5 プログラミング環境
―超大規模並列計算機の性能を活かすプログラミング環境―

応 専

村井　均[*1]　住元真司[*2]　瀧康太郎[*2]　山中栄次[*2]
[*1] 理化学研究所　[*2] 富士通（株）

8万ノード・64万コアを超える超大規模な並列計算機システムの性能を引き出すという困難な作業を遂行するには，機械（計算機）と人（ユーザ）を仲立ちする「プログラミング環境」が重要な役割を果たすと考えられる．

本稿では，コンパイラ，メッセージ通信ライブラリ，開発支援ツール，数学ライブラリなどの，「京」のプログラミング環境について解説する．

これらを利用することにより，ユーザは「京」の性能を引き出すアプリケーションを開発することができる．

「京」におけるプログラミング

並列計算機の性能を引き出すためには，1コア上の逐次処理と，コア間（ノード内）の並列処理およびノード間の並列処理のすべてを効率的に記述する必要がある．

PCクラスタなどの分散メモリ並列システムで動作するアプリケーションプログラムは，ノード内外の区別をすることなくプロセスを単位として，メッセージ通信ライブラリMPIを用いて並列化されている場合が多い．このようなプログラミングモデルはフラットMPIと呼ばれている（図-1上）．

フラットMPIは，記述が容易で移植性に優れるなどの長所を有しているが，超高並列環境では以下の課題があることが知られている．

- 1プロセスが使用する通信用バッファのサイズが増大することでメモリ不足が発生する．
- 主に集団通信で通信データ量が増大し，スケーラビリティが悪化する．

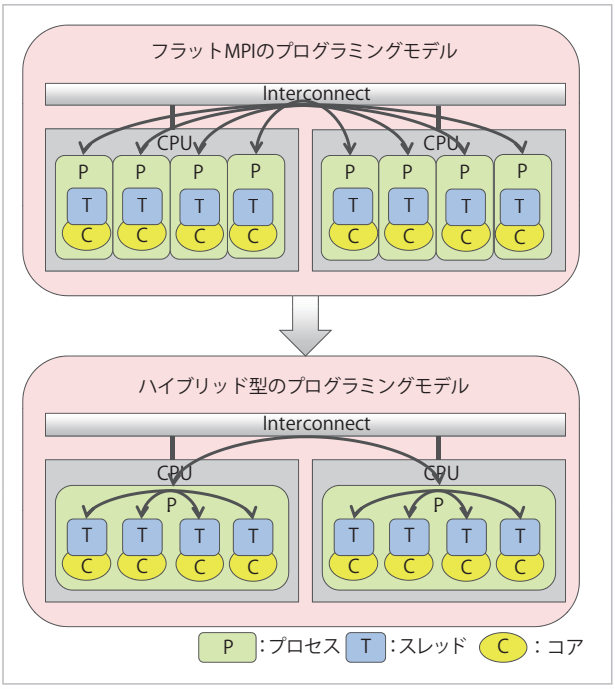


図-1　フラットMPIとハイブリッド並列

したがって，超高並列アプリケーションをターゲットとする「京」では，プロセス並列とスレッド並列を組み合わせたハイブリッド並列モデル（図-1下）を標準のプログラミングモデルとして採用している．ハイブリッド並列では，ノード間をMPIなどを用いてプロセス並列化し，ノード内を自動スレッド並列化機能やOpenMPを用いてスレッド並列化する．

「京」のプログラミング環境を開発する上では，以下の3つの要件を重視した．

- システムの性能を十分に引き出せること
- ユーザが容易にプログラムを開発（コーディング，デバッグ，チューニング）できること
- 一般に流通しているソフトウェア（オープンソー

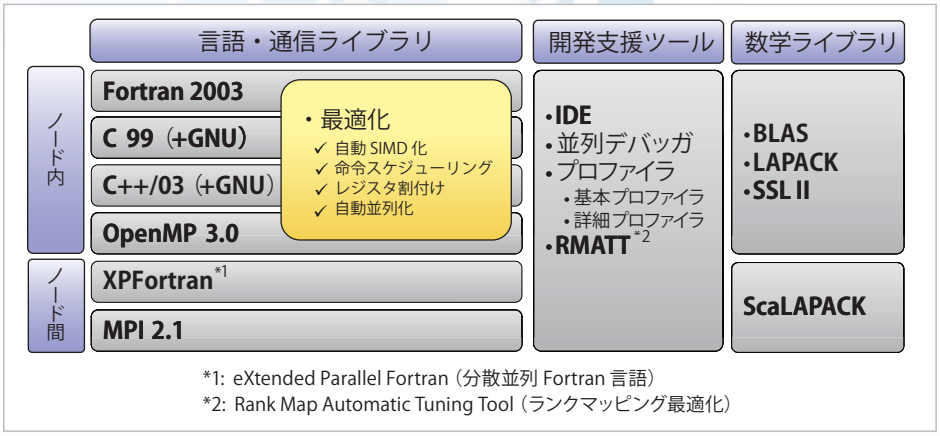


図 -2　プログラミング環境の構成

スを含む）を容易に移植できること

「京」のプログラミング環境の構成を図 -2 に示す．

以降の章で，上記の 3 つの要件に関連するコンパイラの高速化機能と言語仕様，MPI ライブラリ，開発支援ツールおよび数学ライブラリについて説明する．

コンパイラ

●CPU コア向けの高速化機能

- **自動 SIMD 化機能の強化**

「京」のコンパイラは，アプリケーション中のループを対象とする自動 SIMD 化機能とともに，非ループ部分の SIMD 化をする UXSIMD 機能をサポートしている．

UXSIMD 機能は，演算の木構造に基づいて，SIMD 化できる組合せを検出して SIMD 化する機能で，ループを対象とする自動 SIMD 化機能が有効でないようなループの内側の演算列に対しても適用することができる．

これらの機能により，さまざまなアプリケーションに対して SIMD 化による性能向上を実現している．

- **SIMD 組込み関数のサポート**

「京」の C/C++ コンパイラは，Intel 社製の C/C++ コンパイラがサポートしている SIMD 組込み関数と同等の組込み関数を用意している．

オープンソースのアプリケーションには，これらの SIMD 組込み関数を使用して SIMD 化を行っているものがあるが，「京」では，これらのオープンソースアプリケーションも容易に移植することができる．

- **レジスタ割付け・命令スケジューリングの強化**

SPARC64 VIIIfx は，SIMD 命令を使用する場合には 128 個，SIMD 命令を使用しない場合には 256 個と従来のスカラ CPU と比べて非常に多数の浮動小数点レジスタが使用可能なアーキテクチャである．これらの浮動小数点レジスタを有効に活用するためにレジスタ割付けを高度化し，限られたコンパイル時間で適切なレジスタ割付けが可能なようにするとともに，ソフトウェアパイプライニングなどの命令スケジューリング機能を強化している．

- **4 倍精度演算の高速化**

4 倍精度演算の高速化を実現するために，double-double 形式をサポートしている．

double-double 形式は，倍精度の浮動小数点演算を組み合わせて演算を行うため，整数演算でエミュレートする IEEE 形式より高速に演算を実行できる．加えて「京」の double-double 形式は，演算に SIMD 命令を用いるなど SPARC64 VIIIfx 向けに最適化されており，IEEE 形式と比較して，最大で加算で 23 倍，乗算で 39 倍の高速化を実現している．

●自動並列化機能

一般に，2 階層の並列性を抽出する必要があるハイブリッド並列では，フラット並列に比べプログラミングの難易度は高くなる．

「京」では，この問題を解決するために，富士通のスーパーコンピュータ FX1 で実用化された技術である VISIMPACT[1] を発展させて採用した．VISIMPACT は，低オーバヘッドなスレッド並列化を実現する CPU 機構と高度な解析能力を有する自動並列化コンパイラを組み合わせてスレッド並列化を自動的に行う技術である．アプリケーション開発

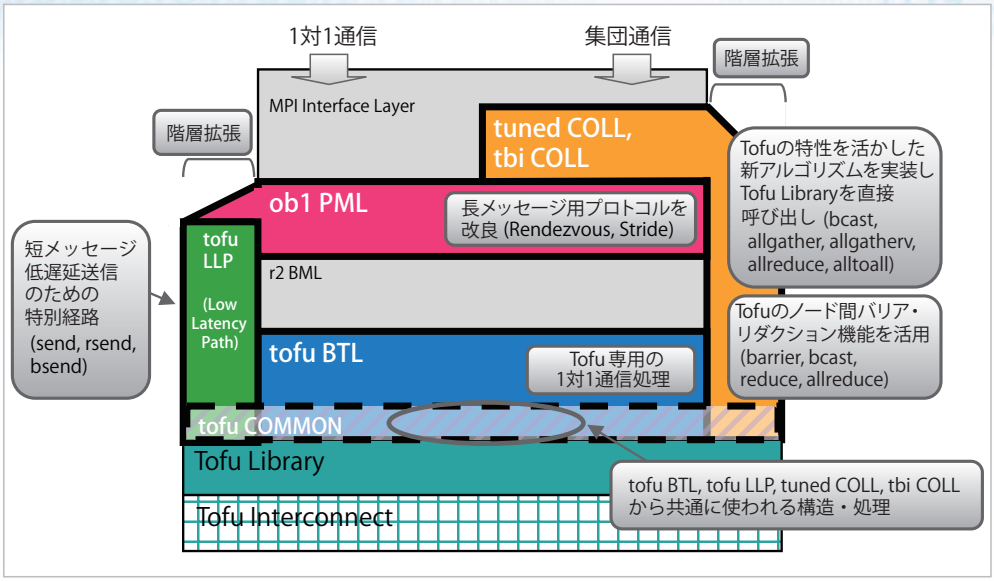


図 -3　MPI ライブラリの実装構造

者は，VISIMPACT を利用することにより，既存のフラット MPI プログラムを再コンパイルするだけでハイブリッド並列化を実現することができる．

●言語仕様

「京」のコンパイラは，オープンソースソフトウェアの移植性を高めるために，Fortran，C，C++ の標準の言語仕様に加えて，GCC 拡張仕様もサポートしている．また，分散メモリ並列処理向けに拡張された Fortran 言語である XPFortran[2]（VPP Fortran [3] と同一仕様）もサポートしている．

MPI ライブラリ

●概要

MPI とは，分散メモリ向け並列処理のデファクト標準であるメッセージ通信ライブラリである．HPC 分野では，MPI ライブラリの良し悪しがシステム性能を大きく左右する．8 万ノードを超える超高並列システムにおいて世界トップクラスの性能を実現する「京」の MPI ライブラリは次のような特徴を持つ．

- **優れた基本性能**

  MPI の基本である 1 対 1 通信性能を高めるため，Tofu インターコネクトの特性を活かす RDMA 通信を利用して 1 対 1 通信を実装している．集団通信についても，RDMA（Remote Direct Memory Access）通信を主体とした 1 対 1 通信をベースに Tofu インターコネクトの持つ複数のネットワークインタフェースを利用し，かつ，メッシュ／トーラス網に適した集団通信アルゴリズムを採用することにより性能を高めている．

- **大規模環境での最適な通信の実現**

  高い通信性能と省メモリ性を可能な限り両立するため，RDMA を利用するとともに，同時に通信を行うプロセスを限定することにより，通信バッファの使用メモリ量を抑制する．

- **使いやすさ**

  6 次元メッシュ／トーラス網を，6 次元の 6 つの軸を組み合わせることで仮想的な 3 次元トーラスとして見せることを可能とした．これにより，3 次元トーラスの既存システムで開発されたアプリケーション移植を容易にするほか，構成可能な 3 次元トーラス形状の選択肢が増える，システムを複数のジョブで分割利用しても 3 次元トーラスシステムとして利用できる，故障ノードが存在しても通信経路を設定することにより故障ノードを意識しないアプリケーション実行が可能になるなどの特長を実現している．

●実装

図 -3 に「京」の MPI ライブラリの実装構造を示す．「京」の MPI ライブラリは，オープンソースの MPI である Open MPI をベースとし，低遅延通信と RDMA ベースの集団通信に対応するための改良を加えたものである．Open MPI の内部インタフェースに合わせた低レベル通信ライブラリ（Tofu

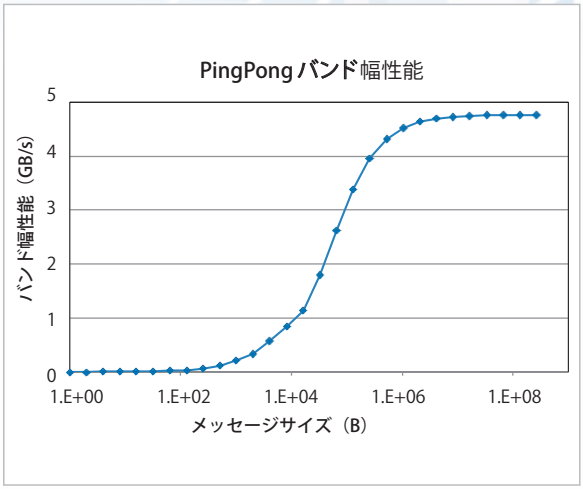


図 -4　MPI ライブラリの 1 対 1 通信の通信バンド幅性能

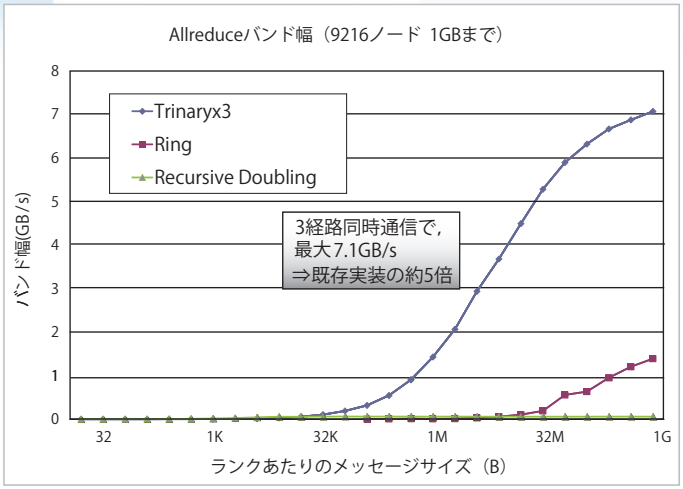


図 -5　MPI ライブラリの Allreduce の性能

Library）を用意し，「京」向けの機能をこの低レベル通信ライブラリ内で可能な限り実現することにより変更を最小限に抑えることを基本方針とし，以下のように実装を進めた．

- **低遅延通信を徹底追及**

  Open MPI は，多様なネットワークに対応するために内部の構造が階層化されて複雑になっている．1 対 1 通信でも PML，BML，BTL[☆1] の 3 つの通信層が介在しており，通信遅延が増加していた．「京」の MPI ライブラリでは，Tofu インターコネクトに不要な階層をショートカットする専用のパス（LLP）を設けて 1 対 1 通信の低遅延化を追求した．

- **RDMA 主体の集団通信実装**

  次の 3 つの理由により一部の集団通信アルゴリズムでは，Open MPI の集団通信の枠組みを用いず新たに開発した集団通信の枠組みを用いて実装している．

  ≫ Tofu のハードウェア Barrier，（Bcast，Reduce，Allreduce[☆2]）への対応

  ≫ RDMA ベースの集団通信への対応

  ≫ 複数ネットワークインタフェースの制御

  さらに，特に頻繁に利用される集団通信の Bcast，Allgather，Allreduce，Alltoall[☆2] について，複数のネットワークインタフェースを用い，かつ，ネットワーク上のパケット衝突を最低限に抑制することができる集団通信アルゴリズムを採用している．

- **バージョンアップへの迅速な対応**

  オープンソースソフトウェアは機能追加やバグフィックスなどバージョンアップが頻繁である．特に Open MPI は，標準化が間近な MPI Version 3.0 に向けての大規模な変更が予定されているため，Open MPI の構造そのものに対しては可能な限り手を入れずに，バージョンアップがそのまま適用可能な実装とすることを重要視した．

● 性能

MPI ライブラリの 1 対 1 通信の片道遅延時間は，隣接の場合，8 バイトメッセージで 1.27usec と高水準な性能を実現している．

1 対 1 通信の通信バンド幅性能は，図 -4 に示した通りである．

図 -4 より，通信バンド幅は，16MB メッセージ時にハードウェアピーク性能 5.0GB/s に対して 4.7GB/s と，Tofu インターコネクトの性能を十分に引き出していることが分かる．

図 -5 に Allreduce の性能を示す．

☆1 PML: Point-to-point Management Layer（1 対 1 通信を制御する層）
BML: BTL Management Layer（複数ハードウェアを制御する層）
BTL: Byte Transfer Layer（インターコネクト依存のデータ送信の層）

☆2 Bcast：1 つのプロセスから他の全プロセスにデータを送る通信（Broadcast，放送）
Reduce：各プロセスのデータを 1 つのプロセスに集計する通信
Allreduce：各プロセスのデータを集計し，結果を各プロセスが持つ通信（Reduce + Bcast と同等）
Allgather：各プロセスのデータを順に結合し，結果を各プロセスが持つ通信
Alltoall：各プロセスのデータを各プロセスが他の全プロセスに送る通信（全対全通信）

Tofu インターコネクトの性能を引き出すため，3つのネットワークインタフェースを用いて通信経路の重なりがないように実装されたアルゴリズム（Trinaryx3）は，7.1GB/s と既存のアルゴリズムに比べて 5 倍の性能を実現している．

開発支援ツール

「京」をターゲットとする超高並列アプリケーションの開発を容易にするためのツールとして，実行時デバッグ機能，会話型デバッガ，プロファイラ，RMATT（ランクマッピング自動チューニングツール）を用意している．

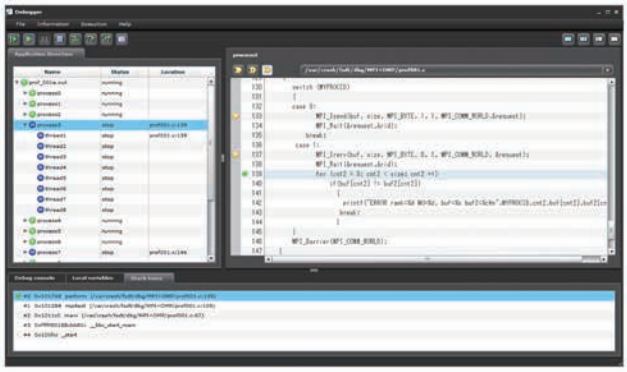
- **実行時デバッグ機能**

実行時デバッグ機能とは，アプリケーションを実行する際に誤りの可能性のある問題（未定義領域の参照，仮引数と実引数の不整合，配列への領域外アクセスなど）をアプリケーションが自律的に検出する機能であり，コンパイル時に問題検出用のライブラリの呼出しをアプリケーションに埋め込むことで実現している．これまでの実行時デバッグ機能は，問題検出用ライブラリの呼出しオーバヘッドによって実行時間が十数倍～数百倍と大幅に増加するために実行時間が長いアプリケーションへの適用が難しかった．「京」の実行時デバッグ機能は，問題検出部分を命令レベルでアプリケーションの命令列に埋め込み，問題を検出した場合にだけ問題処理用のライブラリを呼び出すなど実装を改善することでオーバヘッドを低減し，実行時間の増加を数倍程度に抑えることで，実行時間が長いアプリケーションへの適用性を高めている．

- **会話型デバッガ**

デバッグ時の基本ツールである会話型デバッガについて，コンパイル時に最適化されたアプリケーションへの適用性を高めた．これにより長時間アプリケーションの会話型デバッガによるデバッグが容易となった．

また，超高並列アプリケーションで課題となる並列実行時のデッドロックについては，デッドロックで停止したアプリケーションに対して会話型デバッガで外部からアタッチし，アプリケーションをデバッガの制御下に置き，会話的操作により停止した原因を調査することができる．会話型デバッガのスナップショットを図 -6 に示す．



会話型デバッガでMPI並列プログラムをデバッグしているスナップショットを示す．
各枠には次の情報が表示されている．
左上枠：並列プロセスおよびスレッドごとの走行状態（停止中のものは停止位置）
右上枠：ソースコード上でのブレークポイントの位置や現在の停止位置
下枠　：スタックトレース

図 -6　会話型デバッガのスナップショット

画面左側に各プロセスやスレッドごとの停止位置が表示されており，これらの情報を解析することでデッドロックの原因を切り分けることができる．

- **プロファイラ**

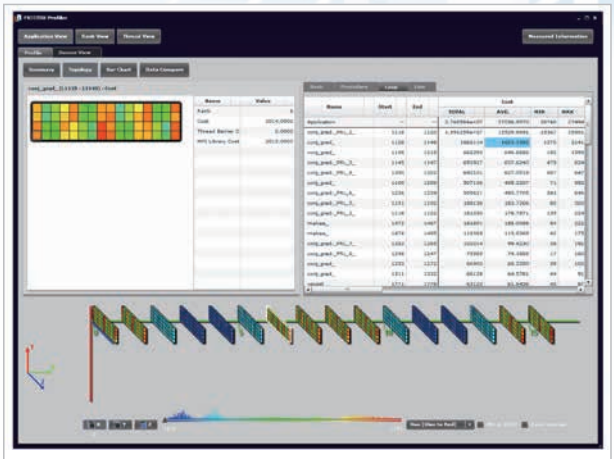
アプリケーションのチューニングを行う際には，実行時の状況を可視化し，性能低下を引き起こしている原因を明らかにすることが必要となる．

「京」では，実行コスト（演算に要する時間）の分布や FLOPS 値などのアプリケーション全体の性能の概要を把握するための基本プロファイラと，アプリケーションの性能向上に重要なホットスポット（実行コストが集中している関数やループ）を詳細に解析するための詳細プロファイラを用意している．

≫ 基本プロファイラ

基本プロファイラは，実行コストの分布や FLOPS 値などのアプリケーション全体の性能状況の概要を把握するために用いる．

基本プロファイラは，特別な準備をしないで容易に使用できるようにタイマー割り込みを用



基本プロファイラを使い性能を分析しているスナップショットを示す．
各枠には次の情報が表示されている．
左上枠：　プロセスごとのコスト情報（部分拡大図）
右上枠：　プログラムループごとの性能値の一覧
下枠：　プロセスごとのコスト情報（全体の3Dトポロジ表示）

図 -7　基本プロファイラを用いた実行コスト分布表示

いたサンプリング方式でアプリケーションをプロファイリングしている．

基本プロファイラを用いることで，チューニングの要否の判断や，ホットスポットの抽出が容易となる．

図 -7 に，基本プロファイラを用いた実行コストの分布表示の例を示す．

図 -7 は，実行コストの分布をトポロジ表示したものである．Tofu の 3D トーラス形状と性能の関係をグラフィカルに認識できるようにデータを可視化することで，超高並列時でも現象の把握が容易となる．

» 詳細プロファイラ

詳細プロファイラは，アプリケーションの性能向上に重要なホットスポットを詳細に解析するために用いる．

詳細プロファイラでは，アプリケーションのソースに測定する個所をマークして指定し，測定項目を選択してアプリケーションを実行することで，指定個所の詳細情報を採取する．

詳細プロファイラでは，SPARC64 VIIIfx の性能計測カウンターを用いて，命令実行，メモリアクセス待ち，演算待ちなどの CPU の動作状態ごとに実行サイクル数を測定して可視化することで，CPU 内のどの処理にボトルネックがあるのかを把握することができる．この分析手法をサイクルアカウンティング分析と呼んでいる．

図 -8 にサイクルアカウンティング分析の例を示す．

図 -8 から，各スレッドでレベル 2 キャッシュ上の浮動小数点データのロードにコストがかかっていること，すなわちレベル 1 キャッシュでの浮動小数点データのロードのミスがボトルネックとなっていることが分かる．

» RMATT

MPI プログラムの通信のチューニングのために，RMATT を用意している．RMATT は，実行時の通信プロファイル情報に基づいてシミュレーションを行い，MPI プロセスのノードへの配置（ランク配置）を自動的に決定し，通信時間を短縮するツールである．オープンソースのアプリケーションなど，適切なランク配置が把握できない場合に効果を発揮する．

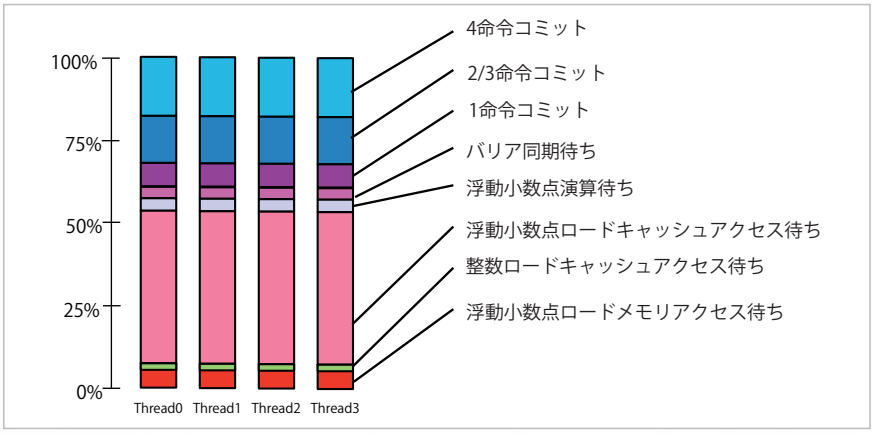


図 -8　サイクルアカウンティング分析

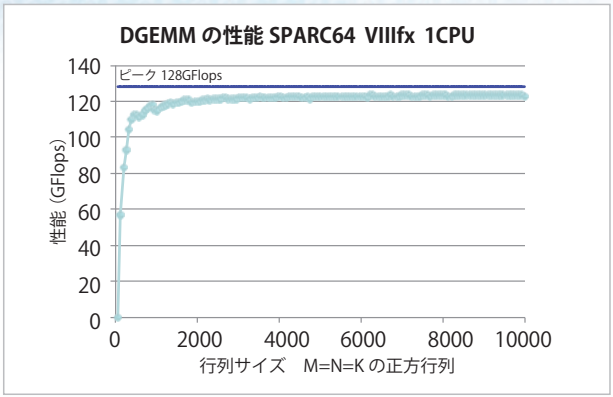


図 -9　DGEMM の性能

数学ライブラリ

線形計算の標準的なライブラリである BLAS, LAPACK, ScaLAPACK については，「京」向けに高度にチューニングしたものを用意している．さらに，BLAS, LAPACK の使用頻度の高いルーチンについてはマルチコア向けにスレッド並列化したものを用意している．

多くのアプリケーションから使用される BLAS の倍精度行列 - 行列積ルーチン（DGEMM）については，図 -9 に示すように，行列のサイズが 1000 程度でピーク性能比 90% を超え，最大でピーク性能比 96.5% という非常に高い性能を達成している．

富士通独自の数学ライブラリ SSL II についても新規アルゴリズムを追加している．具体的には，スレッド並列ライブラリでは，スパース行列の連立 1 次方程式の直接解法および反復解法，スパース行列の固有値問題解法，そして連立 1 階常微分方程式・微分代数方程式のアルゴリズム，分散並列ライブラリでは，複数軸でのデータ分割を可能にすることによって，数万ノードを超える大規模システムでも並列台数効果を得られる 3 次元 FFT（Fast Fourier Transform）を追加している．

まとめ

本稿では，コンパイラ，メッセージ通信ライブラリ，開発支援ツール，数学ライブラリといった，「京」のプログラミング環境について解説した．一般に使われているツール類との互換性を保ちつつ「京」の性能を容易かつ最大限に引き出せることが，これらの特長である．

参考文献

1）井上愛一郎：富士通の HPC への取り組み（2010），参照先：サイエンティフィック・システム研究会：http://www.ssken.gr.jp/MAINSITE/download/newsletter/2010/20100826-sci-1/lecture-5/ppt.pdf
2）永井　亨：XPFortran 入門，名古屋大学情報連携基盤センターニュース，5 (2), pp.129-168（2006）．
3）岩下英俊：VPP Fortran：分散メモリ型並列計算機言語，情報処理学会論文誌，5 (2)，pp.129-168（1995）．

（2012 年 4 月 27 日受付）

村井　均（正会員）　h-murai@riken.jp
1996 年京都大学大学院修士課程修了．1996 ～ 2010 年日本電気（株）．2010 年筑波大学大学院博士課程修了．現在，(独)理化学研究所研究員．博士（工学）．並列プログラミング言語および並列化コンパイラ技術の研究開発に従事．

住元真司（正会員）　sumimoto.shinji@jp.fujitsu.com
1986 年同志社大学工学部卒業，（株）富士通研究所，1997 ～ 2001 年新情報処理開発機構出向を経て，2007 年より富士通（株）勤務．現在，MPI 通信ライブラリ，クラスタファイルシステムなど高性能通信にかかわる HPC システムソフトウェア全般の技術開発に従事．工学博士．

瀧康太郎　taki.koutarou@jp.fujitsu.com
2000 年早稲田大学情報工学科卒業．同年富士通（株）に入社．コンパイラ最適化技術の開発に従事．

山中栄次　e-yamanaka@jp.fujitsu.com
1988 年東京工業大学大学院修士課程修了．同年富士通（株）に入社．並列化コンパイラ技術および並列プログラミング言語の開発に従事．