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] 後藤和茂[†5]

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 Linpack 性能を得る上での課題について論じ，解決法を述べる．16 ノード 256 CPU core を用いた予備実験では，アクセラレータ 8 枚を加えたときに 8.2 %，16 枚を加えたときに 19%の性能向上が観測された．

Performance Evaluation of TSUBAME Heterogeneous Supercomputer with Linpack

Toshio Endo,[†1] Satoshi Matsuoka,[†1,†2] Nobuaki Hashizume,[†3]
Masamichi Nagasaka[†4] and Kazushige Goto [†5]

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. This paper discusses issues to obtain high Linpack performance on heterogeneous systems with general purpose processors and accelerators, and describes solutions. Through preliminary experiments with 256 CPU cores on sixteen nodes, we observed +8.2% speed-up when eight accelerators are added, and +19% with sixteen accelerators.

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一方で，ClearSpeed 社の SIMD アクセラレータボードを備えることにより，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．

Linpack ベンチマークは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，世界ランキングである Top500[2] でも用いられている．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

†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.
†5 テキサス大学オースティン校
The University of Texas at Austin

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，HPL の look-ahead 手法の改良，MPI 実装の spin wait の変更．TSUBAME の 16 ノード 256 CPU core を用いた実験を通して，アクセラレータ併用による性能向上のためにはこれらの技法が必須であることを示す．

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 10Gbps のバンド幅の 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 InfiniBand により 288 ポートの Voltaire ISR9288 スイッチに接続される．スイッチ間は，InifiniBand 24 本により接続される☆☆．

アプリケーションプログラムは，Message passing interface(MPI) の一実装である Voltaire MPI により並列プログラムを動作させることができる．Voltaire MPI は InifiniBand を直接アクセスすることにより高性能を実現する．

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．

☆ 16 ノードのみ，2.6GHz CPU

☆☆ 部屋をまたがって設置されている部分については光ファイバーを利用する

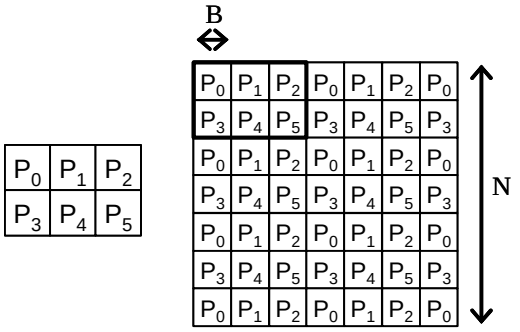

図 2 (左)$P \times Q = 2 \times 3$ のプロセス構成例，(右) 二次元ブロックサイクリック分割

アクセラレータを利用するために，SIMD 並列プログラミング言語 C^n と，BLAS(基本線形演算) ライブラリが提供されている．本論文ではこれらのうち BLAS ライブラリを利用する．

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

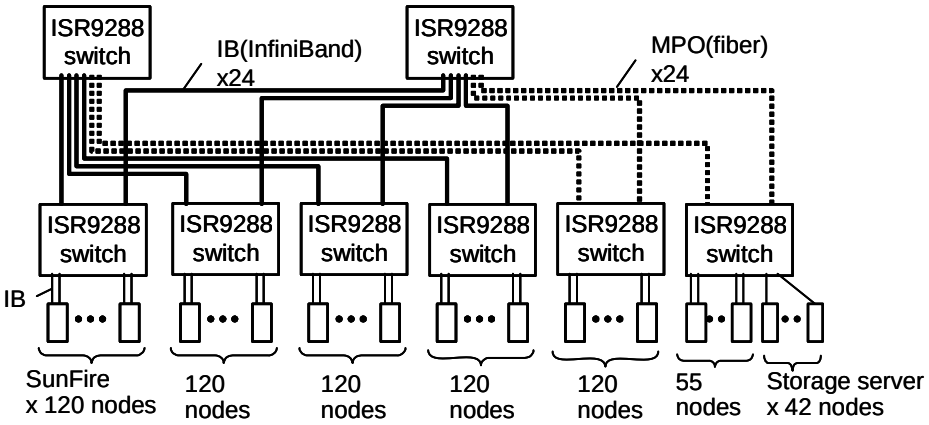

図 1 TSUBAME のシステム構成図

$O(N^2(P+Q))$，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．

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．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．本実行では 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．

実験に用いた 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) とした．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．実験では次のステップ (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 を達成した．システムの Opteron の合計理論性能は 49.87TFlops のため，実効性能は 76.6%となる．

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

3.3 不均一環境における問題点

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，計算量もほぼ均一となる．そのため，異なる速度を持つ汎用 CPU と SIMD アクセラレータを有効利用するための手法が必要となる．

HPL の性能に大きく影響するのは行列積の速度であり，M を $B \leq M \leq N$ とするとき，$M \times B$ 行列と $B \times M$ 行列の積を特に多用する．GOTO BLAS の場合 (2 スレッド利用)，本論文の実験で用いるサイズの行列について，行列積性能は安定的に 8.4 ～ 8.8GFlops である．一方，ClearSpeed により提供される BLAS ライブラリ (以下 CS BLAS) の性能を図 3 に示す．なお，SIMD アクセラレータの PE 数から M, B を 192 の倍数としている．性能は行列サイズによって大きく変動し，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．不均一環境では，GOTO BLAS と CS BLAS の双方を有効利用するために，各プロセスの計

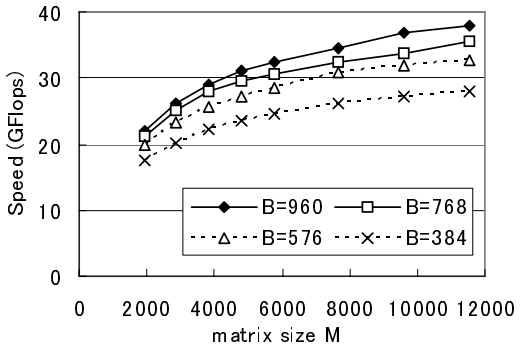

図 3 ClearSpeed BLAS によるサイズ $(M \times B) \times (B \times M)$ の行列積性能

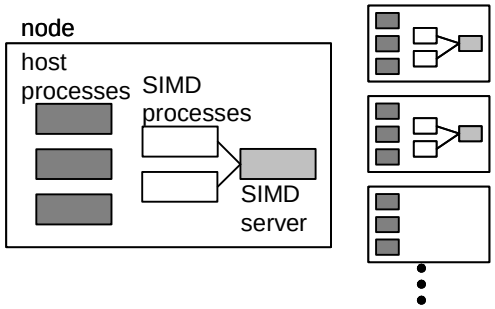

図 4 CPU とアクセラレータを利用するためのプロセス構成

算性能をなるべく均等化する必要がある．

なお，図 3 の性能は理論値 96GFlops に比べると効率 40%以下である．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，現在は半数強のノードのみがアクセラレータを持つ．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

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．それに加えて図 4 のように，CS BLAS を用いるプロセス (**SIMD プロセス**) を導入する．SIMD プロセスは，更新演算時の行列積以外はホストプロセスと同様に動作し，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を目的とする．

ただし，現在の CS BLAS 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．これではプロセス数調整の際の制限が厳しいため，CS BLAS 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．複数の SIMD プロセスは，mmap による共有メモリ上を通して，SIMD サーバへ行列積計算を依頼する．

4.2 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通信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，その度にパネル通信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．このアルゴリズムでは，パネル通信以前の行列積が細切れに行われることになる．図 3 に示したように，CS BLAS 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．

4.3 MPI の spin wait の変更

本論文の実験に用いている Voltaire MPI の現在の実装には，MPI_Send, MPI_Recv などのブロック通信が CPU を消費し続けるという問題がある．これは MPI 関数の内部で spin wait をしているためと考えられる．本章で述べる手法では CPU 数を越える数のスレッドを起動するため，CPU の浪費は望ましくない．

回避策として，ブロック通信関数の代わりに MPI_Isend などのノンブロック通信関数を usleep でスリープしながら断続的に呼び出すように置き換えた．これにより CPU の浪費は抑えられるが，本来よりもメッセージの到着に気づくのが遅れる問題が起こる．将来の Voltaire MPI の改良によりこの回避策は必要なくなるだろう．

5. 不均一環境での性能評価

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．TSUBAME システムのうち，ClearSpeed アクセラレータを持つ 16 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．

図 5 に結果を示す．グラフ中の SIMD16 はアクセラレータを全て併用した場合，SIMD8 は 16 枚のう

ち 8 枚のみを併用した場合，CPU は汎用 CPU のみを利用した場合である．

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，7 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 2 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．なお，ホストプロセス数を 8 としなかったのは，SIMD サーバの CPU 利用率が無視できず，そのための余裕が必要なためである．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 8 つずつ起動した．結局，合計プロセス数は SIMD16 で 160($=8\times20$), SIMD8 で 144($=8\times18$), CPU で 128($=8\times16$) となる (括弧内はプロセス格子サイズ)．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 CPU へバインドした．4.2，4.3 節で述べた技法を，SIMD16 と SIMD8 に適用した．ブロックサイズ B は，試行したうちで最も性能の良いものを選択した．SIMD16 と SIMD8 で $B=576$, CPU で $B=240$ である．

グラフから，アクセラレータを併用することにより，サイズが約 90,000 以上の行列において性能向上が得られていることが判る．$N=184,320$ において，SIMD16 は 1169GFlops, SIMD8 は 1062GFlops を達成している．CPU のみの 981.2GFlops と比較すると，SIMD16 では 19%, SIMD8 では 8.2%の向上が得られた☆．

なお，図 3 によると B が 576 より大きい方が CS BLAS の行列積性能が良いにも関わらず，HPL の性能は下がることが判っている．この原因については調査中であるが，更新計算以外によるボトルネックが B に伴い大きくなるためと推測している．

次に 4.2，4.3 節で述べた技法の影響を図 6 に示す．グラフは全てアクセラレータ 16 枚併用の場合であり，二つの提案技法を用いる場合 (Opt)，用いない場合 (NoOpt) を示す．また，$B=576$, $B=960$ の場合を示す．NoOpt に比べると Opt の性能は高く，提案技法の効果は観測された．特に NoOpt(B=576) の性能の頭打ちが顕著であるが，これは元々の HPL が行列積の細切れ化を行った場合に，その悪影響は B が小さい場合に強く及ぶためと考えられる．

これまでの実験ではノードあたり 7 個のホストプロセス，3 個の SIMD プロセスという比較的細粒度なプロセスを用いた．プロセス粒度を変更した場合の性能を図 7 に示す．Fine はこれまでと同じ設定の場合で，Coarse ではノードあたり 2 個のホストプロセス (GOTO BLAS スレッド数は 7)，1 個の SIMD プロセスを用いた．このグラフにおいては，プロセス粒度はあまり影響しないことが判る．特に，Fine よりも Coarse の方が各プロセスの持つ部分行列が大きくなり，行列積性能は有利と予想されたが，この実験においてはその影響は見られていない．一方，将来 ClearSpeed BLAS の性能が向上した場合には，プロセス数を細かく調整できる Fine の方が有利になると考えられる．

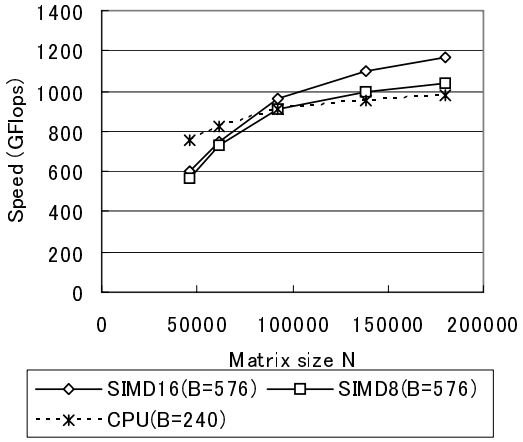

図 5 アクセラレータ併用した場合の 16 ノードの Linpack 性能

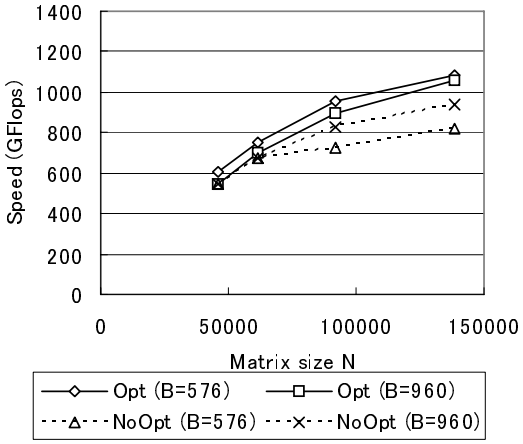

図 6 提案技法の影響の評価．アクセラレータ 16 枚使用時

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

☆ CPU のみでは N, B のさらなるチューニングにより，16 ノードで 1042GFlops，1 ノードで 67.50GFlops を達成している

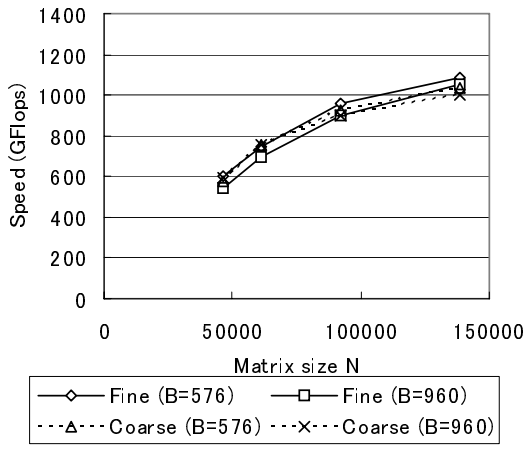

図 7 プロセス粒度を変更した場合の性能．アクセラレータ 16 枚使用時

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

Linpack ベンチマークの実装の一つである HPL を基に改変したプログラムを用い，汎用 CPU である Opteron と ClearSpeed SIMD アクセラレータを備える TSUBAME システムの評価を行った．アクセラレータ併用による性能向上を得るために，パネル通信の lookahead 機構の改変など，HPL への部分的な変更が必要であることを示した．今回の実験ではアクセラレータ側の演算ライブラリとして ClearSpeed BLAS を用いたが，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 16 ノードという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．

今回行った変更は HPL 特有のものであるが，複数プロセスでアクセラレータを共有する手法や，通信と計算を非同期に行うアプローチは，広範囲のアプリケーションに適用可能と考えられる．今後 TSUBAME 上で，アクセラレータを併用する，より一般的な高性能計算を実現するための実行モデルを研究する予定である．

謝辞 本研究にあたって ClearSpeed 社の Chris Piercy 氏，John Gustafson 氏，Sun microsystems 社の Antoine Petitet 氏に協力，助言頂いた．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．

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) Kazushige Goto. High-performance BLAS. `http://www.tacc.utexas.edu/~kgoto/`.
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.