広帯域ネットワークを用いたソフトウェア分散共有メモリの実現と性能評価

西田 晃[†,††]

コモディティハードウェア技術の進展に伴い，高速なネットワークで結合されたソフトウェア分散共有メモリ技術の実現性が高まっている．本研究では，大規模疎行列を対象とした反復解法のスケーラブルな並列実装を実現するため，InfiniBand と PCI Express を組み合わせた広帯域クラスタ環境を構築し，コモディティハードウェア技術の性能と実装上の問題点について検討するとともに，計算ノード内のメモリ帯域幅とノード間の通信帯域幅の確保に重点を置いた設計を行うことにより，疎行列線形演算に適したソフトウェア分散共有メモリの実現可能性について検討した．

Software Distributed Shared Memory with High Bandwidth Network: Production and Evaluation

Akira Nishida[†,††]

The recent development of commodity hardware technologies makes building a software distributed shared memory computing environment a more practical approach for scientific computing that requires the repetitive solutions of large linear systems. In this study, we build a tightly connected PC cluster with PCI Express and InfiniBand technology for the scalable implementation of parallel iterative linear solvers, evaluate the performance and bottlenecks of commodity hardware technologies. The scalability of the implementation of parallel sparse matrix computations considering both the local memory bandwidth of the compute nodes and their internode communication bandwidth is evaluated.

1. はじめに

コモディティハードウェア技術の進展に伴い，高速の PC をネットワークで結合し，仮想的な共有メモリ計算機として用いるソフトウェア分散共有メモリ技術の実現性が高まっている[2),6)]．しかしながら，高速な専用ネットワークを使用した共有メモリ型並列計算機等と比較して，PC クラスタ上でのノード間通信では，通信帯域幅やレイテンシに関するハードウェア上の制約から，十分なスケーラビリティが得られない場合が多い．

特に，大規模疎行列を扱う反復解法においては，間接参照を伴うベクトル間演算は計算量の大部分を占める重要な処理であるが，疎行列反復解法においては，このように大半の処理が内積を含むベクトル間演算，疎行列 - ベクトル間演算から構成されている．したがって，並列化に際しては，これらのベクトル演算が効率的に分散されなくてはならず，メモリ帯域幅と共に，大域的な通信を処理するための高性能な相互結合網が必要となる．本研究では，この問題について調べるため，InfiniBand と PCI Express を組み合わせた広帯域ネットワーク上にソフトウェア分散共有メモリ環境を構築し，大規模疎行列を対象とする反復解法をコモディティハードウェア上に実装する上で障害となるボトルネックについて考察した．

† 東京大学大学院情報理工学系研究科コンピュータ科学専攻
Department of Computer Science, the University of Tokyo
†† 科学技術振興機構 CREST
CREST, JST

2. 背景

PCI Express は従来の PCI バス技術と互換性を持つ次世代のシリアル転送インタフェース規格であり，Intel, NEC 等により 2004 年より実用化されている．PCI バスが 1GB/s の帯域幅を上限とする共有バス方式であったのに対して，PCI Express ではデバイス間を直接接続することができ，また一方向 2.5Gb/s の帯域幅を持つレーンを最大 32 本まで束ねることことにより，双方向で最大 16GB/s の実効帯域幅を実現する．

PCI Express に現時点で対応している高速ネットワーク技術として，InfiniBand を挙げることができる．Mellanox Technologies 社の開発する InfiniHost ホストチャネルアダプタ (HCA) は，1 レーン当たり 5Gbps の帯域幅を持つ DDR InfiniBand を 4 本束ねたポートを持ち (図 1 参照)，8 レーンの PCI Express スロットを利用することにより，双方向で 40Gbps の帯域幅を実現している．

クラスタノードに用いる Opteron プロセッサは

図 1 Mellanox Technologies 社の PCI Express 対応 DDR InfiniHost HCA MHGS18-XT DDR
Fig. 1 InfiniHCA HCA MHGS18-XT DDR for PCI Express bus from Mellanox Technologies, Inc.

AMD 社の開発するマルチプロセッシング対応の 64bit プロセッサである. チップ内にメモリコントローラを内蔵しているため, メモリレイテンシが小さく, メモリ帯域幅がプロセッサ数に比例して増大する点に特徴がある. また, 実験に用いたすべてのノードについて, カーネル 2.6.11 を採用した SuSE Linux 9.3 を用いた. 2.6 以降の Linux カーネルはメモリアフィニティを実装しており, 複数の Opteron プロセッサのメモリ帯域幅を引き出すことができる.

以上の技術的背景から, 本研究では管理コストを考慮して, 4-way 構成の Opteron サーバを計算ノードとして 4 台を接続したクラスタ IBQ を構築するとともに, 相互結合網として Mellanox Technologies 社の 8x PCI Express 対応 single port DDR InfiniHost HCA を用いてクラスタ環境を構築することとし, PCI Express バスの高速性を利用してローカルメモリを省略することにより低コスト化を実現した MHGS18-XT DDR を導入して評価を行った. なおスイッチには 24 ポートの DDR InfiniBand スイッチである MTS2400 DDR を使用した.

2-way サーバをノードとした 8 台構成のクラスタ IBD での評価結果[9] から, 疎行列処理に関して十分なスケーラビリティを得るためには, 少なくとも 1CPU につき 1.5GB/s 以上の双方向帯域幅が必要であることが分かっている. 現状では PCI Express x8 に対応した DDR InfiniBand HCA しかないため, 4-way サーバをノードとして用いた場合に十分なスケーラビリティを得るためには, PCI Express x8 以上の帯域幅を持つスロットを 2 個以上サポートするマザーボードを用いる必要がある. この条件を満たすマザーボードは現状では少ないが, NVIDIA 社の nForce Professional チップセットを用いた Uniwide (Appro), Iwill 社のベアボーンサーバ Uniserver 3346 は, 下のシステム構成に示すように 2 基の PCI Express x16 スロットを持つ. そこで, 本研究ではこれに 2.2GHz, 1MB キャッシュ搭載の Opteron 848 を搭載して, 4 ノード 16 プロセッサからなるクラスタを作成し, 実験に使用した. 図 2 にクラスタノードのシステム構成を示す. なおメモリにはノードあたり 512MB PC3200 DDR SDRAM (ECC Registered) 8 枚を使用した.

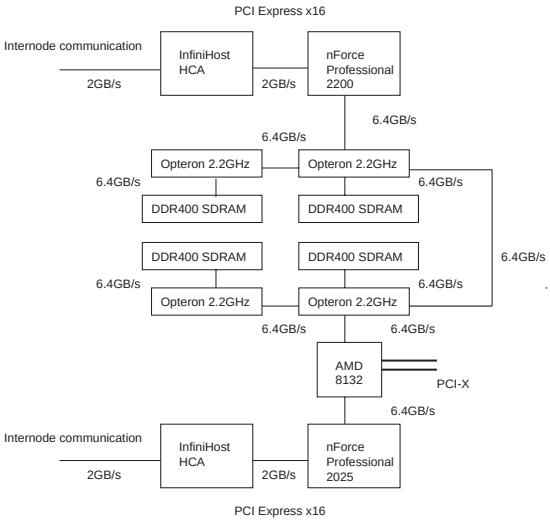

図 2 quad Opteron クラスタノードのシステム構成
Fig. 2 System Diagram of dual Opteron Cluster Node.

図 3 InfiniBand スイッチ MTS2400 DDR で接続された Opteron クラスタ IBQ
Fig. 3 Opteron Cluster IBQ, connected with InfiniBand switch MTS2400 DDR.

本研究では, 平成 17 年 10 月より筑波大学計算科学研究センターの協力を得て, MPI を通信レイヤに用いたソフトウェア分散共有メモリ環境向け OpenMP コンパイラ SCASH-MPI の評価を行っている. ここでは, 今回構築した広帯域クラスタ環境を用いて, InfiniBand ネッ

トワーク上でのソフトウェア分散共有メモリ環境の性能を測定した.

3. 性能評価

SCASH-MPI は, 既存の MPI ライブラリを用いてソフトウェア分散共有メモリ環境を構築することができ, 移植性に優れている. ここでは, 帯域幅とスケーラビリティとの関係を調べるため, InfiniBand に対応した MPI ライブラリである MVAPICH[4] と組み合わせて性能を評価することとした. MVAPICH には通信をモニタして必要に応じて分割し, 複数のカードに分配するストライピング機能が実装されている[5]. 本研究では MVAPICH の最新版である 0.9.6 を使用した. ただし, 現在のところ MHGS-18XT は, 下に示すように Uniserver 3346 との組合せで 2GB/s 弱の性能しか出ていない☆. また, DDR InfiniBand ドライバは現状では Opteron プロセッサ上でのストライピング機能に対応していないため, ここではノードあたり 1 枚の HCA のみを用いて評価を行った.

図 4-5 に MVAPICH 0.9.6 の Opteron/MHGS-18XT クラスタ上での MPI レイテンシと帯域幅を示す. なお, ノード内での MPI レイテンシはノード間に比べて低くなっているが, これは共有メモリ上での MPI 通信の実装によるものである☆☆[3].

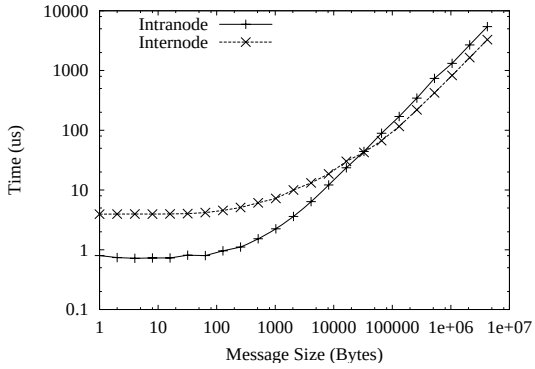

図 4 MVAPICH 0.9.6 の MPI レイテンシ (ノード間は DDR InfiniBand HCA (Mellanox MHGS18-XT) で接続)

Fig. 4 MPI Latency of MVAPICH 0.9.6 (nodes connected with DDR InfiniHost HCA (Mellanox MHGS18-XT)).

3.1 STREAM benchmark

次に, ベクトル演算で重要となる局所的なメモリ帯域幅について, STREAM benchmark[1] を用いて測定

☆ この原因は Uniwide 社にて調査中である.

☆☆ なお, LAM, MPICH もノード内の帯域幅については同程度であった. 現状に関しても, ノード間の帯域幅がノード内に比べて相対的に大きい環境であるといえる.

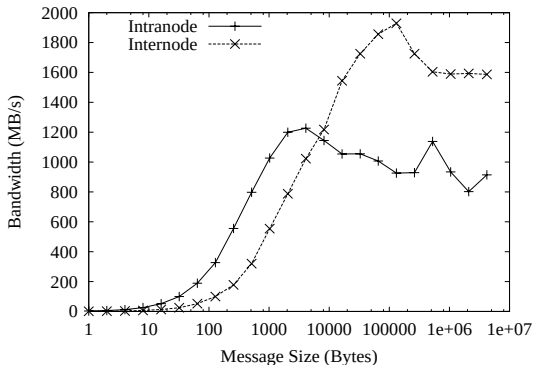

図 5 MVAPICH 0.9.6 の双方向帯域幅 (ノード間は DDR InfiniBand HCA (Mellanox MHGS18-XT) で接続)

Fig. 5 MPI Bidirectional Bandwidth of MVAPICH 0.9.6 (nodes connected with DDR InfiniHost HCA (Mellanox MHGS18-XT)).

した. このベンチマークプログラムでは, 倍精度浮動小数ベクトル間で簡単な演算を行い, 実測値をもとに計算機の実効帯域幅を評価する. 表 1, 図 6に STREAM benchmark の計算内容と主要ループ部分を示す.

```
  /* - MAIN LOOP - repeat NTIMES times - */
  scalar = 3.0;
  for (k=0; k<NTIMES; k++)
    {
  times[0][k] = second();
#pragma omp parallel for
  for (j=0; j<N; j++)
    c[j] = a[j];
  times[0][k] = second() - times[0][k];
  times[1][k] = second();
#pragma omp parallel for
  for (j=0; j<N; j++)
    b[j] = scalar*c[j];
  times[1][k] = second() - times[1][k];
  times[2][k] = second();
#pragma omp parallel for
  for (j=0; j<N; j++)
    c[j] = a[j]+b[j];
  times[2][k] = second() - times[2][k];
  times[3][k] = second();
#pragma omp parallel for
  for (j=0; j<N; j++)
    a[j] = b[j]+scalar*c[j];
  times[3][k] = second() - times[3][k];
    }
```

図 6 STREAM benchmark の主要ループ

Fig. 6 Main loops of STREAM benchmark.

なお, これらの演算は原則として通信が不要であるため, 先に述べたノード間通信性能はここでは影響しない. ここでは OpenMP 版の並列プログラム `stream_d_omp.c` を用い, 複数のネットワーク構成で評価した. まず, 図 7にノード当たり最大 4MPI プロセスで実行した場合の

表 1 STREAM benchmark の計算内容
Table 1 STREAM benchmark types.

Benchmark	Operation	Bytes per iteration
Copy	a[i] = b[i]	16
Scale	a[i] = q * b[i]	16
Add	a[i] = b[i] + c[i]	24
Triad	a[i] = b[i] + q * c[i]	24

STREAM benchmark の台数効果を示す. 問題サイズはプロセス当たり 4,000,000, すなわち約 91.6MB である. 図から分かるように, 1 ノードあたり 3 プロセスを超えるノードがある場合には, スケーラビリティがまったく得られない. プロセッサあたりのメモリ帯域幅は十分であることから, これは SCASH-MPI では MPI プロセスごとに 1 スレッドを通信処理用に割り当てている[8] ためであることが分かる. したがって, 実際には 1 プロセスあたり 2 プロセッサを割り当てて評価を行う必要がある. また, この場合図 8 の性能を得るものの, これは図 9に示す GbE 上での結果とほとんど差がない. GbE 上での MPI レイテンシは最小で 24.27us, 双方向帯域幅は最大で 226MB/s であり, InfiniBand と比較するとかなり低い性能である. したがって, STREAM での性能低下の原因は, 帯域幅や通信レイテンシではなく, バリア同期等の前処理によるオーバヘッドであることを示している. これは, 図 6に示すようにアルゴリズム中の各 for ループに暗黙のバリア同期が含まれていることによっており, これを除去することで, 図 10の性能を得ることができている.

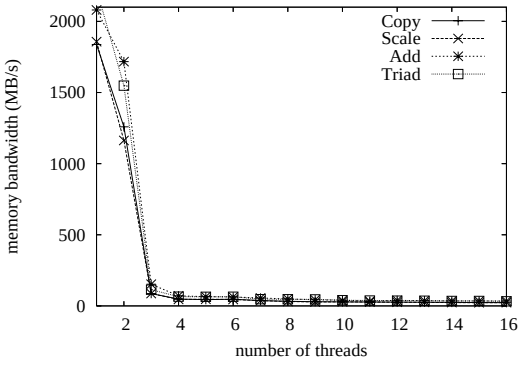

図 7 ノード当たり最大 4MPI プロセスで実行した場合の STREAM benchmark の台数効果
Fig. 7 Speedup of STREAM benckmark results with up to four processes per node.

4. NAS Parallel Benchmark

さらに, 疎行列処理とクラスタの通信性能との関係を調べるため, NAS Parallel Benchmark[7] 3.2 のうち,

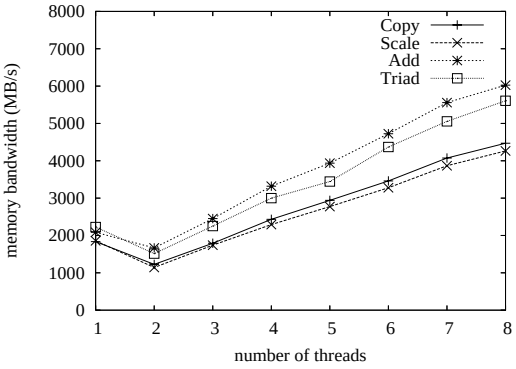

図 8 ノード当たり最大 2MPI プロセスで実行した場合の STREAM benchmark の台数効果
Fig. 8 Speedup of STREAM benckmark results with up to two processes per node.

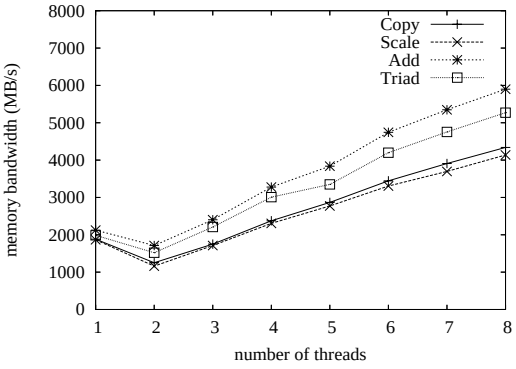

図 9 ノード当たり最大 2MPI プロセスで実行した場合の STREAM benchmark の台数効果 (GbE 上)
Fig. 9 Speedup of STREAM benckmark results with up to two processes per node on GbE.

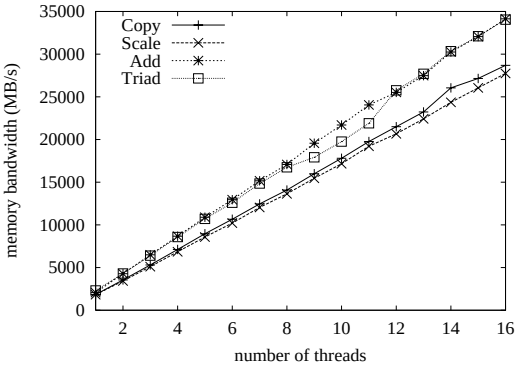

図 10 配列のマッピングを指定するとともに各ループのバリア同期を除去し, ノード当たり 4MPI プロセスで実行した場合の STREAM benchmark の台数効果
Fig. 10 Speedup of STREAM benckmark results with up to four processes per node (Mapping specified and barrier synchronization removed).

本クラスタ上で実行可能な CG kernel の Class S から B までを用いて評価を行った.

```
vector r,p,q,z; /* working vector */
/* solve A*x */
real CGSOLVE(matrix A,vector x, real zeta) {
  int it; real alpha, beta, rho, rho0;
  r = x; p = x; z = 0.0; rho = x * x;
  for(it=0; it < NITCG; it++) {
    q = A*p;
    alpha = rho/(p * q);
    z += alpha * p;
    rho0 = rho;
    r += -alpha * q;
    rho = r*r;
    beta = rho / rho0;
    p += beta*r;
  }
  r = A*z - x;
  zeta = 1.0/x*z;
  x = (1.0/sqrt(z*z))*z;
  return sqrt(r*r);
}
```

図 11 NPB CG 主要ループのアルゴリズム
Fig. 11 Algorithm of NPB CG main loop.

CG kernel では, 対称正定値疎行列の最小固有値を逆反復法と共役勾配法により計算する. 図 11に主要ループのアルゴリズムを示す. ここでは, 行列は行方向に分割され, 通信はベクトルデータについて行われる. ベクトルデータの処理が主であるため, メモリ帯域幅を要する点が特徴である. それぞれのクラスでの元数は S=1,400, W=7,000, A=14,000, B=75,000, C=150,000, また行毎の非零要素数は S=7, W=8, A=11, B=13, C=15 である. なお, このサイズでは送受信される部分ベクトルの大きさは 16 プロセス時で最大 75KB である. 帯域幅の性能に与える影響を調べるため, まず MPI 版 NPB CG kernel の計算性能を DDR InfiniHost HCA を使用した場合について調べる.

CG kernel に関しては, 十分な帯域幅のある環境ではほぼ完全なスケーラビリティが得られることが分かっている[9]. 図 12 に MPI 版 CG kernel の IBD 上での実行結果を示す.

4-way クラスタ上での結果を図 13に示すが, DDR InfiniBand 上では, 2 ポートの SDR InfiniBand ポートを用いた場合に Class C においてほぼ完全なスケーラビリティが得られているのに対して, 若干スケーラビリティに劣っていることが分かる.

次に, 反復ベクトルに関して MPI 版と同様のマッピングを指定し, 1 ノード 2MPI プロセスまで割り当てた場合の OpenMP 版 NPB CG の演算性能を図 14に示す. NPB BT, SP においてはスケールするすることが分かっている[8] が, 図から分かるように, CG においては十分な効果が得られていない. MPI 版の結果から,

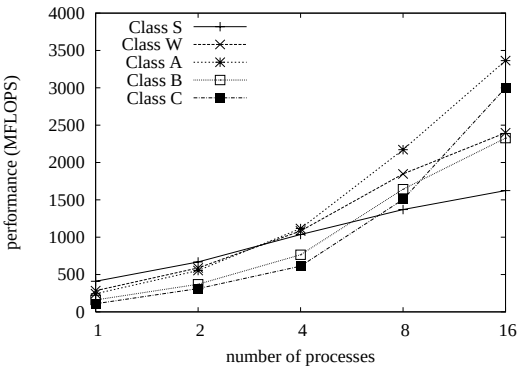

図 12 SDR InfiniHost HCA 上の 2 ポートを使用した場合の NPB CG の演算性能
Fig. 12 Speedup of NPB CG with two ports of SDR InfiniHost HCA.

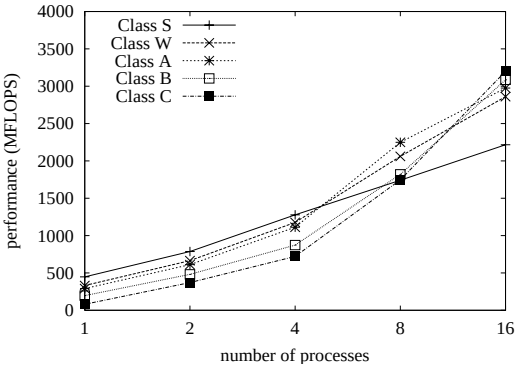

図 13 DDR InfiniHost HCA で接続したクラスタ上での NPB CG (MPI 版) の演算性能
Fig. 13 Speedup of NPB CG (MPI version) with DDR InfiniHost HCA.

ネットワーク帯域幅, 局所メモリ帯域幅に関しては十分な能力がクラスタにあることが分かっているので, CG で性能が低下する原因は, バリア同期等の前処理に必要なコストが他のカーネルに比べて大きいことにあると考えられる.

5. ま と め

本稿では, コモディティハードウェアによる計算環境として PCI Express 対応の InfiniBand 技術を用いた広帯域クラスタ環境を構築し, その性能を評価するとともに, ソフトウェア分散共有メモリ技術の可能性と問題点について検討した. その結果, ノード間の通信帯域幅及びノード内のメモリ帯域幅の確保に重点を置いた計算環境を構築することにより, 4-way サーバを利用したクラスタ環境においても, 疎行列を対象とした反復解法に関してスケーラブルな計算性能を実現できることが分

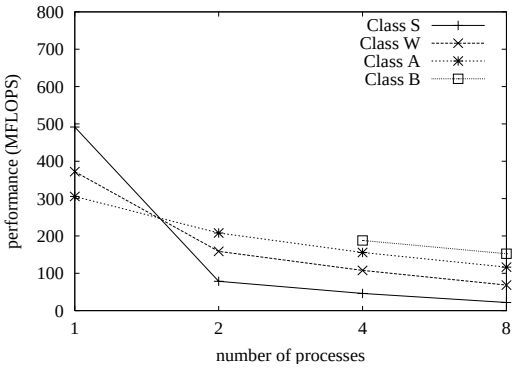

図 14 DDR InfiniHost HCA で接続したクラスタ上での NPB CG (OpenMP 版) の演算性能. 1 ノード 2MPI プロセスまで割り当てた場合

Fig. 14 Speedup of NPB CG (OpenMP version) with DDR InfiniHost HCA, up to two processes per node.

かったが, 一方ソフトウェア分散共有メモリにおいては, ノード内での処理オーバヘッドを一層低減する必要があることが評価結果から明らかとなった.

謝辞 本研究を進めるに当たり, ご議論を頂いた小柳義夫教授, 小柳研究室諸氏に感謝の意を表します. InfiniBand の性能評価にあたっては, Mellanox Technologies, Inc., 株式会社アルティマより, またクラスタノードに関してはアロシステム株式会社, 株式会社リオワークスより多くのサポートを頂きました. なお, 本研究の一部は, 科学研究費補助金特定領域研究 16016225「分散共有メモリクラスタを用いた疎行列線形代数演算ライブラリの効率的な実装技術」, 及び科学技術振興事業団戦略的創造研究推進事業「大規模シミュレーション向け基盤ソフトウェアの開発」プロジェクトの支援によるものである.

参 考 文 献

1) *STREAM: Sustainable Memory Bandwidth in High Performance Computers*, http://www.cs.virginia.edu/stream/.
2) Hennessy, J. I. and Patterson, D. A.: *Computer Architecture: A Quantitative Approach, Third Edition*, Morgan Kaufmann (2003).
3) Jin, H. W., Sur, S., Chai, L. and Panda, D. K.: LiMIC: Support for High-Performance MPI Intra-Node Communication on Linux Cluster, *Proceedings of the International Conference on Parallel Processing* (2005).
4) Liu, J., Chandrasekaran, B., J. Wu, W. J., Kini, S. P., Yu, W., Buntinas, D., Wyckoff, P. and Panda, D. K.: Performance Comparison of MPI implementations over Infiniband, Myrinet and Quadrics, *Proceedings of the International Conference on Supercomputing '03* (2003).
5) Liu, J., Vishnu, A. and Panda, D. K.: Building Multirail InfiniBand Clusters: MPI-Level Design and Performance Evaluation, *Proceedings of the International Conference on Supercomputing '04* (2004).
6) Pfister, G. F.: *In Search of Clusters*, Prentice-Hall, second edition (1998).
7) van der Wijngaart, R.: The NAS Parallel Benchmarks 2.4, Technical Report NAS-02-007, NASA (2002).
8) 小島好紀, 佐藤三久, 朴泰祐, 高橋大介: MPI を通信レイヤに用いるソフトウェア分散共有メモリシステム, 情報処理学会論文誌: コンピューティングシステム, Vol. 46, No. SIG 7, pp. 63–73 (2005).
9) 西田晃: InfiniBand クラスタを用いた疎行列線形代数演算ライブラリの効率的な実装技術, 情報処理学会研究報告, Vol. 2005, No. 81, pp. 97–102 (2005).