽象化によって得られる実行命令数削減効果よりも，BSIM-Logger の内部時計を更新するための BSIM_ADD_TIME 関数呼出しオーバヘッドの影響の方が大きい場合には，オリジナル・コードでの通信プロファイル生成時間と比較してより長くなる．

## 4. 実アプリケーションの抽象化 (ERI)

分子軌道法をはじめとする量子化学計算において，最も大きな計算コストとなるのは二電子積分 (以降，ERI と略す) の計算とその操作である．そこで，量子化学分野のアプリケーションプログラムの性能評価を想定し，ERI 計算用のスケルトン・コードを開発した．

ERI 計算を行うための抽象化したスケルトン・コード (擬似コード) を図 3 に示す．実際に BSIM_ADD_TIME 関数によって抽象化した部分は図中の「ERI 計算」ならびに「Fock 行列への加算」部分である．ERI 計算のコードは，基本的に縮約軌道 (Contracted Shell, CS) の 4 重ループ構造をとる．本スケルトン・コードでは，CS ペアに対する 2 重ループという形で，同様のループ構造を維持している。ERI 計算では，計算量を減らすために，カットオフ処理で生き残った積分のみを計算する．MPI を用いた並列処理は，外側の CS ペアに対するループ (図 3 の ijcs ループ) を静的に分割することで行っている．なお，抽象化コード部分の実行時間見積りに関しては，オリジナル・コードを単一プロセッサで実行し，積分 1 つあたりの計算時間の平均値として求めた．

## 5. 実アプリケーションの抽象化 (FT)

### 5.1 NAS Parallel Benchmarks FT プログラム

NAS Parallel Benchmark プログラムは NASA によって開発された，大規模計算機システムのためのベンチマークプログラム群である[5]．そのなかでも，FT は拡散の偏微分方程式について，式 (1) の 3 次元 Fourier 変換を用いることで解くプログラムであり，3 次元の高速 Fourier 変換 (3D-FFT) のコードを内包している．1 次元 高速 Fourier 変換 (1D-FFT) 部分の計算については Stockham アルゴリズムを使用しており，3D-FFT 計算の入出力となる 3 次元行列について，配列要素の座標軸間の転置と 1D-FFT を繰り返すことで 3D-FFT の計算を行なう．

$$F_{q,r,s}(u) = \sum_{l=0}^{n_z-1}\sum_{k=0}^{n_y-1}\sum_{j=0}^{n_x-1} u_{j,k,l} e^{\frac{-2\pi ijq}{nx}} e^{\frac{-2\pi ikq}{n_y}} e^{\frac{-2\pi ilq}{n_z}} \quad (1)$$

ここで $u_{j,k,l}$ と $F_{q,r,s}$ は 3 次元 Fourier 変換の入力ならびに出力振幅，$n_x$，$n_y$，$n_z$ はそれぞれ $x$，$y$，$z$ 軸についての Fourier 変換の分点数である．

実際のプログラム中では，転置作業については transpose 関数群が，1D-FFT の計算については cffts の関数群がそれぞれ担っている．以降，本稿では，実行時間の大部分を占めるこの 3D-FFT コード部分に焦点を当て議論する．

### 5.2 レベル 1 の抽象化

レベル 1 のスケルトンコードでは，スケルトン化対象演算ブロックとして制御構造を含まないストレートコードのみからなる．抽象化の際には，制御構造を決定する変数を入力変数から正しい値として計算可能となるよう変数依存解析を行なう事が必要であり，入力文から個々の制御変数に至る全ての依存関係を調査しスケルトン化対象の演算ブロックを決定する．本稿においては，人手を介する事で抽象化対象部分を求めた．

スケルトンコードに挿入する見積り時間のついては，

$$T = IC * CPI * CCT \quad (2)$$

の式から求めた．ここで $IC$，$CPI$，$CCT$ はそれぞれ命令数，1 命令当りの所要クロックサイクル数，クロックサイクル時間を表す．$IC$ については，コンパイル済みオブジェクト・コードを逆アセンブルし，各スケルトン化対象演算ブロック毎に静的な命令数を

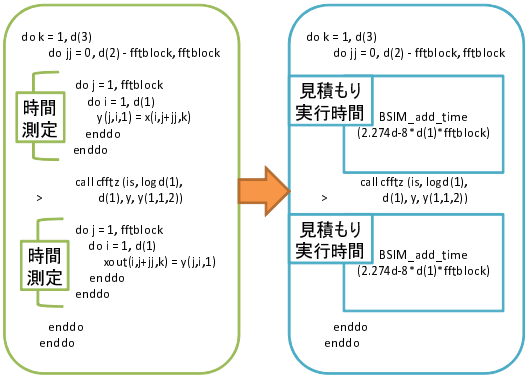


**図 4** レベル 3 での FFT コード抽象化

数えることで求めた．$CPI$ を求める際には，まず，式 (1) の $(nx, ny, nz)$ を FT プログラムに付属の最小入力である $(64, 64, 64)$ と設定した．その上でプログラムを逐次実行にて実際に実行し，全クロックサイクル数と全命令数をハードウェア・カウンタより測定し，その平均 CPI を使用することとした．$CCT$ については使用したプロセッサの動作周波数から求めた．

### 5.3 レベル 3 の抽象化

レベル 3 での FFT プログラム抽象化に関して説明する．抽象化対象の関数 fft の内部では複数関数が呼ばれており，図 4 の左に示すように殆どの関数が多重ループ構造を採る．さらにループ中では別関数が呼ばれ，多重ループの最内では代入または演算が行われる．抽象化の対象として，制御構造を崩さない条件のもと，

(1) MPI 関数をブロックの中に含まない
(2) ループの最内が代入文または演算文のブロック
(3) 他の関数呼び出しを含まない範囲で最大の範囲

を満たす部分を選択した．また，並列計算を実行する際に各プロセスで実行される演算に差はないことを前提に，レベル 3 で抽象化した際に挿入する見積もり時間は以下の方法で決定した．

(1) 抽象化の対象としているブロックの前後に時間測定関数 (MPI_Wtime) を挿入し，プログラム中で対象としているブロックにおけるプロセスあたりの総実行時間を求める．
(2) 対象としているブロック中で最内のループ内にカウンタを用意し，プログラム実行中の各プロセスにおける総演算回数を求める．
(3) 総実行時間を総演算回数の値で割り，最内の計算部一回当たりの演算時間を求める．
(4) 対象としているブロックをコメントアウトし，3 で求めた所要時間とコメントアウトするループ 1 回における演算部の実行回数をかけて見積もり時間とする．

**表 1** 実験環境

| | |
|---|---|
| CPU<br>RAM | Intel Xeon 3.0GHz<br>4GB |
| ノード数 | 8 (2CPUs/node) |
| OS | RedHat Linux 8<br>(Linux kernel 2.4.21)+ SCore 5.8 |
| Compilers | gcc 3.2.3 |
| MPI ライブラリ | MPICH-2.0.4p1 |

## 6. 評価実験

### 6.1 実験環境

第 4 節ならびに第 5 節で説明した 2 種類のアプリケーション・プログラムに関して，通信プロファイル生成に関する正確性ならびに高速性の評価を行った．なお，プログラム・コードの抽象化による影響を解析するため，ネットワークは通信レイテンシ「ゼロ」の理想状態を想定する．具体的には，第 2 節で説明した BSIM-Logger を用いて，オリジナル・コードとそれを抽象化したスケルトン・コードをそれぞれ実行する．以下，評価対象モデルを示す．

- **ERI394, ERI679, ERI1009:** レベル 4 の抽象化に基づく二電子積分計算プログラム．第 4 節で説明したように，抽象化対象コード部分の実行時間は実機実行に基づき求める．各モデルの数字は基底関数の数を表す．
- **FFT-L1C:** レベル 1 の抽象化に基づく FFT プログラム．第 5 で示したように，抽象化対象コード部分の実行時間は，実行命令数と CPI(Clock cycle Per Instruction) に基づき求める．入力クラスは C:$(n_x, n_y, n_z) = (512, 512, 512)$ である．
- **FFT-L3C:** レベル 3 の抽象化に基づく FFT プログラム．第 5 で示したように，抽象化対象コード部分の実行時間は実機実行に基づき求める．入力クラスは C である．

理化学研究所情報基盤センターに設置された RSCC(RIKEN Super Combined Cluster) を用て実験を行った．システム構成を表 1 にまとめる．

## 7. 実験結果

### 7.1 ERI

ERI を対象とした実験結果を図 5 に示す．オリジナル・コードならびにスケルトン・コードの実行に関して，図中の左グラフは予測実行時間を，右グラフは通信プロファイル生成時間を表す．実験結果より，殆ど正確性の低下をもたらすことなく抽象化を実現していることが分かる．実際，抽象化前と後での予測実行時間に関する誤差は $+/-1.5\%$の範囲に収まっている．また，通信プロファイルの取得においては，その所用時間が抽象化前と比較して 3〜9%程度に削減されており，正確性と高速性を両立できていることが分かる．

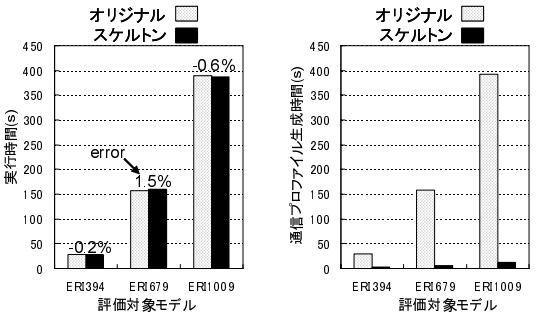


**図 5** ERI の実験結果

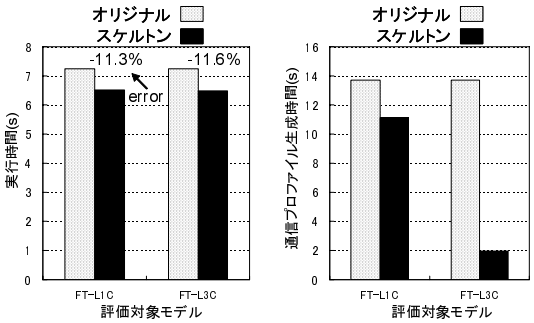


**図 6** FFT の実験結果

### 7.2 FFT

FFT を対象とした実験結果を図 6 に示す．図 5 と同様，左グラフは予測実行時間を，右グラフは通信プロファイル生成時間を表す．

まず，抽象化レベルの違いが通信プロファイルの正確性に与える影響を考察する．図 5 の左グラフで示すように，抽象化レベルの違いによる正確性の大きな差は見られなかった．しかしながら，第 7.1 節で示した ERI とは異なり，抽象化に伴い 11%程度の誤差が発生している．その原因を解析した結果，1D-FFT を計算する cffts 関数群に関する実行時間の見積り精度の低下が原因であった (転置作業を行う transpose 関数の実行時間見積り精度は高かった)．cffts 関数群では比較的深い多重ループ構造を含んでおり，最内ループの実行時間はばらつきが大きい．その結果，実行時間見積り誤差が蓄積すると考える．

次に，通信プロファイル生成時間について議論する．レベル 3 の抽象化では 85% 程度の処理時間削減を達成している．これに対し，レベル 1 の抽象化では 19%程度の削減に留まっている．詳細な原因を解析するため，両方のスケルトン･コードに関して BSIM_ADD_TIME 関数実行回数を測定した．その結果，レベル 1 のスケルトン・コード実行では，レベル 3 の場合と比較して約 $10^4$ 回多く BSIM_ADD_TIME 関数を実行していることが分かった．つまり，第 3.3 節で述べたように，低レベル抽象化では BSIM-Logger 内部時計の更新処理が頻繁に実行されオーバヘッドとなっている．20%程度の通信プロファイル生成時間短縮では，ペタスケール・アプリケーションを対象とする場合には不十分であり，より高レベルな抽象化が必要となる．

## 8. おわりに

本稿では，統合型システム性能評価環境 PSI-SIM において，高速な通信プロファイルの生成を目的とするプログラム抽象化技術を提案した．また，二電子積分計算ならびに FFT プログラムを対象にした適用実験を実施した．その結果，二電子積分計算プログラムに関しては抽象化による効果が極めて高いことが分かった．また，FFT を用いた実験においては，より高レベルの抽象化が重要であることが明らかになった．

大規模なアプリケーション・プログラムを用いた性能評価を可能にするには，1) より高レベルでの積極的な抽象化，ならびに，2) スケルトン・コード実行における使用メモリ量の削減，が重要となる．現在，これらの課題を念頭に置き，幾つかのアプリケーションに関してペタフロップス級スケルトンコードの開発を進めている．また，今回の実験では，通信レイテンシ「ゼロ」の理想ネットワークを想定した．今後は高速ネットワーク・シミュレータを介したより現実的な評価実験を行う予定である．

**謝辞** 本研究は，文部科学省「次世代 IT 基盤構築のための研究開発」，研究開発領域「将来のスーパーコンピューティングのための要素技術の研究開発」(平成 17 年度〜19 年度）における研究開発課題「ペタスケール・システムインターコネクト技術の開発」[4] によるものである．なお，本研究の実験結果は，理化学研究所の RIKEN Super Combined Cluster System を用いて得られたものである．

## 参 考 文 献

1) 柴村英智，薄田竜太郎，本田宏明，稲富雄一，于雲青，井上弘士，青柳睦： 次世代スーパーコンピュータの設計開発に向けたシステム性能評価環境 PSI-SIM，情報処理学会研究報告 2007-HPC-111 (SWoPP'07), 2007 年 8 月.
2) 堀井洋, 山名早人: 実測に基づいた MPI プログラムの実行時間予測， 情報処理学会研究報告 (HPC), No.88, pp.61-66, 2001.
3) 久保田和人, 板倉憲一, 佐藤三久, 朴泰祐：大規模データ並列プログラムの性能予測手法と NPB2.3 の性能評価，情報処理学会論文誌, Vol.40, No.5, pp.2293–2304 (1999).
4) Petascale System Interconnect Project: http://www.psi-project.jp/
5) THE NAS PARALLEL BENCHMARKS, http://www.hpfpc.org/npb.html