2008－HPC－116 (10)
2008／8／5

T2K筑波システムにおけるLinpack性能評価

高橋大介[†] 後藤和茂[††] 朴泰祐[†]
建部修見[†] 佐藤三久[†] 三上和徳[†††]

T2K筑波システムは，648ノード，10,368コアからなる大規模PCクラスタである．本論文では，T2K筑波システムの概要を述べるとともに，Linpack性能評価を行った結果について報告する．T2K筑波システムの625ノード，10,000コアを用いてHPL（High Performance Linpack Benchmark）を実行した．その結果，$N = 1,508,000$に対して76.46 TFlops（実行効率：83.1%）の性能が得られた．

Performance Evaluation of Linpack on T2K-Tsukuba System

Daisuke Takahashi,[†] Kazushige Goto,[††] Taisuke Boku,[†]
Osamu Tatebe,[†] Mitsuhisa Sato[†] and Kazunori Mikami[†††]

The T2K-Tsukuba system is a large-scale PC cluster, which consists of 10,368 cores on 648 nodes. In this paper, we describe an overview of the T2K-Tsukuba system, and report the result of Linpack benchmark on the T2K-Tsukuba system. We performed the HPL (High-Performance Linpack Benckmark) on the T2K-Tsukuba system with 10,000 cores on 625 nodes. Its result was 76.46 TFlops with $N = 1,508,000$, achieving the efficiency of 83.1%.

1. はじめに

T2Kオープンスパコン[1]は，筑波大学，東京大学，京都大学の3大学で基本仕様を共通にしたスーパーコンピュータシステムである．

筑波大学計算科学研究センターでは，2008年6月よりT2K筑波システムの稼働を開始した．T2K筑波システムは，648ノード，10,368コアからなる理論ピーク性能95.39TFlopsの大規模PCクラスタである．

スーパーコンピュータの性能評価に際しては，さまざまなベンチマークプログラムがあるが，Linpackベンチマークは，密行列の連立一次方程式を解く際の浮動小数点演算性能を測定するベンチマークとして現在広く用いられている．世界のスーパーコンピュータTOP500[2]のランキングにも，このベンチマークが採用されている．

本論文では，T2K筑波システムの概要を述べるとともに，Linpack性能評価を行った結果について報告する．

† 筑波大学大学院システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba
†† テキサス州立大学
The University of Texas at Austin
††† クレイ・ジャパン・インク
Cray Japan, Inc.

図1 T2K筑波システムの全景（648ノード，74ラック）

以下，2章でT2K筑波システムについて，3章でHPLについて，4章で性能評価の結果について述べる．最後の5章はまとめである．

2. T2K筑波システム

2.1 T2K筑波システムの概要

T2K筑波システムは，「T2Kオープンスパコン仕様」に基づいた，Appro Xtreme-X3 Serverが648ノード，10,368コアからなる大規模PCクラスタである．ノード間はmulti-railのInfiniBandを用いたFat Tree相互結合網で接続されている．T2K筑波システムの諸元を**表1**に示す．

また**図1**に，全74ラックからなるT2K筑波システムの全景を示す．

表 1　T2K 筑波システムの諸元

ノード台数	648
理論ピーク性能	95.39 TFlops
ノード構成	4 ソケット/ノード
CPU	Quad-Core AMD Opteron 8356 (Barcelona, 2.3 GHz)
L1 キャッシュ	64 KB×4 (命令) +64 KB×4 (データ) (各コア独立)
L2 キャッシュ	512 KB×4 (各コア独立)
L3 キャッシュ	2 MB (コア間共有)
ノード当たりのメモリ容量	32 GB (DDR2 667 MHz)
総メモリ容量	20 TB
ノード当たりの最大メモリバンド幅	42.7 GB/s
ローカルディスク容量	250 GB×4 (SATA-II, RAID-1)
ファイルサーバディスク容量	800 TB (RAID-6)
ネットワークインタフェース	DDR InfiniBand Mellanox ConnectX HCA×4
ネットワークトポロジ	Fat Tree (full-bisection bandwidth)
最大リンクバンド幅	8 GB/s
OS	Red Hat Enterprise Linux version 4 WS (Linux kernel 2.6)
メッセージ通信ライブラリ	MVAPICH (Appro による修正版)
システム規模	総ラック数：74 (ノード，スイッチ他：69 ラック，ファイルサーバ：5 ラック) 総電源容量：745kVA

2.2 計算ノードの構成

T2K 筑波システムの計算ノードのブロックダイアグラムを**図 2** に示す．各ノードには Quad-Core AMD Opteron 8356 (Barcelona, 2.3 GHz) が 4 ソケット搭載されており，16 コアが 32 GB のメモリを共有する．ノード当たりの理論ピーク演算性能は 147.2 GFlops である．

Opteron の NUMA アーキテクチャにより，各プロセッサのメモリバンド幅を効率的に利用することができる．ノード当たりの理論ピークメモリバンド幅は 42.7 GB/s である．各計算ノードは 4 本の InfiniBand 4X DDR，および 2 本の Gigabit Ethernet のインターフェースを持つ．

2.3 インターコネクションネットワーク

T2K 筑波システムでは，4 rail/node の InfiniBand ConnectX リンクが Full-bisection バンド幅の Fat Tree 網によって結合されている．Full-bisection バンド幅は，8 GB/s × 648 = 5.18TB/s である．

ノードからのリンク間通信に制約がないため，4 本のリンクの柔軟な利用が可能になっている．Fat Tree Network は全て 24 port の InfiniBand スイッチ (Flextronix 社製) で構成されており，3 段の Fat Tree となっている．総スイッチ数は 616 台で，総ケーブル数は 8,554 本 (うち 2/3 が電気ケーブル，1/3 が光ケーブル) である．

2.4 共有ファイルシステム

T2K 筑波システムでは，並列処理インターコネクションの Infiniband をストレージネットワークとしても併用している．1 TB (一部 750 GB) HDD を 1,200 台用いて，RAID-6 の冗長性ファイルシステムを構築し，これをベースに Lustre 分散ファイルシステムを構築している．なお，ユーザ利用可能領域は 800 TB であり，ノードのローカルファイルディスクを束ねた Gfarm ファイルシステム[3] の容量は約 600 TB である．

2.5 外部ネットワーク接続

T2K 筑波システムは，SINET3 およびつくば WAN[4] に各々10Gbps で接続されている．このネットワーク接続は T2K 連携および他機関との連携に利用される．

3. HPL

3.1 HPL の概要

High-Performance Linpack Benchmark (以下 HPL)[5] は C 言語で記述された Linpack ベンチマークの分散メモリ型並列計算機用の実装の一つであり，現在広く用いられている．HPL のコンパイルおよび実行には MPI 1.1 準拠のメッセージ通信ライブラリが必要である．また，行列演算ライブラリとして BLAS (Basic Linear Algebra Subprograms)[6] または VSIPL (Vector Signal Image Processing Library)[7] のいずれかが必要である．

HPL においては問題サイズ N や，ブロックサイズ NB，通信パターン等のパラメータを `HPL.dat` というファイルに記述することにより，実行時に様々なパラメータを変更することが可能である．

HPL では，プロセスは $P \times Q$ の二次元のプロセスグリッドに並べられる．係数行列は $P \times Q$ のプロセスグリッドに二次元ブロックサイクリック分割で各ノードに分散して割り当てられ，Right-looking 型の LU 分解アルゴリズムが実行される．

HPL の問題サイズを N とした場合，演算量は $O(N^3)$ となり，通信量は $O(N^2)$ となるので，N が大きくなるに従い計算速度は向上する．

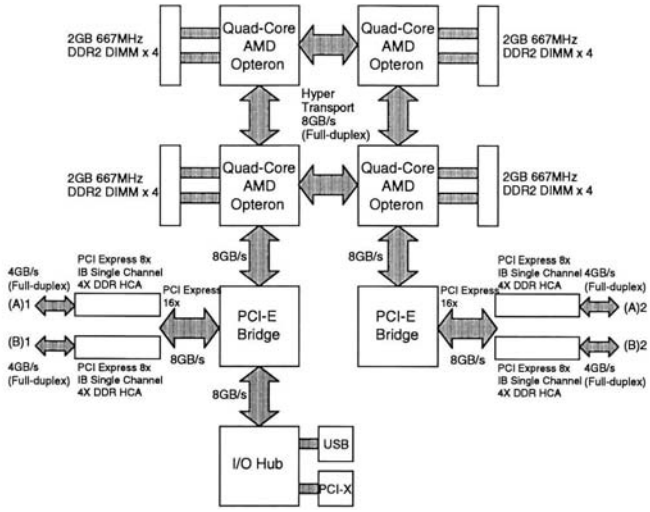

図 2 計算ノードのブロックダイアグラム

3.2 プロセスマッピング

HPL においては，プロセスグリッドの構成が性能に影響することが知られている[8]．T2K 筑波システムの各ノードは，Quad-Core AMD Opteron が 4 ソケットで構成されていることから，HPL を実行する際には大きく分けて以下の 3 種類の実行方法が考えられる．

(1) 1 コアに対して 1 MPI プロセスを用いる（flat MPI，1 ノード 16 MPI プロセス，1 スレッド /MPI で実行）

(2) 1 ソケットに対して 1 MPI プロセスを用いる（MPI+OpenMP，1 ノード 4 MPI プロセス，4 スレッド /MPI で実行）

(3) 1 ノードに対して 1 MPI プロセスを用いる（MPI+OpenMP，1 ノード 1 MPI プロセス，16 スレッド /MPI で実行）

T2K 筑波システムの 64 ノード，1,024 コアを用いて HPL を上記 (1)~(3) の各方法で実行して性能を比較した結果を**図 3** に示す．図 3 より，問題サイズ N が 120,000 以上の場合は (1) の実行方法が，N が 100,000 以下の場合には (2) の実行方法が最も高速であることが分かる．

しかし，(1) と (2) の性能にあまり大きな差はないため，全システムサイズが 10,000 コア程度と非常に大規模であることを考慮し，MPI プロセス数を 1/4 に抑えられる (2) の実行方法を採用した．

また，プロセスマッピングは Row-major とし，行交換で通信が多い方向にノード内 MPI プロセスを割り当てることにした．$P \times Q = 8 \times 4$ とし，1 ソケットに対して 1 MPI プロセスを用いた場合のプロセスグリッドの例を**図 4** に示す．図 4 において，$P_0 \sim P_3$ に網掛けされている部分が，1 ノードに相当する．

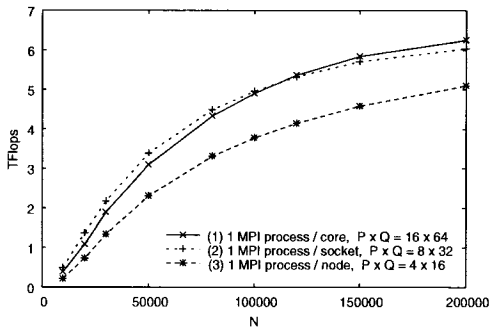

図 3 64 ノード，1,024 コアを用いた場合の HPL 実行方法の比較（GotoBLAS-1.26，MVAPICH0.9.9，NB=232）

P_0	P_8	P_{16}	P_{24}
P_1	P_9	P_{17}	P_{25}
P_2	P_{10}	P_{18}	P_{26}
P_3	P_{11}	P_{19}	P_{27}
P_4	P_{12}	P_{20}	P_{28}
P_5	P_{13}	P_{21}	P_{29}
P_6	P_{14}	P_{22}	P_{30}
P_7	P_{15}	P_{23}	P_{31}

図 4 プロセスグリッドの例（$P \times Q = 8 \times 4$，1 ノード 4 MPI プロセス，4 スレッド /MPI で実行する場合）

4. 性能評価

4.1 性能評価に用いたプログラムおよびライブラリ

Linpack 性能評価のプログラムとしては，オリジナ

```
Column=  7656 Fraction=0.005 Mflops=77416410.86 Wtime=  447.50
Column= 15312 Fraction=0.010 Mflops=78102108.51 Wtime=  882.64
Column= 22736 Fraction=0.015 Mflops=78317580.98 Wtime= 1300.54
...
Column=1198976 Fraction=0.795 Mflops=76802478.63 Wtime=29511.01
Column=1349776 Fraction=0.895 Mflops=76599768.07 Wtime=29811.47
Column=1500576 Fraction=0.995 Mflops=76467233.79 Wtime=29897.67
```

図 5 Intel(R) Optimized MP LINPACK Benchmark for Clusters でコンパイル時に「ASYOUGO」を指定した場合の HPL の出力結果の一部（625 ノード，10,000 コアを使用，$N = 1,508,000$）

ルの HPL 1.0a[5] をベースに Intel が修正，追加した Intel(R) Optimized MP LINPACK Benchmark for Clusters[9] を用いた．

このプログラムではいくつかの拡張が行われているが，コンパイル時に「ASYOUGO」を指定すると，実行が進行するとともに，その都度 Mflops 値と途中経過時間が出力されるという機能が提供されている．この機能を用いることで，最初の数%だけ行の消去を行った場合の性能を確認することができ，HPL パラメータのチューニングを効果的に行うことができる．

また，HPL の実行には数時間を必要とする場合が多いので，この機能により HPL の実行が正常に行われているかどうかを監視することができる．コンパイル時に「ASYOUGO」を指定した場合の HPL の出力結果の一部を**図 5** に示す．なお，この場合に出力される追加情報はわずかであるので，性能にはほとんど影響を与えない．

Linpack の実行においては，BLAS ライブラリがその性能を大きく左右する．今回は GotoBLAS-1.26[10] をベースに，HugeTLB 向けにチューニングした☆ものを用いた．さらに，行交換を行う関数 HPL_dlaswp00N, HPL_dlaswp01N, HPL_dlaswp01T をスレッド化するパッチ☆を当てている．

通信ライブラリには，OpenMPI v1.3[11] の早期リリース版を用いた．T2K 筑波システムでは，multi-rail の InfiniBand をサポートしているが，今回の実行では single-rail として InfiniBand を用いている．

4.2 単一ノードでの予備実行

非常に大規模なクラスタでは，ノードの演算速度にばらつきがある場合がある．HPL においては，最も遅いノードに合わせて実行されることから，他のノードに比べて遅いノードをあらかじめ見つけておくことは重要である．

そこで HPL を実行する前に，各ノードで独立にスレッド版の Linpack を 16 スレッドで実行した．実行に際しては，HPL の実行と同様に，利用可能なメモリ領域をほぼ使い切るような問題サイズ $N = 60,000$ で実行した．その結果，各ノードで約 117GFlops（実行効率：約 79%）の性能が得られた．

☆ 著者の一人である後藤和茂氏による．

```
#!/bin/sh
NSOCKETS=4
RANK=$OMPI_COMM_WORLD_RANK
((SOCKET=RANK%NSOCKETS))
numactl --cpunodebind=$SOCKET \
--membind=$SOCKET ./xhpl
```

図 6 numactl コマンドを用いた HPL の実行スクリプトの例（OpenMPI 1.3 の場合）

表 2 LU 分解アルゴリズムの比較（$N = 300,000$, $P \times Q = 50 \times 50$, NB=232, NBMIN=4, BCAST=2ring-Modified, Column=30160, Fraction=0.100）

LU 分解	再帰的 LU 分解	TFlops
Left-looking	Left-looking	47.43
Left-looking	Right-looking	47.84
Right-looking	Left-looking	47.37
Right-looking	Right-looking	47.67

4.3 numactl コマンド

Linux kernel 2.6 では NUMA（Non Uniform Memory Architecture）向けに，メモリ割り付けを切り替える numactl コマンドがサポートされている．numactl コマンドを用いることで，メモリアロケーションをコントロールすることが可能である．

3.2 節でも述べたように，今回は 1 ソケットに対して 1 MPI プロセスを用いていることから，1 MPI プロセスが 1 ソケットに割り当てられるように，numactl コマンドでメモリアロケーションをコントロールした．

numactl コマンドを用いた HPL の実行スクリプトの例（OpenMPI 1.3 の場合）を**図 6** に示す．この実行スクリプトでは，MPI のランクを環境変数 $OMPI_COMM_WORLD_RANK から受け取り，MPI のランクをソケット数（= 4）で割った余りを求めることで，どの番号のソケットにメモリアロケーションするのかを決定している．

今回 HPL を実行する際には，図 6 の実行スクリプトを mpirun コマンドから実行した．なお，MPI のランクが格納される環境変数は MPI ライブラリによって異なることがあることに注意する．

4.4 HPL のパラメータチューニング

HPL のパラメータチューニングについては，文献

表 3 パネルブロードキャストアルゴリズムの比較（$N = 300,000$, $P \times Q = 50 \times 50$, NB=232, NBMIN=4, Column=30160, Fraction=0.100）

パネルブロードキャストのアルゴリズム	TFlops
1ring	46.79
1ring-Modified	47.11
2ring	46.84
2ring-Modified	47.67

表 4 Swapping threshold の比較（$N = 300,000$, $P \times Q = 50 \times 50$, NB=232, NBMIN=4, Column=30160, Fraction=0.100）

Swapping threshold	TFlops
64	52.87
128	52.33
192	52.71

8), 12), 13) でも指摘されているように，HPL のパラメータは非常に数が多く，最適なパラメータを決定するのは難しい．

4.1 節でも述べたように，Intel(R) Optimized MP LINPACK Benchmark for Clusters でコンパイル時に「ASYOUGO」を指定することにより，実行が進行するとともに，その都度 Mflops 値と途中経過時間が出力されるので，HPL のパラメータチューニングに有効である．しかし，この機能を使ったとしても，メモリを最大限に使った場合の問題サイズ N でパラメータチューニングを行うことは時間の制約もあり容易ではない．

そこで，利用可能なメモリの 1/25 程度のサイズである $N = 300,000$ において，ブロックサイズ NB やパネルブロードキャストのアルゴリズム，NBMIN, Panel factorization, Recursive panel factorization, Swapping threshold などのパラメータを変化させて，パラメータサーチを行った．

その際，T2K 筑波システムの 625 ノードを使ったとしても，$N = 300,000$ とした場合には HPL の実行に約 7 分を要したことから，全てのパラメータについて，その都度 HPL の実行を完了させることは時間の制約から行うことができなかった．そこで，全体の行列の約 10%まで行の消去を行った場合（Column=30160, Fraction=0.100）の性能を比較した．

上記の手法を用いて，LU 分解アルゴリズムを比較した結果を**表 2** に示す．表 2 から，LU 分解には Left-looking を，再帰的 LU 分解には Right-looking を用いた場合に最も性能が高くなることが分かる．

同様にパネルブロードキャストのアルゴリズムを比較した結果を**表 3** に示す．表 3 から，2ring-Modified が最も性能が高くなっていることが分かる．

表 4 に示すように，Swapping threshold を比較した結果，64，128，192 の中では 64 が最も高速であった．また NBMIN については 2 および 4 で計測したが，性能に有意な差は見られなかった．

表 5 625 ノード，10,000 コアを用いた評価における HPL のパラメータ

N	1508000
NB	232
Process mapping	Row-major
P	50
Q	50
Threshold	16.0
Panel factorization	Left-looking
NBMIN	4
NDIV	2
Recursive panel factorization	Right-looking
Broadcast	2ring-Modified
Lookahead depth	0
SWAP	mix
Swapping threshold	64
L1	transposed
U	transposed
Equilibration	yes
Memory alignment in double	0

HPL.dat の最後の行にある「memory alignment in double」というパラメータは，オリジナルの HPL では 0 より大きい値を指定する必要がある[5] が，この値を大きくすると必要メモリ量が増加することから，無駄なメモリを使わないようにパッチ☆を当てている．なお，今回用いた Goto BLAS では alignment の取れていないデータでも性能が低下しないような工夫がされている．

4.5 ラージページの使用

Linpack を実行する際にラージページを用いることについては，文献 14) で有効性が述べられている．ラージページを用いることにより，TLB ミスを減らすことができ，安定した性能を得ることができる．

Linux kernel 2.6 では，HugeTLB が標準でサポートされている．HugeTLB を用いたところ，使わない場合に比べて約 5%の性能向上が認められたので，今回は HPL の行列データを全て HugeTLB 上に確保して実行した．

また，HPL の性能を向上させる上で効果的であるのは，問題サイズ N をできるだけ大きくすることである．問題サイズを N とした場合，全体で $8 \times N \times (N+1)$ バイトのメモリが必要になるとともに，Goto BLAS においてバッファが 1 ノード当たり 1 GB 必要となる．

さらに，カーネルと通常プロセス用に HugeTLB 以外のメモリ領域を 1 GB 程度を残しておく必要があることを考慮し，1 ページを 2 MB として，1 ノード当たり 14,400 ページ（= 28.125 GB）を確保した．問題サイズ N は，プロセスグリッドのサイズ $P(= 50)$ と NB (= 232) の積で割り切れ，かつ 1 ノード当た

☆ 著者の一人である後藤和茂氏による．

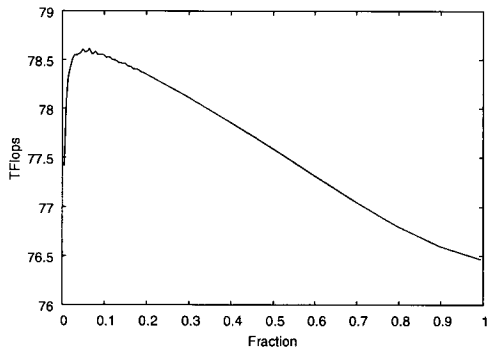

図 7 625 ノード，10,000 コアを用いた HPL 実行における TFlops 値の推移（$N = 1,508,000$）

り 14,400 ページに収まる範囲で最大になるように，$N = 1,508,000$ とした．

4.6 HPL の実行結果

T2K 筑波システムの 625 ノード，10,000 コアを用いて $N = 1,508,000$ の HPL を実行した．用いた HPL のパラメータを**表 5** に示す．

図 7 は，この HPL 実行における TFlops 値の推移を示したものである．図 7 から，Fraction=0.065 付近で約 78.6 TFlops のピークに達した後は，性能がほぼ直線的に低下していることが分かる．また，ブロックサイズ NB を大きくした場合は，最初は高い性能を示すが，LU 分解が進むにつれて性能の低下が大きくなる傾向にある．

最終的に，76.46 TFlops（実行効率：83.1%）の性能が得られた．実行に要した時間は約 8.3 時間である．なお，この結果は 2008 年上半期の TOP500[2] に登録されており，T2K 筑波システムは第 20 位にランクされた．

5. ま と め

本論文では，T2K 筑波システムの概要を述べるとともに，Linpack 性能評価を行った結果について報告した．

T2K 筑波システムの各ノードには Quad-Core AMD Opteron が 4 ソケット搭載されているが，HPL の実行に際しては 1 ソケットに対して 1 MPI プロセスを用いることとし，1 MPI プロセスが 1 ソケットに割り当てられるように，`numactl` コマンドを用いてメモリアロケーションをコントロールした．

さらに，HPL の行列データを全て HugeTLB 上に確保して実行することで安定した性能を得ることができた．また，HPL の実行に際しては問題サイズ N をできるだけ大きくできるように設定することで，高い性能を得ることができた．

T2K 筑波システムの 625 ノード，10,000 コアを用いて HPL を実行した．その結果，$N = 1,508,000$ に対して，76.46 TFlops（実行効率：83.1%）の性能が得られた．

謝辞 T2K 筑波システムにおいて Linpack ベンチマークを実行するにあたり，多くの面でご協力を頂いた，クレイ・ジャパン・インク，Appro International, Inc.，株式会社 HPC ソリューションズ，住商情報システム株式会社および筑波大学計算科学研究センターの関係者諸氏に深く感謝致します．

参 考 文 献

1) T2K Open Supercomputer Alliance. `http://www.open-supercomputer.org/`
2) TOP500 supercomputer sites. `http://www.top500.org/`
3) Gfarm file system. `http://datafarm.apgrid.org/`
4) TSUKUBA WAN. `http://www.tsukuba-wan.ne.jp/`
5) Petitet, A., Whaley, R. C., Dongarra, J. and Cleary, A.: HPL — A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers. `http://www.netlib.org/benchmark/hpl/`
6) BLAS. `http://www.netlib.org/blas/`
7) VSIPL. `http://www.vsipl.org/`
8) 遠藤敏夫, 松岡聡, 橋爪信明, 長坂真路: ヘテロ型スーパーコンピュータ TSUBAME の Linpack による性能評価, 情報処理学会論文誌：コンピューティングシステム, Vol. 48, No. SIG 8(ACS 18), pp. 62–70 (2007).
9) Intel Corporation: *Intel(R) Optimized MP LINPACK Benchmark for Clusters 1.0 (Based on HPL 1.0a from ICL at UTK) Release Notes* (2005).
10) Goto, K.: Goto BLAS. `http://www.tacc.utexas.edu/resources/software/`
11) Open MPI: Open Source High Performance Computing. `http://www.open-mpi.org/`
12) 笹生健, 松岡聡: HPL のパラメータチューニングの解析, 情報処理学会研究報告 2002-HPC-91, pp. 125–130 (2002).
13) 寒川光, 藤本康, 建部修見, 児玉祐悦, 横川三津夫, 工藤知宏, 関口智嗣: AIST スーパークラスタ P-32 の Linpack による性能評価, 情報処理学会研究報告 2004-HPC-99, pp. 163–168 (2004).
14) 成瀬彰, 住元真司, 久門耕一: Xeon プロセッサ向け Linpack ベンチマーク最適化手法とその評価, 情報処理学会論文誌：コンピューティングシステム, Vol. 45, No. SIG 11(ACS 7), pp. 62–70 (2004).