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. 4　169–179 (Oct. 2011)

スーパコンピュータ TSUBAME 2.0 における Linpack 性能 1 ペタフロップス超の達成

遠藤敏夫[†1,†2]　額田彰[†1,†2]　松岡聡[†1,†2,†3]

2010 年 11 月に稼働開始した TSUBAME 2.0 スーパコンピュータは，Intel プロセッサに加え 4,000 以上の NVIDIA GPU を備えるペタスケールのヘテロ型システムである．この TSUBAME 2.0 における Linpack ベンチマークの実行について報告する．本システムは 2CPU と 3GPU を備えた計算ノードを約 1,400 台持ち，それらはフルバイセクションのファットツリー構造を持つ Dual-Rail QDR InfiniBand ネットワークにより接続される．理論演算性能は TSUBAME 1.0 の約 30 倍となる 2.4 PFlops であり，それを TSUBAME 1.0 とほぼ同じ規模の電力で実現している．Linpack ベンチマークのコード改良およびチューニングを GPU を用いた大規模システムの特性に合わせて行い，実行速度として 1.192 PFlops を実現した．この結果は日本のスパコンとしては初めて PFlops を超えるものであり，Top500 スパコンランキングに 4 位にランクされた．さらに電力性能比は 958 MFlops/W であり，Green500 ランキングにおいて the Greenest Production Supercomputer in the World 賞を獲得した．

Achievement of Linpack Performance of over 1 PFlops on TSUBAME 2.0 Supercomputer

Toshio Endo,[†1,†2] Akira Nukada[†1,†2]
and Satoshi Matsuoka[†1,†2,†3]

We report Linpack benchmark results on the TSUBAME 2.0 supercomputer, a large scale heterogeneous system with Intel processors and > 4,000 NVIDIA GPUs, operation of which has started in November 2010. The main part of this system consists of about 1,400 compute nodes, each of which is equipped with two CPUs and three GPUs. The nodes are connected via full bisection fat tree network of Dual-Rail QDR InfiniBand. The theoretical peak performance reaches 2.4 PFlops, 30 times larger than that of the predecessor TSUBAME 1.0, while its power consumption is similar to TSUBAME 1.0. We conducted improvement and tuning of Linpack benchmark considering characteristics of large scale systems with GPUs, and achieved Linpack performance of 1.192 PFlops. This is the first result that exceeds 1 PFlops in Japan, and ranked as 4th in the latest Top500 supercomputer ranking. Also TSUBAME 2.0 has received “the Greenest Production Supercomputer in the World” prize in Green500 ranking for its performance power ratio of 958 MFlops/W.

1. はじめに

ポストペタ・エクサスケールの HPC システムを，現実的な電力・設置面積で実現するうえで，アクセラレータの利用が注目されている．2008 年に Top500 スーパコンピュータランキング[2] で初めて 1 PFlops を達成した LANL RoadRunner システムは，Opteron CPU に加え Sony/Toshiba/IBM PowerXCell 8i プロセッサをアクセラレータとして用いたものであった[8]．そして 2010 年 11 月の Top500 では，本稿で述べる TSUBAME 2.0 を含め，上位 5 システム中の 3 システムが NVIDIA GPU をアクセラレータとして用いている．また本年から来年にかけて CPU の主流となると期待される Intel 社の Sandy-bridge アーキテクチャは GPU を CPU に内蔵している．上述のようなシステムレベルではなくダイレベルの統合ではあるが，高性能なプロセッサコアと，比較的単純化された高演算性能のアクセラレータ（GPU コア）の双方を持つ点は共通といえる．

我々はアクセラレータの本格 HPC 利用について早い段階から取り組んでおり，その成果は 2006 年に東京工業大学に導入されたスーパコンピュータ TSUBAME 1 および 2010 年 11 月に運用開始した TSUBAME 2.0 に活用されている．TSUBAME 2.0 は 2.4 PFlops の理論演算性能を持つ，日本初のペタフロップスの性能を実現したシステムであり，その高い演算性能・電力効率は最新世代の GPU アクセラレータである NVIDIA Tesla M2050 によるところが大きい．さらに，フルバイセクションファットツリー構造のネットワーク，水冷の Modular Cooling System（MCS）による高効率な冷却などの特徴を持つ．

本稿では TSUBAME 2.0 上の Linpack ベンチマーク実行について報告する．Linpack は Top500 のランク決定に使われることでも知られ，密行列連立一次方程式を部分ピボッティングを用いたガウス消去法で解くベンチマークである．用いた手法は我々が TSUBAME 1 用

†1 東京工業大学
Tokyo Institute of Technology
†2 独立行政法人科学技術振興機構，CREST
Japan Science and Technology Agency, CREST
†3 国立情報学研究所
National Institute of Informatics

に開発したアルゴリズム[5] に基づいたものであり，実装は High-performance Linpack [10] を改造する形で行った．その実装は，概要をすでに報告したように[11]，GPU の演算性能を有効活用するために，カーネル演算，MPI 通信および PCI-Express 通信のオーバラップを行っている．TSUBAME 2.0 の 1,357 ノード，4,071GPU を用いたときの実行速度は 1.192 PFlops であった．2010 年 11 月の Top500 では世界 4 位にランクされ，国内では初めて 1 PFlops を超えたシステムとなった．ピーク演算性能（1,357 ノードで 2.288 PFlops）に対する比は 52.1%であり，ピークとの差に関する解析についても報告する．またシステムの消費電力は 1,243.8 kW（Green500 のルールに基づく測定法[1]），電力性能比は 958.35 MFlops/W と，この点でも世界トップクラスを実現している．

2. TSUBAME 2.0 の概要

TSUBAME 2.0 では，1,400 ノード以上の計算ノードと，合計 7.1 PBytes のストレージが QDR InifiniBand により接続されている（**図 1**）．計算ノードは 1,408 台の Thin ノード，24 台の Medium ノード，10 台の Fat ノードからなる．本稿の実験では Thin 計算ノードを用いるため，以下では単に計算ノードと呼ぶ．以下，本稿に関連の深い部分について概要を示す．

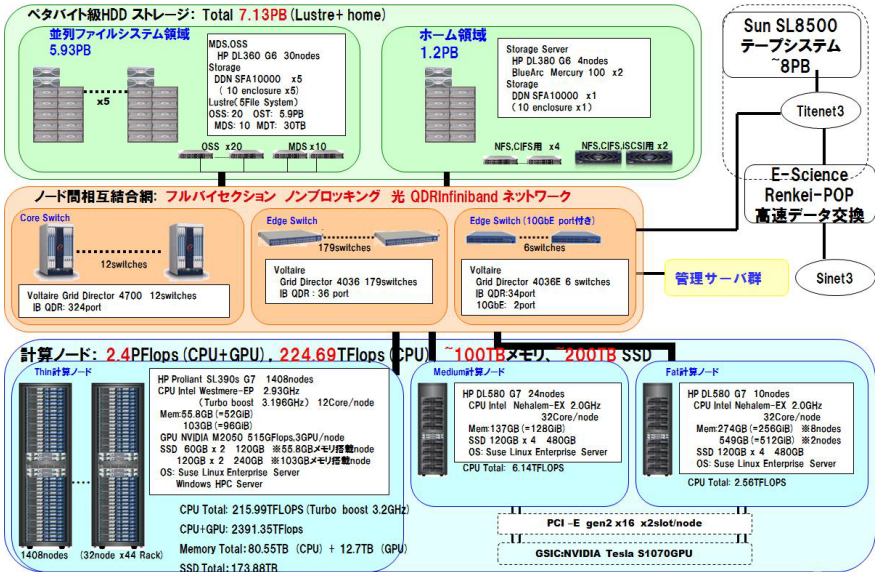

図 1　TSUBAME 2.0 の全体構成図
Fig. 1　Overview of TSUBAME 2.0 system.

各計算ノード Hewlett-Packard Proliant SL390s G7 は 6 コアの Intel Xeon X5670 2.93 GHz プロセッサを 2 個，NVIDIA Tesla M2050 GPU を 3 個搭載する．**図 2** にノード外観を，**図 3** にノード内部構成を示す．メインメモリとしては計 54 GB の DDR3 メモリを搭載する．また 40 Gbps QDR InfiniBand の host channel adapter（HCA）を 2 個持つ．2 個の HCA，3 個の GPU の I/O をまかなうために，ノードは IO Hub（IOH）を 2 個持つ．HCA は両方とも Socket 0 CPU 側（内部構成図の上側）の IO Hub に，それぞれ PCI-Express（PCIe）Gen 2 x8 で接続される．3 個の GPU のうち 1 つは Socket 0 側の IO hub に，2 個は Socket 1 側に，それぞれ PCIe Gen2 x16 で接続される．以上のように，HCA，GPU は PCIe レーンを共有することなく，効率的に通信を行うことができる．

オペレーティングシステムは 64 bit 対応 SuSE Linux Enterprise Server 11 および Windows HPC server 2008 R2 である．本稿の実験では Linux を用いる．

全ノードを接続するインターコネクトは 2 段のスイッチからなるファットツリーであり，フルバイセクション構成である．Dual rail 構成であり，各 rail がファットツリーを構成する．エッジスイッチとして 36 ポートの Voltaire GridDirector 4036 を 185 台持つ．各エッ

図 2　Thin 計算ノード HP Proliant SL390s G7 の外観
Fig. 2　Appearance of a thin compute node: HP Proliant SL390s G7.

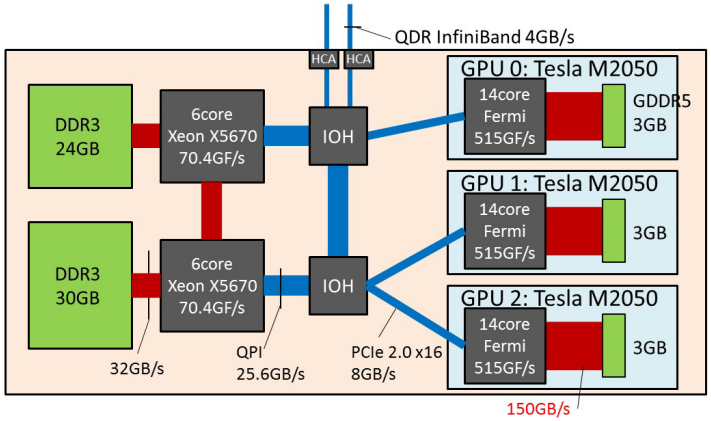

図 3　Thin 計算ノードの内部構成
Fig. 3　Internal structure of a thin compute node.

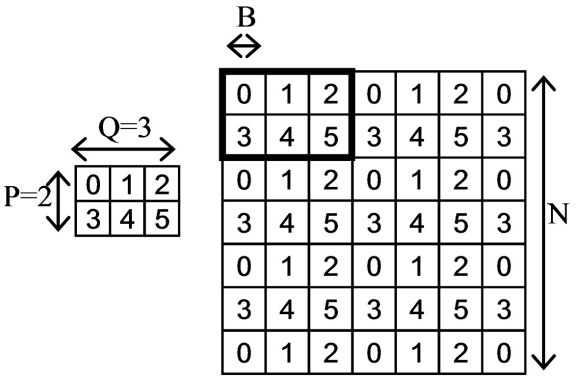

図 4　（左）$P \times Q = 2 \times 3$ プロセスのプロセス格子例，（右）6 プロセス間の $N \times N$ 行列の二次元ブロックサイクリック分割
Fig. 4　(Left) A process grid with $P \times Q = 2 \times 3$ processes. (Right) Two dimensional block cyclic distribution of $N \times N$ matrix on 6 processes.

ジスイッチのポートのうち 18 は上流のコアスイッチ向け，残り 18 は下流のノード向けである．コアスイッチは 324 ポートの GridDirector 4700 である．各 rail につき 6 台，計 12 台存在する．各ノードは 2 本の 40 Gbps QDR InfiniBand によりエッジスイッチに接続される．2 本は Dual rail のそれぞれに接続される．

各ノードは NVIDIA Tesla M2050 と呼ばれる Fermi 世代の GPU を 3GPU 搭載する．各 GPU はストリーミングマルチプロセッサ（SM）を 14 基持ち，各 SM は SIMD 動作する CUDA core を 32 基持つ．また SM 間で共有され，150 GB/s のメモリバンド幅を持つ 3 GB の GDDR5 デバイスメモリが搭載されている．GPU の理論演算性能は，倍精度浮動小数演算では 515 GFlops，単精度では 1.03 TFlops である．Tesla の利用のためには CUDA プログラミング環境が提供されており，拡張された C 言語によるプログラミングを行うことができる．

3. High performance Linpack

本稿では Linpack のよく知られた並列実装である High performance Linpack（HPL）を，ソースコードの一部改変して実行に用いる．HPL は正方密行列を係数とする連立一次方程式をブロック化ガウス消去法で解く，MPI 並列ソフトウェアである．指定された行列サイズ N に対して乱数行列を生成し，方程式を解き，その速度を Flops 値で評価する．

計算に参加するプロセス群は概念的にサイズ $P \times Q$ のプロセス格子を形成し，行列はプロセス格子に従って二次元ブロックサイクリック方式で分散される（**図 4**）．以下，行列サイズを N，ブロックサイズを B とする[⋆1]．計算のほとんどの部分をガウス消去法が占め，以下ではその各ステップ（ステップ番号 k とする）の処理の概略を示す．詳細は文献 10) を参照されたい．

パネル分解：第 k ブロック列はパネル列 L と呼ばれ，その箇所の LU 分解を部分ピボット選択を用いて行う．この過程では L の一部に対して DGEMM（密行列積カーネル）が用いられる．

パネルブロードキャスト：パネル列 L の各ブロックの内容を他プロセスへブロードキャストする．ここではプロセス格子の各行内での通信が発生する．

行交換通信：部分ピボット選択の結果に基づき，行交換を行う．ここではプロセス格子の各列内でピボット行が集約されることにより第 k ブロック行（その箇所を U と呼ぶ）が生成される．HPL においては交換のための通信と同時にプロセス格子列内のプロセスへのブロードキャスト通信が行われ，それにより各プロセスは U を持つこととなる．本稿ではブロードキャストも含め行交換通信と呼ぶ．

⋆1 一般的にブロックサイズは NB と呼ばれるが，$N \times B$ と区別するために本稿では B とする．

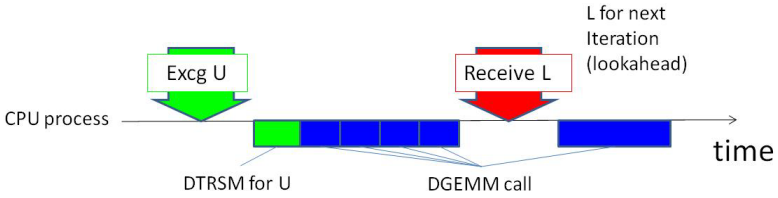

図 5　オリジナルの HPL の 1 ステップのアルゴリズム
Fig. 5　The original HPL algorithm for a single step.

更新計算：まず各プロセスは上記 U に対して三角行列求解（DTRSM）計算を行う．その後パネル列 L と U の内容を用い，行列の未分解部分の更新計算を行う．行列の未分解部分を A_k とすると，各プロセスは自分の持っている部分行列について，$A_k = A_k - L \times U$ という密行列積（DGEMM）計算を行う．

ここで，各 MPI プロセスが一ステップで行う処理を簡略化して**図 5** に示す．なおここでは “パネル分解” は省いている．パネルブロードキャストについては，HPL では lookahead と呼ばれる最適化が採用されている．つまり，ステップ $k+1$ のためのパネル列の通信を，ステップ k のうちに行っておき，通信コストの隠ぺいをしようとするものである．本図に示すアルゴリズムがどのように変更されるかは，後に述べる．

さて上記の処理のうち，パネル分解の計算量総計は $O(N^2B)$，パネルブロードキャストと行交換通信の通信量総計は $O(N^2(P+Q))$，更新計算の計算量総計は $O(N^3)$ である．このことから，最も時間がかかるのは更新計算であり，その傾向は N が大きいほど強いと分かる．そのため，並列 Linpack ベンチマークにおいて良い性能を得るためには，N をメモリ量の限界に近づけるように大きくとり，高速な行列積を行う BLAS 数値演算ライブラリを用いることが一般的に行われている．

4. TSUBAME 2.0 上の設計と実装

4.1 基本設計方針

TSUBAME 2.0 上の Linpack の基本設計方針は既報告の TSUBAME 1.2 上のもの[5] を基にする．その設計上の議論を簡単に述べ，その後に実装について，通信オーバラップ処理に重点を置いて示す．

カーネル演算の主体：カーネル演算である行列積（DGEMM）をどのプロセッサが行うか，各プロセッサ種の演算性能比から議論する．TSUBAME 2.0 においては GPU が理論演算性能の 92%，Xeon が 8%であるため，今回の実験では基本的に GPU をカーネル演算に用いることとした．例外として PCIe 通信コストが相対的に高くなる小さい行列の演算は CPU が行うこととした．

表 1　各システムにおけるノードごとの計算性能とノード間通信性能．典型的な x86 クラスタについても概算を示す．ヘテロ型システムにおいては 1 ノードあたりの，ホスト–アクセラレータ間 PCI 通信性能も示す
Table 1　Computation performance per node and internode communication performance of several systems. Rough estimation of typical x86 clusters is also shown. For heterogeneous systems, PCI communication performance between host and accelerators is also shown.

	理論演算性能 (GFlops)	ノード間通信性能 (GB/s)	PCI 通信性能 (GB/s)
x86 cluster	約 100〜300	約 1〜8	-
RoadRunner	450	2	4
TSUBAME 1.2	157〜330	2	1〜3
TSUBAME 2.0	1,685	8	24

行列データの配置場所：Linpack においては $N \times N$ の行列データを MPI プロセスに分散し保持させる．一方前述のとおり，メモリサイズに収まる範囲で N が大きいほうが高性能のために望ましい．TSUBAME 2.0 においては，ホストメモリが 54 GB，GPU 上のメモリが 3GPU 合計で 9 GB と，後者の方がはるかに小さい．そのため行列データをより大きなホストメモリに配置することとした．このときアクセラレータの演算の際に PCIe 通信が必要となる点に注意が必要である．この点は，行列データをデバイスメモリに置くという RoadRunner[8] における方針とは対照的である．

計算性能と通信性能の比：表 1 に示すように，一般的にヘテロ型システムは通常のクラスタよりに比べ，通信性能が相対的に低くなる傾向にある．TSUBAME 2.0 でもノードあたり 8 GB/s と，絶対通信性能は他の多くのシステムより優れているが，演算性能が約 1.7 TFlops と高いため，相対的にはノード間通信のコストは大きくなる．そのために，通信と計算のオーバラップなどの，通信コストを隠ぺいする技術はこれまでよりも重要となる．

4.2 基本的な実装

以上の方針に基づき HPL バージョン 2.0 のソースコードを改変することにより実装を行った．HPL を構成する各 MPI プロセスは，通常通り CPU 上で動作させる（現状ではそれが唯一の選択肢である）．そして GPU は DGEMM と DTRSM のカーネル演算のためにのみ利用する．現在の実装では 1 つの MPI プロセスが 1 つの GPU を駆動するようにして

いる[⋆1]．

3 章に述べた処理のうち DGEMM もしくは DTRSM を呼び出すのはパネル分解と更新計算部分である．前者については HPL のアルゴリズムは変更せず，DGEMM の高速化のためだけに GPU を用いている．後者についても，DGEMM/DTRSM の高速化のために GPU を用いるが，アルゴリズムの改良を次節で説明する．

さて行列データは通常はホストメモリに置かれているため，DGEMM/DTRSM が呼ばれると以下のような処理が必要となる：(1) 入力データのホストメモリからデバイスメモリへの PCIe 通信，(2) GPU 上での計算，(3) 出力データのデバイスメモリからホストメモリへの PCIe 通信．このとき，通信オーバヘッドの隠ぺいと，デバイスメモリサイズよりも大きいデータへの対応という 2 つの目的のために，行列を細切れにして (1)〜(3) のパイプライン処理を行っている．

この処理において，計算対象となる行列サイズが大きい場合には GPU の演算能力を有効に利用できるが，小さい場合には PCIe 通信のオーバヘッドが相対的に大きくなりオーバラップによっても隠ぺいしきれず，速度上は不利である．この点は特にパネル分解が呼び出す DGEMM において問題となる．現在の実装では，どの辺も 512 以上である行列の計算の場合は上記のように GPU を利用し，それ以外の場合は CPU 上の BLAS を利用することとした．

4.3　細粒度なオーバラップ処理の実装

HPL の各ステップにおいては，DTRSM/DGEMM による更新処理だけでなく行交換通信および次ステップのためのパネルブロードキャストという MPI 通信も行われる．このオーバヘッドは，4.1 節で議論したように特にアクセラレータにより計算性能が高速化されたシステムにおいて相対的に大きくなるため，オリジナルの HPL よりもさらなる隠ぺい処理を行うことが有効である．

我々は，行交換通信および更新処理が異なる列間での依存を持たず並行に行うことが可能である点に注目し，計算と PCIe 通信間のオーバラップに加え MPI 通信もオーバラップすることとした．そのため，U を列方向に分割して細切れに処理するように変更した．つまり，図 5 に述べたアルゴリズムを**図 6** のように変更した．ここでは，各プロセスが持つ U を列方向分割したものを $U_0, U_1, U_2 \cdots$，A_k を列方向分割したものを $A_0, A_1, A_2 \cdots$ と呼んでいる．また，MPI 通信を行うスレッド（thread1）と別に，GPU との PCIe 通信，カーネル呼び出しを行うスレッド（thread2）を生成している．この手法においては，オーバラップにより実行時間の多くにおいて GPU 計算が走ることとなる．オリジナル版と異なり，L の MPI 通信中も計算を行う．GPU が動作していないのは L の PCIe 通信，U_0 の行交換中および PCIe 通信中など，相対的にはごく一部の時間である．

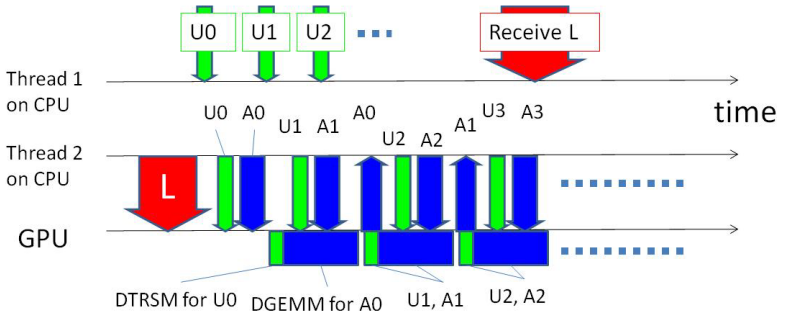

図 6　TSUBAME 2.0 上の HPL の 1 ステップのアルゴリズム
Fig. 6　The modified HPL algorithm on TSUBAME 2.0 for a single step.

なおこの処理を応用し，細切れにした行列の一部を CPU に担当させることにより，カーネル実行に GPU と CPU の双方を用いる版も実装した．しかし大規模 Linpack 実験においては，CPU 併用による速度向上は見られないか，やや性能が下がることが観測された．本来は 5〜8%程度の向上が見込めるはずである．これは CPU によるバス利用と MPI 通信との衝突のためと推測されるが，詳細は今後の課題の 1 つである．

5. 評 価 実 験

5.1　パラメータチューニング

本節では TSUBAME 2.0 上での評価環境とパラメータチューニング手法について述べる．実験に用いたシステムソフトウェアは，SUSE Linux Enterprise 11，OpenMPI 1.4.2，GCC 4.3，CUDA 3.1 である．BLAS ライブラリとしては，Xeon においては GotoBLAS2 1.13 [7]，Tesla GPU においては NVIDIA によって提供された内部バージョンの DGEMM/DTRSM 関数を用いた[6]．これは NVIDIA 公式 BLAS の CUBLAS よりも高速である．Xeon プロセッサの TurboBoost 機能はオフとした．

HPL は多数のパラメータを持ち，性能はその設定に影響される．今回の実験で利用したパラメータを**表 2** に示す．あわせて，東京工業大学の以前のスパコンである TSUBAME

⋆1 複数の GPU を駆動するように変更することは容易である．

表 2　利用した HPL のパラメータおよび Linpack 性能．TSUBAME 2.0 での測定で用いたものに加え，旧システムである TSUBAME 1.2 において，CPU のみを用いた測定およびアクセラレータを含めたシステム全体の測定について示す

Table 2　HPL parameters and the Linpack performance. In addition to ones used on TSUBAME 2.0, the table includes two cases on the predecessor system, TSUBAME 1.2; CPU-only measurement and the whole measurement including all accelerators.

	TSUBAME 1.2 (CPU)	TSUBAME 1.2	TSUBAME 2.0
Matrix size	1,334,160	1,059,839	2,490,368
Block size	240	1,152	1,024
Process mapping	Row-major	Row-major	Row-major
# of processes ($P \times Q$)	36×144	40×50	59×69
Panel factorization	Right-looking	Right-looking	Right-looking
NBMIN, NDIV	4, 2	16, 2	16, 2
Panel broadcast	1ring	1ring	1ring
Look-ahead depth	1	1	1
Swap	Mix (threshold = 240)	Mix (threshold = 16)	Mix (threshold = 16)
L1 Matrix form	transposed	no-transposed	no-transposed
U Matrix form	transposed	no-transposed	no-transposed
Equilibration	yes	yes	no
Alignment	8 double words	8 double words	8 double words
Linpack performance	38.18 TFlops	87.01 TFlops	1.192 PFlops
Theoretical performance	49.87 TFlops	163.2 TFlops	2.288 PFlops
Efficiency	76.6%	53.3%	52.1%

1.2 における測定[5]，および TSUBAME 1.2 でアクセラレータを用いず CPU のみを用いた測定（TSUBAME 1.2(CPU)）で用いたパラメータを示す．TSUBAME 2.0 で主にチューン対象としたのは，カーネル演算性能に大きな影響を及ぼすブロックサイズ，プロセッサへのプロセス割当て，プロセス格子サイズであり，それぞれを以下で議論する．その他のパラメータについては原則として TSUBAME 1.2 と同様のものを用いた．

プロセッサへのプロセス割当て：現在の実装では 1 つの MPI プロセスが 1 つの GPU を駆動するため，TSUBAME 2.0 においては各ノードに 3 プロセス（= GPU 数）を起動することとした．ここで，各プロセスが用いる CPU コアと GPU，およびホストメモリのアフィニティを考慮する，つまり近い箇所にあるようにすることが望ましい．そのため，図 3 の構成から Socket 0 CPU に 1 プロセスをバインドし，Socket 1 CPU に 2 プロセスをバインドし，それぞれ近い方の GPU を用いることとした．メモリのアロケーションポリシは first touch としたため，各プロセスが用いるメモリは，基本的に CPU コアと近いソケット側に置かれる．ただし Socket 0 側と Socket 1 側でプロセス数が異なる（前者は 1，後者は 2）ため，Socket 1 側のメモリ利用があふれ，Socket 0 側から確保される場合がある．後に述べる Linpack 実行の問題サイズでは，その現象は起こらないか，あふれる量は非常に小さかった．なお本実験で各ノード中の 3 つのプロセスは，プロセス格子上で行方向に隣接することとした．そのためプロセス格子の幅 Q は 3 の倍数となる．

以上のような設定とは別の選択肢として，1 つの MPI プロセスがノード内の全 GPU を駆動する実装も考えられる．この場合はプロセスが持つ部分行列が双方のソケットのメモリにまたがりうるので，細粒度にアフィニティを考慮し GPU 間の負荷割当てを行う必要があり，その実装は今後の課題の 1 つである．

プロセス格子サイズ：計算に参加する MPI プロセス数は，用いるノード数の 3 倍であり，これらを $P \times Q$ のサイズの二次元プロセス格子に構成する必要がある．よく知られたように，P と Q の値は近い方が効率的である．PQ を一定とするとき，通信量 $O(N^2(P+Q))$ を小さくするためである．全体実験においては，ノード故障の影響もあり利用可能なノード数は 1,408 台のうち 1,360 台強であった．できるだけこのほとんどを用い，かつプロセス格子を正方形に近づけるために，利用ノード数 1,357，プロセス数 4,071 とし，プロセス格子サイズを 59×69 とした．

DGEMM 性能に基づくブロックサイズの決定：次に Linpack 中のブロックサイズ B のチューニングについて述べる．この点は PCI 通信を必要とするヘテロ型システムにおいては，演算量-PCIe 通信量比を向上させるために特に重要となる．検討のために，GPU 上の DGEMM（行列積）の速度を**図 7** に示す．これは 1GPU 上で前述の NVIDIA 内部カーネルを動作させた結果である．DGEMM の前後で行列データはホストメモリにあるとした．つまり性能は PCIe 通信の影響を含む．行列サイズとしては，Linpack で頻出する行列積のパターンを考慮し，$(M \times B)$ 行列と $(B \times M)$ 行列の積とした．また利用したカーネルの性能は B が 64 の倍数のときに良好であると判明しており，グラフには $B = 512$，1,024，1,280 の場合を示した．一般的に B，M が大きいほうが PCIe 通信コストが相対的に下がるために性能が良く，最大で 350 GFlops 程度である．十分な性能を得られるブロックサイズ B を選択する必要があるが，逆に B が大きすぎると，DGEMM 性能以外への悪影響，たとえば負荷分散の悪化やパネル分解のコスト（$O(N^2B)$）の上昇の影響を受けてしまう．B を 1,024 より大きくしても DGEMM 性能上昇は無視できるほどといえるため，Linpack 実行に用いるブロックサイズは $B = 1{,}024$ とした．

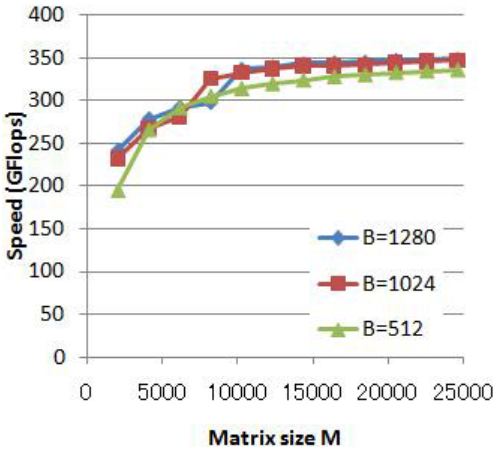

図 7　M2050 1GPU 上の行列積性能．NVIDIA 内部カーネル利用．($M \times B$) 行列と ($B \times M$) 行列の積
Fig. 7　Matrix multiplication performance on a M2050 GPU. The internal kernel by NVIDIA is used. A ($M \times B$) matrix and a ($B \times M$) matrix are multiplied.

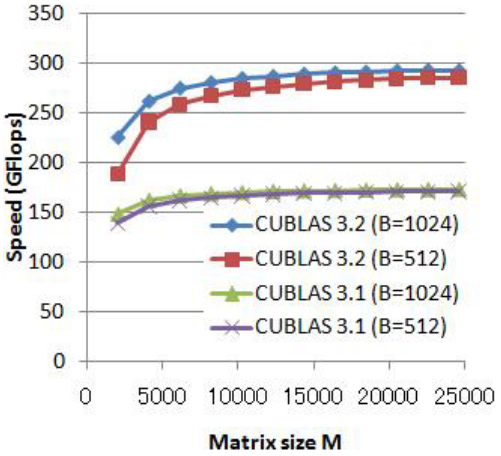

図 8　M2050 1GPU 上の行列積性能．CUBLAS 利用．($M \times B$) 行列と ($B \times M$) 行列の積
Fig. 8　Matrix multiplication performance on a M2050 GPU. CUBLAS is used. A ($M \times B$) matrix and a ($B \times M$) matrix are multiplied.

なお TSUBAME 1.2 においては，用いていた ClearSpeed アクセラレータの特性から B が 96 の倍数のときに効率的である点，および上記のような予備実験の結果を基に，B を 1,152 と決定した．

Equilibration パラメータ：“Equilibration” とは行交換通信の処理において，プロセス間の通信量の均衡化を行う処理である．この処理は一般的には効率化に寄与するため，TSUBAME 1.2 では行う（yes）こととした．一方，TSUBAME 2.0 上では行交換通信を列方向に細切れにしたために，Equilibration 処理にともなう MPI メッセージ数が増加し，性能が低下する現象を確認した．そのため TSUBAME 2.0 では行わない（no）とした．

ここで，DGEMM 性能について補足する．図 7 における DGEMM の最高速度 350 GFlops は，M2050 GPU の理論性能 515 GFlops より大きく下がっている．オンボードの，PCIe の影響を含まない場合でも 360 GFlops 程度（理論性能の 70%）であった．これは，TSUBAME 1.2 でも採用されていた前世代の S1070 GPU においては，理論性能 86.4 GFlops に対し DGEMM 性能 80 GFlops（93%）であったことと対比すると，効率は低いといえる．この点については，NVIDIA 技術者より，M2050 を含む Fermi 世代の GPU ではアーキテクチャの特性から理論値の 75%（= 386 GFlops）が DGEMM 性能の限界である旨の情報提供を受けた．そのため現世代の GPU ではこの効率低下はやむをえない．また比較のために，NVIDIA 公式ライブラリである CUBLAS の DGEMM 関数の同条件での性能を**図 8** に示す．CUBLAS バージョン 3.1 よりも 3.2 の方が大きく性能向上しているものの，それでも内部バージョンのほうが依然高性能と分かった．

5.2　Linpack 実行性能

提案実装のスケーラビリティ評価のために，TSUBAME 2.0 の 128 ノードまでを用いた Linpack 測定結果を**図 9** に示す．縦軸は速度をノード数で割った値である．ノードあたりの行列サイズが 35 GB 程度となるように調整した場合であり，結果は弱スケーリングを示している．8 ノードから 128 ノードまでのすべてにおいてノードあたり 880 GFlops 程度と，良好なスケーラビリティが得られている．4 ノード以下において 5%程度性能が低いという，やや直観に反した結果が得られているが，これは Linpack の性質上ノード数が少ない場合にパネル分解のコストが相対的に大きく見えるという理由により説明可能と考える．

TSUBAME 2.0 全体を用いた Linpack 測定を，システム導入準備期間中である 2010 年 10 月中旬に行った．用いたノード数は前述のとおり，1,408 ノード中の 1,357 ノードである．プロセス数は 4,071，プロセス格子サイズが $P \times Q = 59 \times 69$ である．行列サイズ N については，1 ノードあたりの部分行列が 35 GB 程度となるように調整し，$N = 2{,}490{,}368$ とした．

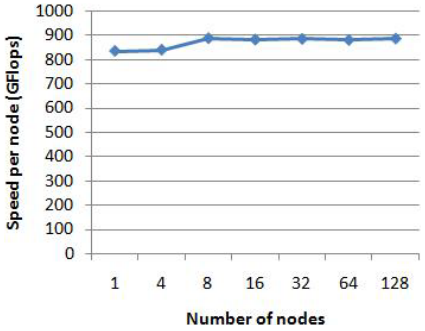

図 9　128 ノード以下での TSUBAME 2.0 上での Linpack 性能．縦軸はノードあたりの性能を示す
Fig. 9　Linpack performance on TSUBAME 2.0 using 128 nodes or less. The vertical axis shows performance per node.

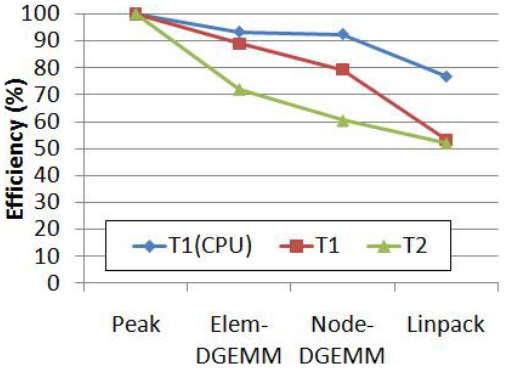

図 10　TSUBAME 2.0，TSUBAME 1.2，TSUBAME 1.2（CPU のみ）上の Linpack の実行効率解析結果
Fig. 10　Analysed results of Linpack efficiency on TSUBAME 2.0, TSUBAME 1.2, TSUBAME 1.2 (CPU only).

このとき，プロセスが担当する部分行列のサイズは最大のプロセスにおいて 43,008 × 36,864 である．

この実行には 8,640 秒かかり，全体性能で 1.192 PFlops，ノードあたりの性能 878 GFlops を達成した．これは国内で初めて 1 PFlops を超えた実行であり，TSUBAME 1.2 の場合の 13.7 倍に相当する．この結果は 2010 年 11 月の Top500 ランキングにおいて世界 4 位にランクされた．なお 1 位の Tianhe-1A，3 位の Nebulae も GPU を用いたヘテロ型システムとなっている．

5.3　更新計算に注目した性能解析

利用した 1,357 ノードの理論演算性能は 2.288 PFlops であるため，Linpack 性能と理論性能の比である実行効率は 52.1%となる．これは多くの大規模スパコンが 70〜80%であるのに対して低く，TSUBAME 1.2 においてアクセラレータを用いた場合の 53.3%に近い．原因の解析を，Linpack 実行時間の多くを占める更新計算における DGEMM の性能に注目して行った．その解析結果を**図 10** に示す．グラフは，TSUBAME 2.0（T2），TSUBAME 1.2（T1）および，TSUBAME 1.2 のうち Opteron CPU のみを用いた場合（T1(CPU)）の 3 通りを示す．

TSUBAME 1.2 の各ノードは，Opteron 16 コアと ClearSpeed アクセラレータ，そして一部のノードが Tesla S1070 GPU を 2GPU 持つ．TSUBAME 2.0 と特に異なる点としては，アクセラレータの種類・個数が異なるノードが混在する（ノード間ヘテロ性），ネットワークはフルバイセクションではなく上流のバンド幅が限られたツリーである，という点があげられる．システムおよび利用した Linpack アルゴリズム詳細については既発表[5] を参照されたい．

最も左側のプロットは理論性能である 100%を示し，最も右側のプロットは Linpack 性能/理論性能を示す．Elem-DGEMM は，各システムの CPU コアもしくはアクセラレータ単体で DGEMM を実行し，それを合計した値に相当する．また Node-DGEMM は，各ノードにおいてアクセラレータおよび CPU で，DGEMM を Linpack 実行時と同様のプロセッサで（TSUBAME 1.2 では全プロセッサ種，TSUBAME 2.0 では GPU のみ）実行した場合に相当する．PCI 通信のコストやバス衝突コストは，Elem-DGEMM と Node-DGEMM の差に含まれる．

グラフから，理論性能と Linpack 性能の乖離の原因は，共通してアクセラレータを用いる TSUBAME 1.2 と TSUBAME 2.0 の間でさえ大きく異なることが分かる．TSUBAME 1.2 においては Node-DGEMM と Linpack の差が最も重大であるが，これにはノード間ヘテロ性の影響や，フルバイセクションでないことによる通信コストの上昇が含まれる．またこのときの実装では，4 章で述べたような細粒度の U 交換のオーバラップを行っていなかったことも原因の 1 つと考えられる．

一方，TSUBAME 2.0 においては Peak と Elem-DGEMM の差が最大である．これは 5.1 節で述べたように，Fermi 世代の GPU において DGEMM の性能が抑えられていることが大きな原因といえる．このため GPU/CPU ハードウェアの変更がない限り，TSUBAME

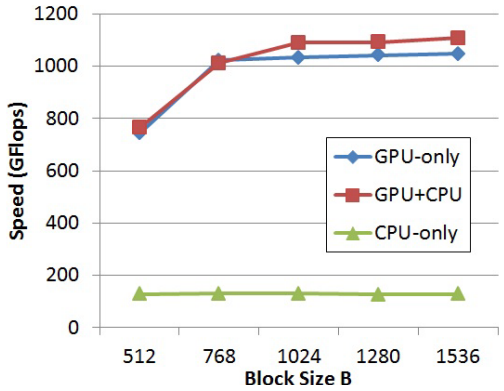

図 11　TSUBAME 2.0 ノード上の合計 DGEMM 性能．3GPU を同時に利用した場合（GPU-only），12CPU コアを同時に利用した場合（CPU-only），GPU と CPU を同時に利用した場合（GPU+CPU）を示す

Fig. 11　Total DGEMM performance of a single TSUBAME 2.0 node. The graph includes when three GPUs are simultaneously used (GPU-only), when 12 CPU cores are used (CPU-only), and both GPUs and CPUs are used (GPU+CPU).

2.0 における Linpack 実行効率は 72%を超えることができない．72%に近づけることが可能か議論するためには，Elem-DGEMM，Node-DGEMM，Linpack 性能の差異について考慮する必要がある．

まず Elem-DGEMM と Node-DGEMM の差異について議論する．この差異は PCIe 通信やバス衝突の影響があることと，後者では CPU コアを用いていないことである．CPU コアの利用有無の影響を，**図 11** に示す．本図は一ノード内の 3GPU を同時に利用した場合（GPU-only），12CPU コアを同時に利用した場合（CPU-only），GPU と CPU を同時に利用した場合（GPU+CPU）の DGEMM 性能を示す．いずれの場合も計算に参加するのは 3 プロセスであり，それぞれが $(24{,}576 \times B)$ 行列と $(B \times 24{,}576)$ 行列の積を行った．なお GPU+CPU においては，PCIe 通信に 3 コアを裂き，DGEMM に参加するのは計 9 コアとした．B = 1,024 以上では CPU 併用の意義はあるという結果であり，B = 1,024 では GPU-only より GPU+CPU のほうが 57 GFlops 高速である．ただしこの差は CPU-only の 129 GFlops よりも小さく，その理由は 3 コア減っている点とバス衝突の双方と推測される．さて，GPU+CPU（B = 1,024）の性能はノードのピーク値の 64.6%であり，GPU-only の 61.2%よりも 3.4%程度高い．よってカーネル計算における CPU 併用により Linpack 性能の実行効率も向上の余地があるといえる．実際の大規模実行においてその向上が得られなかった理由としては，更新計算中にバックグラウンドで起こる MPI 通信も CPU やメモリバスを占有することによる影響や，CPU 併用によりノード性能のジッタリングが起こりやすくなったことと推測しており，その影響を Linpack 実行において十分抑制できるか否かは今後の課題である．

Node-DGEMM と Linpack 性能の差異の要因は複合的であり，またインターコネクト構成の影響など，システム全体を用いた再実験が必要になる点も含まれるため，分析はより困難である．次節では，差異の原因の 1 つとなるパネル分解のオーバヘッドについて，ボトルネックとなっているか検証する．

5.4　パネル分解コストの影響

前節では HPL 中で最も計算量の多い更新計算について解析したが，ここではパネル分解のオーバヘッドに関して議論する．HPL においては，異なるステップのパネル分解処理どうしは並行処理できずクリティカルパス上にあるため，もしその全体実行時間に占める割合が大きいとボトルネックになる．TSUBAME 2.0 全体での実験時のログから，各ステップにおけるパネル分解時間の合計を求めると 2,110 秒であった．これは全体実行時間 8,640 秒の 24.4%にあたる．

この結果は一見パネル分解が大きなボトルネックとなっていることを示すように見えるが，必ずしもそうではない．各ステップにおいてパネル分解に参加するのは，$P \times Q$ プロセスのうちの 1 プロセス列上の P プロセスのみであり，それらのプロセスがパネル分解をしている間は他プロセスは更新計算などの他の処理を行うことができるためである．よって各プロセスがパネル分解に費やす平均時間は $2110/Q = 30.58$ 秒であり，これは全体実行時間の 0.44%となる．

今回の実験ではパネル分解に含まれる DGEMM 演算に GPU を用いることによりパネル分解の効率化を行った．一方パネル分解をすべて CPU で行った場合には，小規模実験においてはパネル分解の時間は 2〜3 倍となった．全体実験において同様にパネル分解時間が 2〜3 倍となるとすると，各プロセスがパネル分解に費やす時間の増加は 1%程度に抑えられる一方，クリティカルパスを長くし，処理時間の隠ぺいをしきれなくなり，全体実行時間を長くすると推測される．以上のように，パネル分解にも GPU を用いた効果はクリティカルパス長に現れたと推測される．

5.5　電 力 性 能

TSUBAME 2.0 全体を用いた Linpack 実行時の消費電力について，分電盤の記録を基に以下のように測定・算出した．実行に先だって，まず TSUBAME 2.0 の機器が接続される

分電盤が記録する電力値が十分安定していることを示すため，以下の確認を行った．ある分電盤が接続される 90 ノードにおいて，負荷プログラムを一定時間実行するという処理を 3 回実行した．それにより 3 回とも分電盤が記録する積算電力は 1%以下の誤差であることを確認した．

Linpack 実行時の消費電力については，ノード・スイッチが接続された全分電盤の積算電力の合計から求めた．このとき，並列ファイルシステム，MCS 空調，チラーは別系統の分電盤であるため含まれていない．ただし，用いた分電盤にはアイドルであったノードも含まれているため，記録値からそれらの電力を減算した．その結果，Linpack 実行中のシステムの平均消費電力は 1,440 kW であった．分電盤レベルの測定であるので，ノードの電源ユニットにおけるロス分は含まれている．

一方でスーパコンピュータの電力性能比のランキングである Green500 [1] には 1,243.8 kW という値を提出している．この値は Green500 の電力測定ルールを遵守すべく，以下のように求められている．まず電力測定の期間は，Linpack 実行中の 20%以上と定められている．Linpack 実行中の最後の 21.3%の期間の平均電力とした．また，エッジスイッチの電力は含む必要があるが，コアスイッチの電力（この場合 36 kW であった）を含まなくてよいと Green500 委員会から回答を得たのでそのようにした．このときの電力と演算性能の比は 958 MFlops/W となり，2010 年 11 月の Green500 において世界 2 位となった[⋆1]．さらに，上位が小規模なプロトタイプシステムであったこともあり，「the Greenest Production Supercomputer in the World」賞を獲得した．

6. おわりに

ペタフロップスの演算性能を持つ TSUBAME 2.0 スーパコンピュータにおいて Linpack を実行し，1.192 PFlops の速度性能，958 MFlops/W の電力性能比を達成した．その性能解析を，旧システムである TSUBAME 1.2 と比較しつつ行い，性能を制約する要因について議論した．TSUBAME 2.0 は速度と電力性能比の双方において世界トップクラスを実現している希有な例であり，実際に 2010 年 11 月の Top500 と Green500 ランキングの双方で 10 位以内であるのは TSUBAME 2.0 のみである．

現在の実装には最適化の余地が残っており，まず GPU と CPU の混合カーネル実行時の性能の安定化を行いたい．また電力性能比を向上させるために CPU/GPU のクロック/電圧と性能の関係に基づいた最適化を行いたい．

プログラミング手法の観点からは，今回の実装のように MPI や CUDA をそのまま用い，通信オーバラップなども手作業で記述するのは手間がかかりすぎであるという認識が広まってきている．その考えのもと，行列演算における 1 つの方向性として，行列データを分割して（たとえばブロック単位）分割データに対するタスク依存関係を DAG の形で記述させ，GPU クラスタ上でタスクスケジューリングを行う StarPU [3] や DPLASMA [4] などのシステムが提案されている．これらは Pivoting なしの Cholesky 分解などでは大きな効果をあげているが，Linpack のように pivoting 処理による行交換などの細粒度の通信が必要な場合に，最適か否かは自明でない．SMPSS/MPI [9] のように，細粒度の send/recv 通信を明示的に記述させることにより Linpack で良好な性能を得ている報告もなされているが，これは CPU クラスタ上のものである．今後の課題として，上記のような技術によりチューニングの手間，別アーキテクチャへの移植の手間の軽減と高性能の両立について検討する予定である．

謝辞　実験にあたって日本電気，日本ヒューレット・パッカード，NVIDIA，マイクロソフト，Voltaire，DDN，東京工業大学学術国際情報センターをはじめとする皆様に多大なご協力をいただきました．本研究の一部は東京工業大学グローバル COE「計算世界観の深化と展開」，JST-CREST「次世代テクノロジのモデル化・最適化による超低消費電力ハイパフォーマンスコンピューティング」，JST-ANR「ポストペタスケールコンピューティングのためのフレームワークとプログラミング」，科学研究費補助金（特定領域研究課題番号 18049028）の援助による．

参考文献

1) The GREEN500 list, available from ⟨http://www.green500.org/⟩.
2) TOP500 supercomputer sites, available from ⟨http://www.top500.org/⟩.
3) Augonnet, C., Thibault, S., Namyst, R. and Wacrenier, P.-A.: StarPU: A unified platform for task scheduling on heterogeneous multicore architectures, *Proc. International Euro-Par Conference on Parallel Processing*, pp.863–874 (2009).
4) Bosilca, G., Bouteiller, A., Danalis, A., Faverge, M., Haidar, A., Herault, T., Kurzak, J., Langou, J., Lemarinier, P., Ltaief, H., Luszczek, P., Yarkhan, A. and Dongarra, J.: Distibuted dense numerical linear algebra algorithms on massively parallel architectures: DPLASMA, Technical Report UT-CS-10-660, University of Tennessee Computer Science (2010).
5) Endo, T., Nukada, A., Matsuoka, S. and Maruyama, N.: Linpack evaluation on a

⋆1 当初の公開の後に修正があり，国立天文台 GRAPE-DR システムが上位にランクされたため，実質 3 位となる．

supercomputer with heterogeneous accelerators, *Proc. IEEE IPDPS10*, p.8 (2010).
6) Fatica, M.: Accelerating Linpack with CUDA on heterogeneous clusters, *Proc. Workshop on General-purpose Computation on Graphics Processing Units* (*GPGPU '09*) (2009).
7) Goto, K. and van de Geijn, R.A.: Anatomy of high-performance matrix multiplication, *ACM Trans. Mathematical Software*, Vol.34, No.3, pp.1–25 (2008).
8) Kistler, M., Gunnels, J., Brokenshire, D. and Benton, B.: Petascale computing with accelerators, *Proc. ACM Symposium on Principles and Practice of Paralle Computing* (*PPoPP09*), pp.241–250 (2009).
9) Marjanovi, V., Labarta, J., Ayguade, E. and Valero, M.: Overlapping communication and computation by using a hybrid MPI/SMPSs approach, *Proc. ACM ICS'10*, pp.5–16 (2010).
10) Petitet, A., Whaley, R.C., Dongarra, J. and Cleary, A.: HPL – A portable implementation of the high-performance Linpack benchmark for distributed-memory computers, available from ⟨http://www.netlib.org/benchmark/hpl/⟩.
11) 遠藤敏夫，額田　彰，松岡　聡：ヘテロ型スーパーコンピュータ TSUBAME 2.0 の Linpack による性能評価，ハイパフォーマンスコンピューティングとアーキテクチャの評価に関する北海道ワークショップ（HOKKE-18），pp.1–6 (2010).

(平成 23 年 1 月 28 日受付)
(平成 23 年 5 月 24 日採録)

遠藤　敏夫（正会員）

1974 年生．2001 年東京大学大学院理学系研究科情報科学専攻博士課程修了．博士（理学）．2007 年より東京工業大学情報理工学研究科グローバル COE「計算世界観の深化と展開」特任准教授．アクセラレータを含むハイブリッド型スーパコンピュータを中心とする高性能計算・省電力計算の研究に従事．日本ソフトウェア科学会，IEEE-CS，ACM 各会員．

額田　　彰（正会員）

1976 年生．2001 年東京大学大学院理学系研究科修士課程修了．2004 年同大学院情報理工学系研究科博士課程退学．同年より科学技術振興機構戦略的創造研究事業の研究員，2007 年より東京工業大学学術国際情報センター産学官連携研究員．FFT 等の数値計算アルゴリズムの高速な実装手法に関心を持つ．IEEE-CS 会員．

松岡　　聡（正会員）

博士（理学）（東京大学）．2001 年より東京工業大学学術国際情報センター教授．専門は高性能システムソフトウェア・グリッド/クラウド・並列処理（GPU・省電力・高信頼）等．NAREGI プロジェクト（2003–2007），科研特定領域・情報爆発（2006–2010），JST-CREST Ultra Low Power HPC（2007–2011）のリーダー/サブリーダー．2006 年のスパコン TSUBAME はわが国トップを 2 年間維持，さらに 2010 年日本初のペタフロップススパコンの TSUBAME 2.0 を構築．OOPSLA 2002，CCGrid2003/2006，Supercomputing 2009 の論文委員長等，国際学会の要職を多々歴任．情報処理学会坂井記念賞（1999 年），学術振興会賞（2006 年），ISC Award（2008 年）等受賞．欧州 ISC フェロー．