2004－HPC－99　(28)
2004／7／31

AIST スーパークラスタ P-32 の Linpack による性能評価

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 スーパークラスタは，産業技術総合研究所のグリッド研究センターに導入されたクラスタシステムで，P-32，M-64，F-32 の 3 システムからなる．3 システムの内最大の計算能力をもつ P-32 クラスタで Linpack ベンチマークを行い，理論ピーク性能値の 75% に該当する 6.155 Tflop/s を実測した．本論文では大規模な Linpack の実行における計算時間および通信時間の振舞いを，HPL プログラムのアルゴリズムに沿って分析し，多くのパラメータの組合せから適切なものを少ない試行回数で見つけ出す方法を述べる．またベンチマークの実施を通して明らかになった，Linux カーネルの NUMA サポートが及ぼす性能への影響について報告する．

Performance Evaluation of AIST Supercluster P-32 by Linpack

Hikaru Samukawa,† Yasushi Fujimoto,† Osamu Tatebe,††
Yuetsu Kodama,†† Mitsuo Yokokawa,†† Tomohiro Kudoh††
and Satoshi Sekiguchi††

AIST supercluster installed at Grid Technology Research Center consists of three systems, P-32, M-64, and F-32. We performed Linpack benchmark on P-32 cluster system which is the largest among the three systems. The measured performance of 6.155 Tflop/s corresponds to 75% of the theoretical peak performance. This paper reports how an appropriate combination of parameters of HPL program was effectively found based upon HPL program analysis for computation and communication time behavior for large scale problems. Then effects of NUMA capability of Linux kernel to the Linpack performance which was revealed through the benchmark.

1. はじめに

AIST スーパークラスタは，Linux をオペレーティングシステムに用いた，全体で 14.6Tflop/s の総演算性能と 9.6TB のメインメモリ，803TB のストレージを持つクラスタ計算機である．システムは，全体の計算性能を重視した「P-32 クラスタ部」，個々の計算機毎のメモリ容量を重視した「M-64 クラスタ部」，複数の独立した計算処理を同時に処理することを目指した「F-32 クラスタ部」の 3 種類のクラスタ部と 20TB の「ストレージ部」から構成されている．これらのがれらは 10 ギガビットイーサネットで相互に接続され，用途に応じて各部を使い分け，あるいは組み合わせて使用することにより最適な性能が得られる[1]．

2. P-32 システム構成

2.1 P-32 スーパークラスタのシステム構成

P-32 クラスタ部は，2 つの AMD 社製 Opteron プロセッサ（クロック周波数 2.0GHz）を有する IBM の e325 サーバー 1,072 台をギガビットイーサネットと Myrinet でつないでいる．全体で 8.6Tflop/s の総演

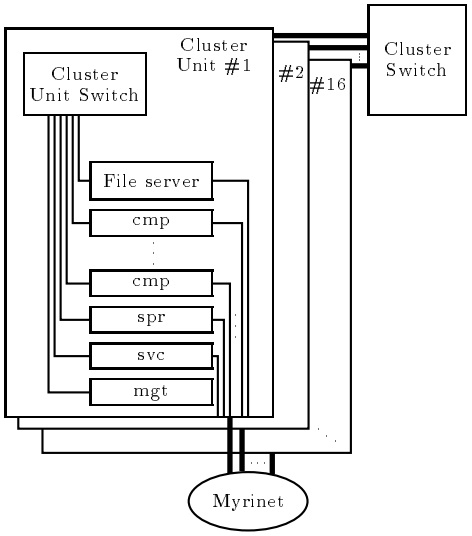

図 1　P-32 クラスタ部構成図
Fig. 1　Configuration of P-32 Cluster part

† 日本アイ・ビー・エム株式会社
IBM Japan, Ltd.
†† 産業技術総合研究所グリッド研究センター
Grid Technology Research Center, AIST

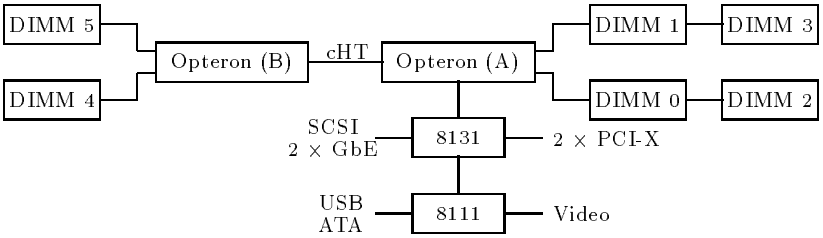

図 2 IBM eServer 325 システムアーキテクチャ
Fig. 2 IBM eServer 325 System Architecture

表 1 P-32 のシステム構成
Table 1 System configuration of P-32.

ノード	1024(cmp) + 16(spr) + 16(svc)
CPU	AMD Opteron 2.0 GHz × 2
L1 キャッシュ	64KB（命令） + 64KB（データ）
L2 キャッシュ	1GB（統合）
チップセット	AMD 8131 + 8111
メモリ	PC2700 Registered ECC 6GB
OS	SuSE Linux for AMD64
カーネル	Linux-2.4.21-143(with SCore patch)
並列環境	SCore Cluster S/W
通信ドライバ	PM
通信媒体	Myrinet

算性能と，6.4TB のメインメモリ，565TB のストレージを持つ．図 1 に構成を示したが，計算ノード（cmp）とスペアノード（spr）は 6GB，サービスノード（svc）は 4GB のメモリを持つ．この $(64+1+1)\times 16 = 1056$ 台が Myrinet に接続する．Myrinet は，各リンクが 2Gbps の物理帯域をもち，各ノードはネットワークインタフェース M3F-PCIXD-2 を 64bit/133MHz の PCI-X バスを介して装着している．ネットワークは 128 ポートのスイッチ 26 台で構成され，高いバイセクション帯域を持っている．P-32 は SCore（一部の機能を除いて）を実装している[2)]☆．なお 1 対 1 の実効性能値は，パケット長 8192B で片方向を実測した結果，212.6MB/s であった．

2.2 e325 システムアーキテクチャ

IBM eServer 325 は科学技術計算に向けた高性能サーバーで，AMD の Opteron プロセッサを 2 ウェイで使用している[3)4)]．IBM eServer 325 は SMP 構成（複数のプロセッサでメモリコントローラを共有する方式）を採らず，メモリコントローラをそれぞれのプロセッサチップに内蔵する方式を採っている．こらはプロセッサの高周波数化にメモリ性能が追随しやすいように設計された結果である．図 2 に e325 の構成を示したが，Opteron プロッセサ（のメモリコントローラ）から直接 DIMM へリンクが伸びるのが特長である．DIMM は容量 1GB で 333MHz の DDR である．2 つの Opteron を接続するのは両プロセッサに内蔵された Coherent HyperTransport（cHT）ユニットを結ぶリンクである．この方式では，Opteron(A) からは DIMM 0,1,2,3 がローカルで，DIMM 4,5 はリモートになり，Opteron(B) からは反対になる．メモリアクセス時間はローカルとリモートでは約 80ns と 115ns の差がある．またメモリ帯域ではローカルに対し 5.3GB/s の性能があり，2 ウェイの場合，ローカル／ローカルでは 10.6GB/s だが，リモート／リモートとクロスすると cHT の帯域である 6.4GB/s に低下する．

☆ 現時点では SCore の Opteron サーバ上での実装は限定的なものである．

3. HPL

HPL（High-Performance Linpack Benchmark）は Linpack ベンチマーク実装の一つである[5)]．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，密行列連立 1 次方程式の求解の実行時間により性能を評価する．問題のサイズ n をパラメータで指定すると，HPL プログラムが擬似乱数で行列を生成する．n は可能なかぎり大きくとることで Gflop/s 性能値を上げることができるが，P-32 で 6GB×1024 では 6TB のメモリが使用でき，n は 60 から 80 万となり，実行時間も 10 時間を越える．なおプログラミング言語は C である．

3.1 基本アルゴリズム

基本アルゴリズムにはブロック化された Right-Looking（外積）型ガウス消去法を用いている．計測が長時間に及ぶため，何度も試行してパラメータ選択を行なうことが許されない．そこでまず素朴な軸選択のない形でアルゴリズムを記述し，次数が数 10 万の規模でのこのアルゴリズムでの演算量，通信時間，並列度を考察した．これを本節と次節に述べる．

次数 n の行列 A を，ブロックサイズ b の小行列 A_{IJ}（ただし $I = 1:N$，$J = 1:N$ で $N = n/b$ と割り切れるものとする）に分割した場合，LU 分解は，

$$U_{IJ} = L_{II}^{-1}\left(A_{IJ} - \sum_{K=1}^{I-1} L_{IK}U_{KJ}\right), \quad (I \le J)$$

$$L_{IJ} = \left(A_{IJ} - \sum_{K=1}^{J-1} L_{IK}U_{KJ}\right)U_{JJ}^{-1}, \quad (I \ge J)$$

と書き表すことができるが，外積型では添え字 K を最

外側に用いたループ構成をとる[6]．軸選択を省略した形で述べると，まず $A_{11} \to L_{11}U_{11}$ と主小行列を LU 分解し，その下に位置する小行列を $A_{2:N,1} \to L_{2:N,1}$ に変換する（U_{11} を用いた代入計算）．次に A_{11} の右に位置する小行列を $A_{1,2:N} \to U_{1,2:N}$ に変換する（L_{11} を用いた代入計算）．$L_{2:N,1}$ と $U_{1,2:N}$ が求まると，右下の正方形部分を階数 b 更新によって $A_{2:N,2:N} \to A^{(1)}_{2:N,2:N}$ と更新するが，この操作は元の問題を次数 n から $n-b$ に消去したことに相当する．したがって $A^{(1)}_{2:N,2:N}$ を改めて縮小された問題と見做して同じ操作を繰り返すことで LU 分解を完結する．

HPL では小行列の行列演算は BLAS（Basic Linear Algebra Subprograms）インターフェースのライブラリで実行可能に書かれている．上記のアルゴリズム中，$A_{1,2:N} \to U_{1,2:N}$ は TRSM（Triangular-matrix Solve with Multiple columns）で，階数 b 更新は行列行列積和 GEMM（General Matrix Matrix multiplication）で行なうことができるが，これらは BLAS-3 の，キャッシュブロック化された高速なライブラリで実行することができる．なお，$A_{2:N,1} \to L_{2:N,1}$ は軸選択のために TRSM は使えず，BLAS-1 と 2 のルーチンを用いる．

浮動小数点演算量は TRSM が $n(b-1)(n-b)/2$，GEMM が $n(n-b)(2n-b)/3$ になるので，問題が大きくなると大部分の演算が GEMM に集中する．たとえば b が 200 程度で n が 60 万では，$599800^2/600000^2 = 99.93\%$ に達する（総演算量を $2n^3/3$ とした）．

3.2 並列化アルゴリズム

係数行列 A を 2 次元ブロックサイクリック分割する．プロセスグリッドを $p \times q$ で表す．図 3 に 12 ウェイで $p \times q = 3 \times 4$ とした場合を示した．HPL では 12 ウェイを 2 次元プロセスグリッドにマップする場合，00，10，... の順にとる Row Major と，00，01，... の順にとる Column Major の 2 通りが選択可能である．P-32 は 2 ウェイのノードなので，Row Major 順として，軸選択とそれに伴う行交換で通信が多い方向にノード内並列を利用した．図 3 で小行列上に記した 2 桁の数字が，その小行列を所有するプロセス番号である．斜線を付した 9 つの小行列はプロセス 11 が所有する A_{11}，$A_{41},\ldots,A_{79}$ であるが，これらの小行列は図右に示したように，各プロセスのメモリ上（ローカルの視野）では A_{11} の 1 列目に続いて A_{41} の 1 列目が置かれているので，ブロックサイズ $3b$ の正方行列として扱うことができる．

HPL では各段で消去する小行列 $A^{(K-1)}_{K:N,K}$ をパネルと呼び，それをワークエリアに置いて計算を進める．パネルの分解 $A_{11} \to L_{11}U_{11}$ と $A_{2:N,1} \to L_{2:N,1}$ には p ウェイの並列度があり，これが完了するとパネル分解を担当したプロセスは $L_{K:N,K}$ をプロセスグリッドの横方向に Bcast する．L_{KK} を受け取った q プロセスは $U_{K,K+1:N} = L^{-1}_{KK}A^{(K-1)}_{K,K+1:N}$ を計算して，これをプロセスグリッドの縦方向に Bcast する．図 4 に $K=1$ でプロセス 11 が L_{10}, L_{40}, L_{70} と U_{01}, U_{05}, U_{09} を受け取って

$$\begin{pmatrix} A_{11} & A_{15} & A_{19} \\ A_{41} & A_{45} & A_{49} \\ A_{71} & A_{75} & A_{79} \end{pmatrix} - \begin{pmatrix} L_{10} \\ L_{40} \\ L_{70} \end{pmatrix} \begin{pmatrix} U_{01} & U_{05} & U_{09} \end{pmatrix}$$

の階数 b 更新を行なう様子をグローバルの視野で示した．この更新計算には pq ウェイの並列度がある．2 つの Bcast の通信に要する理論的な時間はそれぞれ，$(n-b)^2\lceil \log_2 p \rceil/2q$ と $(n-b)^2\lceil \log_2 q \rceil/2p$ と考えられる．HPL ではこの Bcast を単一ノード間の通信（Secd/Recv）を使って実装しており，Increasing-1ring，Increasing-2ring，Bandwidth-reducing の 3 種類と，それぞれに対し次のパネルの分解を担当するプロセスを Bcast グループから除外する Modified 版の合計 6 種類の Bcast トポロジから選択できる．P-32 の計測では p や q が大きいので，2ring-Modified を用いた．

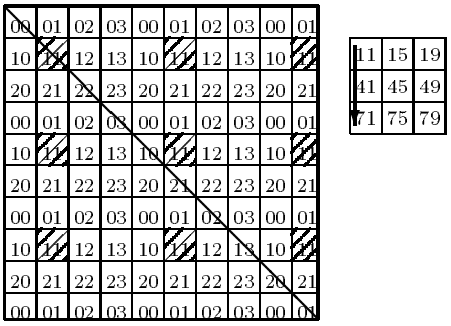

図 3 2 次元ブロックサイクリック分割とプロセス 11 のもつ小行列
Fig. 3 2-dimensional block-cyclic array decomposition and submatrices owned by process 11

演算量の大部分を占める階数 b 更新に着目すると，演算量が $O(n^3)$ で並列度が pq であるの対し，通信時間は $O(n^2)$ で並列度が p または q である．したがって HPL の解く問題が大規模になればなるほど計算速度は向上する．またブロックサイズ b は，n を固定して考えれば，大きくすると通信が速くなり，小さくすると計算が速くなる傾向にある．ただしこれはここに述べた素朴な並列ブロック化アルゴリズムでの検討であって，次節に述べる HPL のパネル内部のブロック化や，BLAS ライブラリの実装で必ずしもこの傾向から外れるパラメータ設定もある．

3.3 HPL のパラメータ

HPL は上で述べたブロックサイズ b を配列分割に用いるが，BLAS ライブラリの GEMM や TRSM に渡される時は，複数の小行列が接続して大きなサイズになっている．ライブラリは改めて実行環境に適切なブロックサイズを自身で選択できる．HPL では，生成された行列から L や U を作り，これを後続の計算で使用する際，L^t や U^t に転置するオプションが

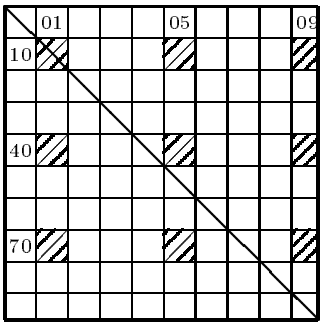

図 4 行列 A_{IJ} の更新計算
Fig. 4 Update computation for A_{IJ}

用意されている．また，Panel Factorization のアルゴリズム自身がさらに再帰的にブロック化されており，可変サイズのサブブロックサイズを使用している．Panel Factorization のアルゴリズムは Left-Looking，Right-Looking，クラウト法から選択でき，サブブロック化のアルゴリズムも同様に 3 つの中から選ぶことができる．サブブロックサイズを規定するパラメータは NDIV と NBMIN である．

以上はパネル内部の分解方法に関するものであるが，LU 分解ではパネルを分解している間は，それを担当している p 個のプロセスだけが仕事を行なっていて，他の $q(p-1)$ 個のプロセスは待っている．これが並列化を阻害するので，先に述べた Bcast トポロジの Modified オプションのように，次のパネル分解を行うプロセスの負荷を軽減する方策が用意されている．その 1 つにパネルの先読み段数（Look-ahead depth）があり，次のパネル分解を行うプロセスは階数 b 更新を $(n-b)\times(n-b)$ の全小行列に対して行なうのではなく，次のパネル分の $(n-b)\times b$ に対してだけ行い，先にパネル分解に入って，パネル分解後に残りの更新を行なう．これは先読み段数が 1 に設定された場合であり，2 やそれ以上にも設定することができる．

軸選択に対応する行交換の通信方法では，Binary-exchange，Spread-roll，Mixed から選択する．

4. HPL の計測

HPL を大規模システムで実行する前に，いつでも利用可能な（例えば 64 ノード程度の）環境で HPL 自身，また性能の分析に有効な関連計算を実施しておくことは，性能の予測には必須である．

4.1 BLAS ライブラリの選択

BLAS ライブラリは ATLAS と Goto ライブラリ（libgoto_opt64-r0.93）を試したが，後者がやや速かったのでこれを使用した[7]．コンパイラは gcc 3.2.2 を用いた．単一ノードで Goto ライブラリの DGEMM を測定すると，Opteron プロセッサのピーク性能の 85%の性能である 6.8Gflop/s が得られた．

表 2 HPL パラメータと結果
Table 2 HPL Parameters and Results.

HPL パラメータ	値	意味 (単位)
N	678912	行列の次数 n
NB	204	ブロックサイズ b
P,Q	32,64	プロセスグリッド
L1,U	T,T	転置，転置
PFACT	L	Left-looking
RFACT	L	Left-looking
NDIV	2	サブブロック化
NBMIN	2	サブブロック化
Btop	3	2ringM
R_{max}	6,155	Gflop/s
R_{max}/R_{peak}	75.134	%

4.2 HPL のパラメータチューニング

4.2.1 単一ノードでの計測

$p\times q = 1\times 2$ として，単一ノードで可能な限り大きな n について HPL を実行すると，Opteron プロセッサのピーク性能の 82.5%の性能である 6.6Gflop/s が得られた．

4.2.2 MPI 並列での計測

行列行列積を計算する MPI プログラムを Goto ライブラリの GEMM ルーチンを使用して計測し，単一ノードに通信が加わった場合の性能の影響を知ることができる．また，HPL の擬似乱数による行列生成をフランク行列に変更して，軸選択に伴う行交換の発生しない問題を解くことで，行交換に要する通信時間を知ることができる．

4.2.3 最適なパラメータの探索

これらの傾向から HPL のパラメータと性能に関する大まかな傾向を掴めても，パラメータ空間が広く，なかなか最適に近いパラメータ設定は得にくい．そこで利用可能な最大のノード数を調べ，p と q は，$p=q$ にできるだけ近い，$p\le q$ を，p は行交換に Binary exchange を使用するので 2 のべきになるように選んだ．そして，HPL プログラムに途中経過時間を出力するステートメントを挿入し，測定のための予約マシンタイムのはじめに数ブロックの消去だけを行なうジョブを繰返し実行して，n，b，Bcast トポロジ，先読み段数の順序でパラメータサーチを行なった．こうして選択されたパラメータから，負荷が均等になるように n を微調整（n が $b\times p$ で，$n+1$ が $b\times q$ で割り切れるのが理想）した．

4.3 並列計算の測定結果と OS の NUMA 機能

4.3.1 並列計算の測定結果

Adaptec 社のシリアル ATA ドライバの都合で，6GB のメモリのうち 4GB しか利用できない環境でまず測定を実施した．使用したパラメータと測定結果を表 2 に示す．

64 ビットアドレス対応のドライバ導入後の 6GB のメモリのすべてが利用できる状態では，問題のサイズ

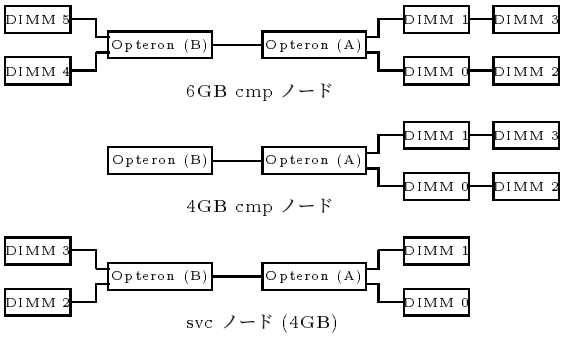

図 5 6GB 構成と 2 種類の 4GB 構成
Fig. 5 6GB and two types of 4GB configuration.

を $\sqrt{1.5}$ 倍にできるので，数パーセントの性能向上が期待できる．しかし並列計算の計測では 4GB の時の性能を上回ることはできなかった．これは Linux のカーネルが NUMA 対応になっていないため，プロセスの性能にバラツキがでて，並列計算ではその最も遅いプロセスに足を引きずられる形になって，問題サイズによる性能向上分を帳消しにしていたためと考えられる．以下，その分析について述べる．

4.3.2 OS の NUMA 認知機能

図 5 に今回テストした 3 種類のノードを示した．上図は 1024 台の計算ノード（cmp）と 16 台のスペア（spr）ノードの構成で，図 2 に示した状態である．中図は 6 個の DIMM のうちアドレスが上位の 2 つが使用できない状態のノードである．この場合 Opteron(A) からは全メモリがローカル，Opteron(B) からは全メモリがリモートである．下図は 16 台のサービスノード（svc）で，4GB のメモリを 2GB ずつ対称に装備している．これらの環境で $p \times q = 1 \times 2$ として，単一ノードで n を変更して HPL を実行した結果を図 6 に示した．“cmp 4GB” で示した計測値は $n = 5000$ から 21000 でメモリ不足が原因で性能が降下しはじめるまでほとんど揺れずに一定のカーブの上に載っている．これに対して “cmp 6GB” と “svc 4GB” では 5%程度の性能の揺れが見られる．この原因は HPL プログラムがブロックサイズごとの消去のループの中で，パネルのためのワークエリアを malloc で獲得しているが，これがローカルに獲得される場合とリモートに獲得される場合とが混在しているためと考えられる．HPL のメモリに対する要求は次のようになっている．

```
malloc (A)
matrix generation on A
timer start
Solve Ax=b start
   for(j=0; j<N; j+=nb){
    malloc (Panel)
    Panel Factorize
    Update by Panel
    free (Panel)
   }
Solve end
timer stop
```

問題サイズ n が例えば 20000 の場合，1 つのノードで実行される 2 つの HPL プログラムは $(20,000^2 \times 8/2$ より，それぞれ 1.6GB を行列 A のために malloc する．Linux の OS はすでに 1GB ほどのメモリを使用しているので，これ以上のメモリ要求は 4GB の実メモリの環境では性能に影響を与える．4GB の cmp では Opteron(A) からは常にローカル，Opteron(B) からは常にリモートのメモリを使うので，性能に揺れは生じない．6GB の cmp と svc では，最もアクセス頻度の高いパネル領域が，どちらのプロセッサからもローカルに取られるときが最も速く，どちらからもリモートに取られると最も遅く，それ以外の場合はその中間にある．ループの中で malloc と free を繰り返すので，性能に揺らぎが生ずる．6GB の cmp で 4GB を越えると揺らぎが小さくなるのは，パネル領域が DIMM 4 と DIMM 5 に取られる確率が高くなっているためと考えられる．

Linux には NUMA に対応する，ページ要求を発行したプロセッサのローカルに実メモリを取るメモリ・アフィニティ機能を持つものがある．これは BIOS の ACPI Static Resource Allocation Table（SRAT）をイネイブルにすることで，OS にどのメモリがどのプロセッサにローカル／リモートであるかを教えることを通して実現される[9]．上記の分析内容を確かめる目的で，NUMA を認知するカーネル 2.4.19-249 により $n = 21000$ から 25000 までを 1000 ずつ増加させながら 4 回ずつ同様の計測を行なった結果，性能の揺れは 3 桁目以降にしか現れなかった．例えば $n = 24000$ では 6.513×2 回，6.574，6.575Gflop/s であり，$n = 25000$ では 4 回とも 6.580Gflop/s であった．

HPL はブロック化アルゴリズムを採用しており，演算の大部分はキャッシュブロック化が実現された BLAS-3 ライブラリで処理されるため，メモリ性能の差がジョブ全体の性能にあまり大きな影響を与えない．しかし他のベンチマーク問題，たとえば SPEC CFP2000 Rate ではこの差はより顕著に現れるものがある．さらにメモリ性能の差がはっきり現れるものに STREAM ベンチマークがある．IBM eServer 325 の 2 スレッドで綿密は計測を行なった結果が公開されている[9]．これによると NUMA を認知する Linux と，認知しない SMP 対応のみの Linux の比較が，7GB/s 対 3.8GB/s という大きな開きになっている．

5. おわりに

Linpack ベンチマークは “Linpack 100” に始まり，次数 1000 の問題をプログラムの改変を認める “Towards Peak Perforamnce（TPP）” と呼ばれる仕様が追加され，HPL は 3 代目にあたる．ベンチマークとして常にその時代の科学技術計算のマシンの性能を

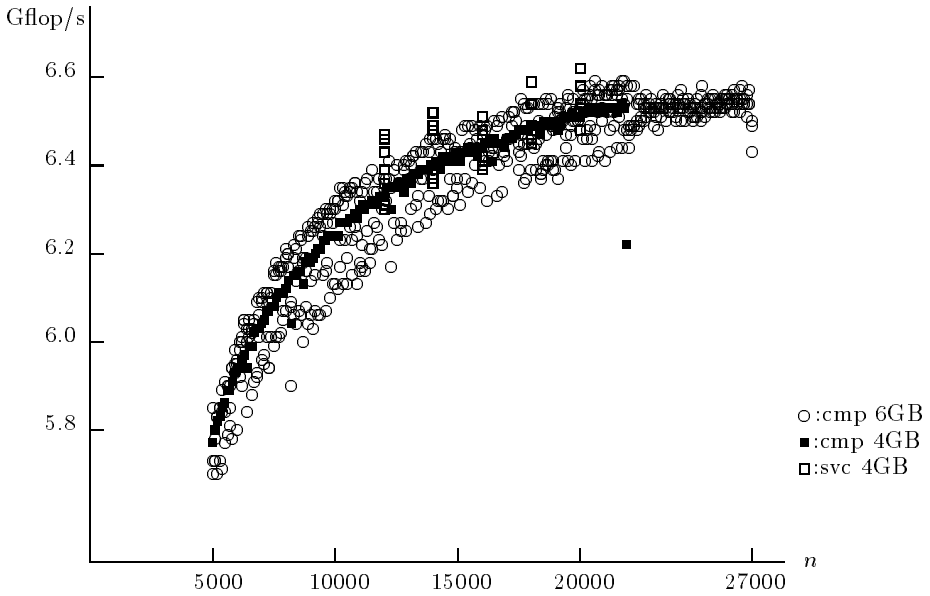

図 6 3 種類のシステムでの 1 ノードでの HPL の測定結果
Fig. 6 Single node HPL Performance Comparison on three systems.

評価する役割を担ってきたが，現実のアプリケーションの一部として実際に使用されるか，という観点では，徐々にそのような用途から遠ざかっているように思われる．たとえば TPP では 1000 × 1000 の行列を境界要素法のアプリケーションでそのまま利用できるようなライブラリが，そのままベンチマークで使用されていた．現在ではアプリケーションの採用するアルゴリズムもより問題に特化した形をとり，境界要素法で次数が数 10 万の問題を解く場合は，多重極展開（FMM）によるアプローチをとるであろう．HPL 自身にも，問題サイズを自由に設定できるとか，1 本の右辺ベクトルを LU 分解と同時に消去するとか，分解された三角行列を転置したフォーマットで格納できるというようなオプションが許されており，数値計算ライブラリとして見ると，汎用的な用途から離れつつある．その反面，大規模な並列計算機の安定稼動や性能の検証の目的には重要な役割を担っている．今回，Linux の NUMA 対応の性能への影響を見つけることができたことはその好例といえるであろう．HPL が MPI 通信の機能のうち，Send/Recv を中心としたプリミティブな機能しか使用していないのも，システムの初期の状態を考慮したものかもしれない．Linpack ベンチマークではかならずしも HPL の使用を限定してはいないので，そのシステムがサポートする MPI ライブラリの Bcast や非同期通信（Isend/Irecv）を使用するプログラムもあわせて計測できることを，今後の課題としてあげておきたい．

謝辞 NUMA 対応の Linux カーネルで HPL の計測していただいた IBM eServer xSeries Performance Development and Analysis の Douglas M. Pase 博士に感謝します．

参 考 文 献

1) http://unit.aist.go.jp/aist_j/press_release/pr2004/pr20040510/pr20040510.html
2) 石川 裕, 佐藤三久, 堀 敦史, 住元真司, 原田 浩, 高橋俊行: Linux で並列処理をしよう–SCore で作るスーパーコンピューター–, 共立出版 (2002).
3) IBM eServer 325, ftp://ftp.software.ibm.com/pc/pccbbs/pc_servers_pdf/e325spec.pdf.
4) AMD Opteron Processor Data Sheet, http://www.amd.com/us-en/assets/contact_type/white_papers_and_tech_docs/23932.pdf.
5) HPL Algorithm, http://www.netlib.org/benchmark/hpl/algorithm.html.
6) 寒川 光: RISC 超高速化プログラミング技法, 共立出版 (1995).
7) High-Performance BLAS by Kazushige Goto, http://www.cs.utexas.edu/users/flame/goto/.
8) 建部修見: LINPACK ベンチマーク, bit Vol.33,No.2, 共立出版 (2001).
9) Pase, D. : Performance of the IBM eServer 325 for Scientific and Technical Applications, ftp://ftp.pc.ibm.com/pub/special/serverperformance/e325_performance_032904.pdf.