情報処理学会論文誌　コンピューティングシステム　Vol. 4　No. 1　1–23 (Feb. 2011)

大規模スーパコンピュータ向けシステム性能評価環境の構築

井上弘士[†1]　安藤壽茂[†3]　薄田竜太郎[†2]
山村周史[†3]　柴村英智[†2]　三輪英樹[†3]
本田宏明[†2]　稲富雄一[†1]　眞木淳[†2]
平尾智也[†2]　青柳睦[†1]　村上和彰[†1]
石附茂[†3]　小松秀実[†3]　安島雄一郎[†3]
三吉郁夫[†3]　清水俊幸[†3]　黒川原佳[†4]

本稿では，大規模スーパコンピュータ開発のための新しいシステム性能評価法を提案する．また，それを実現するために開発した各種ソフトウェア・ツールの詳細を述べる．さらに，実在するテラフロップス級スーパコンピュータを対象とした精度評価実験，ならびに，仮想ペタスケール・スーパコンピュータを対象とした性能予測実験を行うことにより，本環境の有効性を明らかにする．実在する計算機システムを利用して，2〜3 桁高性能な計算機システムの性能を予測するためには，これらの間に存在する大きな性能差を埋める必要がある．そこで我々は，1）プログラムコードの抽象化技術の導入，2）抽象化をサポートするためのプロセッサならびに相互結合網のシミュレータ開発，3）抽象化したプログラムを実行するための仮想超並列実行環境の構築，を行った．ギガフロップス級の PC クラスタを用いて，実在するテラフロップス級スーパコンピュータの性能を予測した結果，性能予測誤差はわずか 10%〜20%程度であることが分かった．また，理論ピーク性能 2.1 ペタフロップスの仮想スーパコンピュータを対象とした実験を行い，4 時間〜6 時間といった現実的な時間での性能予測が可能であることを示した．

Developing Performance Evaluation Environment for Large Scale Supercomputers

Koji Inoue,[†1] Hisa Ando ,[†3] Ryutaro Susukita,[†2]
Shuji Yamamura,[†3] Hidetomo Shibamura,[†2]
Hideki Miwa,[†3] Hiroaki Honda,[†2] Yuichi Inadomi,[†1]
Jun Maki,[†2] Tomoya Hirao,[†2] Mutsumi Aoyagi,[†1]
Kazuaki Murakami,[†1] Shigeru Ishizuki,[†3]
Hidemi Komatsu,[†3] Yuichiro Ajima,[†3] Ikuo Miyoshi,[†3]
Toshiyuki Shimizu[†3] and Motoyoshi Kurokawa[†4]

This paper proposes a novel approach to predict the performance of large scale supercomputers, and shows the details of several software tools developed in this project. It is not easy to accurately predict the performance of large-scale systems such as peta-scale supercomputers. This is because we need to exploit existing computer systems, the performance of which is two or three orders of magnitude lower than the target system. To bridge the performance gap, a technique to highly abstract the application program codes is introduced. This approach makes it possible to achieve high-speed, accurate performance predictions. In addition, three software tools are developed to evaluate the performance of processors, interconnects, and also whole systems. The experiments demonstrate that the error was only from 10% to 20% when the performance of a tera-scale supercomputer is predicted by a giga-scale PC cluster. Another case study is to predict the performance of an un-existing peta-scale supercomputer by using an existing tera-scale machine. The results show that the performance can be predicted by spending only from four to six hours.

†1 九州大学
Kyushu University
†2 財団法人九州先端科学技術研究所
Institute of Systems, Information Technologies and Nanotechnologies
†3 富士通株式会社
Fujitsu Limited
†4 独立行政法人理化学研究所情報基盤センター
Advanced Center for Computing and Communication, RIKEN

1. はじめに

スーパコンピュータは計算科学ならびに計算機科学の発展を支える重要かつ重宝な道具である．計算科学においては，スーパコンピュータを利用した大規模シミュレーションを行うことで，人類がいまだ解明できていない様々な諸問題の解決を目指す．一方，計算機科学の分野では，世界に類のない高性能な計算機システムを実現することにより，マイクロプロセッサやメモリシステムといったハードウェアからOSや各種ライブラリのミドルウェアに至るまで，計算機システムに必要不可欠な様々な構成要素に関する技術開発が加速される．そして，このようなスーパコンピュータを「使う技術」と「造る技術」は両輪となり，今日の情報化社会を支えている．たとえば，薬や自動車など我々の社会生活に深く浸透している製品の開発においてはスーパコンピュータを用いたシミュレーションが多く用いられる．また，クラウドコンピューティング時代においては情報社会インフラとしての基幹系計算機サーバが必要不可欠であり，その実現にはスーパコンピュータ開発において確立された高性能化技術，低消費電力化技術，高信頼化技術，実装技術など様々な知識と知見を直接的に活用できる．さらに，これらは要素技術としての側面も持ち合わせるため，次世代情報家電や携帯端末機器に代表される組み込みシステムへの貢献も期待される．このような背景の中，世界各国においてもスーパコンピュータ開発が加速しており，特に近年では，日本や欧米だけでなく，中国をはじめ韓国，インド，ブラジルといった国々の台頭が顕著になっている．

このように，スーパコンピュータの開発は学術的/社会的に様々な恩恵をもたらす一方，その開発には莫大な費用を要する．特に，テラフロップスからペタフロップス，さらには，エクサフロップス級の性能を目指すためにはシステムそのものが大規模化/複雑化せざるをえず，開発コストの増大はより深刻な問題となる．そのため，開発後の大幅なシステム変更は事実上不可能であり，製造前の仕様決定段階において広大な設計空間を探索し可能な限り最適なシステム構成を決定しなければならない．これを実現するためには，様々なシステム構成を想定した性能予測が必要となる．特に，ハードウェア構成のみで決定される理論ピーク性能ではなく，アプリケーション・プログラムを動作させた際の実効性能を正確に見積もらなければならない．しかしながら，現状では設計者の経験に頼るところが大きく，多くの場合は過去の開発実績に基づく机上での見積りにとどまっている．通常，アプリケーション・プログラム実行の振舞いはソフトウェア特性だけでなくハードウェア構成にも依存する．特にスーパコンピュータのような超並列計算処理を行う場合には，プロセス間通信能力に応じて実行時の挙動が変化する．このような動的な振舞いとそれがシステム性能へ与える影響を机上予測することは容易でなく，その結果として最悪の場合にはシステム製造後に所望の性能を満足できない可能性もある．したがって，仕様設計の段階で様々なシステム構成の性能を予測し，かつ，その詳細を解析するための環境構築が急務の課題となる．

そこで我々は，上述した課題を解決しスーパコンピュータ開発を加速させるべく，大規模スーパコンピュータ向け性能評価環境の構築を行った．本研究の主たる貢献は次の2点にある．まず第1点目は，超高性能プロセッサならびに大規模インターコネクト用シミュレーション・ツール群の開発である．これらのツールを用いることにより，スーパコンピュータの中で最も重要な構成要素である計算ノードとそれらを相互に接続する結合網に関するアーキテクチャ探索が可能となる．第2点目は，プログラム実行コードの高度抽象化技術の導入とそれに基づく仮想超並列実行環境の構築である．これにより，既存の小規模計算機システム（たとえばテラフロップス級マシン）を用いて，未知の大規模スーパコンピュータ（たとえばペタフロップス級マシン）の実効性能を予測することが可能となる．なお，本稿は，平成17年度～平成21年度の5年間にわたりに実施した「スーパコンピュータ開発に向けた性能評価環境の構築」の研究成果をまとめたものである[1),5),8),26)]．開発したツールの一部は平成22年夏を目処に一般公開する予定であり，本研究で確立した性能評価技術ならびにツール群は，将来にわたるスーパコンピュータの性能評価だけでなく，様々な大規模計算機システムの開発に広く応用可能であると考える．

以降，本稿では，開発した大規模スーパコンピュータ向け性能評価環境の詳細を示す．2章ではスーパコンピュータの性能予測に関する既存技術の問題点を整理し，解決すべき課題を明確にする．次に，3章では本研究において実際に開発した各種ツール群とその評価結果を紹介し，4章で仮想ペタスケール・スーパコンピュータを前提とした性能予測実験を行う．5章では提案手法の適用可能性に関する考察と今後の課題を述べ，最後に6章で本稿をまとめる．以降，本稿では，性能予測対象となる計算機システムを「**ターゲットマシン**」，性能予測を実施する計算機システムを「**ホストマシン**」と呼ぶ（つまり，ホストマシンを用いてターゲットマシンの性能を予測する）．

2. 大規模スーパコンピュータ向け性能予測技術と解決すべき課題

計算機システムの実効性能を正しく予測するためには，システムを構成するハードウェアの特性だけでなく，実行対象となるアプリケーション・ソフトウェアの特性も同時に考慮しなければならない．これに加え，対象がスーパコンピュータの場合には数万から数十万の計算ノードを有するため，いかにして現実的な時間内での性能予測を可能にするかが重要なポ

イントとなる．このような課題を解決すべく，これまでに様々な研究開発が行われてきた．これらは主に，「性能モデリング方式」と「シミュレーション方式」に大別できる．

性能モデリング方式では，ターゲットマシンでのアプリケーション実行時間をモデル化し，ハードウェア特性やアプリケーション特性に基づきモデル中に出現するパラメータ値を決定する．たとえば，プログラム実行時間を計算時間と通信時間に分離し，アプリケーションごとに問題サイズやプロセス数などをパラメータ値とした性能モデルを作成する．これにより，少ないプロセス数の測定結果などから大規模構成の性能を予測することが可能となる[14),20)]．また，トレース情報に基づきシングルプロセッサ性能と通信性能を取得しこれらを性能式へ代入する方法も提案されている[15),18),25)]．その他の関連研究としては，メモリバンド幅をキャッシュ・ヒット率の補間可能な関数としてモデル化する手法[27)]やニューラルネットワークを用いた学習により性能モデルを自動生成する手法[21)]，類似性の高い複数の既存マシンでの実行結果より性能予測対象マシンでの性能モデルを作成する手法[22)]などがあげられる．これらに加え，各演算処理や通信に対応した代表遅延時間を用いて実行時間を算出する MPIETE[2),7)]，統計情報を用いる Parafrase[9)]なども提案されている．これらモデリングに基づく性能予測方式の共通点は，小規模もしくは類似の実在する計算機環境を用いて性能に関する基本情報を採取し，実行対象となるアプリケーションの特性を考慮してプログラム実行時間を数式（性能モデル）で表現する点にある．したがって，この性能モデルを 1 度作成してしまえば，その後の性能予測に要する時間は無視できる程度に小さい．しかしながら，これら性能モデルには時間的概念が含まれないため，プログラム実行の挙動が単純かつ静的に決定される場合にはきわめて有効であるが，プロセス間ロードインバランスなどの動的変動要因が性能へ影響する場合には対応することが難しい．

一方，シミュレーション方式による性能予測では実際にアプリケーションを実行するため，動的な実行の振舞いの変化が生じた場合においても比較的精度良く性能を予測できる．たとえば，代表的な研究開発事例として，EXCIT+INSPIRE[3)]，INSIGHT[4)]，CSIM[9)]，MPI-SIM[23)]，BigSim[29)]などがあげられる．これらの共通点はシステムの振舞いを細部にわたり性能予測する方針がとられていることであり，その多くは**図 1** (A) に示すように評価対象となる並列プログラムをホストマシン上で実行し通信ログを採取する．そして，これを入力としたインターコネクト・シミュレーションを実施することでシステム性能（アプリケーション・プログラムの実行時間）を予測する．しかしながら，世界最速スーパコンピュータの性能予測を行う場合には，各種シミュレーションに利用可能な計算資源との性能ギャップが存在し，特に以下の問題が生じる．

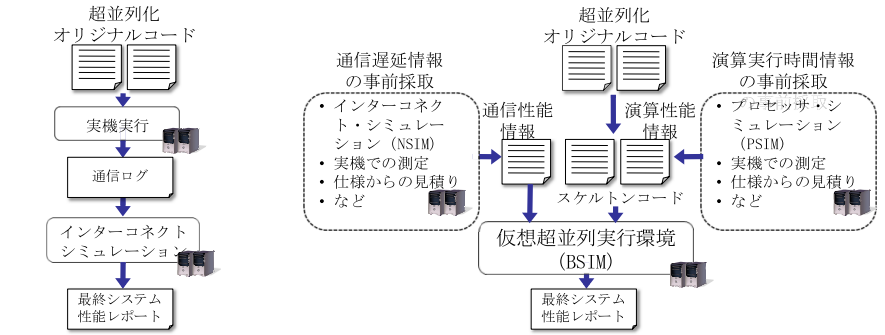


図 1　性能評価フロー
Fig. 1　Performance evaluation flow.

【問題点 1】ホストマシンのリソース不足に起因するプログラム実行限界：ターゲットマシンの性能を正しく予測するためには，それに見合った大規模なアプリケーション・プログラムと入力サイズを想定し，その実行によって発生する各種イベント情報（プロセス間通信のパターンなど）を収集する必要がある．しかしながら，ホストマシン上で当該プログラムをそのまま実行する場合，ハードウェアのリソース不足（特にメモリ容量不足）が深刻化する．一般にターゲットマシンはホストマシンより多くの計算ノードを有する．したがって，計算ノードあたりの使用メモリ容量が同一と仮定した場合，計算ノード数に比例してシステム全体での総使用メモリ容量が増加する．その結果，メモリ不足によりホストマシン上でのターゲットマシン用プログラムの実行がきわめて困難となる．

【問題点 2】通信ログ採取限界：数万から数十万の計算ノードを搭載するスーパコンピュータでは多くの通信が発生する．そのため，通信ログ採取時間の長期化や通信ログ記憶のためのハードディスク容量不足といった問題が顕著化する．たとえば，我々の試算では，4,096 計算ノード（64 GB メモリ/ノード）を有するシステムにおいて HPL（High Performance Linpack）を実行した場合には通信ログサイズが 1 TB 程度になる．これは計算ノード数の増加にともない増大するため，大規模なターゲットマシンを対象とする場合はより深刻となる．

【問題点 3】ソフトウェア・シミュレーションの実行限界：小規模なシステムを対象とする場合，一般的には当該システムのソフトウェア・シミュレータを開発し，その上で

アプリケーション・プログラムを実行することでシステム性能を見積もる．しかしながら，その速度は実機と比較して 3〜4 桁以上遅いため，数万から数十万ノードの計算機資源を有し，かつ，それに応じた規模のアプリケーション・プログラムを実行対象とする場合には非現実的である．たとえば，フリットレベルのインターコネクト・シミュレータ[5] を利用し，計算ノード数 4,096（3 次元トーラス）での全体全通信をシミュレーションした結果，その完了に約 9 時間を要している．これは計算ノード数の増加にともない増大するため，前述した通信ログ生成問題と同様，数万から数十万規模のターゲットシステムを対象とする場合はより深刻になる．また，さらに大規模なアプリケーション・プログラムにおいては複数回の通信が生じるため，現行のインターコネクト・シミュレータを用いたプログラム実行性能の予測は容易でない．この問題は，スーパコンピュータの主要構成要素であるプロセッサのシミュレーションにおいても同様である．

今後，さらなる性能向上を目的として，スーパコンピュータの構成はより大規模化かつ複雑になると予想される．このような状況においては，プロセス間ロードインバランスや通信遅延の変動など非決定的なイベントが性能へ与える影響はより大きくなることが予想されるため，シミュレーション方式による性能予測の必要性は今後より高くなる．

3. 大規模システム性能予測環境の開発

2 章で述べた問題点を解決するためには，性能評価に関して従来手法とは異なる新しいアプローチが必要となる．本章では，「規模の壁（2 章の問題点 1 と問題点 2)」と「速度の壁（2 章の問題点 3)」を打破する新しい性能評価法とそれを可能にする各種ツール群の詳細を述べる．

3.1 基本コンセプトと特徴

2 章で述べたシミュレーション方式における問題点を解決し，実在するホストマシンを利用して開発対象の大規模ターゲットマシン性能を予測するためには，巨大な通信ログファイルを必要とせず，かつ，ターゲットマシン上でのプログラム実行の振舞いを高速に模擬できる性能予測技術の確立が必要となる．これらの要求を満足するため，我々は大規模システム性能予測技術とそれを用いた各種ツール群の開発を行った．提案する性能予測フローを図 1 (B) に示す．従来方式とは異なり，ターゲットマシンを想定したプロセッサでの演算時間情報，ならびに，通信遅延情報を事前に採取しておく．そして，これらの値を利用し，ホストマシン上に構築した仮想超並列実行環境でターゲットマシン用のアプリケーション・プログラムを実行することでターゲットマシンの性能を予測する．図 1 (A) に示す従来方式では，評価対象プログラムの実行において，プロセッサでの演算処理やノード間通信のインスタンスすべてをシミュレーション対象とする．たとえば，ループ内に集団通信が含まれている場合，集団通信イベントが発生するたびにその遅延時間をインターコネクト・シミュレーションにより求める．そのため，より正確に性能を予測できる反面，大量のイベントが発生する大規模スーパコンピュータを想定した場合には膨大な時間が必要となる．この問題を解決するため，本研究では以下のアプローチをとる．

- **実行対象プログラムコードの高度な抽象化：**性能予測の目的はあるプログラムを実行した際の実行時間（もしくは実効性能）を正しく推測することであり，システムの機能を検証するわけではない．したがって，当該プログラムの出力結果の正当性は保証しなくてよい．そこで，プログラム実行の出力結果は保証せず，その代わりにプログラム実行の振舞い（実行される制御フローや通信発生パターンなど）を可能な限り忠実に再現できる性能評価専用プログラムコードを導入する．これにより，対象アプリケーション・プログラムの実行に必要となる処理量，通信量，ならびに，メモリ使用量を大幅に削減することが可能となる（2 章の問題点 1 を解決)．
- **各演算処理や通信に対応した代表遅延時間の利用：**多くのプログラムは基本的にループ構造をとる．そのため，同一のプログラムコード部分が繰り返し実行されるという性質を有する．そこで，繰り返し実行される演算処理や通信のそれぞれに関して，詳細シミュレーションにより所要時間の代表値を決定する．以降，プログラム実行において同一処理が出現した場合にはつねに事前決定した代表値を用いる．これによりインスタンスごとの詳細な実行時間見積り作業が不要となり性能予測時間を大幅に短縮できる（2 章の問題点 3 を解決)．
- **ホストマシンでのネイティブ実行可能な仮想超並列実行環境の開発：**高度に抽象化されたアプリケーション・プログラムを実在するホストマシンで直接実行する．具体的には，性能予測対象となるターゲットマシンでの実行と同数のプロセスをホストマシン上で起動し，各演算ならびに通信時間の代表値に基づきプログラムの実行を進める．これにより，ターゲットマシン上でのプログラム実行の振舞いを可能な限り忠実に再現し性能を予測する．また，事前に採取した代表遅延時間を用いて直接的にプログラム実行を進めるため，従来方式のような通信ログなどの取得は不要となる（2 章で示した問題点 2 を解決)．

本性能予測方式の新規性は，2 章で説明した「性能モデリング方式」と「シミュレーション予測方式」のハイブリッド方式を採用した性能評価環境を構築した点にある．繰り返し実行される同一の演算や通信といったミクロな処理に関しては実行時間のばらつきが小さいと考え，該当するプログラムコード部分を性能モデルで置き換える（プログラムコードの抽象化）．一方，プロセス間の同期や並列実行などのマクロな処理については，抽象化したコードを実際に実行することでその挙動を正しく反映させる．具体的には，ターゲットマシンでの実行と同じ数のプロセスを，ホストマシン上に構築した仮想超並列実行環境で実行する．したがって，性能モデリング方式では対処できない動的な実行の振舞いを反映できる．また，高負荷部分に関しては性能モデルで表現しているため，その実行においてはホストマシンのハードウェア資源ならびに時間をほとんど必要とせず，前述したような従来型シミュレーション方式において生じる 3 つの問題を回避可能である．さらに，従来のシミュレーション方式では図 1 (A) に示すように「計算に要する時間」と「通信に要する時間」を逐次的に求めるため，通信遅延の変動が計算時間に影響する場合には対応できない．これに対し提案方式では，図 1 (B) で示すように計算ノードとインターコネクトの挙動を同時かつ時系列に即して模擬するため，計算時間と通信時間が互いに影響し合う場合においても対応可能となる．

本方式は，図 1 (B) で示すように，性能評価専用プログラムコード（図中ではスケルトンコードと表記）の作成において性能モデリングのための演算実行時間に関する情報が必要となる．これは，ターゲットマシンの計算ノード性能を表すものであり，ターゲットマシンが実在する場合には実機により測定することができる．一方，ターゲットマシンの計算ノード（特にプロセッサ）が存在しない場合には，プロセッサ・シミュレータなどを用いた性能推定が必要となる．3.3 節で説明する PSIM は，将来のスーパコンピュータを想定したプロセッサ・シミュレータであり，この演算実行時間情報を取得するために利用できる．また，仮想超並列実行環境（BSIM）は通信性能情報を参照しながらアプリケーション実行の時刻を進める．この際，通信遅延時間に関する情報が必要であり，計算ノードと同様，既存もしくは類似のインターコネクトが存在しない場合にはシミュレーションなどにより通信性能を求める必要がある．3.4 節で説明する NSIM は大規模インターコネクトをシミュレート可能なツールであり，この通信遅延情報の取得に使用可能である．

3.2　アプリケーション・プログラムコードの抽象化

3.1 節で述べた基本コンセプトを実現するためには，実行対象となるプログラムコードを個々の演算処理部分と通信部分へ分割し，かつ，それら個別の代表遅延時間との対応付けが必要となる．これを可能にするため，本節ではスケルトンコードと呼ぶ高度に抽象化した性能評価専用プログラムコードを導入する．

3.2.1　スケルトンコードの導入

ターゲットマシンの動的な振舞いを反映させ，その性能をより正確に予測するためには，何らかの方法で対象アプリケーション・プログラムを実行する必要がある．しかしながら，2 章で説明したように，ターゲットマシンでの実行を想定したアプリケーション・プログラム（とその問題サイズ）をそのままホストマシンで実行することはきわめて難しい．この問題を解決するため，アプリケーション・プログラムのオリジナルコードに抽象化を施し，プログラムの実行結果は保証しないが，その実行時間を推定可能な性能評価専用コードを作成する．本稿ではこれを「スケルトンコード」と呼ぶ．スケルトンコードは主に以下の特徴を有する．

- **演算部分の抽象化：**オリジナルコードを演算部分（連続実行される計算コード部分）と通信部分に分割する．そして，演算部分を実行時間というきわめて高い抽象度で表現する．つまり，演算部分をコメント化し，それに対応するターゲットマシンでの推定実行時間で置き換える．これにより，プログラム実行の振舞いを損なうことなく，短時間でのスケルトンコード実行が可能となる．
- **疑似通信ライブラリの利用：**実行結果を保証する必要がないため，ホストマシン上で対象アプリケーション・プログラムを実行する場合には必ずしも実際に通信を行う必要はない．そのため BSIM がサポートする特別な通信ライブラリ（LMPI ライブラリ）を利用する．このライブラリはその遅延時間のみを性能予測に反映させており，これによりホストマシンでの通信量を大幅に削減し，高速なスケルトンコードの実行を可能にする．
- **使用メモリ容量の削減：**実行結果を保証する必要がないため，オリジナルコードの実行と同じ容量のメモリを使用する必要はない．また，1 つのノードに多数のランクを発生させることにより通常のプログラム実行では発生しないメモリ不足の状態が起こりうる．そこで，ホストマシンでの実行を可能にすべく，使用メモリ容量を削減する．

3.2.2　スケルトンコード作成方法

スケルトンコード作成方法を典型的なコード**図 2** に示す．前項の説明のとおり，スケルトンコード作成のためには主に以下の 3 つの作業が必要である．

1. **演算部分の抽象化**：図 2（右）に示すようにプログラムブロックのコメント化ならびに BSIM_Add_time 関数の挿入によって行う．この BSIM_Add_time 関数の引数には当該

オリジナルコード

```
.......
MPI_Send(...);

double a[1000];
for (i=0; i<1000; i++)
  a[i] = func(i);


MPI_Recv(...);
.......
```

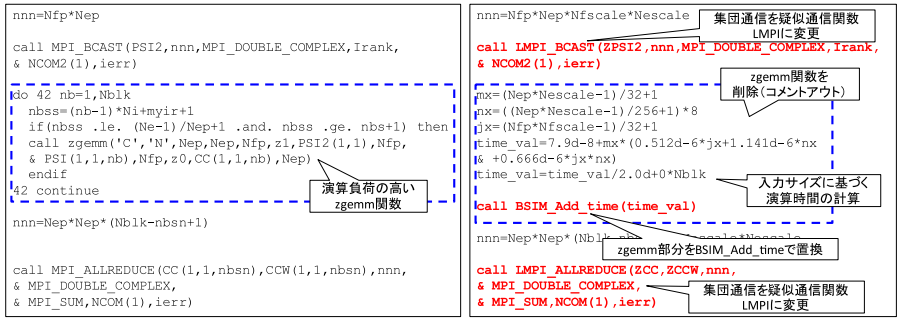
スケルトンコード

```
.......
LMPI_Send(...);
/* コメント
double a[1000];
for (i=0; i<1000; i++)
  a[i] = func(i);
*/
BSIM_Add_time(203e-9);
LMPI_Recv(...);
.......
```

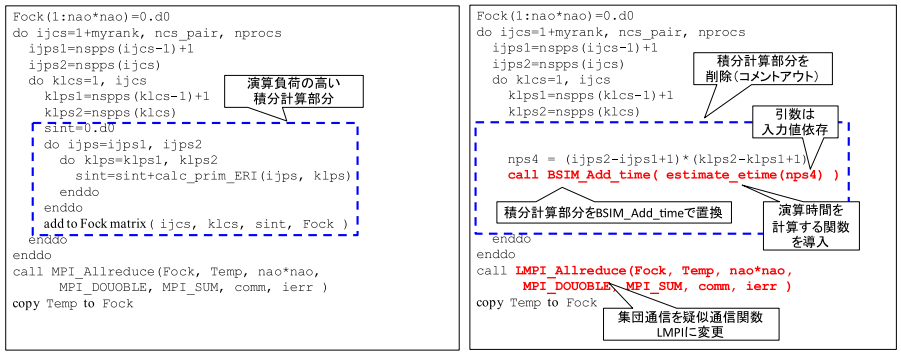
図 2　スケルトンコード化
Fig. 2　Skeleton Code generation.

コメント化された部分の実行時間を記述する．この際，ブロックにループ文などがあり実行時に入力により動的に実行時間が大きく変化する場合では，引数としては絶対実行時間だけでなくモデリングに基づく実行時間評価式を指定する．

2. **通信量の削減（MPI 通信ライブラリ呼び出しの接頭語変換）：**スケルトンコードにおける MPI 通信を LMPI 通信ライブラリへの呼び出しへと変更する．コード変更には MPI_ で始まる通信関数に対し接頭文字を追加し LMPI_ とするのみでよく，引数などの変更は必要ない．なお，LMPI は仮想超並列実行環境 BSIM が解釈可能なライブラリである．実行時には，ホストマシン上で実際の通信を行うことなく通信遅延時間を性能予測に反映させることができる．詳細は 3.5.1 項で説明する．
3. **使用メモリ容量の削減：**基本的には演算部分の抽象化を行う際に不必要となる配列宣言を削除する．また，コメント化箇所以外に配列を使用している場合では，配列宣言部とデータアクセス部を単一のスカラデータに変更する方法や，小規模問題サイズ向けにメモリ領域を確保して演算量や通信量をスケールする方法などがある．ただし，オリジナルコードが有する実行の振舞いを維持できるよう注意する必要がある．



（A）PHASE のオリジナルコード（左）とスケルトンコード（右）



（B）二電子積分計算 のオリジナルコード（左）とスケルトンコード（右）

図 3　PHASE と ERI のスケルトンコード例
Fig. 3　Examples of Skeleton Code (PHASE and ERI).

3.2.3　スケルトンコードとその実行例

具体例として，PHASE ならびに ERI（2 電子積分計算）のスケルトンコードを**図 3** に示す．PHASE は第 1 原理（量子力学）による半導体などの固体の電子状態計算を通じて多様な物性・機能をナノ構造から予測・解明するプログラムであり，計算科学上の重要なプログラムとなっている．図 3 (A) のスケルトン化においては，演算処理である zgemm 関数を BSIM_Add_time に，また，2 カ所の集団通信（MPI_BCAST と MPI_ALLREDUCE）を LMPI に置換している．ここで，BSIM_Add_time 関数への引数となる zgemm 実行時間に関しては，スケーリングパラメータ Nescale と Nfscale を導入した実行時間モデル（変数

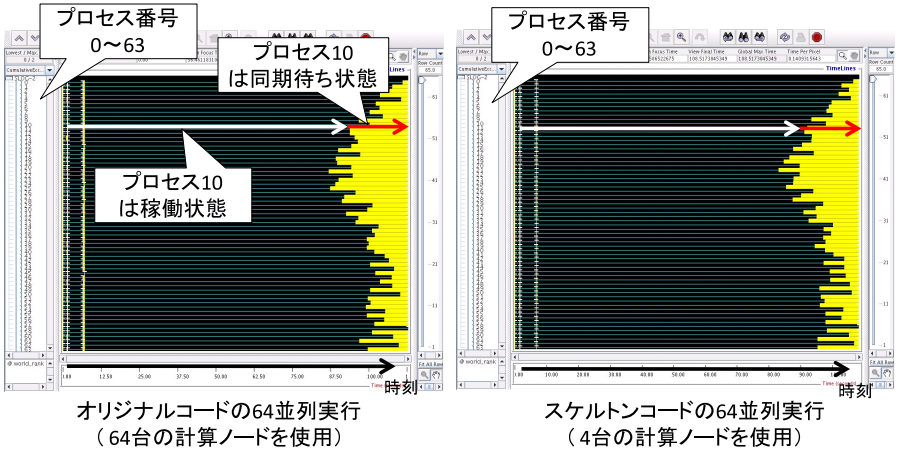


図 4　2 電子積分計算の実行（オリジナルコード vs. スケルトンコード）
Fig. 4　Execution behavior of ERI (Original code vs. Skeleton Code).

time_val への代入式）として与える．これにより，問題サイズを変更した場合でも同一のスケルトンコードを用いた性能評価が可能となる．一方，2 電子積分計算は分子軌道法をはじめとした量子化学計算において最も計算負荷の高い処理の 1 つであり，そのカーネル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．通常は計算量の削減を目的として事前に $O(N^2)$ の計算量を持つ CS ペアに対するカットオフテーブルを作成し，カットオフ処理後に残った積分のみを計算対象とする．図 3 (B) に示すオリジナルコードでは，外側の CS ペアに対するループ（変数 ijcs のループ）を静的に分割することで行っている．ここでは，4 重ループの最内部に位置する原始 2 電子積分関数（calc_prim_ERI），ならびに，2 重ループ内に存在するフォック行列生成関数（Fock_matrix）を抽象化し BSIM_Add_time で置き換えている．ここでも PHASE と同様，引数である抽象化部分の実行時間は入力変数の値に基づき計算する．また，ループ実行完了後に必要となる All_Reduce 通信も LMPI 関数に置き換えることで実行の高速化を狙う．

2 電子積分計算を例とし，そのオリジナルコードならびにスケルトンコードを並列実行した際の様子を**図 4** に示す．横軸は経過時間を表しており，縦軸は MPI のランク（プロセス番号）である．各プロセスにおいて，黒い部分は 2 電子積分計算を行っている時間，黄色い部分は MPI 通信の同期待ち時間である．使用した計算機は理研スーパコンバインドクラスタ（RSCC）の Linux クラスタ部分であり，入力データはグリシン 15 量体（$(Gly)_{15}$）で基底関数は 6-31G*とした（原子数 = 108，基底関数数 = 1,009）．また，スケルトンコードの実行においては 3.5 節で説明する仮想超並列実行環境を用いた．左図は 64 台の計算ノードを使用したオリジナルコードの実行であり，そのときの実行所要時間は約 103 秒であった．これに対し，右図は 4 台の計算ノードを用いた場合のスケルトンコード実行の様子であり，その実行は約 6.7 秒で終了した．つまり，1/8 の計算機資源であるにもかかわらず，約 15 倍の高速化を達成していることになる．また，スケルトンコードで再現された実行の挙動は 64 台の計算ノードを用いたオリジナルコードときわめて類似しており，実行時間の予測誤差はわずか 2%程度である．このように，プログラムコードの高度な抽象化を導入することで高速かつ正確な性能予測が可能となる．

3.3　超高性能プロセッサ・シミュレータ PSIM の開発

3.2 節で説明したスケルトンコードを作成するためには，演算部分の実行時間を正しく見積もらなければならない．予測対象となるターゲットマシン向けプロセッサが実在する場合には実機を用いた測定が可能である．しかしながら，新規プロセッサの開発を考える際には何らかの見積り手段が必要となる．本節では，この要求を満足するために開発したプロセッサ・シミュレータ PSIM について説明する．本シミュレータは SPARC 命令セットを対象としており，本節で説明するマイクロアーキテクチャのシミュレーションが可能である．

3.3.1　超高性能 SIMD 拡張スカラプロセッサ・アーキテクチャ

近年，大規模シミュレーションをはじめとして，HPC 分野における計算能力への要求は高まる一方である．多数のプロセッサを相互に接続して大規模計算機システムを構成する場合，そのネットワーク構成やシステム全体の電力，運用管理などの観点から，計算ノードプロセッサ単体は従来以上に高性能かつ電力（面積）あたりで高効率であることが望まれる．これを実現するために，既存スカラ型のコアプロセッサを基本として，浮動小数点演算ユニットを多数個搭載する SIMD 拡張スカラプロセッサ・アーキテクチャを考案した[8]．高い浮動小数点（以降，FP と略記）演算性能を高いエネルギー効率で実現するために，多数の FP 演算器での SIMD 計算による演算処理の効率化，ならびに，メモリアクセス処理の効率化に関して注力した．まず，ピーク演算処理性能を高めるために FP 演算器数を増加させる．従来のスカラコアを複数個単純に並べるアーキテクチャもありうるが，制御部のハードウェアコストを抑えるために，命令制御部は単一のままで多数の FP 演算器を接続する SIMD 処理方式を採用した．アプリケーション実行において，SIMD 処理により並列計算部を高速化すると逐次計算部の処理時間がボトルネックとなる．そこで，逐次計算に対して

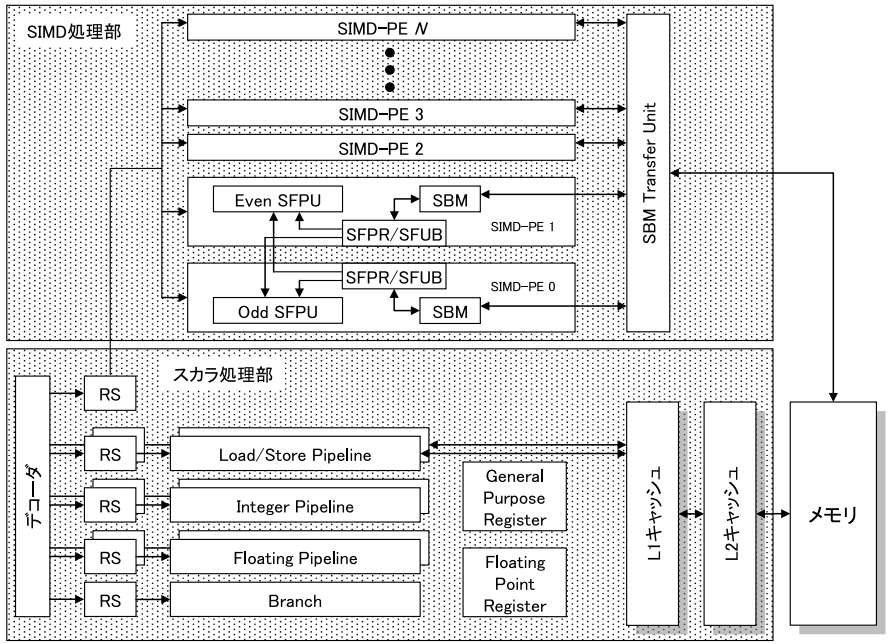


図 5　SIMD 拡張型プロセッサの構成
Fig. 5　Structure of SIMD extended processor.

も高い性能を持つ単一スカラコアを搭載することとした．このような構成により，スカラコアを単純に並べるよりも高い電力効率とピーク性能が実現できる．次に，メモリアクセスの効率化を図るために，SIMD 部に搭載したオンチップメモリ（ローカルメモリ）とスカラコアのキャッシュを有効利用するというアプローチを採用した．SIMD 処理用の大量のデータは，メモリからローカルメモリに直接高速転送する．また，行列の転置処理のようなアドレスが不連続なデータはスカラコアでキャッシュに取り込み SIMD 処理部に転送する．ベクトル型プロセッサなどのストライドアクセスではメモリからのデータの一部しか使用することができないが，キャッシュをバッファとして使うことによりメインメモリバンド幅を有効に利用することができる．

図 5 に SIMD 拡張スカラプロセッサの構成を示す．プロセッサコアは，主として以下に示すスカラコア部と SIMD 演算部からなる．スカラコア部は SPARC64 V[30] をベースとしており，SPARC-V9 アーキテクチャで定義された命令を実行する部分である．また，これに加え，SIMD 処理部で実行する命令の制御（発行やコミット制御など）も行う．一方，SIMD 処理部は SPARC-V9 アーキテクチャで定義されている FSQRT 命令および FDIV 命令を除くすべての FP 演算命令と後述する新規 SIMD 命令を並列実行する．以下，主な特徴をまとめる．

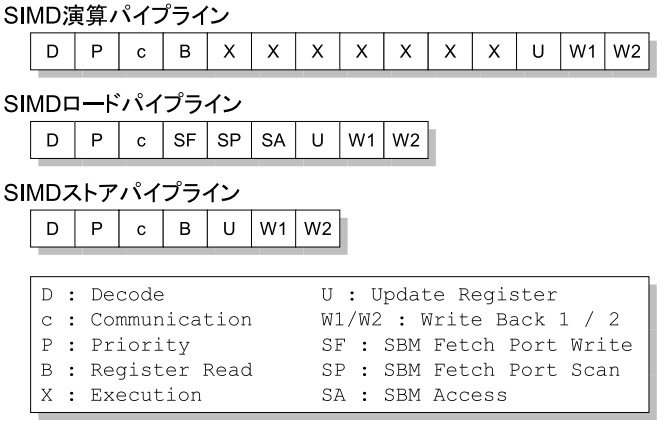


図 6　SIMD 実行パイプラインの構成
Fig. 6　Structure of SIMD execution pipeline.

- **最大 32 個の SIMD-PE（SIMD Processing Element）を搭載：**各 SIMD-PE は 1 個の FP 演算器（SFPU），SIMD FP レジスタ（SFPR）および SIMD Buffer Memory（SBM）を有する．SFPU は，1 本の SIMD 演算パイプライン，2 本の SIMD ロードパイプライン，ならびに，1 本の SIMD ストアパイプラインで構成される．デコードステージ以降のパイプライン構成を**図 6** に示す．基本的に SPARC64 V の実行パイプラインと同様の動作を行うが，スカラコア部から SIMD-PE への命令送信やコミット指令の送信が長距離配線となるため，「c ステージ（Communication）ステージ」ならびに「W1 ステージ（Write back 1）ステージ」を追加した．C ステージはスカラコアから SIMD-PE へのデータ転送に，また，W1 ステージはコミット処理のための指示を SIMD 処理部からスカラ処理部に転送するために必要となる．なお，SIMD-PE での命令はアウトオブオーダで実行され，コミットの制御はスカラコア部で行う．

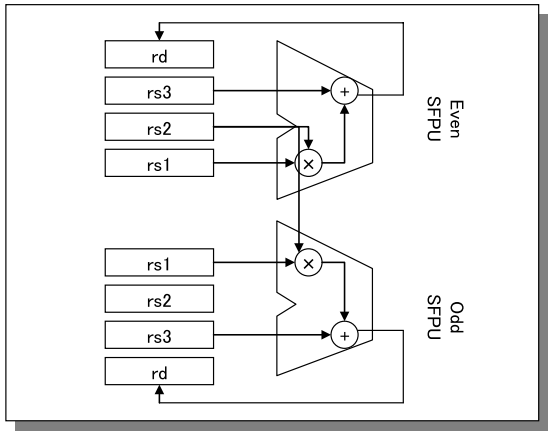


図 7　SIMD-PE ペアの演算
Fig. 7　Example of execution with SIMD-PE pair.

- **全 SIMD-PE への命令ブロードキャスト：**SIMD-PE はスカラ部のデコーダと RS（Reservation Station）に直結されており，プロセッサコアの SIMD 状態ビットをデコーダがチェックする．このビットがセットされた状態で FP 演算命令が現れた場合は，SIMD 命令と解釈してすべての SIMD-PE に同一の命令を発行する．一方，SIMD 状態ビットがリセットされた状態では，FP 演算命令はスカラ処理部の FP 演算器で処理され，SIMD 処理部は動作しない．
- **クロス演算のサポート：**2 つの SFPU を「Even SFPU」と「Odd SFPU」としてペアで用いる．これにより，それぞれの SFPR に格納されている値を相互に参照（クロス参照）して演算することが可能となる．**図 7** に演算例（FXMADDD 命令）を示す．この命令では，奇数 SIMD-PE が他方の rs2（第 2 ソースオペランド）のレジスタ値を入力として演算を行い，自身の SFPR に演算結果を格納する．この機能は，特に，SIMD 処理部において複素数演算を実行する場合に有効利用できる．
- **SIMD Buffer Memory（SBM）の搭載：**スカラ処理部の L1 キャッシュに相当する小容量のバッファメモリ領域．SIMD-PE は SBM に格納されているデータを SFPR に転送し，それをオペランドとして演算を行う．キャッシュメモリとは異なり，別アドレス空間のローカルメモリとしてプログラムによってアドレス指定してアクセスできる．格納するデータをプログラムで完全制御できるため，効率的な SIMD 演算を実現するようにデータを配置することが可能となる．

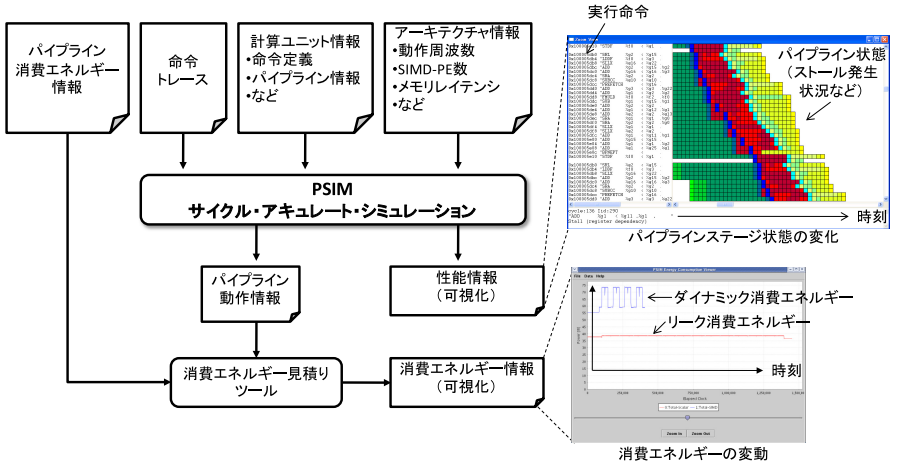


図 8　PSIM の入力と出力
Fig. 8　Inputs and outputs of PSIM.

なお，新規追加した SIMD 拡張命令を出力する FORTRAN コンパイラも合わせて開発した．

3.3.2　PSIM の特徴とその実装

3.3.1 項で提案したマイクロプロセッサの実チップは存在しない．そこで，本アーキテクチャの性能や消費エネルギーを評価するため，サイクルアキュレートなプロセッサ・シミュレータを開発した．プロセッサ内部の各種機能ユニットやその動作をクロックサイクル単位でアーキテクチャシミュレーションする．これにより，各種性能評価情報や消費エネルギー情報を取得することができる．また，マイクロプロセッサを構成する各種資源などの仕様をシミュレータへの入力パラメータとして柔軟に変更でき，コストや性能，電力などを考慮したアーキテクチャ探索が可能となる．**図 8** にシミュレータへの入力ならびに出力を示す．性能に関しては，シミュレーション対象命令トレース部分の実行時間だけでなく，各パイプライン・ステージ状態の変化を可視化している．図 8 の右上の画面には命令列実行における各パイプライン・ステージの様子を表示しており，これによりパイプライン・ストール発生箇所の特定やその原因追跡が可能となる．一方，消費エネルギーに関しては SPARC64 V プ

表 1　プロセッサ構成
Table 1　Processor configuration.

	パラメータ名	パラメータ値
スカラコア部	命令発行幅	4命令 / サイクル
	演算パイプライン数	INT : 2, FP : 2, LD/ST : 2
	INT/FPリザベーションステーション	INT: 8エントリ×2，FP: 8エントリ×2
	リオーダバッファ	64エントリ
	L1キャッシュ	命令128KB（連想度2），データ128KB（連想度2）
SIMD処理部	SIMD-PE数	16
	SIMD Reservation Station	16エントリ
	SBM	64KB×16
	SBMからメモリへのoutstandingリクエスト数	16エントリ
メモリ	L2キャッシュ	共有6MB（連想度12）
	メモリアクセスレイテンシ	120クロックサイクル

ロセッサの設計データを分析し，各パイプライン・ステージ動作時の消費エネルギー情報を求めた．これに加え，プロセッサ・シミュレーションにより得たパイプライン動作情報を用いることで時刻経過における消費エネルギーを算出することができる．

ここで，既存のマイクロプロセッサ・シミュレーション技術と比較する．代表的なツールとしては SimpleScalar [13] やマルチコアのシミュレーションを前提とした M5 [16] があげられる．また，消費電力に関しては Wattch [17] や SimplePower [28] などが開発されている．これらプロセッサ・シミュレータにおいては本稿で示した大規模な SIMD 拡張は行われていない．これに対し，PSIM は今後の高性能プロセッサにおいて重要となる大規模な SIMD 実行をサポートしており，かつ，それに対応したコンパイラも開発している．また，消費電力の見積りにおいては，パイプライン単位で動作するラッチ数を把握するという従来とは異なるアプローチをとっている．

3.3.3　シミュレータを用いたプロセッサ性能と消費エネルギー評価

開発したシミュレータを用いて提案する SIMD 拡張型スカラプロセッサの性能ならびに消費エネルギーを評価した．プロセッサの構成は**表 1** のとおりである．対象アプリケーションとしては，実数の行列積を計算する dgemm，複素数の行列積を計算する zgemm，ならびに，フーリエ変換を計算する fft を用いた．dgemm はスパコン性能測定に広く用いられる HPL（High Performance Linpack）の計算処理の大部分を占める関数である．一方，zgemm と fft は 3.2.2 項で説明した PHASE において主要な計算部分となる．なお，本性能評価においては，スカラコアからのメモリアクセスはすべてプリフェッチにより必要なデータを事前にキャッシュに格納することが可能であると仮定する．評価結果を**表 2** に示す．ここでは，90 nm バルク CMOS テクノロジ，プロセッサ動作周波数は 2 GHz，電源電圧は 1.0 V を想定している．この結果より，dgemm に関しては 7 倍近い性能向上を達成しており，かつ，消費エネルギーも半分以下となっている．また，ハードウェアコストの見積りを行った結果，SIMD 拡張型プロセッサの面積はスカラコアの約 1.5 倍程度であることが分かった．

表 2　SIMD 拡張型プロセッサの性能と消費エネルギー
Table 2　Performance and energy consumption of SIMD extended scalar processor.

	実行時間			消費エネルギー		
	スカラコア [ms]	+16SIMD [ms]	性能倍率	スカラコア [mJ]	+16SIMD [mJ]	消費エネルギー倍率
dgemm 1024元	368.9	53.12	6.9x	27.9	13.1	0.47x
zgemm 1024元	1114.6	185.82	6.0x	93.3	34.4	0.37x
fft 1024点x8	0.1764	0.0668	2.6x	7.59	4.13	0.54x

消費エネルギーに関しては理想クロックゲートを前提（リーク電力込み）

3.4　超大規模インターコネクト・シミュレータ NSIM の開発

3.2.2 項で説明したスケルトンコードの作成においては，演算部分の実行時間だけでなく，通信遅延時間（特に長い時間を要する集団通信）の見積りも重要となる．多くの場合でインターコネクトの規模や機能は世代ごとに異なるため，何らかの見積り手段が必要である．本節では，大規模な相互結合網を対象とした通信遅延時間の見積りが可能な高速インターコネクト・シミュレータ NSIM の詳細を説明する．

3.4.1　NSIM の特徴

近年のスーパコンピュータの設計開発においては，プロセッサやメモリシステムに代表される計算ノードだけでなく，それらを相互に接続するインターコネクトの設計がきわめて重要となる．数万から数十万といった多数の計算ノードを接続するためのネットワーク・トポロジ，スイッチやルータにおける各種パラメータ（バッファ容量など），インジェクション・レートなど様々な最適化を行う必要がある．システム開発前の性能評価や事前最適化を可能にすべく，これまでにもいくつかのインターコネクト・シミュレータが開発されてきた[4),6),10),19)]．これらのシミュレータではシステムの挙動を細部にわたりモデル化する方針がとられており，より精度の高い性能予測が可能となる．しかしながら，その一方で，近年のスーパコンピュータは飛躍的な性能向上とともに実行するアプリケーションも含めて大規

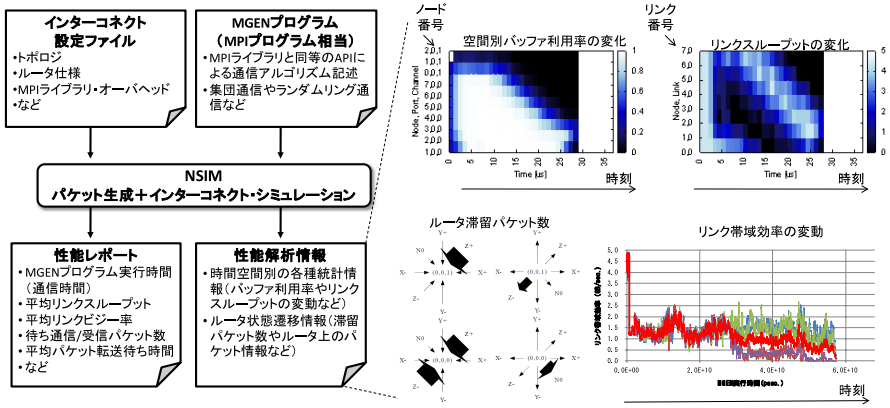


図 9　NSIM の入出力
Fig. 9　Inputs and outputs of NSIM.

模化かつ複雑化の一途をたどっており，従来手法による性能評価ではシミュレーションに費やす時間は非常に大きくなるといった問題が生じる．

そこで我々は，大規模なインターコネクトを高速かつ実用時間内で評価するためにインターコネクト・シミュレータ NSIM を開発した．従来，パケット転送方式であるバーチャル・カットスルーをシミュレーションする際にフリットの挙動を詳細にモデル化していた．NSIM では，これを単一パケットの挙動として集約することで大幅な高速化を達成している．また，スイッチやルータの動作をサイクルレベルで模擬するのではなく，「1 パケットのルータ通過時間」のように粒度の粗い処理単位で遅延時間を積算する．このように，抽象度を従来よりも高くしたハードウェアや通信遅延時間のモデル化により高速なシミュレーションを実現している．さらに，NSIM は MPI で実装しており，デスクトップ PC からスーパコンピュータまで，幅広い実行プラットフォーム上で用途に応じた利用形態をとれる．

NSIM の入出力を**図 9** に示す．NSIM は以下の 2 つのファイルを入力とする．

- **インターコネクト設定ファイル：**シミュレーション対象となるインターコネクトの構成や仕様を指定する．サポートするトポロジは，一般的な 1 次元/2 次元/3 次元メッシュ網，1 次元/2 次元/3 次元トーラス網，Fat-Tree 網，ならびに，文献 11) のような 6 次元メッシュである．また，各ルータやスイッチの遅延時間といったハードウェア情報，

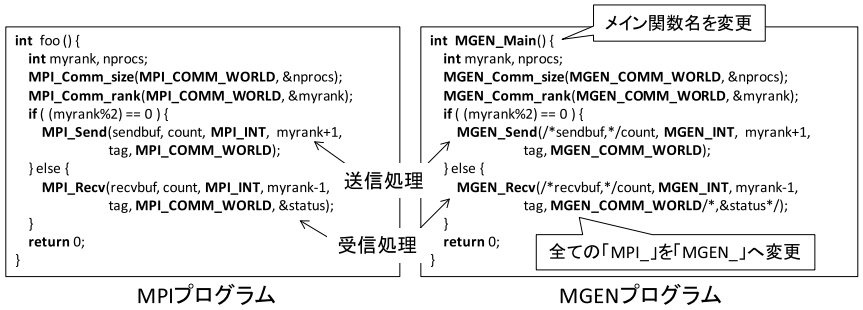


図 10　MGEN プログラムの例
Fig. 10　Example of MGEN program.

さらには，MPI 通信ライブラリ実行にともなうオーバヘッド時間など，インターコネクトに関する各種パラメータ値を設定することができる．

- **MGEN プログラム：**通信性能の評価を目的とした並列化プログラムであり，専用の API を用いて集団通信アルゴリズムを記述する．**図 10** に示すように，NSIM では MPI 互換の API（MGEN API と呼ぶ）を提供しており，ユーザは MPI プログラムを記述するのと同等の感覚で MGEN プログラムを作成できる．実際，MPI と MGEN API との違いは宣言のプレフィクスのみである．したがって，MPI プログラムにおいて文字列「MPI_」を「MGEN_」に置換し，かつ，main() 関数名を MGEN_Main() 関数に置き換えるだけでよい．

一方，出力に関しては以下のような情報がある．

- **性能レポート：**MGEN プログラム実行時間（つまり，MGEN プログラムに記述した全通信の完了までに要する時間）や平均リンクスループットなどの各種統計情報を出力する．これに加え，すべてのノードで受信したパケット数，リンクの帯域効率，リンクのビジー率などもシミュレーション結果として出力される．
- **性能解析情報：**より詳細な分析を行うための詳細情報を出力する．具体的には，時間的/空間的に観測ポイントを指定し，該当するインターコネクト部分の挙動を観測することができる．たとえば，相互結合網のリンクスループット，リンク転送量，バッファ利用率，パケットの転送遅延時間に関する統計情報，仮想チャネルバッファに滞留して

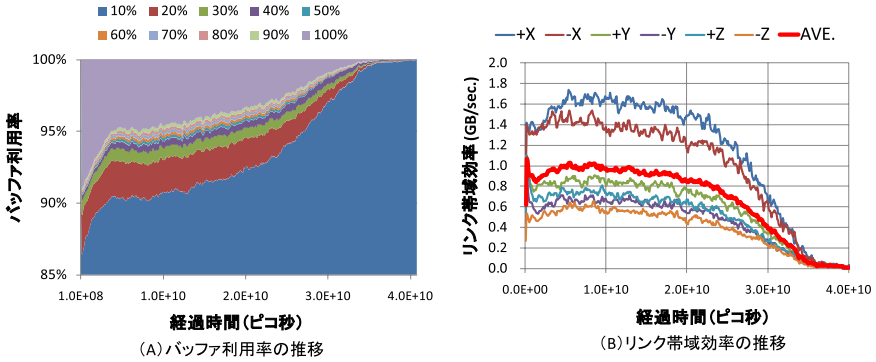


図 11　インターコネクト内部状態の可視化
Fig. 11　Visualizing inside of interconnect.

いるパケットの情報，などを出力可能である．これらの情報を可視化することにより，通信混雑や輻輳の原因解析などを支援する．ランダムリング通信を対象とした 3 次元トーラス網のシミュレーション結果例を**図 11** に示す．図 11 (A) は，単位時間あたりの仮想チャネルバッファの利用率を 10 階級で分け，その割合の時系列変化を示したものである．すなわち，プログラム実行にともなうバッファの利用状況の推移を把握することができ，ランダムリング通信では，ほとんどのバッファが 10%以下の利用率であることが分かる．同様に，図 11 (B) は，リンク帯域効率の時系列変化を座標方向（±X，±Y，±Z）ごと，ならびに，それらの平均を示している．

従来開発されてきたインターコネクト・シミュレータのほとんどは，あらかじめ取得した通信ログを入力としてシミュレーションを行う．そのため，将来の通信パターンが送信/受信タイミングに依存する場合には対応できないという問題があった．これに対し，NSIM では MGEN プログラムを入力とし，動的にメッセージならびにパケットを生成する．すなわち，自己メッセージ/パケット生成機能を搭載しており，インターコネクト・シミュレーションとメッセージ/パケット生成をインタラクティブに実施する．これにより，通信遅延時間に依存して将来の通信パターンが異なる場合にも対応できる．また，このような特徴を利用することで，通信中に発生する OS ジッタやパケットペーシングを反映した性能評価も可能となる．さらに，従来研究では，プログラム実行を通して平均した帯域効率を示すことが多いが，NSIM は高い観測性を持つため様々な観点からインターコネクトの挙動を俯瞰することができる．

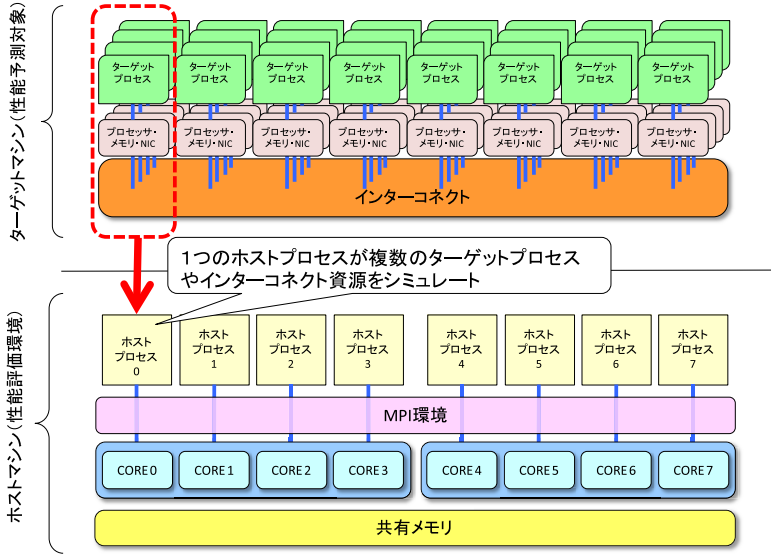


図 12　8 個のプロセッサコアを用いた NSIM の並列実行の様子
Fig. 12　Parallel execution of NSIM with 8 cores.

ここで，既存のインターコネクト性能予測技術と比較する．代表的な研究開発として，INSIGHT [4] や INSPIRE [6]，IBM BlueGene/L 用シミュレータ [10]，BigNetSim [19] などがあげられる．これらは通信ログを入力とするため 2 章で述べた問題が生じる．これに対し，3.4 節で説明した NSIM は，MPI 相当のプログラム実行による通信イベント生成とインターコネクト・シミュレーションをインタラクティブに行うため通信ログを必要としない．一方，ある程度の精度低下を許容し高速化を実現する方法として FSIN [24] があるが，このシミュレータは並列化されていないため現在のマルチコア環境や PC クラスタの計算能力を十分に活用することはできない．

3.4.2　内部構成と動作フロー

NSIM は，パケットレベルでの並列離散事象シミュレーションを行う．ここで，離散事象シミュレーションとは，対象システムの状態を変化させる事象（イベント）を離散的な時刻に発生させるシミュレーション方式である．**図 12** に示すように，NSIM は複数プロセッサ上で並列に離散事象シミュレーションを行う．ここで，「ターゲットプロセス」とは

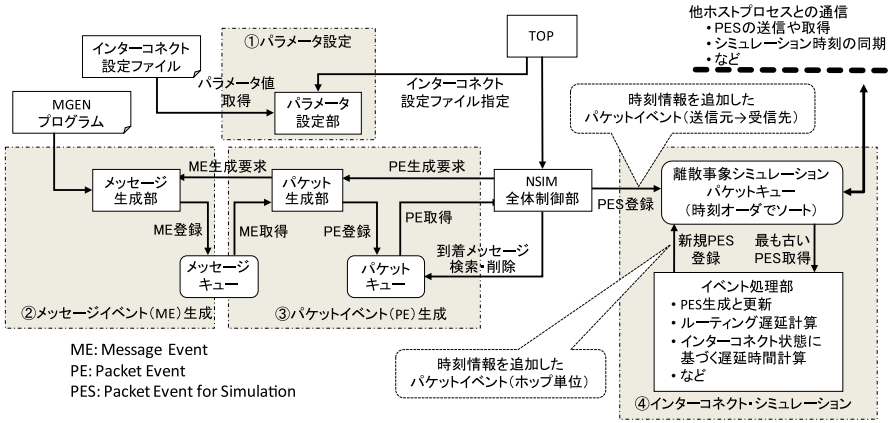


図 13　NSIM 内部構成とシミュレーション・フロー
Fig. 13　Organization and simulation flow of NSIM.

MGEN プログラムに記述されたシミュレーション対象となるプロセスを表し，「ホストプロセス」は並列化された NSIM 実行のためのプロセスである．この図の例では，32 並列で実行可能なターゲットマシンを 8 個のプロセッサコアでシミュレーションする場合を示しており，ターゲットプロセス数が 32，ホストプロセス数が 8 となる．なお，NSIM は複数ターゲットプロセスを単一ホストプロセスに割り当てるため，異なるホストプロセスに割り当てられたターゲットプロセス間で通信が発生する場合には，インターコネクト・シミュレーション中にホストプロセス間での通信が必要となる．NSIM の内部構成と実行フローを**図 13** に示す．NSIM は，主に，① パラメータ設定，② メッセージイベント生成，③ パケットイベント生成，④ インターコネクト・シミュレーションの 4 つの主要処理部で構成される．それぞれの処理内容は以下のとおりである．

1. **各種パラメータの設定：**インターコネクト設定ファイルより各種パラメータ値を読み出しシミュレーションの対象となるインターコネクト情報として設定する．
2. **メッセージの生成：**MGEN プログラムにおいて，MGEN API で記述されたメッセージ送信もしくは受信に対応したメッセージレベルのイベント（以降，メッセージイベントと呼ぶ）を生成する．また，当該イベントをメッセージ・キューに投入する．このメッセージ生成は，キューが一杯になる，もしくは，受信用のメッセージイベントがキューイングされるまで継続される．
3. **パケットの生成：**メッセージ・キューからメッセージイベントを取り出してパケットレベルのイベント（以降，パケットイベントと呼ぶ）へ分解し，パケット・キューに投入する．メッセージ・キュー内のすべてのイベントの変換処理が終了した場合（すなわち，メッセージ・キューにメッセージイベントがなくなった場合）には処理フロー 2. へ戻る．
4. **インターコネクト・シミュレーション：**パケット・キューからパケットイベントを取り出す．そして，シミュレーション時刻（タイムスタンプ）を付加して離散事象シミュレーションの操作対象となるイベント（以降，シミュレーション用パケットイベントと呼ぶ）を生成する．また，当該イベントを離散事象シミュレーション・パケットキューへ投入する．キュー内のイベントはタイムスタンプの値でソートされており，キューの先頭となる時刻の最も古いイベントから順に処理される．先頭イベントがパケット送信を表す場合には，1 ホップごとの転送に対応する新たなシミュレーション用パケットイベントを生成してキューに再投入する．一方，受信に相当するイベントの場合には受信すべきパケットの到着を待つ．このような操作は図 13 のイベント処理部で行われ，その際にインターコネクト・トポロジや通信衝突などの影響がタイムスタンプに反映される．1 個のメッセージを構成する全パケットのイベント処理が完了すると，当該メッセージイベントを消滅させる．このような処理を繰り返すことでシミュレーション時刻を進めていく．もし，シミュレーションの進行によりメッセージ・キュー内にメッセージイベントがなくなった場合には処理フロー 2. へ戻る．

3.4.3　NSIM の精度ならびに性能評価

NSIM のインターコネクト・シミュレーション能力を評価するため，3 次元トーラス網を前提とした評価実験を行った．ここでは，様々な状況での使用を想定し，2 種類の通信パターン，ならびに，4 種類の実験環境を用いてインターコネクト・シミュレーションを実施した．通信パターンである評価アプリケーションに関しては，HPC Challenge の Communication bandwidth and latency ベンチマークで用いられているランダムリング通信（メッセージサイズは 1 MB，ベンチマーク内のループ回数は 5），ならびに，多くの通信が発生する全対全通信（メッセージサイズは 4 B，Pairwise Exchange アルゴリズム）を選択した．実験結果を**表 3** に示す．クアッドコアプロセッサを 2 基搭載したメモリ共有型ワークステーションを使用した場合には 8 K〜16 K プロセスを対象としたシミュレーションを行うことができ，このときの所要時間はいずれも 3 時間以内であった．一方，PC クラスタを用いた実験では 128 K プロセスのランダムリング通信に関するシミュレーションを完了しており，所

表 3　3 次元トーラス網を対象とした NSIM シミュレーション所要時間
Table 3　NSIM execution time for 3D-torus simulations.

通信パタン	評価用ホストマシン		1K〜8Kターゲットプロセスを対象とした場合のシミュレーション所要時間 [分]							
	機種	コア数	1K	2K	4K	8K	16K	32K	64K	128K
ランダムリング通信	PCクラスタ（Intel Xeon 3.0 GHz）	128	27.1	40.6	53.0	105.6	163.2	239.6	544.2	1,025.0
		64	21.3	33.2	45.4	96.1	162.2	273.0	665.0	3,357.8
	デスクトップPC（Intel Xeon 3.2 GHz）	4	9.9	23.4	50.6	166.5	—	—	—	—
全対全通信（Pairwise Exchange）	デスクトップPC（Intel Xeon 2.8 GHz）	8	0.2	0.7	3.3	17.5	119.1	—	—	—

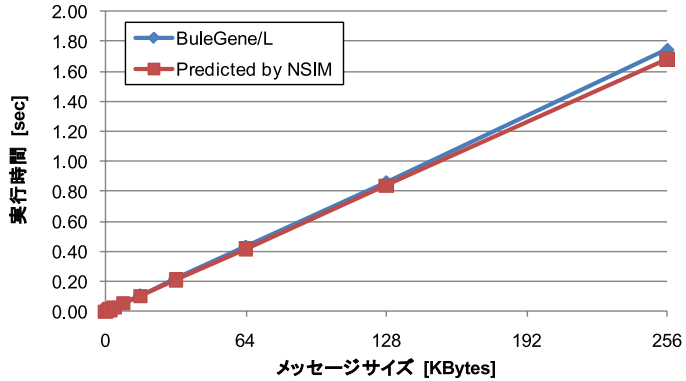


図 14　NSIM の通信性能予測精度
Fig. 14　Accuracy of NSIM simulation.

要時間は 128 コア使用の場合で約 17 時間である．これは，現実的な時間内でのシミュレーションであると考える．

また，ターゲットマシンとして IBM 社の BlueGene/L を利用した精度評価を行った．ここで，使用したノード数は 128 であり，Bruck アルゴリズムの全対全通信を対象としており，各種インターコネクト・パラメータに関しては文献 12) を参考に決定している．また，1 ホップあたりの通信遅延は 90 ns，スイッチあたり 154 MB/s とした．実験結果を**図 14** に示す．縦軸は全対全通信実行時間，横軸はメッセージサイズである．青実線が BlueGene/L の実機で測定した値，赤実線が NSIM による予測値である．メッセージサイズが大きくなると若干誤差が大きくなっているものの，精度良く通信時間を予測できていることが分かる．実際，メッセージサイズが 2 K バイトより大きい場合，誤差は 5%未満であった．なお，メッセージサイズが 64 KB，128 KB，256 KB のときのシミュレーション時間は，それぞれ，44 秒，145 秒，923 秒であった．

3.5　仮想超並列実行環境 BSIM の開発

3.1 節で示した本性能評価法の基本コンセプトを具現化するためには，3.2 節で導入したスケルトンコードを実行し，かつ，性能予測対象であるターゲットマシン上での実行時間を予測する環境が必要となる．そこで，既存の実機マシン上で直接実行可能な仮想超並列実行環境 BSIM を開発した．本節ではその詳細を説明する．

3.5.1　BSIM の機能

BSIM（Base SIMulator）は，Multi-Processing Environment（MPE）を拡張して構築されたスケルトンコードの仮想超並列実行環境である[26]．たとえば，3.2.2 項で示した ERI スケルトンコードの実行においては，4 台の計算ノードを有するホストマシンを用いて，64 台の計算ノードからなるターゲットマシンの性能を予測している．この場合，BSIM は実際に 64 個のプロセスを起動し，4 台の計算ノードで実行する．各プロセスには個別の仮想タイマが割り当てられ，これらを適切に更新することで 64 台の計算ノードを用いた並列実行時の時間経過を模擬する．このような機能を実装すべく，BSIM は以下に示す 2 種類の API をサポートしている．

- **BSIM_Add_time**：各プロセスの時刻を進めるための API であり，引数として渡される値を当該プロセスの仮想タイマに積算する．これにより，スケルトンコードにおいて抽象化された計算コード部分の実行を行うことなくプログラム実行時刻を進めることができる．
- **LMPI**：通信レイテンシを考慮したプログラム実行時間予測を可能にするための API であり，各 MPI 関数に対応する LMPI 関数を用意している．スケルトンコード中で本関数が呼び出された際には，実際に計算ノード間での通信は行わず（もしくはきわめて短いメッセージの通信のみを行う），当該通信に関するレイテンシを仮想タイマに積算する．ここで，通信レイテンシは入力ファイルである通信性能情報でモデル化しており，送信/受信プロセス番号や通信データサイズ，インターコネクト・トポロジなどに応じて適切な値を取得する．たとえば，受信通信レイテンシに関しては，メッセージサイズ，送信元ランク，ならびに，受信先ランクをパラメータとする関数で表現してい

表 4　テラフロップス・ターゲット性能予測環境
Table 4　Target and host machines for tera-scale performance prediction.

システム名		ノード数（ジョブ当たり）	プロセッサ数（ノード当たり）	プロセッサ仕様	メモリ容量（ノード当たり）	インターコネクト	理論ピーク性能
ターゲットマシン	PRIMEQUEST 580	16	32	Dual-Core 1.6 GHz Itanium 2	128 GB	InfiniBand	6.6 TFlop/s
ホストマシン	PRIMERGY RX200 S2	16	2	3 GHz Xeon	7 GB	Gigabit Ethernet	192 GFlop/s
	Desktop PC	2	2	1.6 GHz Xeon	10 GB	Gigabit Ethernet	51 GFlop/s

る．本 API の利用により，たとえば集団通信のように長い時間を要する処理を省略しスケルトンコードの実行時間を短縮できる．

なお，通信に依存関係（たとえば送信とそれに対応する受信）がある場合には，スケルトンコード実行中に送信プロセスと受信プロセスの仮想タイマを参照してこれらを適切に更新する．したがって，通信依存に起因する待ち時間の発生も正しく反映される．このほかにも，BSIM は通信ログ出力機能を有しており，広く用いられている Jumpshot といった可視化ツールを使用してプログラム実行の詳細を解析することができる．また，通信イベントだけでなく，特定のイベント発生時刻を通信ログに出力する機能もサポートしており，プログラム・チューニングにも利用可能である．

3.5.2　テラフロップス級スーパコンピュータを対象とした性能予測実験

実在するテラフロップス・ターゲットマシンを対象とし，BSIM による性能予測実験を実施した．これにより，BSIM の性能予測精度ならびに予測所要時間を評価する．実験環境を**表 4** にまとめる．性能予測の対象となるターゲットマシンとしては，九州大学情報基盤研究開発センターに設置されたスーパコンピュータ PRIMEQUEST580（以降 PQ580 と略す）を利用した．1,024 個のプロセッサコアによる並列計算が可能であり，ピーク性能は 6.6 テラフロップスである．一方，性能予測を実施するホストマシンは 2 種類のギガフロップス級 PC クラスタを用いており，それぞれの性能は 192 ギガフロップスならびに 51 ギガフロップスである．性能予測対象となるベンチマーク・プログラムとしては，HPL（密行列連立一次方程式），PHASE（第 1 原理擬ポテンシャルバンド計算ソフトウエア），ならびに，FMO-ERI（2 電子積分計算）を用いた．以下，PHASE を対象とした実験手順の詳細を示す．なお，HPL や FMO-ERI に関しても基本的な内容は同じである．

1. **超並列化オリジナルコード開発：**1,024 個のプロセッサコアを用いることを前提としオリジナルコードの超並列化を行う．ここでは，波数方向とバンド方向の 2 方向への並列化を同時に行う 2 次元分割とすることにより，数万ノードに拡張できる超並列化を行った．また，高負荷部として抽出した 3 つの処理部（擬ポテンシャル積部，Gram-Schmidt 直交化部，3 次元 FFT 部）を結合しカーネルコードとした．これら 3 つの超並列処理部では，それぞれが使用するデータの並びが異なるため，カーネルコード間でデータ並べ替えのためのプロセス間データ通信が大量に発生する．
2. **スケルトンコード作成：**上記 1. で並列化したオリジナルコードに対応するスケルトンコードを人手で作成する．具体的には，3.2.2 項の図 3 (A) で示したようなスケルトンコード化を上記 1. で作成したカーネルコードに適用する．なお，3.2.2 項の図 3 (A) で示した実行時間式の各係数に関しては，PQ580 の計算ノード 1 台（正確には 1 コア）を用いた簡易逐次実行に基づきフィッティング関数を作成することで求めている．
3. **通信性能情報の作成：**LMPI 関数実行時に仮想タイマへ加算すべき通信遅延時間算出のためのデータベースを作成する．BSIM では，送信，受信，集団通信ごとに通信遅延時間見積りを行う．通信遅延時間に関して，送信の場合は遅延ゼロ，受信の場合は「$l + s/w$」のモデル式で近似した．ここで，l は最小通信遅延時間，s はメッセージサイズ，w はバンド幅を表す．l ならびに w の値に関しては，PQ580 を用いた簡易実行による実測に基づき，それぞれ 40 マイクロ秒ならびに 100 MB/s とした．一方，集団通信に関しては**図 15** に示すような基礎データを実機上で事前に採取する．図 15 において，縦軸は集団通信遅延時間，横軸は通信対象となるデータサイズ，各線はコミュニケータ・サイズ（プロセス数）である．そして，このような実測データに基づき集団通信遅延時間の評価関数をテーブル形式で作成した．本関数は，コミュニケータ・サイズとデータサイズを入力とし，それに対応する通信遅延時間を返す．なお，テーブル内に対応するエントリが存在しない場合は内挿により近似値を求め出力する．
4. **ターゲットマシン性能の予測：**表 4 で示した PRIMERGY をホストマシンとし，上記 2. で作成開発したスケルトンコードを BSIM 環境下において実行する．これにより，表 4 に示す 6.6 テラフロップス・ターゲットマシンでのプログラム実行時間を予測する．

性能予測結果を**表 5**，**表 6**，ならびに，**表 7** に示す．それぞれの表において，「実測」は

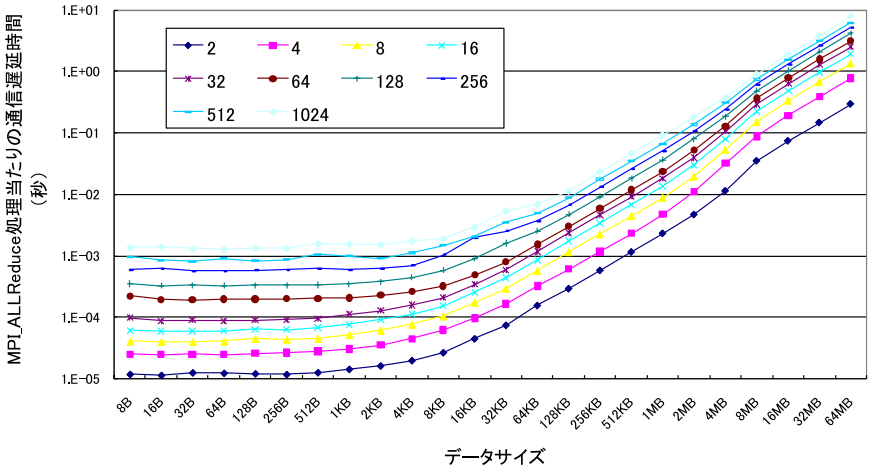


図 15　PQ580 における Allreduce 集団通信の遅延時間
Fig. 15　Allreduce latencies on PQ580.

表 5　PQ580 を対象とした性能予測結果（PHASE）
Table 5　Performance prediction results for PQ580 (PHASE).

バンド数	プロセス数（要確認）	全実行時間（演算＋通信）			演算のみの実行時間		
		予測 [sec]	実測 [sec]	誤差	予測 [sec]	実測 [sec]	誤差
4,096	1,024	23.5	28.8	−19%	11.6	12.3	−5%
8,192	1,024	84.0	97.8	−14%	39.8	39.2	2%
16,384	1,024	301	336	−10%	146	148	−1%

PQ580 によるオリジナルコード（並列化済み）の実行時間を表す．これに対し，「予測」はホストマシンを用いてスケルトンコードを実行することにより得た予測実行時間である．まず，性能予測精度に関して議論する．HPL に関しては，通信を含む全実行時間に関して ±10%程度，通信を無視した演算時間のみに着目した場合は ±5%未満の誤差でターゲットマシンの性能を予測している．これは，スケルトンコードによるプログラム抽象化を行った場合においてもなお，オリジナルコードを用いたターゲットマシンでの実行挙動を正しく再現できていることを意味する．一方，PHASE に関しては，HPL と同様に演算時間の予測は高い精

表 6　PQ580 を対象とした性能予測結果（HPL）
Table 6　Performance prediction results for PQ580 (HPL).

問題サイズ	プロセス数	ブロックサイズ	全実行時間（演算＋通信）			演算のみの実行時間		
			予測 [sec]	実測 [sec]	誤差	予測 [sec]	実測 [sec]	誤差
160,000	1,024	120	707	770	−8%	468	481	−3%
240,000	1,024	120	2,047	2,155	−5%	1,548	1,580	−2%
320,000	1,024	120	4,484	4,348	3%	3,630	3,624	<1%
160,000	256	120	2,168	2,043	6%	1,814	1,780	2%
160,000	512	120	1,156	1,119	3%	915	905	1%
320,000	1,024	180	4,543	4,562	<1%	3,684	3,764	−1%

表 7　PQ580 を対象とした性能予測結果（FMO-ERI）
Table 7　Performance prediction results for PQ580 (FMO-ERI).

対象分子	基底関数	全実行時間（演算＋通信）			演算のみの実行時間		
		予測 [sec]	実測 [sec]	誤差	予測 [sec]	実測 [sec]	誤差
Lysozyme	STO-3G 1961原子 60005関数	889.6	926.6	4%	880.6	889.7	1%

度で実現できており，全実行時間に関しても 10%〜20%の誤差である．大規模スーパコンピュータの性能予測において 10%〜20%程度の誤差は十分許容できる範囲であると考える．いずれのプログラムにおいても，演算時間と比較して通信遅延時間の見積り誤差が大きく，特に多くの集団通信を必要とする PHASE ではこの影響が顕著に現れている．FMO-ERI に関しても同様の傾向にあるが，本プログラムでは全実行時間のうち 95%以上が演算処理となるため，通信時間の予測誤差の影響は小さい．一般に計算ノードをまたぐ通信時間は様々な外乱による影響を受けるため，実機での実行においてもばらつきが生じる．したがって，より精度を高めるためにはこのような外乱を考慮した通信性能モデリングが必要となり，これは今後の課題である．次に，ホストマシンによるスケルトンコード実行時間（つまり，性能予測に要する時間）について考察する．HPL ならびに PHASE それぞれの実行に関しては最大でも 1 時間以内，FMO-ERI の場合は 8 分程度でスケルトンコード実行を完

了した．これは，スケルトン化により演算実行時間や通信時間を大幅に短縮したためである．また，場合によってはターゲットマシンでの実行時間より短い時間でホストマシンでのスケルトンコード実行を完了しており，大規模化への対応が可能であることを示している．

なお，本評価においては，仮想超並列実行環境での性能予測精度検証を主な目的としたため，演算時間モデリングならびに通信時間モデリングにおいてはターゲットマシンであるPQ580 の一部を使用した．開発対象システムが既存システムの拡張を前提とする場合には，本評価とほぼ同様の手順で性能評価を実施することができる．一方，既存システムとは大きく異なるアーキテクチャ（インターコネクト・トポロジやプロセッサなど）を前提として新規システムを開発する場合には，本評価手順の「2：スケルトンコードの開発」と「3：通信性能情報の作成」においては，3.3 節や 3.4 節で紹介した各種シミュレータを用いることで対応できる．

4. 仮想ペタスケール・スーパコンピュータを対象とした大規模性能予測

本研究で開発した性能評価環境を用いた大規模性能予測を実施した．本章ではその詳細を報告する．

4.1 実験環境

2010 年現在，世界最高を誇るスーパコンピュータの性能は 1 ペタフロップスを超えている．そこで，本研究で開発した性能評価環境の有効性を評価するため，仮想ペタスケール・スーパコンピュータを想定した性能予測実験を行った．具体的には，**表 8** に示す理論ピーク性能 2.1 ペタフロップスのターゲットマシン構成を前提とした超並列プログラムの実行時間を予測する．ここで，性能予測を実施するホストマシンに関しては，理論ピーク性能 1.6 テラフロップス（理化学研究所 Riken Super Combined Cluster）ならびに 192 ギガフロップス（九州大学 PRIMERGY）の 2 種類の実在マシンを使用した．ホストマシンに対するターゲットマシンの性能差は，それぞれ，1,312 倍ならびに 10,937 倍となる．性能予測対象であるターゲットマシンに関しては，3.3 節で示した SIMD 拡張型プロセッサコアを 8 個搭載する場合を想定した．コアあたり 16 個の SIMD 演算ユニット（浮動小数点加算と浮動小数点乗算）を搭載する．これを 8 コア搭載することで理論ピーク性能が 512 ギガフロップスのプロセッサチップとなる．これを 1 つの計算ノードとし，4,096 の計算ノードを 3 次元トーラス・インターコネクトで相互結合する．なお，本インターコネクトにおいては，リンクあたりのバンド幅は 20 GB/s（片方向あたり 10 GB/s）を想定しており，インターコネクト/演算ノード性能比は 0.24 B/Flop（20 GB/s × 6 方向リンク/512 GFlop/s）となる．実行対象となるベンチマーク・プログラムは HPL（行列サイズ 131 万元）ならびに PHASE（バンド数 65,536）である．

表 8　大規模性能予測実験におけるターゲットならびにホストマシン構成
Table 8　Target and host machines for peta-scale performance prediction.

システム名	ノード数（ジョブ当たり）	プロセッサ数（ノード当たり）	プロセッサ仕様	メモリ容量（ノード当たり）	インターコネクト	理論ピーク性能
仮想ペタスケール・スパコン（ターゲット）	4,096	1	「2GHz 4命令発行スーパスカラ＋16個のSIMD FMAユニット」×8コア（理論ピーク：512GFlop/s）	64 GB	3次元トーラス（16×16×16）（BiSec BW:10.24TB/s）	2.1 PFlop/s
RSCC Linux Cluster（ホスト：HPL）	128	2	3.06 GHz Xeon	4 GB	Gigabit Ethernet	1.6 TFlop/s
PRIMERGY RX200 S2（ホスト：PHASE）	16	2	3 GHz Xeon	7 GB	Gigabit Ethernet	192 GFlop/s

本評価が対象とするペタスケール・ターゲットマシンは実在しない．そこで，スケルトンコードの作成において必要となる演算コード部分の実行時間推定に関しては，3.3 節で紹介したサイクルアキュレート・プロセッサ・シミュレータ PSIM を使用した．具体的には，各ベンチマーク・プログラムにおいて小規模問題サイズでの実行を行い（たとえば，HPL の DGEMM では数百元程度），その結果に基づき性能モデル（演算実行に要する時間の式）を導出している．PSIM は 1 秒あたり約 25,000 命令をシミュレート可能であり，小規模問題サイズを対象とした PSIM シミュレーションは約 1 分以下で完了した．一方，通信遅延時間に関しては想定したインターコネクト仕様に基づき見積もる．具体的には，3.4 節で説明した NSIM は使用せず，仮定したスイッチ・レイテンシとバンド幅に基づき計算ノード間での平均通信遅延時間を計算し（スイッチ通過遅延時間は 100 ns，筐体内/間のケーブル遅延時間は 5 ns/25 ns を想定），通信対象となる計算ノード番号とメッセージサイズに基づき通信遅延情報を決定した．

4.2 性能予測結果

性能予測結果を**表 9** に示す．ここで，3 列目の「性能予測に要する時間」とは，ホストマシン上でのスケルトンコード実行に要する時間である．実験の結果，想定したペタスケール・システムの性能は，HPL に関して通信遅延がない場合で 1.07 ペタフロップス，3D-Torus ネットワークでの通信遅延を考慮した場合で 1.02 ペタフロップスであることが分かった．また，これらの性能予測に要する時間は約 6 時間であった．PHASE カーネルの場合にお

表 9　ペタスケール・ターゲットマシンの性能予測結果
Table 9　Performance prediction results for peta-scale target machine.

ベンチマーク	ホストマシン		ターゲットマシン性能（予測）			
	理論ピーク性能	性能予測所要時間	通信遅延無し		通信遅延有り	
			実行時間	実効性能	実行時間	実効性能
HPL	1.6 TFlop/s	約6時間	1,397秒	1.07 PFlop/s	1,477秒	1.02 Pflop/s
PHASE	192 GFlop/s	約4時間	165秒	0.65 PFlop/s	---------	---------

いても，実効性能は 0.65 ペタフロップスであり，その性能予測を約 4 時間で実現している．このように，本研究で開発した性能評価環境を用いることにより，理論ピーク性能値では 1,300 倍（2.1 PFlops/1.6 TFlops）〜10,000 倍（2.1 PFlops/210 GFlops），計算ノード数では 32 倍〜256 倍ほど大規模なターゲットマシンの性能を現実時間内で予測することができた．なお，本実験で用いた通信性能の見積り方式では集団通信時間をモデル化することが難しい．そのため，通信遅延を考慮した評価結果に関しては集団通信を使用しない HPL のみとした．この問題は 3.4 節で示した NSIM を用いれば解決可能であり，今後追加実験を行う予定である．

5. 提案方式の適用可能範囲に関する考察と今後の課題

今後のペタスケールからエクサスケールへとより高性能なスーパコンピュータを対象とするためには，様々なシステム・アーキテクチャを想定した性能予測の実現が必要となる．そこで本章では，現状における提案方式の適用可能範囲を明確にするとともに，より汎用的かつ実用的な利用に向け解決すべき課題を整理する．

まず，提案する性能予測方式の適用可能性をハードウェア・アーキテクチャの観点から考察する．図 2 (B) に示す BSIM において精度の高い性能予測を実現するためには，アプリケーション・コード中のスケルトン化部分に関して適切な演算性能モデリングを行う必要がある．もし，ターゲットマシンに搭載されるマイクロプロセッサが使用可能であれば，3.5.2 項で示した性能評価事例と同様に実測に基づくモデリングを行える．ターゲットマシン用のマイクロプロセッサが使用不可能（開発予定であり存在しない場合など）であれば，プロセッサ・シミュレータを用いる必要がある．3.3 節で説明した PSIM は SPARC アーキテクチャのみをサポートしているため，その他の命令セットや大幅に異なるマイクロアーキテクチャを想定した評価は実施できない．一方，BSIM が性能予測に用いる通信性能情報に関しては，開発対象となるターゲットマシンと同規模のインターコネクトは実在しないのが一般的であるため，より精度の高い性能予測を実現するには 3.4 節で示した NSIM を用いる必要がある．NSIM では現在のスーパコンピュータで採用されている 3 次元トーラスや Fat-Tree といった標準的なトポロジをサポートしており，また，各種ルータやスイッチの性能を設定ファイルで変更できるため，比較的幅広いインターコネクト・アーキテクチャに対応できる．ただし，現状ではコミュニケータ分割は未サポートのため，複数コミュニケータでの同時通信が存在する場合は対応できない．これに加え，BlueGene/L のように集団通信専用ネットワークが用意されているような特殊アーキテクチャを有する場合も対象外となる．

次に，様々なシステム構成に対する提案方式の適用可能性を議論する．近年，より高い性能を実現する有望な手段として，GPU などのアクセラレータの活用が注目されている．実際，すでに TOP500 の上位にはアクセラレータ搭載型スーパコンピュータがランクインしており，この傾向はより顕著になりつつある．また，汎用プロセッサにおいても，コア数の増加やメモリ階層の拡張，MRAM や PCRAM といった新メモリデバイスの活用などが期待を集めている．BSIM を用いた性能予測のフレームワークにおいて，このようなハードウェア構成に関して特に制限はない．3.2 節で説明したスケルトンコードを作成可能であれば，基本的にどのようなアーキテクチャでも評価対象とすることができる．たとえば，アクセラレータ搭載型計算ノードを前提とした場合には，スケルトン化対象コード部分をアクセラレータ性能モデルで置き換えればよい．したがって，「様々な特性を有する新プロセッサ・アーキテクチャやハードウェアの性能をいかにして精度良くモデル化するか？」がポイントとなる．この対策としては，各種プロセッサ・シミュレータの新規開発や，文献 22) で提案されている既存プロセッサをベースとした性能モデリングの応用など，新たな性能モデリング技術の導入が必要となる．一方，インターコネクトの基本的なトポロジはトーラス網やメッシュ網，Fat-Tree 網といった形態に収束しつつあり，基本的な機能としては NSIM で十分対応可能と考える．しかしながら，さらなる計算ノード数の増加を見据えた場合，コミュニケータ分割のサポートや，より高機能な専用ネットワークを用いた集団通信の実現などが必須になると予想され，これにともない NSIM を改良する必要がある．

最後に，本稿で提案する性能予測環境をより実用的に活用するために今後解決すべき課題を述べる．まず，システム規模の観点から議論する．3.5.2 項で説明した性能予測精度評価では，ターゲットシステムは 6.6 TFlop/s，ホストシステムは 192 GFlop/s または 52 GFlop/s の実験環境であった．これらの性能差は 34〜126 倍であり，小規模なシステムを用いた性能評価の実現可能性を示すことができた．しかしながら，2010 年時点で世界トップクラス

のスーパコンピュータの性能は 1 ペタフロップスを超えており，3.5.2 項で対象としたターゲットシステムとの性能差は大きい．したがって，今後，数百テラフロップス〜ペタフロップスといったより大規模な実在ターゲットマシンを対象とした精度評価実験を実施する必要がある．次に，性能予測精度について考察する．3.5.2 項で述べたように，通信性能の予測誤差が比較的大きく，特に集団通信に関する遅延時間モデルの精度改善が必要不可欠である．今後，プロセス配置や通信ミドルウェアの実行オーバヘッドなどを考慮した通信性能のモデル化が必要になると考える．これに加え，様々なベンチマーク・プログラムを用いたさらなる精度検証を実施し各種改善を行うとともに，本環境での精度保証範囲をより明確にしなければならない．

6. まとめと将来展望

本稿では，平成 17 年度から 5 年間にわたって実施したスーパコンピュータの性能評価技術に関する研究成果を示した．スーパコンピュータの実効性能を正しく予測するためには，開発対象となるターゲットマシンを想定したうえで，アプリケーション・プログラム実行の振舞いを考慮しなければならない．そのためには，性能評価に使用できる既存のホストマシン上でターゲットマシンの動作を模擬する必要があるが，これらのマシン間には 2〜3 桁以上の性能ギャップが存在する．本研究の最大の挑戦は「いかにしてこの性能ギャップを解消し，現実的な時間内でターゲットマシン性能予測を可能にするか？」にある．これを実現すべく，我々は処理内容を高度に抽象化したプログラムであるスケルトンコードを導入し，かつ，それを用いた性能評価を行うための各種ツールを開発した．その結果，ホストマシンとターゲットマシンの間に 1,300 倍〜10,000 倍の性能ギャップがある場合でも性能予測が可能であることを示した．また，既存のテラフロップス級マシンを用いた実証実験を行い，高い精度で実行時間を予測できることを明らかにした．本研究で開発した PSIM に関しては SPARC 命令セットを前提としたシミュレーション，また，NSIM に関しては一般的な 1 次元/2 次元/3 次元メッシュ網，1 次元/2 次元/3 次元トーラス網，Fat-Tree 網，ならびに，文献 11) のような 6 次元メッシュといったトポロジのみを対象とする．一方，3.5 節で示した性能予測手法そのものに関しては，スケルトンコードで計算時間を見積り可能であれば様々なアーキテクチャを対象とすることが可能である．

わが国における計算科学ならびに計算機科学分野の継続的かつさらなる発展を実現するためには，世界 No.1 の道具を「使う技術」と「造る技術」を両輪として研究開発を進める必要がある．本稿では，特に「造る技術」に焦点を当て，そのための性能評価手法を示した．スーパコンピュータの性能向上はとどまることを知らず，その姿は年々変化している．たとえば，計算ノードに関してはベクトル型からスカラ型，近年では GPU などのアクセラレータ活用へと向かっている．また，搭載される構成要素数（プロセッサやメモリなど）は増加の一途をたどっており，今後もさらなる大規模化ならびに複雑化が進むと予想される．このような状況においては，新規スーパコンピュータ開発の初期段階において，性能や消費電力，設置面積，各種コストなど様々な制約を満足する適切な構成法を検討することがより重要となる．そのためには，各種評価技術のさらなる発展と継続的な研究開発の実施が必要不可欠である．

今後，本研究で開発したツール群を可能な限り一般公開し，スーパコンピュータ開発とその利用をサポートする大きな流れを形成することにより，科学技術分野の発展と新しい知の発見への貢献を目指す．また，近年の地球温暖化問題を契機に，スーパコンピュータの低消費電力化ならびに低消費エネルギー化が求められている．そこで，今後は，消費電力ならびに消費エネルギーの評価も可能とすべく各種ツールを改良する予定である．さらに，本研究で開発した各種ツールをプログラム・チューニングに利用することを検討する．実際，本研究においては，仮想超並列実行環境 BSIM を用いた並列実行解析を行い，その結果をプログラム最適化へとフィードバックした．そこで，より高機能なプログラム実行解析機能を実装することにより「スーパコンピュータを使う技術」への貢献も狙う．

謝辞　本研究を遂行するにあたり多くのご協力をいただいた米国富士通研究所の木村康則氏，富士通株式会社の安里彰氏，松本孝之氏，折居茂夫氏，坂本真理子氏をはじめ，PSI プロジェクト・メンバの諸氏に感謝いたします．本研究は，一部，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「将来のスーパコンピューティングのための要素技術の研究開発」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」，ならびに，科学研究費補助金若手 A（課題番号：21680005）による．なお，本研究の実験結果の一部は，九州大学情報基盤研究開発センターの研究用計算機システムを用いて取得したことを付記する．

参 考 文 献

1) 井上弘士，薄田竜太郎，安藤壽茂，石附　茂，小松秀実，稲富雄一，本田宏明，山村周史，柴村英智，于　雲青，青柳　睦，木村康則，村上和彰：大規模システム評価環境 PSI-SIM：数千個のマルチコア・プロセッサを搭載したペタスケールコンピュータの性能予測，情報処理学会研究報告，計算機アーキテクチャ研究会報告，Vol.2008, No.39, pp.51–56 (2008).

2) 岩渕寿寛，杉田　秀，山名早人：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，情報処理学会研究報告，2005-HPC-101, pp.175–180 (2005).
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) 柴村英智，久我守弘，末吉敏則：超並列計算機のための相互結合網シミュレータ，情報処理学会論文誌，Vol.35, No.4, pp.589–599 (1994).
5) 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：PSI-SIM：大規模並列システムの性能解析に向けた並列相互結合網シミュレータ，電子情報通信学会技術研究報告（CPSY），CPSY2007-32, Vol.107, No.276, pp.45–50 (2007).
6) 原田智紀，曽根　猛，朴　泰祐，中村　宏，中澤喜三郎：並列処理用ネットワークのための性能評価用シミュレータ生成系 INSPIRE，情報処理学会研究報告，ARC-95-113-9, pp.65–72 (1995).
7) 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，2003-HPC-097, Vol.2004, No.20, pp.55–60 (2004).
8) 山村周史，青木　孝，安藤壽茂：大規模科学技術計算向け SIMD 拡張スカラプロセッサの提案とその評価，情報処理学会研究報告，2007-ARC-174, pp.61–66 (2007).
9) Abu-Sufah, W. and Kwok, A.Y.: Performance Prediction Tools for CEDAR: A Multiprocessor Supercomputer, *Proc. 12th International Symposium on Computer Architecture*, pp.406–413 (1985).
10) Adiga, N.R., Blumrich, M.A., Chen, D., Coteus, P., Gara, A., Giampapa, M.E., Heidelberger, P., Singh, S., Steinmacher-Burow, B.D., Takken, T., Tsao, M. and Vranas, P.: Blue Gene/L torus interconnection network, *IBM Journal of Research & Development*, Vol.49, No.2/3, pp.265–276 (2005).
11) Ajima, Y., Sumimoto, S. and Shimizu, T.: Tofu: A 6D Mesh/Torus Interconnect for Exascale Computers, *IEEE Computer*, Vol.42, No.11, pp.36–40 (2009).
12) Almasi, G., Archer, C., Castaños, J.G., Gunnels, J.A., Erway, C.C., Heidelberger, P., Martorell, X., Moreira, J.E., Pinnow, K., Ratterman, J., Steinmacher-Burow, B.D., Gropp, W. and Toonen, B.: Design and implementation of message-passing services for the Blue Gene/L supercomputer, *IBM Journal of Research & Development*, Vol.49, No.2/3, pp.393–406 (2005).
13) Austin, T., Larson, E. and Ernst, D.: Simplescalar: An Infrastructure for Computer System Modeling, *IEEE Computer*, Vol.35, No.2, pp.59–67 (2002).
14) Barker, K.J., Davis, K., Hoisie, A., Kerbyson, D.J., Lang, M., Pakin, S. and Sancho, J.C.: Using Performance Modeling to Design Large-Scale Systems, *IEEE Computer*, Vol.42, No.11 (2009).
15) Bailey, D.H. and Snavely, A.: Performance Modeling: Understanding the Past and Predicting the Future, LNCS 3648, pp.185–195 (2005).
16) Binkert, N.L., Dreslinski, R.G., Hsu, L.R., Lim, K.T., Saidi, A.G. and Reinhardt, S.K.: The M5 Simulator: Modeling Networked Systems, *IEEE Micro*, Vol.26, No.4, pp.52–60 (2006).
17) Brooks, D., Tiwari, V. and Martonosi, M.: Wattch: A Framework for Architectural-Level Power Analysis and Optimizations, *Proc. 27th International Symposium on Computer Architecture*, pp.83–94 (2000).
18) Carrington, L., Snavely, A., Gao, X. and Wolter, N.: A Performance Prediction Framework for Scientific Applications, *ICCS Workshop on Performance Modeling and Analysis* (2003).
19) Choudhury, N., Mehta, T., Wilmarth, T.L., Bohm, E.J. and Kale, L.V.: Scaling an optimistic parallel simulation of large-scale interconnection networks, *Proc. Winter Simulation Conference*, pp.4–7 (Dec. 2005).
20) Davis, D., Barker, K.J. and Kerbyson, D.J.: Performance Prediction via Modeling: A Case Study of The ORNL CRAY XT4 Upgrade, *World scientific, Parallel Processing Letters*, Vol.19, No.4, pp.619–639 (2009).
21) Ipek, E., Supinski, B.R., Schulz, M. and McKee, S.A.: An Approach to Performance Prediction for Parallel Applications, LNCS, Volume 3648/2005, pp.196–205 (2005).
22) Pfeiffer, W. and Wright, N.J.: Modeling and Predicting Application Performance on Parallel Computers Using HPC Challenge Benchmarks, *Proc. 22nd IEEE International Parallel and Distributed Processing Symposium* (April 2008).
23) Prakash, S. and Bagrodia, R.L.: MPISIM: using parallel simulation to evaluate MPI programs, *Proc. 30th conference on Winter simulation*, pp.467–474 (1998).
24) Ridruejo, F.J. and Miguel-Alonso, J.: INSEE: an Interconnection Network Simulation and Evaluation Environment, LNCS, Vol.3648, pp.1014–1023 (Aug. 2005).
25) Snavely, A., Carrington, L., Wolter, N., Badia, R. and Purkayastha, A.: A Framework for Performance Modeling and Prediction, *Proc. 2002 ACM/IEEE conference on Supercomputing*, pp.1–17 (2002).
26) Susukita, R., Ando, H., Aoyagi, M., Honda, H., Inadomi, Y., Inoue, K., Ishizuki, S., Kimura, Y., Komatsu, H., Kurokawa, M., Murakami, K., Shibamura, H., Yamamura, S. and Yu, Y.: Performance Prediction of Large-scale Parallel System and Application using Macro-level Simulation, *International Conference for High Performance Computing, Networking, Storage and Analysis* (*SC08*) (Nov. 2008).
27) Tikir, M.M., Carrington, L., Strohmaier, E. and Snavely, A.: A Genetic Algorithms Approach to Modeling the Performance of Memory-bound Computations, *Proc. International Conference for High Performance Computing, Networking, Storage and Analysis* (Nov. 2007).
28) Vijaykrishnan, N., Kandemir, M., Irwin, M.J., Kim, H.S. and Ye, W.: Energy-

Driven Integrated Hardware-Software Optimizations Using SimplePower, *International Symposium on Computer Architecture*, pp.95–106 (2000).
29) Zheng, G., Kakulapati, G. and Kal'e, L.V.: BigSim: A Parallel Simulator for Performance Prediction of Extremely Large Parallel Machines, *18th International Parallel and Distributed Processing Symposium*, p.78b (2004).
30) UNIX サーバ用プロセッサ：SPARC64 V (Aug. 2004).
http://primeserver.fujitsu.com/primepower/catalog/data/pdf/sparc64_v_j.pdf

(平成 22 年 8 月 4 日受付)
(平成 22 年 12 月 7 日採録)

井上　弘士（正会員）

昭和 46 年生．平成 8 年九州工業大学大学院情報工学研究科修士課程修了．同年横河電機（株）入社．平成 9 年より（財）九州システム情報技術研究所研究助手．平成 11 年の 1 年間 Halo LSI Design & Device Technology, Inc. で訪問研究員としてフラッシュ・メモリの開発に従事．平成 13 年九州大学で工学博士を取得．同年福岡大学工学部電子情報工学科助手．平成 16 年より九州大学大学院システム情報科学研究院助教授．平成 19 年 4 月より，同大学准教授，現在に至る．高性能/低消費電力プロセッサ/メモリ・アーキテクチャ，ディペンダブル・アーキテクチャ，3 次元積層アーキテクチャ，性能評価，等に関する研究に従事．電子情報通信学会，ACM，IEEE 各会員．

安藤　壽茂

昭和 45 年東京工業大学大学院理工学研究科電子工学専攻修士課程修了．同年富士通株式会社に入社．以降，ハイエンドコンピュータの開発に従事．平成 4 年から平成 9 年にかけて米国 HAL コンピュータシステム社に出向し SPARC プロセサの開発に従事．その後，富士通に戻り，コンピュータシステム事業本部技師長等を務める．平成 21 年に富士通を退社し，現在は，電気通信大学非常勤講師およびテクニカルライターとして執筆活動を行っている．電子情報通信学会，ACM，IEEE 各会員．

薄田竜太郎

平成 9 年京都大学より理学博士を取得．その後，理化学研究所において分子動力学専用計算機の開発に参加．また専用計算機を用いた大規模物理シミュレーション，天文シミュレーション技術の研究を行う．現在，九州先端科学技術研究所において大規模並列システム/アプリケーションの性能予測，ネットワークシミュレータの開発に従事している．日本物理学会，日本天文学会各会員．

山村　周史（正会員）

平成 10 年京都工芸繊維大学大学院電子情報工学科修士課程修了．平成 13 年同大学院情報・生産科学専攻博士課程修了．博士（工学）．同年富士通（株）入社．（株）富士通研究所でプロセッサアーキテクチャ，システム性能評価・チューニングの研究に従事．平成 19 年より富士通（株）でプロセッサの開発に参加し，SPARC64VII，SPARC64 VIIIfx の二次キャッシュ部の開発を行い現在に至る．プロセッサアーキテクチャに関する研究に興味を持つ．電子情報通信学会，IEEE 各会員．

柴村　英智（正会員）

平成 2 年詫間電波高等工業専門学校情報工学科卒業．平成 4 年九州工業大学情報工学部知能情報工学卒業．平成 6 年同大学大学院情報工学研究科修士課程修了．同年同大学マイクロ化総合技術センター助手．平成 10 年熊本大学工学部助手．平成 18 年より（財）九州先端科学技術研究所研究員．博士（工学）．現在，HPC，リコンフィギャラブルシステム，計算機アーキテクチャ等の研究に従事．IEEE，電子情報通信学会各会員．

三輪　英樹（正会員）

昭和 55 年生．平成 20 年九州大学大学院システム情報科学府情報理学専攻博士課程単位取得退学．同年より九州先端科学技術研究所特任研究員．インターコネクトシミュレータの開発に従事．平成 22 年富士通株式会社入社．現在に至る．

本田　宏明（正会員）

昭和 45 年生．平成 12 年北海道大学理学研究科化学第二学科より理学博士を取得．同年（現）みずほ情報総研研究員，平成 17 年九州大学研究戦略企画室助手，平成 18 年九州大学情報基盤研究開発センター特任准教授，平成 20 年九州先端科学技術研究所特任研究員，平成 22 年より九州大学システム情報科学研究院特任准教授．現在に至る．量子化学物性理論，量子化学における効率的分子積分計算，アクセラレータを利用した量子化学計算，単一磁束量子回路によるアクセラレータに関する研究に従事．日本化学会，分子科学会，ACM 各会員．

稲富　雄一（正会員）

昭和 44 年生．平成 10 年筑波大学大学院博士課程化学研究科修了．同年筑波大学化学系技官．平成 12 年より（株）富士総合研究所特別研究員．平成 14 年より産業技術総合研究所研究員．平成 16 年より科学技術振興機構研究員．平成 18 年より九州大学情報基盤センター学術研究員．平成 20 年より九州先端科学技術研究所特任研究員．平成 21 年より九州大学情報基盤研究開発センター学術研究員．現在に至る．大規模分子軌道計算プログラムの開発，アプリケーションプログラム最適化に従事．日本化学会，分子科学会各会員．

眞木　　淳

昭和 42 年生．平成 11 年北海道大学大学院理学研究科化学第二専攻博士後期課程修了．同年北海道大学で理学博士を取得．同年科学技術振興事業団計算科学技術研究員．平成 13 年分子科学研究所非常勤研究員．平成 15 年同研究所産学官連携研究員．平成 17 年九州大学情報基盤センター産学官連携研究員．平成 20 年より九州先端科学技術研究所特任研究員．現在に至る．アプリケーションの開発，最適化の研究に従事．

平尾　智也

昭和 52 年生．平成 13 年奈良先端科学技術大学院大学情報科学研究科修士課程修了．特定分野向け高性能計算機の開発のほか，各種組み込みシステムの開発に従事．平成 21 年より財団法人九州先端科学技術研究所特任研究員．

青柳　　睦（正会員）

昭和 34 年生．昭和 59 年慶応義塾大学大学院理工学研究科修士課程修了．昭和 61 年名古屋大学大学院理学研究科博士課程単位取得退学．理学博士（昭和 62 年名古屋大学）．米国アルゴンヌ国立研究所博士研究員，工業技術院化学技術研究所（現：産業技術総合研究所）主任研究員，岡崎国立共同研究機構分子科学研究所助教授を経て，九州大学教授．分子科学計算，計算科学，連成計算，分散計算，性能評価に関する研究に従事．

村上　和彰（正会員）

昭和 35 年生．昭和 59 年京都大学大学院工学研究科情報工学専攻修士課程修了．同年富士通（株）入社．汎用大型計算機の研究開発に従事．昭和 62 年九州大学助手．平成 6 年九州大学助教授．現在九州大学大学院システム情報科学研究院情報理学部門教授，情報基盤研究開発センター長，情報統括本部長．計算機アーキテクチャ，並列処理，システム LSI 設計技術，等に関する研究に従事．工学博士．平成 3 年情報処理学会研究賞，平成 4 年情報処理学会論文賞，平成 9 年坂井記念特別賞，平成 12 年日経 BP 社 IP アワード，平成 12 年情報処理学会創立 40 周年記念論文賞，平成 14 年電子情報通信学会業績賞をそれぞれ受賞．

石附　　茂

昭和 43 年生．平成 4 年弘前大学理学部物理学科卒業．同年富士通株式会社入社．スーパコンピュータ向けプログラム高速化業務に従事．平成 17〜19 年文部科学省による「ペタスケールシステムインターコネクト（PSI）」プロジェクト研究員．

小松　秀実

昭和 32 年生．昭和 63 年東京大学大学院理学系研究科天文学専攻博士課程修了．理学博士．同年富士通株式会社入社．スーパコンピュータ向けプログラム高速化業務に従事．平成 17〜19 年文部科学省による「ペタスケールシステムインターコネクト（PSI）」プロジェクト研究員．

安島雄一郎（正会員）

平成 9 年東京大学工学部電気工学科卒業．平成 14 年同大学大学院工学系研究科博士課程修了．博士（工学）．同年（株）富士通研究所入社．現在，富士通（株）次世代テクニカルコンピューティング開発本部に勤務．インターコネクトアーキテクチャの開発に従事．

三吉　郁夫（正会員）

昭和 45 年生．平成 7 年京都大学大学院工学研究科情報工学専攻修士課程修了．同年富士通株式会社入社．現在，次世代スパコンの性能評価業務に従事．

清水　俊幸（正会員）

昭和 61 年東京工業大学工学部電子物理学科卒業，昭和 63 年同大学大学院理工学研究科修士課程修了．同年（株）富士通研究所入社．並列計算機アーキテクチャの研究に従事．平成 8 年 4 月より 1 年間，米ウィスコンシン大学マディソン校客員研究員．平成 22 年 4 月より九州大学情報基盤研究開発センター客員教授を兼務．現在，富士通（株）次世代テクニカルコンピューティング開発本部に勤務．HPC システム，インターコネクトアーキテクチャの開発に従事．電子情報通信学会会員．

黒川　原佳（正会員）

平成 14 年北陸先端科学技術大学院大学博士後期課程修了．同年から理化学研究所に勤務．博士（情報科学）．主に並列 CFD，並列分散処理，システム評価，運用技術・設計のシステム化に興味を持つ．スーパコンピュータ・システムの設計，運用管理，利用者支援に従事．IEEE CS，日本機械学会各会員．