スーパーコンピュータ 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[†,††] 額田彰[†,††] 松岡聡[†,†††,††]

Intel プロセッサに加え NVIDIA GPU を備え，2010 年 11 月に稼働開始したヘテロ型スパコンである 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 1400 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 2.4PFlops 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行い，実行速度として 1.192PFlops 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．

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

Toshio Endo,[†,††] Akira Nukada[†,††] and Satoshi Matsuoka [†,†††,††]

We report Linpack benchmark results on the TSUBAME 2.0 supercomputer, a large scale heterogeneous system with Intel processors and NVIDIA GPUs, operation of which has started in November 2010. The main part of this system consists of about 1400 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 2.4PFlops, 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 1.192PFlops. This is the first result that exceeds 1PFlops in Japan, and ranked as 4th in the latest Top500 supercomputer ranking.

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・設置面積にて実現する上で，アクセラレータの利用が注目されている．2008 年に Top500 スーパーコンピュータランキング[2] で初めて 1PFlops 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．

著者ら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ーコンピュータ 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 2.4PFlops 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

† 東京工業大学
Tokyo Institute of Technology
†† JST, CREST
††† 国立情報学研究所
National Institute of Informatics

図 2 Thin 計算ノードの外観

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 TSUBAME1 用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 1357 ノード，4071GPU を用いたときの実行速度は 1.192PFlops 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 1PFlops を超えたシステムとなった．ピーク演算性能 (1357 ノードで 2.288PFlops) に対する比は 52.1% であり，ピークとの差についての解析についても報告する．またシステムの消費電力は 1243.8kW(Green500 のルールに基づく測定法[1])，電力性能比は 958.35MFlops/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，1400 ノード以上の計算ノードと，合計 7.1PBytes 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 1408 台の Thin ノード，24 台の Medium ノード，10 台の Fat ノードから成る．本論文の実験では 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場合がある．以下，本論文に関連の深い部分について概要を示す．

Thin 計算ノード: 各計算ノード Hewlett-Packard Proliant SL390s G7 は 6 コアの Intel Xeon X5670 2.93GHz 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 54GB の DDR3 メモリを搭載する．また 40Gbps 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 のうち一つは 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．

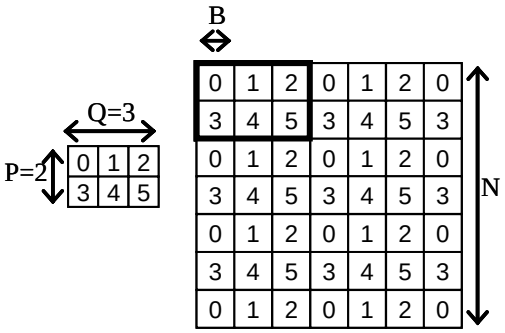

図 4 (左)$P \times Q = 2 \times 3$ プロセスのプロセス格子例，(右)6 プロセスによる $N \times N$ 行列の二次元ブロックサイクリック分割

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

フルバイセクションネットワーク: インターコネクトは 2 段のスイッチから成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 40Gbps 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．

Tesla M2050 GPU: 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，150Gbytes/s のメモリバンド幅を持つ 3GB 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 515GFlops，単精度では 1.03TFlops 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 の良く知られた並列実装である

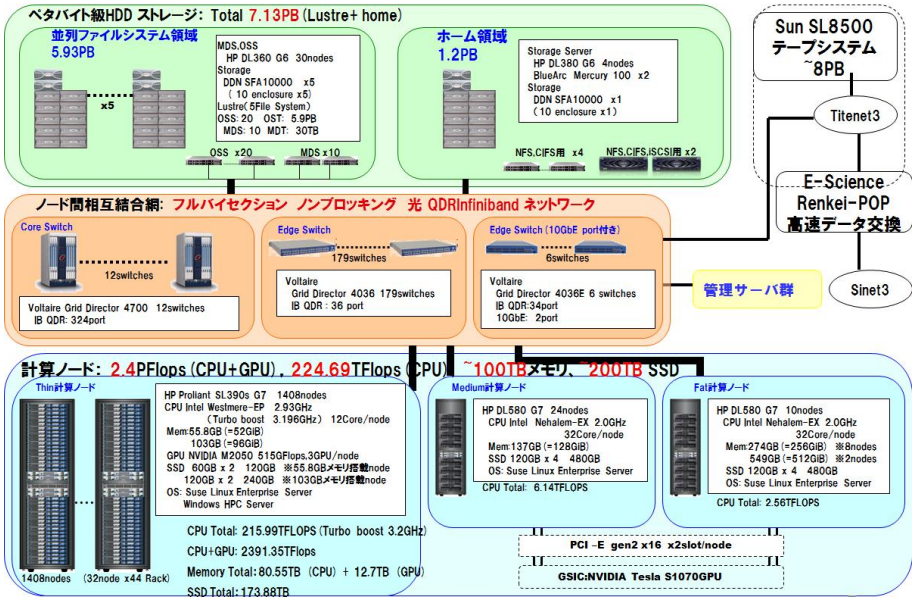

図 1 TSUBAME 2.0 の全体構成図

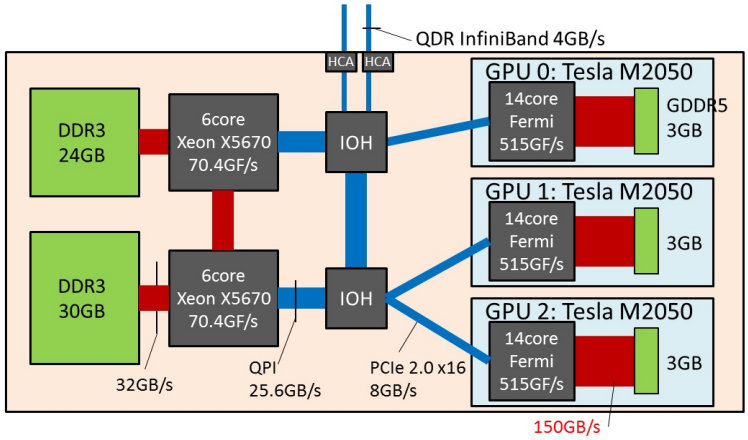

図 3 Thin 計算ノードの内部構成

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☆．計算のほとんどの部分をガウス消去法が占め，その各ステップ (ステップ番号 k とする) は，以下のような処理からなる．

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．

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 においては集約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る．

☆ 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

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．

各 MPI プロセスが行う処理を図 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 が大きい程強いと分かる．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．

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．一方前述の通り，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 54GB，GPU 上のメモリが 3GPU 合計で 9GB 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 8GB/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 1.7TFlops 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 実装とオーバラップの最適化

ここでは TSUBAME 2.0 上の HPL ソースコードの改変について述べる．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 はカーネル演算のためにのみ利用する．行列データは前述のように通常はホストメモリに置かれるため，DGEMM/DTRSM 演算の際には一部ずつデバイスメモリに PCIe を介し送信し，GPU 側で計算する．ここではパイプライン処理により，計算と PCIe 通信のオーバラップを行う．さらには MPI 通信もオーバラップ可能とするため，U を列方向分割して行交換処理を細切れに処理可能なように変更した．つまり，図 5 に述べたアルゴリズムは図 6 のように変更された．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．

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一つである．

また現在の実装では一つの MPI プロセスが一つの GPU を駆動するようにしているが，複数の GPU を駆動するように変更することは容易である．

5. 評 価 実 験

5.1 予備実験とチューニング

まず TSUBAME 2.0 上での予備実験結果とそれに基

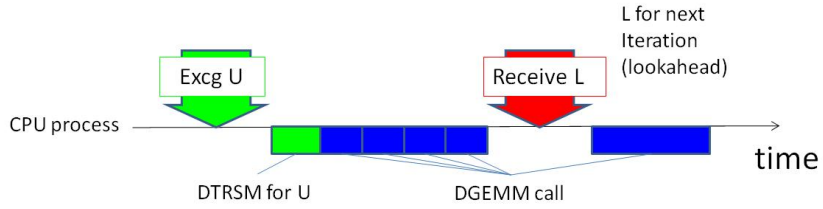

図 5 オリジナル HPL の 1 ステップのアルゴリズムの模式図

表 1 各システムにおけるノード毎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

	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	1685	8	24

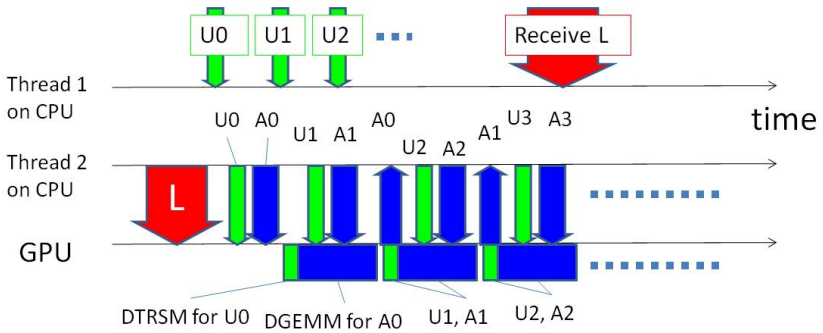

図 6 TSUBAME 2.0 上の HPL の 1 ステップのアルゴリズムの模式図

づくチューニングについて述べる．実験に用いたシステムソフトウェアは，SUSE Linux Enterprise 11, Open-MPI 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 とは異なる．Xeon プロセッサの TurboBoost 機能はオフとした．

プロセス割り当て: 今回の実験では，一つの MPI プロセスが一 GPU を駆動し，各ノードに 3 プロセス (=GPU 数) を起動することとした．この場合，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 に応じて 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．メモリのアロケーションポリシーは 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．以上のような実行設定とは別の選択肢として，一 MPI プロセスがノード内の全 GPU を駆動することも考えられる．性能比較はまだ行っていないが，この場合はメモリのアフィニティの設定がより複雑になると予想される．

ブロックサイズと DGEMM 性能: 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 上で上記の NVIDIA 内部カーネルを動作させたものである．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

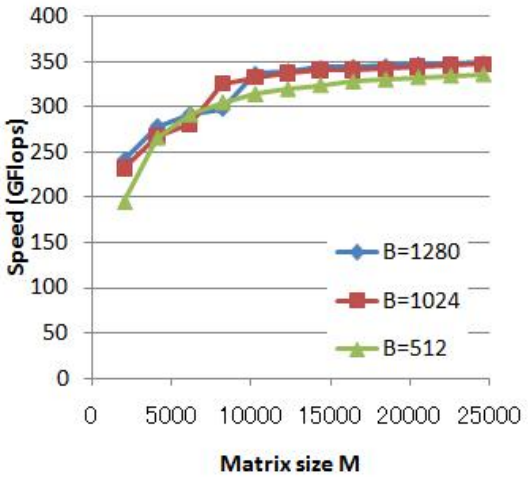

図 7 M2050 1GPU 上の行列積性能．NVIDIA 内部カーネル利用．$(M \times B)$ 行列と $(B \times M)$ 行列の積．

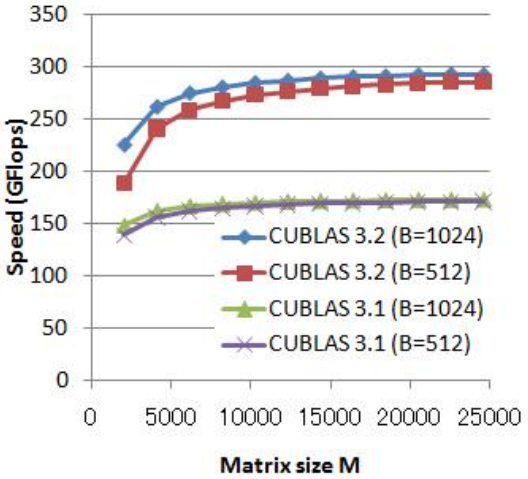

図 8 M2050 1GPU 上の行列積性能．CUBLAS 利用．$(M \times B)$ 行列と $(B \times M)$ 行列の積．

ンを考慮し，$(M \times B)$ 行列と $(B \times M)$ 行列の積とした．グラフから分かるように，一般的に B, M が大きいほうが性能が良く，350GFlops 程度である．これは PCIe 通信コストが相対的に下がるためである．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，B が大きすぎると，負荷分散の悪化やパネル分解のコストの上昇の影響を受けてしまう．B を 1024 より大きくしても性能上昇は無視できるほどと言えるため，Linpack 実行にもちいるブロックサイズは $B = 1024$ とした．

DGEMM 性能について補足: 図 7 における最高速度 350GFlops は，M2050 GPU の理論性能 515GFlops 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 360GFlops 程度であった．これは前世代の S1070 GPU で理論性能 86.4GFlops に対し DGEMM 80GFlops 以上であったのと対象的である．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%(=386GFlops) が DGEMM 性能の限界である旨の情報提供を受けた．また比較のために，NVIDIA 公式ライブラリである CUBLAS の同条件での性能を図 8 に示す．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

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 35GB 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 ノードまでの全てにおいてノードあたり 880GFlops 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．

TSUBAME2.0 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．1408 ノード中の 1357 ノードを実行に用いた．このときプロセス数 (=GPU 数) は 4071 となり，このプロセスを $P \times Q = 59 \times 69$ の格子に構成した．利用パラメータは，$N = 2,490,368, B = 1024$ となっている．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 $43,008 \times 36,864$ であり，1 ノードあたり (3 プロセス) では 35.4GB 程度を占める．

この実行により 1.192 PFlops，ノードあたり 878 GFlops を達成した．これは国内で初めて 1PFlops を超えた実行であり，TSUBAME 1.2 の場合の 13.7 倍に相当する．実行時間は 8640 秒であった．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．なお一位の 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 実行効率の解析

1357 ノードの理論演算性能は 2.288PFlops 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 TSUBAME 2.0 の前身であり，同じくアクセラレータを備えたシステムである TSUBAME 1.2 時の 53%に近いが，その原因は大きく異なることが分かった．原因を解析するために，DGEMM 性能に注目し，その解析結果を図 10 に示す．グラフは，TSUBAME 2.0, TSUBAME 1.2 および，TSUBAME 1.2 のうち Opteron CPU のみを用いた場合の三通りを比較する．

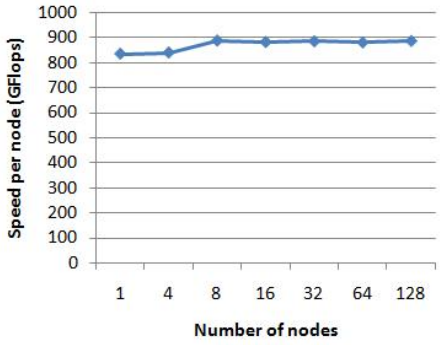

図 9 256 ノード以下での Linpack 性能．縦軸はノードあたりの性能を示す．

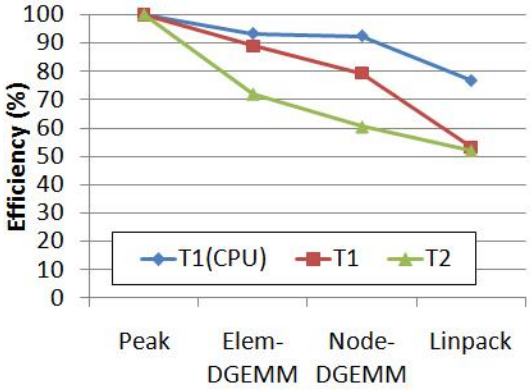

図 10 TSUBAME 2.0, TSUBAME 1.2, TSUBAME 1.2(CPU のみ) 上の Linpack の実行効率解析結果

なお 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，という点が挙げられる．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．

TSUBAME 1.2 と TSUBAME 2.0 における理論性能と Linpack 性能の乖離の原因は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．またこの時の実装では，4 節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一つと考えられる．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言える．現状のハードウェアにおいて性能向上を行うためには，Elem-DGEMM，Node-DGEMM, Linpack 性能の差異を小さくしていく必要があり，その詳細な解析を今後行う予定である．

5.4 電力性能

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．Linpack 実行に先だって，まず TSUBAME 2.0 の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三回実行した．それにより三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 1440kW 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．

一方でスーパー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 1243.8kW 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 36kW 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．この時の電力と演算性能の比は 958MFlops/W となり，2010 年 11 月の Green500 において世界 2 位となった☆．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．

☆ 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

6. おわりに

ペタフロップスの演算性能を持つ TSUBAME 2.0 スーパーコンピュータにおいて Linpack を実行し，1.192PFlops の速度性能，958MFlops/W の電力性能比を達成した．こ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，実際に最新の両方のランキングで 5 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の効率化と，それに対する MPI 通信の影響の軽減を行いたい．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，行列演算におけるひとつの方向性として，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頂きました．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. http://www.green500.org/.
2) TOP500 supercomputer sites. http://www.top500.org/.
3) Cedric Augonnet, Samuel Thibault, Raymond Namyst, and Pierre-Andre Wacrenier. StarPU: A unified platform for task scheduling on heterogeneous multicore architectures. In *Proceedings of International Euro-Par Conference on Parallel Processing*, pages 863–874, 2009.
4) G. Bosilca, A. Bouteiller, A. Danalis, M. Faverge, A. Haidar, T. Herault, J. Kurzak, J. Langou, P. Lemarinier, H. Ltaief, P. Luszczek, A. Yarkhan, and J. Dongarra. Distibuted dense numerical linear algebra algorithms on massively parallel architectures: DPLASMA. Technical Report UT-CS-10-660, University of Tennessee Computer Science, 2010.
5) Toshio Endo, Akira Nukada, Satoshi Matsuoka, and Naoya Maruyama. Linpack evaluation on a supercomputer with heterogeneous accelerators. In *Proceedings of IEEE IPDPS10*, page 8pages, 2010.
6) Massimiliano Fatica. Accelerating Linpack with CUDA on heterogeneous clusters. In *Proceedings of Workshop on General-purpose Computation on Graphics Processing Units (GPGPU '09)*, 2009.
7) K. Goto and R. A. van de Geijn. Anatomy of high-performance matrix multiplication. *ACM Transactions on Mathematical Software*, 34(3):1–25, 2008.
8) Michael Kistler, John Gunnels, Daniel Brokenshire, and Brad Benton. Petascale computing with accelerators. In *Proceedings of ACM Symposium on Principles and Practice of Paralle Computing (PPoPP09)*, pages 241–250, 2009.
9) Vladimir Marjanovi, Jesus Labarta, Eduard Ayguade, and Mateo Valero. Overlapping communication and computation by using a hybrid MPI/SMPSs approach. In *Proceedings of ACM ICS'10*, pages 5–16, 2010.
10) A. Petitet, R. C. Whaley, J. Dongarra, and A. Cleary. HPL - a portable implementation of the high-performance Linpack benchmark for distributed-memory computers. 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. pages 1–6, 2010. 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).