ヘテロ型スーパーコンピュータ TSUBAME の Linpack による性能評価

遠藤敏夫[†1] 松岡聡[†1,†2]
橋爪信明[†3] 長坂真路[†4]

TSUBAME スーパーコンピュータは，655 ノード 10480 Opteron core と 360 枚の ClearSpeed SIMD アクセラレータボードを備えるヘテロ型の大規模クラスタシステムである．本論文では Linpack の並列実装である HPL を TSUBAME 上で効率的に動作させる技法を述べ，性能評価を行う．均一環境を主な対象とした HPL に対し，比較的軽微な修正を行うという方針を採る．提案技法は，細粒度なプロセスによる不均一な計算資源の共有，非同期通信の採用などを含む．修正した HPL を用いてシステムの評価を行ったところ，47.38TFlops を達成した．この結果はヘテロな環境における Linpack の性能としては世界最速のものである．本研究の成果は，今後増加が予想されるヘテロ型スーパーコンピュータの大規模並列計算への適用可能性を示している．

Performance Evaluation of TSUBAME Heterogeneous Supercomputer with Linpack

Toshio Endo,[†1] Satoshi Matsuoka,[†1,†2] Nobuaki Hashizume[†3] and Masamichi Nagasaka [†4]

The TSUBAME supercomputer is a heterogeneous large-scale cluster system, which is equipped with 10480 Opteron CPU cores on 655 nodes and 360 ClearSpeed SIMD accelerator boards. This paper describes techniques to run HPL, which is a parallel Linpack implementation, on the TSUBAME system efficiently, and evaluates the performance. The techniques include sharing heterogeneous computing resources among fine grained processes, and using asynchronous communications. Through the evaluation of the system with the modified HPL, we have observed 47.38TFlops, which is the world's fastest Linpack performance on heterogeneous systems. The result of this work shows that heterogeneous supercomputers, which are expected to be much more popular in the near future, are promising for large scale parallel computations.

1. はじめに

大規模計算機システムを構築する上で，近年のプロセッサの消費電力の上昇が大きな問題となっている．システム全体の電力を現実的な範囲に抑えつつ，広範囲な応用に対して高性能なアーキテクチャとして，汎用的なプロセッサと，より用途を特化した省電力型プロセッサを併用する，不均一なシステムが注目されている．例として，IBM の Roadrunner スーパーコンピュータ (2008 年完成予定) や東京工業大学の TSUBAME スーパーコンピュータが挙げられる．

TSUBAME スーパーコンピュータは，東京工業大学学術国際情報センターにおいて 2006 年 4 月に稼動開始した，現在国内最速のシステムである．このシステムは Intel 互換 CPU である AMD Opteron と Linux OS を採用することにより，汎用性とプログラミングの容易さを確保する．さらに，ClearSpeed 社の SIMD アクセラレータボードをも備えることにより，スペース性能比，電力性能比を向上させることをねらっている．システムが備える 10480 の Opteron CPU core の理論性能は 50TFlops，360 枚のアクセラレータの理論性能は 35TFlops であり，合計で 85TFlops である．

スーパーコンピュータの科学技術計算性能の指標として，Linpack ベンチマークの性能による世界ランキングである Top500[2] が広く知られている．TSUBAME は 2006 年 6 月のランキングにおいて 38.18TFlops を記録し，世界 7 位としてランクされた．この測定に

†1 東京工業大学
Tokyo Institute of Technology
†2 国立情報学研究所
National Institute of Informatics
†3 サン・マイクロシステムズ (株)
Sun Microsystems K.K.
†4 日本電気 (株)
NEC Corp.

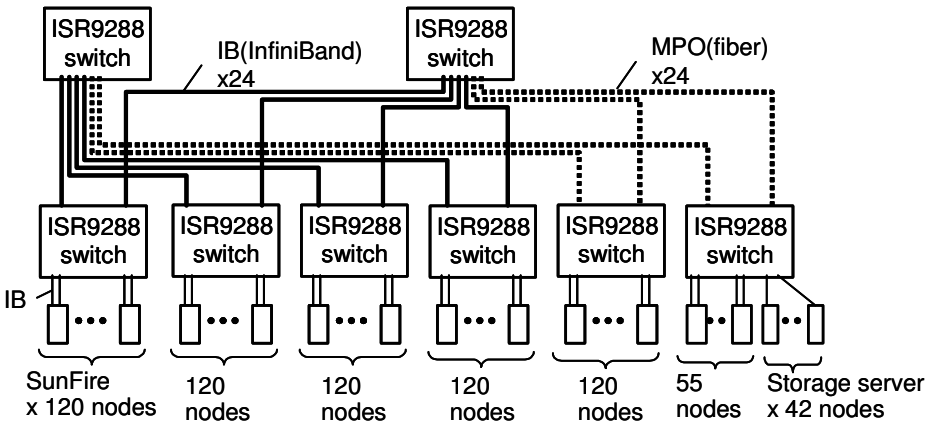

図 1 TSUBAME のシステム構成図

用いられたのは Opteron のみであるため，アクセラレータを併用することによりさらなる性能向上が期待できる．

本論文では，汎用 CPU とアクセラレータの双方を用いて，TSUBAME システム全体の評価結果を報告する．Linpack の並列実装である High-Performance Linpack(HPL)[8] を基とするが，これは本来均一な環境のために設計されている．ヘテロな環境に適合させるため，以下の技法を提案する：CPU とアクセラレータ間の負荷分散，アクセラレータの有無による起動プロセス数の調整，HPL の look-ahead 手法の改良，通信の out-of-order 化．また，アクセラレータの特性に適合するチューニングについても報告する．これらの技法を用いて，TSUBAME のようなヘテロ型スーパーコンピュータで良好な性能を達成可能であることを示す．

2. TSUBAME システム

TSUBAME では，655 ノードの計算サーバ SunFire X4600 と，合計 1.1PBytes のストレージサーバが InifiniBand により接続されている (図 1)．以下，本論文に関連の深い部分について概要を示す．

ファットノード型計算サーバ: 各 SunFire ノードは，dual core 2.4GHz Opteron CPU ☆ を 8 個を持ち，16 CPU core が 32GB のメモリを共有する．またノードは InifiniBand host channel adapter (HCA) を 2 つ持つ．オペレーティングシステムは 64bit 対応 SuSE Linux Enterprise Server 9 であり，Linux カーネルバージョンは 2.6.5 である．また 655 ノードのうち 360 ノードが，PCI-X バスによって接続される ClearSpeed アクセラレータボードを持つ．

高速インターコネクト: 各ノードは 2 本の 10Gbps InfiniBand により 288 ポートの Voltaire ISR9288 スイッチ群に接続される．スイッチ間は，InifiniBand 24 本により接続される

並列プログラムを動作させるために，Message passing interface(MPI) の一実装である Voltaire MPI などを用いることができる．Voltaire MPI は InifiniBand を直接アクセスすることにより高性能を実現している．

SIMD アクセラレータ: ClearSpeed Advance Accelerator Board[1] は，理論性能 96GFlops の演算能力を持つ PCI-X ボードである．アクセラレータボードは 2 つの ClearSpeed CSX600 SIMD プロセッサと 1GB の DRAM を持つ．各プロセッサには，0.5GFlops の演算能力を持つ 96 個の PE が含まれる．なお，アクセラレータ上の演算の入出力データは，1.06GBytes/s の PCI-X バスを介してホストと通信する必要がある．アクセラレータあたりの消費電力は約 25W，360 枚で約 9kW となっており，これはシステム全体のピーク消費電力の 1.2MW の 1%程度に相当する．

アクセラレータを利用する手段として，SIMD 並列プログラミング言語 C^n，基本線形演算を行う CSXL ライブラリ，高速フーリエ変換を行う CSFFT ライブラリなどが提供されている．本論文ではこれらのうち CSXL ライブラリを利用する．CSXL ライブラリには BLAS API のうち dgemm(倍精度行列積) のみが実装されており，それ以外の BLAS 関数については GOTO BLAS などの CPU を利用するライブラリが自動的に呼ばれる．

☆ 16 ノードのみ，2.6GHz CPU

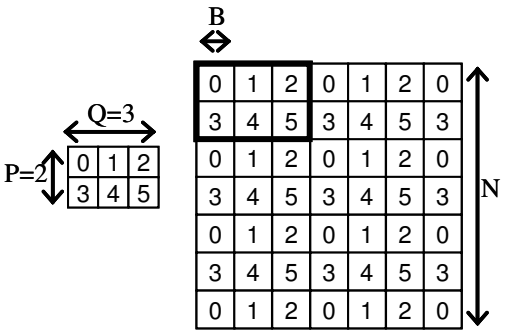

図 2 (左)$P \times Q = 2 \times 3$ プロセスのプロセス格子例，(右)6 プロセスによる $N \times N$ 行列の二次元ブロックサイクリック分割

3. HPL の概要

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 2)．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の通信が発生する．

更新計算: パネル列と，行交換後の第 k ブロック行の内容を用い，行列の未分解部分の更新計算を行う．行列積演算が発生する．

以上のうち，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良く行われている．

Matrix size	1334160
Block size	240
Process mapping	Row-major
# of processes ($P \times Q$)	36×144
Panel factorization	Right-looking
NBMIN, NDIV	4, 2
Panel broadcast	1ring
Look-ahead depth	1
Swap	Mix (threshold=240)
Matrix form	L1 trans, U trans
Equilibration	yes
Alignment	8 double words

表 1 10,368CPU を用いた評価における HPL のパラメータ

4. 予 備 評 価

4.1 Opteron を用いた性能評価

ここで TSUBAME システムの 648 ノード 10,368CPU core を用いた HPL の実行結果を示す．MPI ライブラリとしては Voltaire MPI を，数値演算ライブラリとしては，GOTO BLAS[6] を用いた．GOTO BLAS ライブラリはユーザが指定した数のスレッドを用い，行列演算をノード内で並列化して行うことができる．この評価では GOTO BLAS が用いるスレッド数を 2 とし，各ノードに 8 プロセスずつ起動することにより，ノード内の 16 CPU core を利用する．

この評価に用いられた HPL のパラメータを表 1 に示す．HPL は通信方法やデータ配置などについての様々なチューニング項目を持つ．パネルブロードキャストのアルゴリズムとしては，占有バンド幅を下げることを優先し，リング型トポロジー (1ring) を選択した．また，Look-ahead は HPL が備える最適化機構の一つであり，各ステップの更新計算の最中に以降のステップのパネルブロードキャストを行うことにより，通信待ちの時間を削減するものである．オーバラップ段数 (look-ahead depth) を 1 とした．

行列サイズ $1,334,160$，プロセス数 $36 \times 144 = 5184$ の実験において，速度 38.18TFlops が得られた．このときの実行時間は約 11.5 時間である．648 ノードの Opteron の合計理論ピーク性能は 49.87TFlops であり，それに対する比率は 76.6%となる．

なお，655 ノード全てを利用しなかった理由は，中途半端なノード数の場合にはプロセス格子の形状が悪くなり，かえって性能が悪くなると考えられるためである．

4.2 各計算資源の性能

TSUBAME では半数強のノードが理論性能 96GFlops のアクセラレータを持っている．また，各ノードが

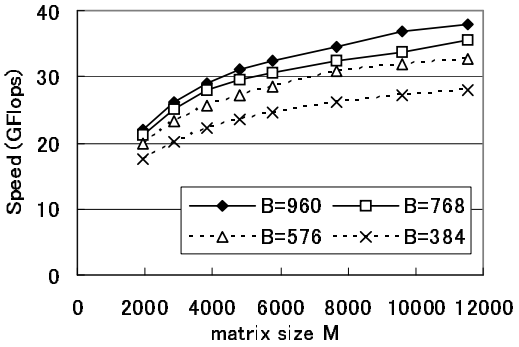

図 3 CSXL ライブラリによるサイズ $(M \times B) \times (B \times M)$ の行列積性能

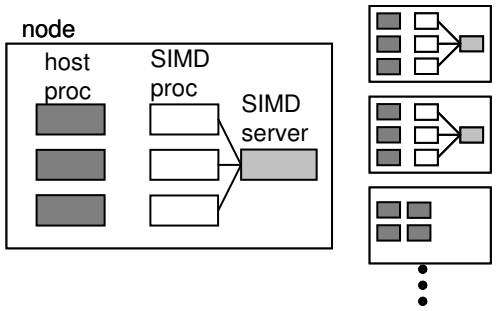

図 4 CPU とアクセラレータを利用するためのプロセス構成

持つ CPU core の理論性能は 4.8GFlops × 16 = 76.8GFlops となっている．両者の性能のオーダが近くなっているため，このシステムで良好な性能を得るためには CPU, アクセラレータのいずれか一方では不足であり，双方を用いる必要がある．

CPU とアクセラレータを併用する手法を説明する前に，ここではそれぞれの性能を述べる．HPL の性能に大きく影響する，行列積演算の性能に注目する．特に HPL では $B \ll M$ であるような B, M に対して，$M \times B$ 行列と $B \times M$ 行列の積が重要である．

まず前節の評価でも用いた，GOTO BLAS の行列積性能を述べる．$B = 240 \sim 960$, $M \geq 1920$ として TSUBAME の 1 ノード上で計測したところ，2 スレッド利用時に 8.5 ∼ 8.8GFlops，4 スレッド利用時に 16.3 ∼ 17.2GFlops であった．これは理論ピーク性能の 85 ∼ 92%程度である．また，B, M の差による速度変動は小さいことが分かった．

次に ClearSpeed により提供される CSXL ライブラリの行列積性能を図 3 に示す．B, M ともに 192 の倍数の場合を調べたが，これはアクセラレータが持つ PE 数が 192 であり，行列サイズがその倍数であるときに性能が良いためである．グラフから速度は B, M に大きく依存し，両者が大きいほど良いことが分かる．これは，行列サイズが大きいほど，演算量が PCI-X 上の通信量に比例して多くなるためと考えられる．グラフの範囲では，速度は 18GFlops($M = 1920, B = 384$) から 38GFlops($M = 11520, B = 960$) となっている．

なお，現在の CSXL ライブラリの性能は理論ピーク性能の半分以下となっている．この原因は PCI-X バスのバンド幅の低さに加え，現在のライブラリが充分最適化されていないためと考えられ，将来の版で改善されることが期待される[7]．

5. 提 案 手 法

5.1 不均一環境における課題

これまで述べたように，TSUBAME は性能・特性の異なる計算資源を持っている．一方 HPL では，各プロセスにほぼ均一の行列データが分散されるため，各プロセスに均一性能の計算資源が割り当てられるときに効率よく動作する．我々のねらいは HPL の構造に大きな変更を加えずに，CPU とアクセラレータの双方を有効利用することである．

HPL においては行列積の負荷分散に最も注意をはらう必要がある．CPU とアクセラレータを用いる一つの方法として，BLAS ライブラリの行列積関数を改造し，一つの行列積演算を双方の間で負荷分散することが考えられる[10]．この手法は CSXL ライブラリにも取り入られており，ユーザに指定された割合の計算をアクセラレータに，残りを CPU に行わせることが可能である．しかしながら，TSUBAME においてはそれだけでは不足である．アクセラレータを持つノードと持たないノードが計算に参加するため，速度差が出てしまうためである．このような環境でも，全てのプロセスの計算速度をそろえるための手法が更に必要となる．

5.2 基 本 方 針

各プロセスの速度をそろえつつ不均一なノードを活用するために，各ノードで起動するプロセス数を変更するという方針を採る．採用するプロセス構成は図 4 の通りであり，**ホストプロセス**と **SIMD プロセス**と呼ばれる二種類のプロセス群が HPL の計算に参加する．両者の違いは，前者は行列積を含む全ての演算を CPU で行い，後者は行列積演算をアクセラレータを用いて行うという点であり，それ以外は同様に動作する．アクセラレータを持たないノードでは，ホストプロセスのみを起動する．我々のねらいは，両者の数を調節することにより，全プロセスが利用する計算性能

を均等に近付けることである．

以上のようなプロセス構成では，各ノードにおいて複数の SIMD プロセスが一枚のアクセラレータを共用することになる．しかし，現在の ClearSpeed アクセラレータは同時に単一のプロセスによってしかアクセスできないという制限を持つ．この問題を解決するために，アクセラレータを直接アクセスするための**SIMD サーバ**を設け，アクセラレータにつき 1 つずつ起動する．SIMD プロセスは行列積を行う際には，SIMD サーバへ演算を依頼する．なお，行列積ごとの行列データのコピーを避けるために，SIMD サーバと SIMD プロセス群は mmap によりメモリを共有し，その上に行列データが配置される．

5.3 CPU とアクセラレータの負荷分散

アクセラレータのないノードでは，(ホストプロセス数)×(GOTO BLAS スレッド数) =(ノードの CPU core 数) となるようにプロセス数を設定する．アクセラレータのあるノードでも同数のホストプロセスを起動し，さらに SIMD プロセスを追加するのが一見良いと思われたが，これでは性能が下がることが分かった．

この理由は，SIMD サーバが CSXL ライブラリを介してアクセラレータと通信を行う際の CPU 利用率が無視できないためである．ホストプロセスとアクセラレータ通信の干渉を無くすためにホストプロセスを 1 つ減らすこととしたが，今度はアイドルな CPU core が生じてしまう．1CPU core をアクセラレータとの通信に専念させるとして，GOTO BLAS スレッド数が 2 のときは 1 core が，4 のときは 3 core がアイドルとなる．これらを使い切るために，SIMD サーバはアクセラレータと余剰 CPU の双方を用いることとした．そのために SIMD サーバは依頼された行列積の領域を適切な比率で分割し，片方をアクセラレータに，もう片方をアイドルな CPU に行わせる．

5.4 Look-ahead の改良

4.1 節で述べたように，HPL は更新計算とパネルブロードキャストをオーバラップさせる最適化を採用しているが，以下に述べるような問題がある．現在の実装では，更新対象部分のうち一部について行列積を行い，その度にパネルブロードキャストメッセージが到着しているかチェックするという処理を繰り返す．パネル通信が終った後は残りの更新対象部分を一度の行列積関数で計算する．この手法により計算と通信のオーバラップが可能になる一方で，行列積計算は細切れに行われることになる．この細切れ化は GOTO BLAS を用いるプロセスではほとんど問題とならないが，図 3 に示したように，CSXL ライブラリでは行列積対象が小さいときに大きく性能が低下してしまうため，望ましくない．

そこで，行列積の細切れ化を防ぎつつオーバラップを可能にするため，以下のような簡単な変更を行った．各プロセスは 1 つ子スレッドを作っておき，更新計算の際の行列積関数呼び出しを子スレッドに行わせる．その間メインスレッドはパネルブロードキャストのための通信を行う．これにより，一度の行列積関数呼び出しにより大きな範囲の更新が可能となり，性能低下を防ぐことができる．

5.5 通信の out-of-order 化

HPL では，行交換通信と次ステップ以降のパネルブロードキャストという，二種類の通信が同時に発生しうる．しかし現在の実装では，一方の通信が開始するとブロックし，もう一方の通信は必ず待たされる．この点を MPI_Isend, MPI_Irecv などのノンブロッキング通信を用いることにより，out-of-order に通信が進むようにした．

6. TSUBAME の性能評価

TSUBAME システム上で，CPU と SIMD アクセラレータの両方を用いた場合の Linpack 性能を報告する．

6.1 評 価 条 件

多くの HPL パラメータについては表 1 に示すものと同様のものを使用した．用いたライブラリは，前と同様に Voltaire MPI と GOTO BLAS である．以降では変更したパラメータや，プロセス配置について述べる．

ブロックサイズ: CPU のみの評価はブロックサイズ B を 240 としたが，これはアクセラレータの CSXL ライブラリで良い行列積性能を得るには小さすぎる．一方，HPL 全体の性能としては，B が大きいほど行列積以外のパネル分解等のコストが相対的に上昇し，性能低下する場合がある．実験では，$B = 960$ とした．

プロセス粒度: 各ホストプロセスが利用する GOTO BLAS のスレッド数を決定するために，以下の評価を行った．アクセラレータ併用し，$B = 960$ の場合で，スレッド数を調節する場合の評価を行った．60 ノード利用で，各プロセスのスレッド数を 2, 4, 8 と変化させた．結果を表 2 に示す．問題サイズ N は同じで 276480 である．この中では，4 スレッドの時が性能が良いため，これを採用する．

2 スレッドの場合に性能がやや落ちているが，これは各プロセスの持つ部分行列が小さいことにより，(1)CSXL ライブラリの行列積性能の低下，(2) ブロッ

#threads	2	4	8
speed	4428	4760	4405

表 2 プロセス粒度を変更した場合の速度 (GFlops)．60 ノード，アクセラレータ 60 枚使用．$N = 276480$.

クサイクリック分割によるプロセス間の仕事量の不均一の拡大，のためと考えられる．逆に 8 スレッドの場合は，HPL 中の行列積以外の，並列化されない部分のコストの影響が大きくなったのではないかと考えられる．

ノード毎のプロセス数: アクセラレータのないノードでは，4 つのホストプロセスを起動し，これにより 16 CPU core を用いる．アクセラレータつきのノードでは，5.3 節で述べたようにホストプロセスを 1 つ減らし，3 つとする．1 CPU core をアクセラレータの通信に専念させ，残りの計算資源であるアクセラレータと 3 CPU core を SIMD プロセスで分け合う．これらの行列積性能の和は 48GFlops 前後であり，ちょうどホストプロセス 3 つ分に相当する．そのため SIMD プロセス数も 3 とした．なお実験においては，各ホストプロセスを，Linux 2.6 の sched_setaffinity システムコールにより適切な数の CPU core へバインドした．

プロセス配置: HPL においてはプロセス格子の構成が性能に影響する．特に今回は不均一なプロセスをどう配置するか考慮する必要がある．予備評価の結果，ホストプロセスと SIMD プロセスをプロセス格子の列 (縦方向) 上で混在させると性能が低下することが分かった．これは HPL においては，プロセス列方向の同期通信が多いため，プロセス間で性能に差が生じるときに同期待ちが大きくなるためと考えられる．そのため行方向に混在させることにした．

648 ノード時のプロセス格子は 36×92 であり，計 3312 プロセスで実験を行った．

6.2 評価準備

予備実験の段階で，アクセラレータを用いた HPL の実行が，実行最後の残差チェックに失敗することがあることが分かった．この残差チェックは，HPL が連立一次方程式 $Ax = b$ を解いたあとに，$||Ax - b||$ が充分に小さいことを確かめるものである．HPL は直接解法を用いているため，データ化けなどが一回でも起こると最後まで影響する．各ノードで独立に HPL を多数回走らせた結果，360 枚中数枚のアクセラレータで，残差チェックの失敗が起こることがわかった．頻度にばらつきはあるが，数十分に一回問題を起こすボードが多かった．

本評価での失敗をさけるために，以下の対処を行った．一回でも残差チェック失敗となったボードは初期不良と見なし，交換を行った．安全率を向上させるため，アクセラレータのクロック周波数を 250MHz から 233MHz に落として以下の評価を行った．なお，この周波数低下のためにアクセラレータ一枚あたりのピーク性能は 89.6GFlops，648 ノード時のシステム全体のピーク性能は 82.125TFlops となっている．

6.3 評価結果

60–648 ノードでの性能評価の結果を表 3 に示す．表中の Full Acc は全ノードがアクセラレータを持ち全て利用する場合，Half Acc は半分のノードにおいてのみアクセラレータを利用した場合である．ただし 648 ノードの Half Acc においては 55%にあたる 360 ノードがアクセラレータを持つ．行列サイズ N として，物理メモリサイズを超えないようななるべく大きい値を設定した．

Half Acc においては多くの場合ノードあたり 72–73GFlops の性能となっており，648 ノードにおいては 47.38TFlops を達成している．これは CPU のみの場合 ($B = 240$) の 38.18TFlops と比べると約 24%の向上となっている．またアクセラレータ一枚あたり 25.5GFlops の向上となっている．4GFlops 強の性能を持つ 1CPU core を通信専用としたので，アクセラレータが寄与した性能は一枚あたり約 30GFlops と見られる．なお，240nodes でノードあたりの性能が低くなっている原因は不明だが，行列サイズまたはプロセス配置のチューニング不足と考えられる．

Full Acc においてはノードあたり 85–87GFlops の性能となっており，350 ノードで 29.86TFlops を達成している．350 ノードで CPU のみを測定した性能は 21.61TFlops となっており，これに対するアクセラレータあたりの向上は 23.6GFlops となっている．

なお仮に TSUBAME の全ノードがアクセラレータを装備した場合，ノードあたり 85GFlops が実現できたとすると，性能は 55.1TFlops となると推測される．

Half Acc の場合に設定した行列サイズ N は，いずれの場合も Full Acc より小さくなっている．これは以下の理由による．各ノードで起動されるプロセス数は，アクセラレータを用いるノードにおいては 6 個，用いないノードでは 4 個である．それにも関わらず各ノードに搭載されている物理メモリは均一の 32GB であるため，N を増やしていく時に，アクセラレータを用いるノードの方が先に物理メモリが一杯になってしまう．このことは，物理メモリサイズを計算性能と比例させたときに，メモリを効率良く利用できることを示している．

		Half Acc	Full Acc
60nodes	speed	4366	5203
	(/node)	(72.8)	(86.7)
	N	345600	391680
240nodes	speed	15640	20510
	(/node)	(65.2)	(85.5)
	N	737280	783360
350nodes	speed	25160	29860
	(/node)	(71.9)	(85.3)
	N	864000	921600
648nodes (*1)	speed	47380	-
	(/node)	(73.1)	-
	N	1148160	-

表 3 アクセラレータ併用した場合の Linpack 性能．表には速度 (GFlops)，ノードあたりの速度 (GFlops)，行列サイズ N を示す．(*1)648 ノードのうち 360 ノードがアクセラレータを持つ

次に，行列サイズと性能の関係を評価する．図 5 は 60 ノードによる評価結果を示し，グラフの横軸は行列サイズ N を，縦軸は速度 (GFlops) を示す．Full Acc, Half Acc は上と同様で，No Acc は CPU のみを利用した場合である．No Acc については $B = 240, B = 960$ の場合を示す．各実験でのプロセス数は，Full Acc で $360(= 15 \times 24)$, Half Acc で $300(= 15 \times 20)$, No Acc で $240(= 15 \times 16)$ となる (括弧内はプロセス格子サイズ)．

Full Acc は $N = 391,680$ において 5203GFlops を，Half Acc は $N = 345,600$ において 4366GFlops を達成している．No Acc(B=240) の最高速度 3800GFlops ($N = 391,680$) に比べ，Full Acc では 37%, Half Acc では 14%の向上となっている．なお，Half Acc の $N = 391,680$ のデータが無いのは，上述したメモリサイズの制限のためである．

全ての場合で行列サイズが大きいほど高速になっているが，その伸び率は異なる．これは CSXL ライブラリの，速度が行列サイズに大きく依存するという特性のためと考えられる．N が 90000 程度のように小さいときはアクセラレータを併用する方が遅くなっている．

なお，60 ノード時の Half Acc の向上率 (14%) よりも上述の 648 ノード時の向上率 (24%) のほうが大きいという，直観に反した結果となっているが，理由は以下のように推測される．Half Acc と No Acc(B=960) の 60 ノード，$N = 345,600$ における速度どうしを比較すると，差は 25.3%であり，これは 648 ノードでも同様の向上率と推測される．60 ノードにおいては，No Acc(B=960) と No Acc(B=240) の差が大きいために，Half Acc と No Acc(B=240) の差は 14%まで小さくなっている．648 ノードで大きい N を用いると，B の差による性能への影響が相対的に小さくなり，結果として Half Acc と No Acc(B=240) の差が大きくなっているのではないかと考えられる．

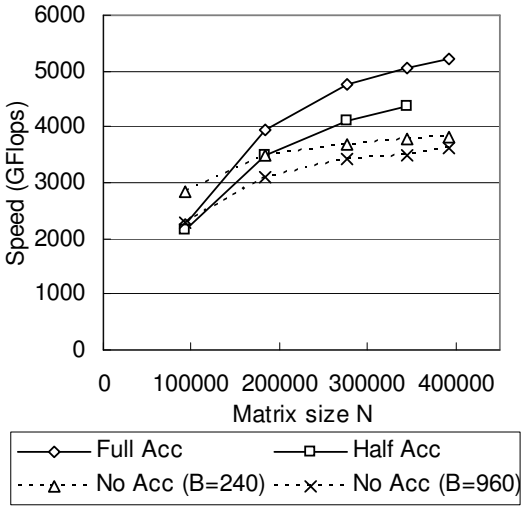

図 5 アクセラレータ併用した場合の 60 ノードでの Linpack 性能．横軸に行列サイズ，縦軸に速度を示す

7. 関連研究

不均一なクラスタ環境において行列演算を効率的に行うアプローチの一つは，各プロセスにノード性能に比例したサイズの部分行列を割り当てることである[3),4)]．このために二次元再帰分割をはじめとする分割法が提案，評価されてきた．しかし既存の多くの並列プログラムは均一的な分割を行っており，それを不均一な分割に変更するにはプログラム全般に渡る変更が必要である．

本研究のアプローチはより笹生ら[9)] のものに近い．その研究では HPL の一部を改変し，高性能ノードにおけるプロセスが他プロセスの整数倍の部分行列を持てるようにした．本研究は SIMD アクセラレータを持つファットノードクラスタで大規模評価を行い，細粒度プロセスが実用的であることを示した点が異なる．

Graphics processing unit(GPU) が高いベクトル処理性能を持つことに注目し，数値計算に利用する研究が広まっている．Galoppo ら[5)] は GPU を用いた LU 分解アルゴリズムを提案・評価している．また，大島ら[10)] は，CPU と GPU の双方を利用するヘテロ環境において，HPL の性能評価を行っている．これらの研究は GPU としては一つのみを用いている一方，本研究では数百枚のアクセラレータと約 10000 の CPU core の併用により効率的に HPL が実行可能であるこ

とを示した．

8. おわりに

汎用 CPU と用途を特化した計算資源を組み合わせるアプローチは，電力性能比を向上させる上で有望と考えられ，TSUBAME や Roadrunner のほかにも文部科学省・理化学研究所が計画する次世代スーパーコンピュータでも導入が検討されている．このようなシステムを有効利用するために，異なる種類の計算資源に別個の (もしくは，関連するが独立性の高い) 計算を割り当てるアプローチも考えられるが，本研究では単独の並列プログラムのために異なる計算資源を併用することに注目した．そして TSUBAME 上の Opteron と ClearSpeed アクセラレータの双方を効率的に利用して HPL を動作させる技法を述べた．

修正した HPL を用いた TSUBAME システム全体の評価により，43.78TFlops が得られた．CPU のみの性能と比べると 24%の速度向上を実現しており，それに伴う電力消費の増加は 1%程度にとどまっている．この結果は 2006 年 11 月の Top500 ランキングでは 9 位にランクされた．他計算機センターのリソース増強のために前回よりランクは下がっているものの，不均一な計算資源を用いた Linpack の性能としては，我々の知る限り現時点で世界最速のものである．

本論文の手法は，ノード間で計算資源に差がある場合でさえも有効であることを示した．また，将来 CSXL ライブラリの性能向上や計算資源の増強があった場合にも容易に対応可能であるし，GPU を数値計算に用いるシステムや将来増加が見込まれるヘテロ型スーパーコンピュータにも適用が可能と考えられる．

本研究では HPL を対象に評価を行ったが，以下で本研究の手法を適用可能なアプリケーションについて議論する．まず，プログラムのカーネル部分がアクセラレータにより充分に高速化可能な計算であり，そのカーネルがライブラリの形で切り分けられることが望ましい．切り分けについては必須条件ではないが，切り分けが困難な場合には細粒度なパイプライン化など，計算に特化した作りこみが必要となると考えられる．また，独立に計算できるカーネル部分の計算量が通信量に比べ充分多いことが，高性能のためには必要である．この点はシステム設計に大きく依存するが，TSUBAME においては PCI-X のバンド幅に注目する必要がある．なお今回の HPL の実験では PCI-X のバンド幅の影響をできるだけ小さくするために，ブロックサイズを大きくすることにした．

今後の課題としては，より一般的な高性能計算のために，アクセラレータ等を含むヘテロ型システムにおける並列計算の性能モデルの構築，より広範囲の並列演算による性能評価，異なる計算資源を利用するスケジューラの構築などが挙げられる．

謝辞本研究にあたってご助言いただいた，GOTO BLAS ライブラリの作者であるテキサス大学 後藤和茂氏に感謝致します．実験にあたって日本電気 (株)，Sun microsystems(株)，ClearSpeed 社，Voltaire 社，東京工業大学学術国際情報センターの皆様に多大なご協力を頂きました．本研究の一部は科学研究費補助金 (特定領域研究 課題番号 18049028, 若手研究 (B) 課題番号 17700050) の援助による．

参考文献

1) ClearSpeed Technology Inc. `http://www.clearspeed.com/`.
2) TOP500 supercomputer sites. `http://www.top500.org/`.
3) Phyllis E. Crandall and Michael J. Quinn. Block data decomposition for data-parallel programming on a heterogeneous workstation network. In *Proceedings of IEEE International Symposium on High Performance Distributed Computing (HPDC)*, pages 42–49, 1993.
4) Toshio Endo, Kenji Kaneda, Kenjiro Taura, and Akinori Yonezawa. High performance LU factorization for non-dedicated clusters. In *Proc. of CCGrid*, 2004.
5) N. Galoppo, N. K. Govindaraju, M. Henson, and D.Manocha. LU-GPU: Efficient algorithms for solving dense linear systems on graphics hardware. In *Proceedings of ACM/IEEE SC05 Conference*, 2005.
6) Kazushige Goto. Goto BLAS. `http://www.tacc.utexas.edu/resources/software/`.
7) John Gustafson. ClearSpeed Technology Inc., private communication.
8) 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/`.
9) 笹生 健, 松岡 聡, and 建部 修見. ヘテロなクラスタ環境における並列 LINPACK アルゴリズム. In **並列処理シンポジウム** *JSPP2002* **論文集**, pages 71–78, 2002.
10) 大島 聡史, 吉瀬 謙二, 片桐 孝洋, and 弓場 敏嗣. CPU と GPU を用いた並列 GEMM 演算の提案と実装. In **先進的計算基盤システムシンポジウム** *SACSIS2006* **論文集**, pages 41–50, 2006.