HPLのパラメータチューニングの解析

笹生　健[†]　松岡　聡[††]

LINPACK ベンチマークの実装の一つである HPL プロセスは，Top500 リストの順位付けを決定する性能評価のために多くの計算機上で用いられている．HPL はパラメータを設定することでシステムの性能を引き出した評価が可能という特徴があるが，HPL のパラメータは非常に数が多く，プロセッサのキャッシュサイズやメモリサイズ，ネットワーク性能等，並列システムのアーキテクチャに依存するため，最適なパラメータを決定するのは困難である．そのため HPL を使用するユーザーにとって様々なアーキテクチャの並列システム上における HPL のパラメータ設定に関する情報は非常に有益である．本稿では第 19 回 Top500 リストにおいて 47 位を達成した PrestoIII クラスタが Top500 の際に使用した HPL のパラメータを公開すると共に，PrestoIII クラスタ上での様々なパラメータ設定による比較を行った．それにより最適なパラメータ設定の方針に関する知見を得た．

Performance Tuning High-Performance Linpack (HPL)

Takeru Sasou [†] and Satoshi Matsuoka[††]

HPL is one of the implementation of LINPACK benchmark and is used for performance evaluation of Top500 by many users. We can achieve good performance by tuning parametars, but it is difficult to determine the best parameter since HPL has many parameters. So, the information about a parameter setup of HPL on various parallel systems of is very useful for users. In this paper, we exhibit the configuration of HPL when PrestoIII cluster ranked as the 47th place in the 19th Top500 list, and evaluate in all kinds of parametar setting on PrestoIII cluster. Therefore, we acquired the knowledge about the line of the best parameter tuning.

1. はじめに

世界中の高性能システムの勢力分布，先端技術の達成度，各国別の計算機保有動向，ベンダの技術の推移など，多方面にわたる動向を調査する事を目的として，世界中の高性能計算機システムの性能 Top500 リスト[1] がある．Top500 リストの順位付けを決定する性能値は Tennessee 大学の Dongarra によって公表されている LINPACK ベンチマーク[2] の Highly Parallel Computing の結果であり，このクラスでは数値解の精度のみ要求され，問題サイズや解法は規定されていない．Highly Parallel Computing の実装の一つに HPL(High-Performance LINPACK Benchmark)[3] がある．HPL1 プロセスは実行時に様々なパラメータを設定する事でシステムの性能を引き出した評価が可能という特徴があり，Top500 リストの上位に位置する多くの計算機で使用されている．HPL1 プロセスで高い性能を発揮するためには適切なパラメータ設定が欠かせないが，HPL のパラメータは非常に数が多く，プロセッサのキャッシュサイズやメモリサイズ，ネットワーク性能等，並列システムのアーキテクチャに依存するため，最適なパラメータを決定するのは困難である．HPL のソースに添付されているパラメータ設定のガイドラインはあくまで経験的なものとして書かれおり，様々なアーキテクチャの並列システム上における HPL のパラメータ設定に関する情報がより多く存在する事は HPL を使用するユーザーにとって有益である．

そこで，本稿では第 19 回 Top500 リストにおいて 47 位を達成した PrestoIII クラスタ[4] が Top500 の際に使用した HPL のパラメータを公開すると共に，PrestoIII クラスタ上での様々なパラメータ設定による比較を行った．それにより最適なパラメータ設定の方針に関する知見を得た．

2. HPL

HPL は，分散メモリ型並列計算機用のベンチマークソフトウェアであり，ガウス消去法を用いた密行列連立 1 次方程式の求解の際の実行時間で性能を評価する．HPL は C 言語で実装されており，実行には MPI1.1[5]

† 東京工業大学
Tokyo Institute of Technology
†† 東京工業大学 学術国際情報センター
Tokyo Institute of Technology, GSIC

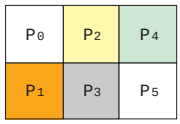

図 1 プロセス格子の例 (2 × 3)

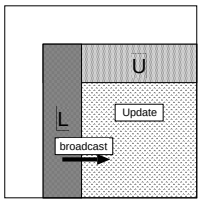

図 2 ブロック LU 分解

準拠のメッセージ通信ライブラリを必要とする．また行列演算に BLAS[6],VSIPL[7] のどちらかを使用する．HPL は様々なパラメータを計算機の特性に合わせて設定したり，高度に最適化された行列演算カーネルを組み込む事で，より高い性能を得る事ができるようになっている．

2.1 アルゴリズム

HPL では，まずプロセスを図 1 の様にプロセス格子という 2 次元の格子状に並べる。次に，係数行列を複数の正方形ブロックに分割して，2 次元プロセス格子上に対してブロックサイクリックに割り当てる．LU 分解のメインループは right-looking の LU 分解アルゴリズム 2 であり，その処理は`Panel Factorization`，`Panel Broadcast`，`Update`，`Backward Substitution` というフェイズから構成される．それぞれにおいてはパネル列 LU 分解，分解済パネルの送信，未分解小行列の更新計算，後退代入演算による $Ux = y$ の求解を行なう．

2.2 パラメータ

HPL では以下の 16 項目についてパラメータを設定できる．性能に大きく影響を与えると思われるのは問題サイズ，ブロックサイズ，プロセス格子のサイズ，LU 分解アルゴリズム，Broadcast のトポロジー，等である．

- 問題サイズ N
- ブロックサイズ NB
- プロセス格子のサイズ (P, Q)
- 解のチェックにおける残差の境界値
- Panel Factorization のアルゴリズム
- 再帰的 Panel Factorization のアルゴリズム
- 再帰的 Factorization における subpanel 数
- 再帰的 Factorization における subpanel 幅の最小値
- Panel Broadcast のトポロジー
- Look-ahead の深さ
- Update における通信トポロジー
- long における U の平衡化処理の有無
- mix における行数の境界値
- L1 Panel の保持の仕方
- U Panel の保持の仕方
- メモリの alignment

2.3 HPL の計算量

HPL では通信は全て 1 対 1 で行う．レイテンシを α，スループットを β，メッセージサイズを L とすると，通信時間 T_c は以下のようになる．

$$T_c = \alpha + \beta L \tag{1}$$

BLAS の Level1(vector-vector)，Level2(vector-matrix)，Level3(matrix-matrix) による floating operation の実行時間をそれぞれ γ_1，γ_2，γ_3 とする．

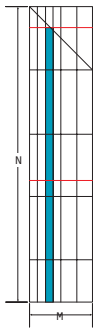

図 3 Panel Factorization

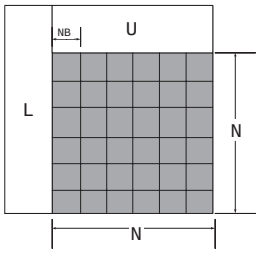

図 4 Update

Panel Factorization フェイズ 3 において $M \times N$ の小行列を P 個のプロセッサで LU 分解する場合を考えると，matrix-matrix 演算の再帰的 LU 分解を行うので floating operation は Level3 BLAS を使用することになり，$O(N^3)$，$O(N^2)$ の項に注目すると $\gamma_3 \left(\frac{M}{P} - \frac{N}{3}\right) N^2$ となる．また，ピボット選択のためにプロセス間でサイズが $2N$ のデータを binary-exchange 通信 ($\log P$ ステップ) でやりとりするので通信時間は $N \log P(\alpha + 2\beta N)$．Panel Broadcast のトポロジーを increasing-1ring(modified) と仮定すると，メッセージ送信は 1 回のみとなり $\alpha + \frac{\beta MN}{P}$ となる．以上から Panel Factorization の実行時間は以下のようになる．

$$T_{pfact}(M, N) = \gamma_3 \left(\frac{M}{P} - \frac{N}{3}\right) N^2 + N \log P(\alpha + 2\beta N) + \alpha + \frac{\beta MN}{P} \tag{2}$$

Update フェイズ (図 4) において $N \times N$ の小行列を更新する場合，プロセス格子を $P \times Q$，ブロックサイズを N_B とすると，Upanel の分解は $\gamma_3 \left(\frac{N \cdot N_B^2}{Q}\right)$ となり，消去計算は $\gamma_3 \left(\frac{2N^2 \cdot N_B}{PQ}\right)$ となる．Upanel の列方向 broadcast を Bandwidth-reducing 通信で行うと仮定すると，$\log P$ ステップの spread フェイズと $P - 1$ ステップの rolling フェイズで $N \cdot N_B$ のメッセージが送受信されるので，Update フェイズの実行時間は以下のようになる．

$$T_{update}(N, N_B) = \gamma_3 \left(\frac{N \cdot N_B^2}{Q} + \frac{2N^2 \cdot N_B}{PQ}\right) + \alpha(\log P + P - 1) + \frac{2\beta N \cdot N_B}{Q} \tag{3}$$

Backward substitution では $y \leftarrow y - Ux$ の vector-matrix 計算を行うので，$\frac{\gamma_2 N^2}{PQ}$．長さ N_B の vector を左のプロセス列に送信した後にそのプロセス列内で vector を broadcast するという操作を $\frac{N}{N_B}$ 回行うので，Backward substitution の実行時間は以下のようになる．

$$T_{backs}(N, N_B) = \frac{\gamma_2 N^2}{PQ} + N\left(\frac{\alpha}{N_B} + 2\beta\right) \tag{4}$$

(2)，(3)，(4) から HPL の合計の実行時間は以下のようになる．

図 5 prestoIII クラスタ

表 1 PrestoIII クラスタの仕様

ノード数	256
CPU	AMD Athlon MP 1900+ ×512 プロセッサ (2 プロセッサ/ノード)
マザーボード	Tyan TigerMP (AMD760 Chipset) or Tyan Thunder K7 (AMD760 Chipset)
メモリ	合計 192GB (768MB/ノード)
ハードディスク	合計 20TB
ネットワーク	100BASE-TX 1 系統, Myricom Myrinet2000 1 系統

$$\sum_{k=0}^{N}[T_{pfact}(N-k\cdot N_B,N_B) + T_{update}(N-(k-1)\cdot N_B,N_B)] + T_{backs}(N,N_B) \quad (5)$$

主要な α，β，γ_3 の項に注目すると以下のようになる．

$$T_{HPL} = \gamma_3\frac{2N^3}{3PQ} + \beta\frac{N^2(2P+Q)}{2PQ} + \alpha\frac{N((N_B+1)\log P + P)}{N_B} \quad (6)$$

3. パラメータ設定による比較

3.1 実験環境:PrestoIII クラスタ

PrestoIII クラスタ (図 5) は松岡研究室が所有する 5 台目のクラスタである．PrestoIII クラスタは HPL を用いた性能測定により 716.1GFlops を達成した．これにより第 19 回 Top500 において 47 位を達成し，x86 アーキテクチャのクラスタとしては世界 2 位，日本国内 1 位という結果を納めた．この結果は大学の研究室レベルでテラ級の性能を達成できる事を実証するものである．また，近い将来に，多くの研究室がテラ級のシステムを用いて研究を行い，新しい科学的発見に結びつく事を期待させる結果でもある．

PrestoIII クラスタの仕様を**表 1** に示す．PrestoIII クラスタは CPU に AMD Athlon™MP プロセッサ 1900+を使用し，768MB のメモリを搭載した Dual CPU の PC256 ノードから構成されている．各ノードは Myricom 社の Myrinet2000[9] を使用した高速ネットワークで接続されている．

3.2 問題サイズ

HPL で消費するメモリの大部分は行列データであり，その量は $N\times(N+1)$×64bit となる．その他に

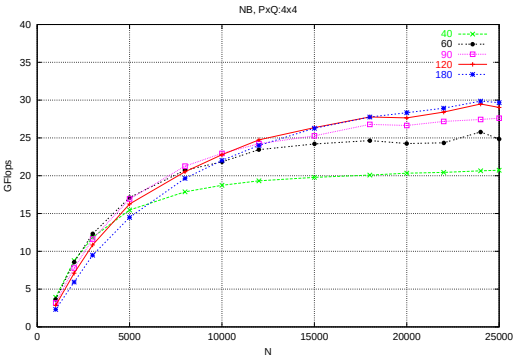

図 6 ブロックサイズ NB による性能比較 $(P,Q)=(4,4)$

表 2 ブロックサイズ NB による性能比較 $(P,Q)=(4,4)$

N	NB=40	NB=60	NB=90	NB=120	NB=180
1000	3.914	3.660	3.178	2.850	2.318
2000	8.776	8.569	7.786	7.103	5.932
3000	11.85	12.31	11.62	10.85	9.485
5000	15.47	17.11	16.92	16.26	14.49
8000	17.87	20.72	21.27	20.52	19.65
10000	18.73	21.83	22.95	22.78	22.02
12000	19.32	23.45	24.23	24.74	24.01
15000	19.78	24.20	25.28	26.36	26.28
18000	20.09	24.65	26.79	27.77	27.76
20000	20.33	24.26	26.65	27.65	28.33
22000	20.44	24.35	27.18	28.42	28.93
24000	20.65	25.79	27.44	29.49	29.87
25000	20.72	24.85	27.61	29.03	29.67

HPL のバイナリ，shmem 通信用の領域，計算中の作業領域などでメモリを消費する．基本的には問題サイズの増大に伴って性能の向上が見られるが，消費するメモリ領域サイズが使用可能な物理メモリ量を超えるとメモリのスワップが起きるためにより性能が低下する．

3.3 ブロックサイズ

PrestoIII の 8 ノード 16CPU を使用し，(P,Q)=(4,4) においてブロックサイズ，問題サイズを変えて測定した結果を図 6，**表 2** に示す．実験の結果では問題サイズが大きくなるほど最適なブロックサイズも大きくなっている．ブロックサイズは小さくする事によって各プロセスのロードバランスが良くなる一方，総ブロック数が増大する事によりプロセス間の通信回数も増大するためトレードオフの関係にある．また，ブロックサイズは CPU のキャッシュサイズに応じた値を選ぶ方が性能が高くなる．今回行列演算ライブラリには Athlon MP 1900+(L1 キャッシュ64kB，L2 キャッシュ256KB) に最適化した ATLAS3.3.14[10] を使用しており，インストールログによると L1 matmul のブロックサイズは 60 なので，ブロックサイズにはその倍数を使用している．

3.4 プロセス格子のサイズ

16 プロセッサ使用して (P,Q) の全ての組み合わせにおいて測定した結果を図 7, **表 3** に示す。プロセス格子のサイズ P,Q はなるべく正方形に近い方が良い．(6) 式の $\gamma_3\frac{|P+Q}{PQ}N^2$ の項があるためである．プロセッサ数の都合で正方形にできない場合は，$P<Q$ とし

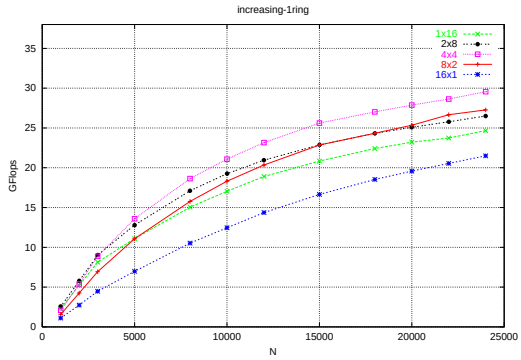

図 7 プロセス格子のサイズによる性能比較 (Broadcast:1ring)

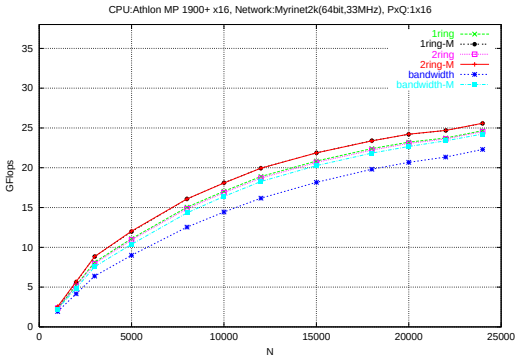

図 8 $(P, Q) = (1, 16)$

表 3 プロセス格子のサイズによる性能比較 (Broadcast:1ring)

N	1 × 16	2 × 8	4 × 4	8 × 2	16 × 1
1000	2.319	2.561	2.101	1.571	1.116
2000	5.212	5.779	5.392	4.249	2.745
3000	8.135	9.029	8.864	6.937	4.475
5000	11.09	12.80	13.59	11.08	6.988
8000	15.05	17.11	18.64	15.77	10.53
10000	17.05	19.25	21.08	18.31	12.46
12000	18.90	20.95	23.16	20.35	14.37
15000	20.85	22.87	25.61	22.85	16.65
18000	22.41	24.33	27.02	24.34	18.53
20000	23.22	25.08	27.88	25.34	19.58
22000	23.74	25.78	28.64	26.66	20.55
24000	24.66	26.50	29.56	27.27	21.50

表 4 prestoIII 1CPU での LU 分解アルゴリズムの比較 (N=9000, NB=180, NDIV=2, NBMIN=2)

再帰的 LU 分解	LU 分解	GFlops
crout	crout	2.421
	left-looking	2.418
	right-looking	2.422
left-looking	crout	2.417
	left-looking	2.417
	right-looking	2.416
right-looking	crout	2.420
	left-looking	2.419
	right-looking	2.423

た方が (6) 式の $\beta\frac{N^2(2P+Q)}{2PQ}$ の項が小さくなるので性能が高い.

3.5 Panel Factorization のアルゴリズム

HPL では LU 分解と再帰的 LU 分解のそれぞれにおいて，crout，left-looking，right-looking の 3 種類のアルゴリズムを選択できる．一般的に，left-looking は参照するデータの局所性が強く，キャッシュサイズが大きい時に有利であり，right-looking はループアンローリングによる多段同時消去が効く等の特徴がある．

PrestoIII の 1 プロセッサで再帰的 LU 分解アルゴリズム，LU 分解アルゴリズムの全ての組み合わせにおける比較実験の結果を図**表 4** に示す．実験は問題サイズを 9000，ブロックサイズを 180，再帰的 LU 分解の subpanel 数を 2，subpanel 幅の最小値を 2 とした．実験では再帰的 LU 分解，LU 分解共に right-looking という組み合わせが最も性能が高いという結果が出た．しかしながら，それぞれの差は高々数 MFlops であり，アルゴリズムの違いによる性能差はそれほど大きくない。

3.6 Panel Broadcast のトポロジー

Panel Broadcast のトポロジーには increasing-1ring，increasing-2ring，Bandwidth-reducing の 3 種類と，次の Panel Factorization を行うプロセスにメッセージ送信をさせないようにした modified 版がそれぞれ 3 種類の計 6 種類から選択できる．

16 プロセッサ使用した (P,Q) の全ての組み合わせにおいて，Panel Broadcast トポロジーの比較実験の結果を図 8〜12, **表 5**〜**表 9** に示す．normal のトポロジーと modified 版を比べると，

normal メッセージ受信→メッセージ送信→ Update → Panel Factorization

modified メッセージ受信→ Update → Panel Factorization

となり modified 版の方がメッセージ送信が無い分，速くなる．increasing-2ring は全体の通信ステップ数を減らす事ができるので，図 8, 表 5 に見られるように，Q が大きい場合に有利である．実験では increasing-1ring(modified) と increasing-2ring(modified) のどちらかが最適なトポロジーとなっている．HPL のサイトでもこの 2 つのトポロジーが多くの場合高性能を達成するパラメータであると述べられており，この実験でもそれが確認できる．

Bandwidth-reducing は broadcast するメッセージを Q 等分して，まず最初に Q 個のプロセスに分散し，隣り合うプロセス間で相互にメッセージの断片を Q-1 回やり取りする．この通信の特徴は送受信するメッセージサイズが $\frac{1}{Q}$ になるため，プロセッサ性能に比べてネットワーク性能が圧倒的に低い状況では Bandwidth-reducing の方が性能が良くなる．PrestoIII において使用するネットワークを 100BASE-TX の Ethernet として，(P,Q)=(1,16) で測定を行ったところ (図 13, **表 10**)，Bandwidth-reducing が最も高い性能を発揮した．

表 5 $(P, Q) = (1, 16)$

N	1ring	1ring-M	2ring	2ring-M	long	long-M
1000	2.319	2.464	2.357	2.556	1.934	2.248
2000	5.212	5.623	5.127	5.686	4.158	4.854
3000	8.135	8.837	7.950	8.838	6.391	7.561
5000	11.09	11.97	10.94	12.04	9.000	10.38
8000	15.05	16.07	14.86	16.09	12.54	14.33
10000	17.05	18.11	16.87	18.09	14.44	16.39
12000	18.90	19.95	18.70	19.95	16.18	18.28
15000	20.85	21.88	20.67	21.87	18.17	20.27
18000	22.41	23.40	22.22	23.38	19.81	21.84
20000	23.22	24.20	23.06	24.22	20.68	22.67
22000	23.74	24.69	23.62	24.69	21.34	23.39
24000	24.66	25.57	24.55	25.60	22.30	24.24

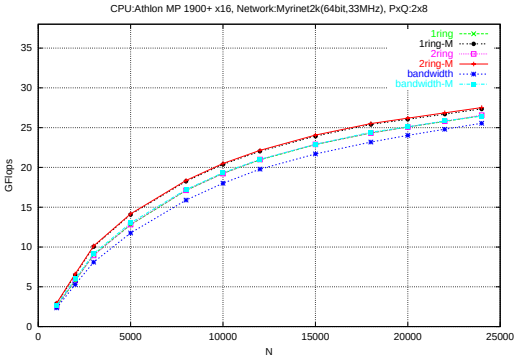

図 9 $(P, Q) = (2, 8)$

表 6 $(P, Q) = (2, 8)$

N	1ring	1ring-M	2ring	2ring-M	long	long-M
1000	2.561	2.885	2.581	2.960	2.354	2.683
2000	5.779	6.488	5.790	6.611	5.293	6.005
3000	9.029	10.05	8.941	10.14	8.094	9.174
5000	12.80	14.09	12.82	14.17	11.76	13.03
8000	17.11	18.26	17.13	18.37	15.90	17.23
10000	19.25	20.38	19.23	20.50	18.01	19.34
12000	20.95	22.03	21.00	22.15	19.79	21.02
15000	22.87	23.93	22.90	24.07	21.70	22.91
18000	24.33	25.40	24.32	25.51	23.19	24.43
20000	25.08	26.05	25.08	26.19	24.03	25.11
22000	25.78	26.69	25.80	26.86	24.80	25.85
24000	26.50	27.36	26.57	27.52	25.55	26.45

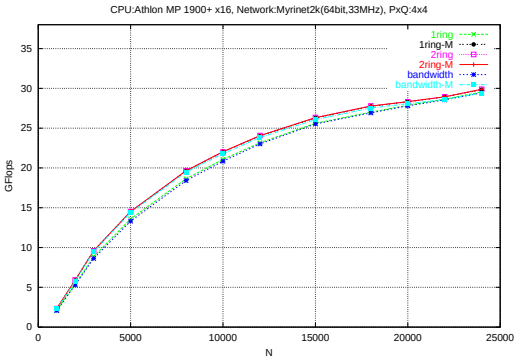

図 10 $(P, Q) = (4, 4)$

表 7 $(P, Q) = (4, 4)$

N	1ring	1ring-M	2ring	2ring-M	long	long-M
1000	2.101	2.326	2.337	2.330	2.079	2.282
2000	5.392	5.885	5.935	5.894	5.295	5.784
3000	8.864	9.591	9.589	9.589	8.627	9.501
5000	13.59	14.49	14.52	14.52	13.32	14.40
8000	18.64	19.63	19.66	19.64	18.41	19.49
10000	21.08	22.01	22.01	22.05	20.85	21.79
12000	23.16	24.02	24.08	24.06	23.02	23.83
15000	25.61	26.28	26.35	26.31	25.52	26.08
18000	27.02	27.75	27.79	27.78	26.92	27.53
20000	27.88	28.32	28.32	28.34	27.80	28.08
22000	28.64	28.93	28.94	28.94	28.55	28.60
24000	29.56	29.83	29.89	29.88	29.44	29.40

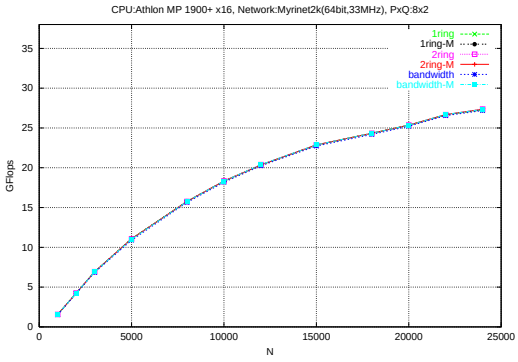

図 11 $(P, Q) = (8, 2)$

表 8 $(P, Q) = (8, 2)$

N	1ring	1ring-M	2ring	2ring-M	long	long-M
1000	1.571	1.576	1.569	1.572	1.540	1.569
2000	4.249	4.229	4.251	4.229	4.195	4.235
3000	6.937	6.937	6.940	6.928	6.851	6.914
5000	11.08	11.08	11.06	11.07	10.90	11.02
8000	15.77	15.79	15.76	15.77	15.64	15.76
10000	18.31	18.33	18.34	18.33	18.18	18.32
12000	20.35	20.36	20.39	20.36	20.25	20.39
15000	22.85	22.85	22.88	22.88	22.73	22.88
18000	24.34	24.35	24.32	24.34	24.18	24.34
20000	25.34	25.33	25.36	25.36	25.22	25.35
22000	26.66	26.66	26.67	26.68	26.55	26.69
24000	27.27	27.28	27.37	27.36	27.19	27.33

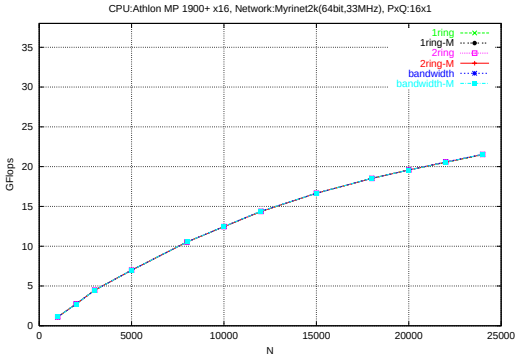

図 12 $(P, Q) = (16, 1)$

4. 第 19 回 Top500 における PrestoIII のパラメータ

第 19 回 Top500 で PrestoIII が 716.1GFlops を記録した際のパラメータを以下に示す.

- 問題サイズ N:100,000
- ブロックサイズ NB:180
- プロセス格子のサイズ (P, Q):20 × 24
- 解のチェックにおける残差の境界値:16.0
- Panel Factorization のアルゴリズム:right-looking
- 再帰的 Panel Factorization のアルゴリズム:right-looking
- 再帰的 Factorization における subpanel 数:2
- 再帰的 Factorization における subpanel 幅の最小値:2
- Panel Broadcast のトポロジー:increasing-2ring(modified)
- Look-ahead の深さ:0
- Update における通信トポロジー:mix
- long における U の平衡化処理の有無:有
- mix における行数の境界値:64
- L1 Panel の保持の仕方:transposed

表 9 $(P,Q) = (16,1)$

N	1ring	1ring-M	2ring	2ring-M	long	long-M
1000	1.116	1.120	1.115	1.117	1.117	1.112
2000	2.745	2.742	2.749	2.744	2.744	2.733
3000	4.475	4.480	4.481	4.471	4.466	4.467
5000	6.988	6.989	6.986	6.988	6.986	6.992
8000	10.53	10.53	10.53	10.53	10.53	10.54
10000	12.46	12.46	12.47	12.46	12.48	12.47
12000	14.37	14.36	14.37	14.37	14.41	14.40
15000	16.65	16.66	16.66	16.66	16.66	16.66
18000	18.53	18.54	18.52	18.52	18.54	18.54
20000	19.58	19.58	19.57	19.58	19.58	19.57
22000	20.55	20.56	20.59	20.54	20.59	20.54
24000	21.50	21.50	21.53	21.53	21.55	21.55

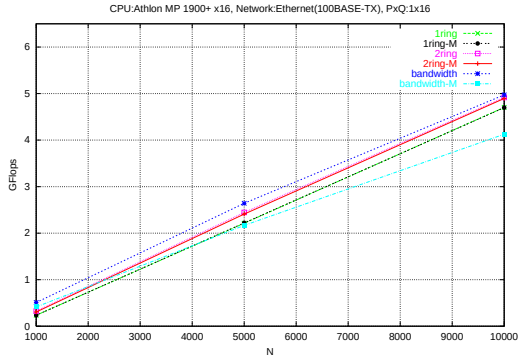

図 13 $(P,Q) = (1,16)$, Network=Ethernet(100BASE-TX)

表 10 $(P,Q) = (1,16)$, Network=Ethernet(100BASE-TX)

N	1ring	1ring-M	2ring	2ring-M	long	long-M
1000	0.2354	0.2357	0.3244	0.3054	0.5088	0.4131
5000	2.213	2.219	2.445	2.410	2.642	2.167
10000	4.703	4.699	4.917	4.895	4.972	4.126

- U Panel の保持の仕方:transposed
- メモリの alignment:8

BLAS ライブラリは 3.3 節で使用した ATLAS のバイナリを使用している．問題サイズを除いて，各パラメータは今回の実験結果から得られる最良のものと同じパラメータ選択が為されている．Top500 時は全てのノードを安定して稼働させる事ができなかったためメモリ使用量が少ない問題サイズとなっているが，全ノード全メモリを使用した場合，これまでの結果から性能を概算すると N=150,000 程度で 0.9〜1TFlops を達成できるものと予想される．

5. おわりに

PrestoIII クラスタ上で HPL のパラメータ設定についての比較実験を行った．今回の結果は HPL のソースに添付されたガイドラインで推奨するパラメータと同じであり，最適なパラメータ設定は以下の通りであった．

- ブロックサイズは CPU のキャッシュサイズに応じて最適な値を探す．
- プロセス格子のサイズ (P,Q) は正方形に近い方が良く，$P \leq Q$ の方が好ましい．
- Panel Factorization のアルゴリズム，再帰的 Panel Factorization のアルゴリズムは right-looking の組み合わせが最も性能が高かったが，パラメータの違いでそれほど差が無い．
- Panel Broadcast のトポロジーは increasing-1ring(modified) か increasing-2ring(modified) のどちらかが最も性能が高い．ただし，CPU 性能が高いがネットワーク性能が低い場合は long が最適である事もある．

特にブロックサイズの決定が最も困難であるが，これは ATLAS のインストールログを参考にし，その整数倍をいくつか試して決定するのが良い．

今後は松岡研のクラスターチームのサイト[4] 上で，松岡研のクラスタにおける HPL を始めとした様々なベンチマークの結果や分析を公開していく予定である．

参 考 文 献

1) TOP500 Supercomputer Sites:TOP500 Supercomputer Sites, http://www.top500.org/
2) http://www.netlib.org/benchmark/top500/lists/linpack.html
3) Antoine Petitet, R. Clint Whaley, Jack J. Dongarra, and Andy Cleary. *HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers*. Innovative Computing Laboratory, September 2000. Available at http://ics.cs.utk.edu/hpl/
4) http://cluster-team.is.titech.ac.jp/
5) MPICH Home Page, http://www-unix.mcs.anl.gov/mpi/mpich/
6) BLAS (Basic Linear Algebra Subprograms), http://www.netlib.org/blas/
7) Vector Signal Image Processing Library, http://www.vsipl.org/
8) http://datarafm.apgrid.org/
9) http://www.myri.com/
10) Automatically Tuned Linear Algebra Software (ATLAS), http://math-atlas.sourceforge.net/
11) Jack J. Dongarra, J. Bunch, Cleve Moler, and G. W. Stewart. *LINPACK User's Guide*. SIAM, Philadelphia, PA, 1979.