社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

[招待講演] アクセラレータを用いたヘテロ型スーパーコンピュータ上の並列計算

遠藤　敏夫†

† 東京工業大学 学術国際情報センター　〒 152–8550 東京都目黒区大岡山 2–12–1
E-mail: †endo@gsic.titech.ac.jp

あらまし　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．TSUBAME は Linpack ベンチマークで 38.18TFlops を記録し，この結果により 2006 年 6 月の Top500 ランキングに 7 位としてランクされた．しかしその測定には Opteron のみが使われ，アクセラレータは用いられていない．本論文は汎用 CPU とアクセラレータによる不均一な環境において並列 Linpack を効率的に動作させることを目的とする．均一環境を主な対象とした Linpack の並列実装である HPL に対し，比較的軽微な変更を行うことにより不均一環境へ対応する技法を述べ，さらに最適化技法を提案する．60 ノード 960 CPU core を用いた予備実験では，アクセラレータ 30 枚を加えたときに 14%，60 枚を加えたときに 37%の性能向上が観測された．また，232 ノードとアクセラレータ 232 枚を用いて 18.36TFlops を達成した．
キーワード　高性能計算，ヘテロ型システム，SIMD アクセラレータ，連立一次方程式

[Invited talk] Parallel Computation on Heterogeneous Supercomputer with Accelerators

Toshio ENDO†

† Global Scientific Information and Computing Center, Tokyo Institute of Technology
Oo-okayama 2–12–1, Meguro-ku, Tokyo, 152–8550 Japan
E-mail: †endo@gsic.titech.ac.jp

Abstract　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. The TSUBAME system has achieved 38.18TFlops with Linpack benchmark and is ranked 7th in the Top500 supercomputer ranking in June 2006, even though the measurement is done without any accelerator boards. The purpose of this paper is to run parallel Linpack effectively on heterogeneous environments. This paper describes techniques to run HPL, which is a parallel Linpack implementation designed for homogeneous environment, on heterogeneous environments effectively by making slight changes. Through preliminary experiments with 960 CPU cores on 60 nodes, we observed +14% speed-up when 30 accelerators are added, and +37% with 60 accelerators. And our method achieved 18.36TFlops with 232 nodes and 232 accelerators.
Key words　high performance computing, heterogeneous systems, SIMD accelerators, linear equation solver

1. はじめに

東京工業大学学術国際情報センターは 2006 年 4 月に国内最速のスーパーコンピュータ TSUBAME を導入した．このシステムは東工大キャンパスグリッドの中心として，グランドチャレンジシミュレーションを始めとする科学技術計算に加え，全学ストレージサービスや事務系サーバを含めたホスティングサービスにも活用されることが期待されている．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，科学技術計算におけるスペース性能比，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，国内トップとしてランクされた．この測定に用いられたのは 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た不均一 (ヘテロ) な環境における Linpack ベンチマークの結果を報告する．実験には Linpack の並列実装である High-Performance Linpack(HPL) [8] を用いたが，これは本来均一な環境のために設計されている．不均一な環境に適合させるため，以下の技法を提案する：アクセラレータの有無による起動プロセス数の調整，CPU とアクセラレータ間の負荷分散，HPL の look-ahead 手法の改良，通信の out-of-order 化．TSUBAME の 60 ノードから 232 ノードを用いた実験を通して，これらの技法の効果を示す．

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 (注1) 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．

(注1)：16 ノードのみ，2.6GHz CPU

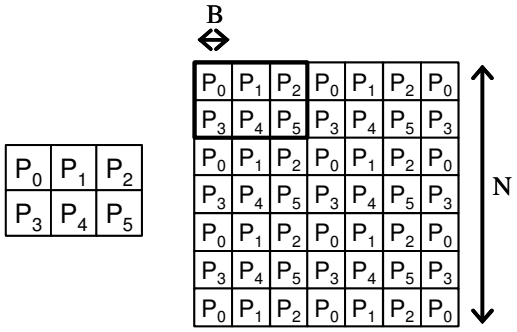

図 2　(左)$P \times Q = 2 \times 3$ のプロセス構成例，(右) 二次元ブロックサイクリック分割

3. HPL

3.1 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数値演算ライブラリを用いることにより，性能を上げることができる．

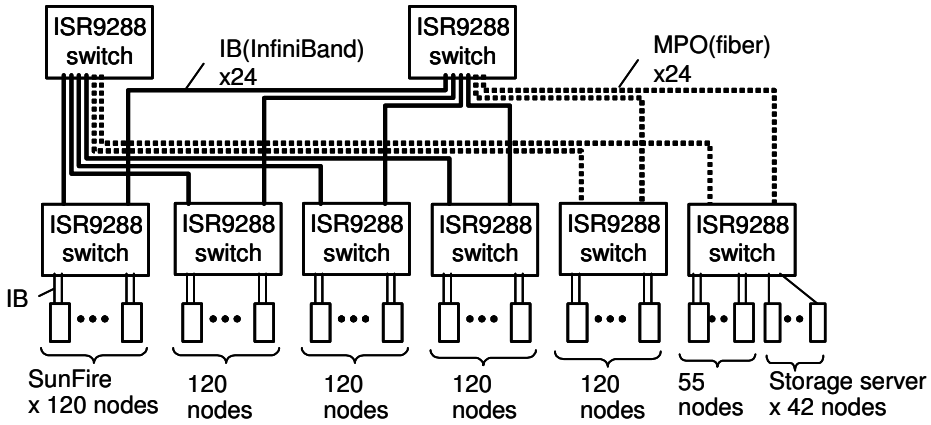

図 1　TSUBAME のシステム構成図

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

3.2　10,368CPU を用いた性能評価

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(注2)．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 [5] 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．パネルブロードキャストのアルゴリズムは，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) である．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．この評価では次のステップ (depth=1) のブロードキャストとオーバラップさせることとした．

行列サイズ $1,334,160$，プロセス数 $36 \times 144 = 5184$ の実験において，実行時間約 11.5 時間，速度 38.18TFlops が得られた．システムの Opteron の合計理論ピーク性能 49.87TFlops に対する比率は 76.6%となる．

(注2)：この評価は Sun microsystems 橋爪信明氏を中心に行われた

3.3　不均一環境における課題

前節では TSUBAME の汎用 CPU を用いたが，これに加えて SIMD アクセラレータも併用することにより，さらなる性能向上が期待される．しかしながら，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．つまり，均一な環境を対象として設計された並列プログラムである．我々のねらいはこの HPL に大きな変更を加えずに，計算速度・特性が異なる 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 の性能に大きく影響するのは行列積の速度であり，特に $B \ll M$ とするとき，$M \times B$ 行列と $B \times M$ 行列の積を多用する．GOTO BLAS の行列積の性能は，$B = 240 \sim 960$, $M \geqq 1920$ として TSUBAME 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．一方，ClearSpeed により提供される CSXL ライブラリの行列積の性能を図 3 に示す．性能は行列サイズに大きく依存するが，このグラフの範囲では 18GFlops($M = 1920, B = 384$) から 38GFlops($M = 11520, B = 960$) となることが分かる．

以上のように，CPU とアクセラレータは計算速度も特性も異なる．次節では，これらの計算資源をプロセス間で分割し，各プロセスの負荷がほぼ均一となるような技法を述べる．

なお，現在の CSXL ライブラリの性能は理論値 96GFlops の 40%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，現在のライブラリの計算スケジューリングと通信スケジューリングの問題であり，将来の版で改善されることが期待される [6].

4.　提 案 手 法

TSUBAME システムにおいては，汎用 CPU とアクセラレータが混在するという不均一性に加え，アクセラレータを持つノード (360 台) と持たないノード (295 台) が混在する．このような環境で，均一環境向けに設計されている HPL を，大きく変更せずに効率的に動作させるため，以下の技法を提案する．

4.1　プロセス構成

提案するプロセス構成を図 4 に示す．これまでと同様に GOTO BLAS を用いるプロセスを**ホストプロセス**と呼ぶ．それに加えてアクセラレータを用いるプロセス (**SIMD プロセス**

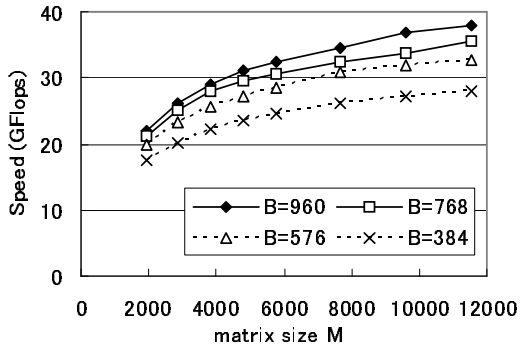

図 3　CSXL ライブラリによるサイズ $(M \times B) \times (B \times M)$ の行列積性能

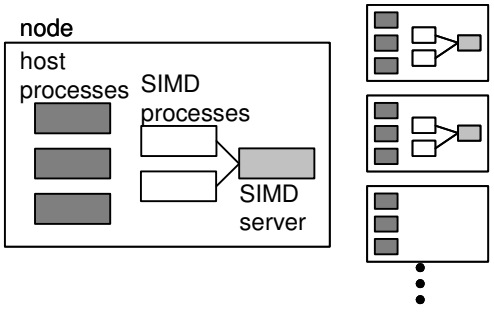

図 4　CPU とアクセラレータを利用するためのプロセス構成

) を導入する．SIMD プロセスは，行列積以外の部分についてはホストプロセスと同様に動作するが，行列積の部分についてのみ，アクセラレータに依頼する．HPL の演算量のほとんどは行列積で占められるため，SIMD プロセスによる CPU 利用率は非常に低い．このようなホストプロセスと SIMD プロセスの双方が並列プログラム実行に参加する．両者の数を調節することにより，各プロセスの計算能力を均等に近付けることにより，全体の性能を向上させることがねらいである．

ただし，現在の CSXL ライブラリ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．この制限下で SIMD プロセスを複数走らせるために，CSXL ライブラリを直接アクセスするための **SIMD サーバ**を設け，アクセラレータを持つ各ノードに 1 つずつ起動する．SIMD プロセスは直接行列積を行う代わりに，SIMD サーバへ行列積計算を依頼する．なお，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．

4.2　CPU とアクセラレータの負荷分散

各ノードのプロセス数を決定する際に，(ホストプロセス数)×(GOTO BLAS スレッド数) =(ノードの CPU core 数) とするのが一見良いと思われるが，これでは性能が下がることが分かった．これは SIMD サーバが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．余裕を持たせる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．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．これらを使うために，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 に行わせることとした．

4.3　Look-ahead の改良

3.2 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ぐためにスレッド化を行い，更新計算の行列積関数呼び出しを，通信とは別スレッドで行うこととする．

4.4　通信の out-of-order 化

HPL では，行交換通信とパネルブロードキャストが同時に発生しうるが，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て，通信を out-

of-order に行うことを可能とした．

5. 不均一環境での性能評価

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．

5.1 評価条件

TSUBAME システムのうち，ClearSpeed アクセラレータを持つ一部のノードを実験に用いた．HPL パラメータは，明示的に示すもの以外は表 1 に示すものを用いた．

プロセス構成は 4.1 節で述べた通りとする．アクセラレータを用いる各ノードに，3 つのホストプロセスを起動し，それぞれの GOTO BLAS のスレッド数を 4 とした．また，プロセス間の性能をなるべくそろえるよう，SIMD プロセスの数を 3 とした．そして SIMD サーバは，4.2 節で述べたように，アクセラレータと 3 つの CPU core を行列積に用いる．アクセラレータを用いないノードには，ホストプロセスのみを 4 つずつ起動した．

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．ブロックサイズ B としては，試行したうちで最も性能の良いものを選択した．アクセラレータを併用する場合には $B = 960$, CPU のみの場合には $B = 240$ である．

5.2 評価結果

図 5 に 60 ノードの実験結果を示す．グラフの横軸は行列サイズ N を，縦軸は速度 (GFlops) を示す．グラフ中の Full Acc はアクセラレータを全て併用した場合，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，No Acc は汎用 CPU のみを利用した場合である．各実験での合計プロセス数は，Full Acc で 360(= 15 × 24), Half Acc で 300(= 15 × 20), No Acc で 240(= 15 × 16) となる (括弧内はプロセス格子サイズ)．

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 ライブラリは GOTO BLAS よりも，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．アクセラレータを併用する場合には，行列サイズが約 18 万以上のときに CPU のみの場合よりも効果が得られていることが判る．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 の最高速度 3800GFlops ($N = 391,680$) と比較すると，Full Acc では 37%, Half Acc では 14%の向上が得られた．

Half Acc の $N = 391,680$ のデータが無いのは，メモリサイズの制限のためである．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メモリサイズは均一の 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において先に物理メモリが一杯になってしまう．Half Acc の向上率 14%が，Full Acc の向上率 37%の半分未満となっていることの主な理由はこの点と考えられる．

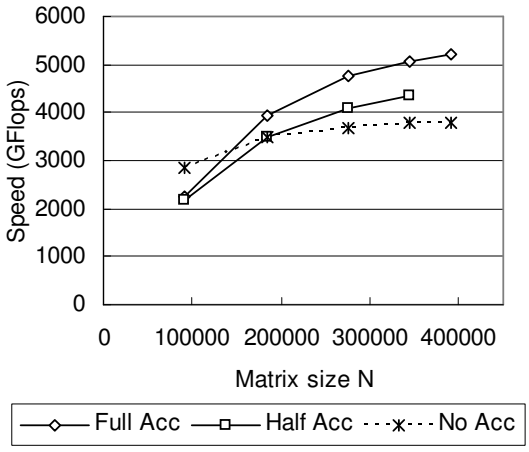

図 5 アクセラレータ併用した場合の 60 ノードでの Linpack 性能

		Half Acc	Full Acc
60nodes	speed	4366	5203
	(/node)	(72.8)	(86.7)
	#nodes	60	60
	N	345600	391680
116–120nodes	speed	8078	9659
	(/node)	(69.6)	(80.5)
	#nodes	116	120
	N	460800	483840
232nodes	speed	-	18360
	(/node)	-	(79.1)
	#nodes	-	232
	N	-	576400

表 2 60–232 ノードでの性能．表には速度 (GFlops)，ノードあたりの速度 (GFlops)，ノード数，行列サイズ N を示す．

より多数のノードの結果を表 2 に示す．ただし実験期間の制限により，未測定のケースや，Full Acc と Half Acc でノード数が異なるケースを含む．Full Acc の場合，232 ノードで 18.36TFlops，ノードあたりの速度は 79.1GFlops/node を達成している．60 ノードの No Acc(63.3GFlops/node) を基準にすると，今回の実験の範囲では，Half Acc では 10–14%，Full Acc では 25–37%の向上が見られる．

5.3 プロセス粒度の影響

これまでの実験では各ホストプロセスは 4 スレッド (=4 CPU core) を用いていた．このスレッド数を 2, 4, 8 と変更した場合の性能を表 3 に示す．60 ノードで，アクセラレータを全て用いた場合である．予備実験により，ホストプロセスと SIMD プロセスの速度が均等に近くなるよう，各ノードのプロセス数は以下のようにした．

- 2 スレッド時: 6 ホストプロセス + 5 SIMD プロセス
- 4 スレッド時: 3 ホストプロセス + 3 SIMD プロセス
- 8 スレッド時: 1 ホストプロセス + 2 SIMD プロセス

また，SIMD サーバはアイドルな CPU core を使うよう調整されている．この実験では，4 スレッドの場合が最も性能が良かった．1 プロセス 2 スレッドの場合は，各プロセスの部分行

#threads	2	4	8
speed	4428	4760	4405

表 3　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．$N = 276480$.

列が小さくなり，CSXL ライブラリの性能が下がったと考えられる．逆に，1 プロセス 8 スレッドの場合は，HPL 中の行列積以外の，並列化されていない部分の影響が大きくなったのではないかと考えられる．

6. 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った点，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．

岸本らは各 CPU に異なる数の HPL プロセスを割り当てた場合の性能モデルを提案し，少ない数の実験試行から，ふさわしいプロセス数を推定する手法を提案した [7]．このモデルにおいては，本研究で指摘したようなプログラムの実装上の問題点については触れていない．

7. お わ り に

TSUBAME が採用している，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，京速計算機や IBM の Roadrunner などの次世代スーパーコンピュータでも採用される見込みである．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．そしてテストケースとして HPL を採用し，TSUBAME 上の Opteron と ClearSpeed アクセラレータの双方を効率的に利用する技法を提案した．

提案した技法は，アクセラレータの有無による起動プロセス数の調整，CPU とアクセラレータ間の負荷分散，HPL の look-ahead 手法の改良，通信の out-of-order 化などである．これらにより，均一な環境を主な対象とする HPL に対し，比較的小さな変更で良好な性能向上が得られることを示した．本論分の手法は，全ノードがアクセラレータを持たない場合でさえも有効である．

なお，今回の実験で用いた CSXL ライブラリの性能は充分とは言えないが，将来の版で性能向上が期待されている．その場合にも，本論文の手法はプロセス数調整により容易に適応可能である．また，今回はシステムの一部を用いた実験を行ったが，今後システム全体の評価を行っていきたい．

本研究では HPL を対象に研究を行ったが，提案手法が効果的に適用可能な条件は，以下のように考えられる: (1) 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ること，(2) 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．HPL においては，CPU とアクセラレータ間の通信量 $O(n^2)$ に対して計算量は $O(n^2B)$ であり，ブロックサイズ B が充分に大きければ条件 (2) を満たすと言える．今後の課題の一つとして，アクセラレータを併用するヘテロ型システム上で，より一般的な高性能計算のための実行モデルの構築が挙げられる．

謝辞　本研究は東京工業大学/NII 松岡聡氏，Sun microsystems(株) 橋爪信明氏，日本電気 (株) 長坂真路氏，テキサス大学 後藤和茂氏との共同研究である．本研究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に多大なご協力をいただいた．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．

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] Kazushige Goto. Goto BLAS. `http://www.tacc.utexas.edu/resources/software/`.

[6] John Gustafson. ClearSpeed Technology Inc., private communication.

[7] Yoshinori Kishimoto and Shuichi Ichikawa. An execution-time estimation model for heterogeneous clusters. In *Proceedings of IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, 2004.

[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.