疑似グローバルクロックを用いた高精度実行時間測定

早川 潔†　岩根　雅彦†　関口 智嗣§

処理性能の高精度な測定は, より正確な評価を可能にし, 通信時間の隠蔽などといった並列処理の効率化を可能にする. 一般的に処理性能を測定するためには, 各ノードの実行開始時刻を正確に揃えなければならない.Beowulf クラスタでは, 各ノードの実行開始時刻を揃えるために,MPI などの通信ライブラリの Barrier 関数が用いられる. しかし,Beowulf クラスタに実装される Barrier では, ある程度の誤差が生じてしまう. そこで, 本稿では, 疑似グローバルクロックカウンタシステム (PGCCS) を用いた高精度実行時間測定システムを実装し, 性能評価を行った. その結果, 本システムは, 従来の方式に比べ少ない誤差で実行時間を測定できた.

Accurate Performance Measurement with The Pseudo Global Clock Counter System

Kiyoshi Hayakawa †, Masahiko Iwane † and Satoshi Sekiguchi §

Accurate performance analysis of collective communication is useful on performance evaluation and prediction of the Beowulf cluster systems. In order to measure execution time accurately, each node have to take the first step with execution by barrier. But it is difficult for each node to take the first step with execution each other, since it receives packets indicating barrier completion through Ethernet (i.e. MPI_Barrier) in different time. This paper describes the pseudo global clock counter system(PGCCS) that allows us to measure execution time accurately. Time difference errors of the PGCCS was smaller than that of conventional methods.

1　はじめに

汎用部品で構成された Beowulf クラスタシステムは, そのシステム構築が比較的安価でかつ容易なため, 注目を集めている. また, 市販マイクロプロセッサの性能が急激に向上しており, そのプロセッサを使用する Beowulf クラスタシステムはより高速な並列処理を可能にしている. よって, Beowulf クラスタでの通信性能をはじめとする種々の処理性能に関する研究 [2][3] は重要な研究の一つであり, 処理性能の高精度な測定は, より正確な評価を可能にする [5]. また, 処理時間を正確に測定し, その測定値を基にしてアプリケーション内の各処理の実行時間を予測することにより, 並列処理の効率化 (通信時間の隠蔽など) を計ることができる.

処理性能を測定するためには, 各ノードの実行開始時刻を正確に揃えなければならないが, ある程度の誤差が生じてしまう. 各ノードの実行開始時刻を揃えるために,MPI に代表される通信ライブラリの Barrier 関数 (MPI_Barrier) が用いられる. Beowulf クラスタにおける通信ライブラリは,Ethernet を使用して Barrier を行っている場合が多い.Ethernet を使用した Barrier 同期では, パケット転送時間や通信レイヤでの処理時間のオーバーヘッドが大きいため, ある程度の精度（数 100μs 程度）では揃えられるが, それ以上の精度は望めない.

そこで, 本稿では,PC をベースした Beowulf クラスタシステム（SCCB[1] -Cluster）における高精度実行時間測定システムとして疑似グローバルクロックカウンタシステム (PGCCS)[2] を提案する. 高精度な測定を可能にするため, 高速なバリア同期を可能にした SCC[3] ボードを Beowulf クラスタに搭載し, そのボード上に PGCCS を構築した. 性能評価として,PGCCS のオーバーヘッド, および SCCB-Cluster の collective 通信時間を PGCCS で測定し, 測定誤

†:九州工業大学　工学部 電気工学科
Department of Electronic and Computer engineering Kyushu Institute of Technology
§: 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology

[1] Sync-Comm Controller Beowulf
[2] Pseudo Global Clock Counter System
[3] Sync-Comm Controller

差を他の 2 方式と比較した.

2 SCCB-Clusterシステム

SCCB-ClusterはPCボード（PICMG規格のボード CPU：PentiumIII600MHz）を 100Base/TX の Switching HUB で結合したクラスタシステムである.OS は Linux (kernel version:2.2.15), 通信ライブラリは,MPICH (version 1.2.0) である. 各ノードには,SCC[1] と呼ばれる高速なバリア同期を可能にしたボードが実装されている. 各 SCC は Sync-Comm Network と呼ばれる同期・通信用ネットワークで結合されている.

2.1 SCCボード

SCC ボードは, メッセージパッシング型の通信処理および拡張型バリア同期処理を高速に行うための PCI ボードである [1]. 本稿では,SCC の同期処理機能と連携した PGCCS を構築する.

SCC ボードは,Control LSI および Sync-Comm Network 用コネクタ（Link_A および Link_B）のみで構成されている.Control LSI は, 同期・通信処理をハードウェアで実装することにより, 高速化を実現する.Sync-Comm Network 用コネクタは,Sync-Comm Network 用コネクタの Link_A と Link_B を接続する（図 1 参照）.

図 2 に Control LSI 内のブロック図を示す.Control LSI は,PCI バス-内部バスブリッジ（PCI BUS-Internal BUS Bridge）, 同期制御部（Sync Control）, 通信制御部（Comm Control）, 疑似グローバルクロック部 (Pseudo Global Clock Counter System), およびネットワーク制御部（Network Control）で構成される.

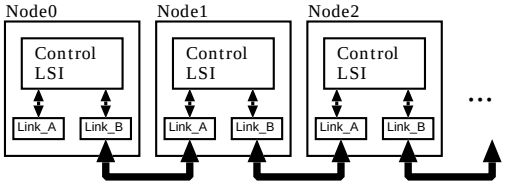

図 1: Sync-Comm ネットワーク

PCI バス・内部バスブリッジは,PCI バスプロトコルと内部バスプロトコルをインターフェースし,PCI のマスター／ターゲット機能をサポートする. 同期制御部は,Sync-Comm Network を使用した同期処理アクセスを行い, 任意参加バリア,Fuzzy バリアおよび RBC 同期機構 [1] の同期を処理する. 通信制御部は, パケットデータ処理を行う. 疑似グローバルクロック部は, 同期制御部と連携し, 高精度実行時間測定に必要な処理をハードウェアで行う. ネットワーク制御部は, 同期信号と通信信号とを Sync-Comm Network 上で融合させるための制御を行う.

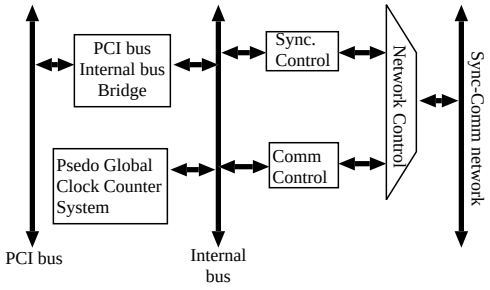

図 2: Control LSI 内のブロック図

2.2 Sync-Comm Networkでの同期信号

同期時における Sync-Comm Network の信号線は, 図 3 のように接続される. 同期時の信号線は, 最大 16 台で構成されるマイクロクラスタ内での同期信号線とそのマイクロクラスタ間での同期信号線に分かれる.

同期成立の検出は, まず, マイクロクラスタ内での同期成立を検出し, その後にマイクロクラスタ間での同期成立を検出する. マイクロクラスタ内では, Sync_n 信号 (n はマイクロクラスタ内でのノード番号) を使用し, マイクロクラスタを構成する各ノード間の同期成立を検出する.

Sync_n 信号は, マイクロクラスタ内ノード番号 n のノードが同期ポイントに到達していることを示す信号である. この信号線は, マイクロクラスタ内のみのバス接続である.

マイクロクラスタ間では,Divided 信号,Part-sync-comp 信号,All-sync-comp 信号を使用して, クラスタ間での同期成立を検出する. Divided 信号は, クラスタシステムがマイクロクラスタに分割されていること（ノード数が 17 台以上であること）を示す信号,Part-sync-comp 信号はマイクロクラスタ内の

同期が成立したことを示す信号,All-sync-comp 信号は, クラスタ全体の同期が成立したことを示す信号である.Divided 信号線,All-sync-comp 信号線は, クラスタ全体のバス接続である. Part-sync-comp 信号線は, デージチェーン接続である.

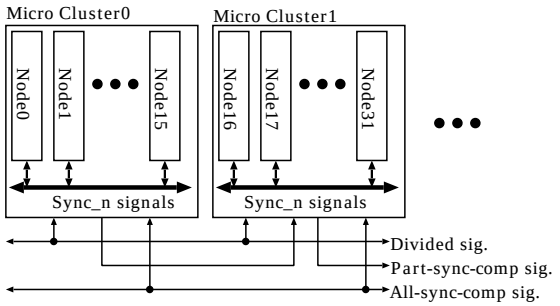

図 3: Sync-Comm Network の同期信号

2.3 同期制御部の機能

同期制御部は, 同期成立検出回路（synchronization completion detector）および RBC 回路で構成される（図 4 参照）.

マクロクラスタ内での任意参加バリアをサポートするために, 同期成立検出回路には, 同期グループレジスタ（sync-group reg）（16 ビット）が用意されている. また, 同期の成立を知らせるために, 同期フラグ（S-flag）が用意されている.

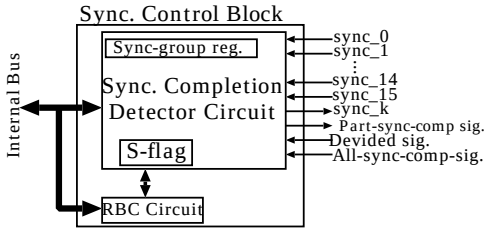

図 4: 同期制御部の構成 (Node 番号 k の場合)

3 PGCCS

一般的に, 実行時間測定には, 各ノードの CPU 内に搭載されているクロックカウンタを使用するが, それらのカウンタを使用した実行時間測定には, ある程度の誤差が生じてしまう. その誤差が生じる原因と解決手段となる PGCCS について言及する.

3.1 バリアを使用する実行時間測定

バリア同期関数として,1)Ethernet を利用した MPI_Barrier を実行した場合, 2)SCC を利用した SCC_Barrier を実行した場合, それぞれの測定誤差について述べる.

各ノードの実行開始時刻を揃えるために,MPI の Barrier 関数 (MPI_Barrier) を用いる場合,CPU がバリア同期の実行・クロック値の読み込みを行う.Ethernet を使用して Barrier を実行するため, CPU は PCI バスを介してバリア同期のためのパケット情報を Ethernet カードへ送り, 他のノードのバリア同期用パケット受信することにより, バリア同期の検出を行う. その後に CPU 内のクロック値を読み込む (図 5 参照).

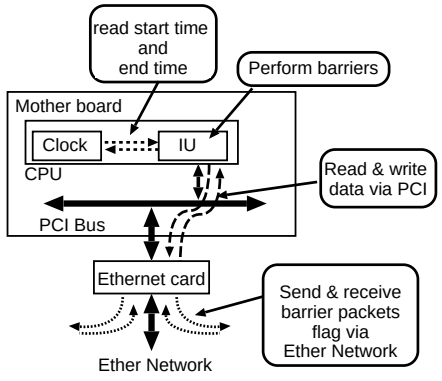

図 5: MPI_Barrier を利用した場合の動作

各ノードの実行開始時刻を揃えるために,SCC を利用する SCC_Barrier 関数を利用する場合, バリア同期処理は SCC が行い, カウンタの値の読み取りを CPU が行う. CPU は SCC がバリア同期の処理を終了したか否かをチェックし, 終了したら, カウンタの値を読み込む (図 6 参照）.

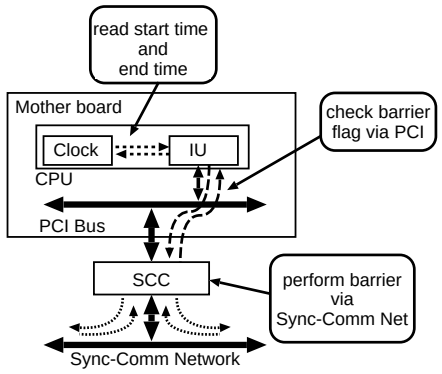

図 6: SCC の高速バリア同期を利用した場合の動作

3.2 CPU 内のカウンタを使用した測定方法での誤差

上記の 2 つ方法を使用して各ノードの実行開始時刻を揃えたとしても, ある程度誤差が生じてしまう. クラスタのような疎結合の場合, バリア同期に関する情報を通信するのに時間がかかり, 全ノードのバリアポイントに到達したことを検知する時刻が各ノード毎で異なってしまう. また, 同期が完了した後,CPU がクロックカウンタの値を読み込む処理において, 各ノードのシステム状態が異なるためその読み込み時刻も異なってしまう. よって, 各ノードのバリア命令終了時刻が異なってしまう（図 7 参照）.

これらの実行時間測定の誤差を解決するため,PGCCS を提案する.

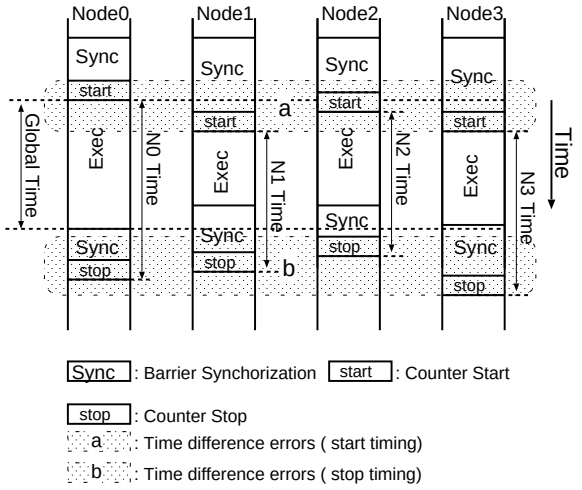

図 7: 同期終了時刻が異なることによる計測誤差

3.3 PGCCS の構成

PGCCS では, SCC はバリア同期およびクロックカウンタのスタート・ストップ・読み込みをすべて行う (図 8 参照).

図 9 に PGCCS の構成を示す. 疑似グローバルクロック部は,25MHz のクロック（OSC）, クロックカウンタ (32bit), タイムスタンプメモリおよびカウンタコントローラで構成されている. カウンタコントローラがバリア同期とカウントスタートおよびストップを連動して行う. つまり, カウンタコントローラが Synchronization Control 内の S-flag を常に監視し,S-flag がリセットされたら (同期が成立したら), カウンタをスタートさせたり, ストップさせる. タイミングシミュレーションでは, 同期が成立してからタイマスタート・ストップまで, 最大 240ns かかる. また, ユーザープログラムの要求にしたがって, カウンタの値をタイムスタンプメモリに格納したり, そのメモリからデータを読み出す.

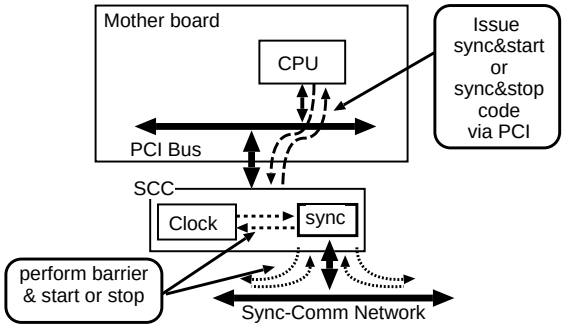

図 8: PGCCS を利用した場合の動作

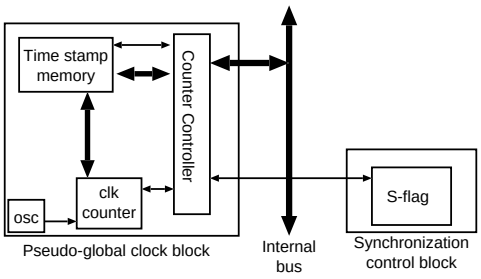

図 9: PGCCS の構成

PGCCS の場合, 各ノードの測定値には, ほとんど誤差はない[4] . しかし, 厳密には,Global クロックと PGCCS の値との間に誤差が生じてしまう. 誤差の原因の 1 つとして, カウンターをスタートさせた後の命令の終了時刻のずれが挙げられる.S-flag がリセットされた後にカウンターをスタートさせるが, それと同時に命令の終了処理も行われる. その終了処理にかかる時間は,SCCB-cluster の場合, 最悪 1.6μs かかる. その時間が global クロックとの誤差となる. また, 終了時のバリア同期を開始してからタイマーをストップさせるまでの時間も誤差の原因の 1 つである.

[4] 同期が成立してからタイマースタート・ストップまで, 最悪 240ns かかるので, その時間のずれが誤差になる

4 PGCCSを用いた実行時間測定

高精度測定の評価として,collective 通信の実行時間の測定を PGCCS を用いて行った. 前述した CPU 内のクロックを使用する 2 つの方法の測定結果と比較した. なお, 本稿では SCCB-Cluster の 16 ノードを使用して測定した.

PGCCS の性能評価として,PGCCS 関数の実行時間を測定した.Barrier&start コードと Barrier&end コードを連続して実行し, 計測される時間を測定した. 測定結果は,16 ノードで 3.1μs (各ノードの平均) であり, 各ノードの誤差は± 200ns であった.

4.1 MPI collective通信における高精度測定の評価

PGCCS を利用し,MPI の collective 通信の実行時間を測定した. 測定した collective 通信は,MPI_Reduce, および MPI_Allreduce である. 測定は 5 回行い, 各回ごとに各ノードの実行時間測定結果を平均し, その平均値からの誤差を各ノード毎に求めた. 同様の測定を前述の 2 方式でも行った.

MPI_Reduce の実行時間を測定したときの測定誤差の度数分布を図 10,11, および 12 に示す.

PGCCS での誤差は, ± 0.04%以内に収まっているが SCC の高速バリア同期を使用した測定では± 0.24%以内と 6 倍の誤差が生じた.MPI_Barrier を用いた測定では, ± 1.8%以内と比較的大きな誤差が生じてしまった.

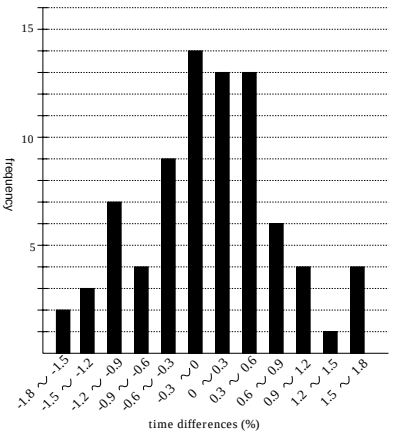

図 10: MPI_Barrier を使用した場合の測定誤差（測定関数：MPI_Reduce）

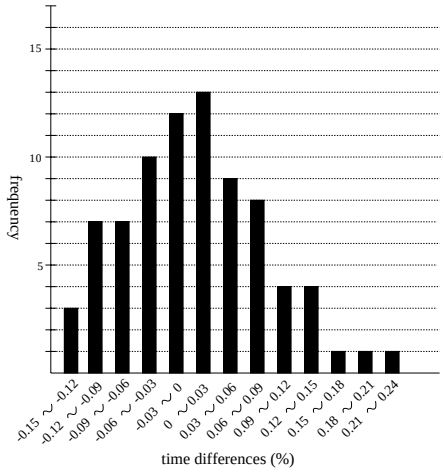

図 11: SCC_Barrier を使用した場合の測定誤差（測定関数：MPI_Reduce）

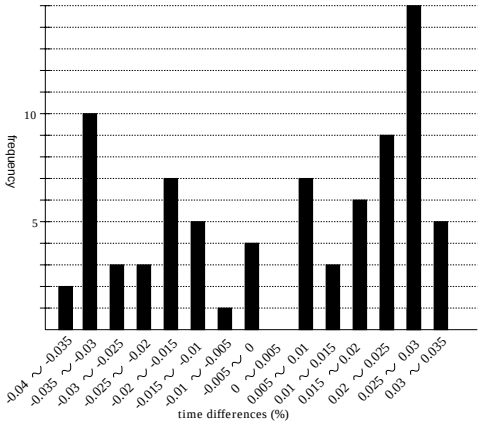

図 12: PGCCS で測定した場合の測定誤差（測定関数：MPI_Reduce）

MPI_Allreduce の実行時間を測定したときの測定誤差の度数分布を図 13 および 14 に示す. PGCCS で測定した場合の誤差範囲は± 0.0025%と比較的少ない誤差を得た. また, 前述の MPI_Reduce の測定では,0.02%の誤差に収まっているノードは全体の 45%に留まったが,MPI_Allreduce の測定では, 全てのノードが 0.02%の誤差に収まっている. これは,PGCCS の誤差は計測関数 (計測時間) にかかわらず,ほぼ同じ時間の誤差を生じさせているためである. つまり,MPI_reduce の測定時間は, 1 word 転送という比較的小さい転送を行ったため,MPI_Allreduce より短くなった. しかし, 計測誤差はほぼ同じ時間であるため, 相対的に誤差の%は増える結果となった.

5 おわりに

PC をベースした Beowulf クラスタシステム（SCCB-Cluster system）における高精度実行時間測定システムを提案した. 高精度な測定を可能にするため,Beowulf クラスタに SCC ボードを搭載した. また, その SCC ボード上において PGCCS を構築した.

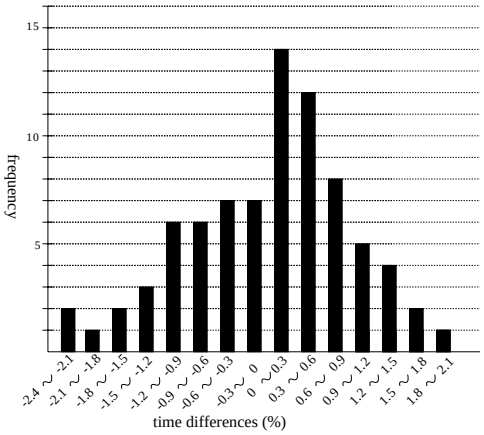

図 13: MPI_Barrier を使用した場合の測定誤差（測定関数：MPI_Allreduce）

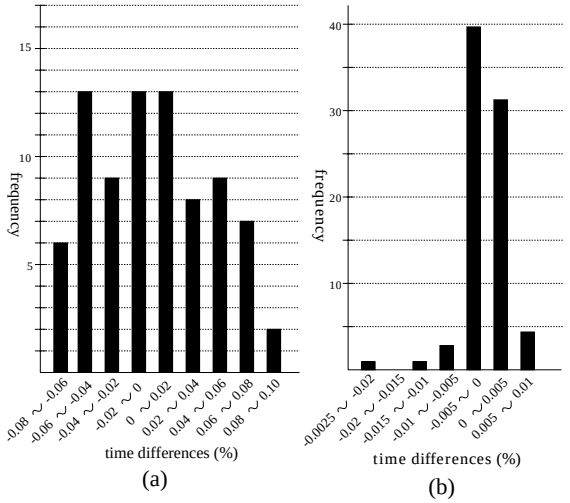

図 14: SCC_Barrier を使用した場合の測定誤差 (図中 (a)) および PGCCS で測定した場合の測定誤の測定誤差 (図中 (b))（測定関数：MPI_Allreduce）

性能評価として,MPI の collective 通信の実行時間を測定した. その結果,PGCCS の測定誤差が± 0.04% 以内であり, 他の 2 方式に比べ, およそ 1/6〜1/200 であった. PGCCS で測定した場合, 実行時間が長くなるにつれて誤差も少なくなる結果を得た. また,SCC の高速バリア同期を使用した測定でも, ± 0.24%以内おさまっており,PGCCS までは顕著に現れなかったが, 実行時間の長い関数では, 誤差が少なくなるという結果を得た.MPI_Barrier を使用した場合, 比較的実行時間が短い関数の測定には向いていないと思われる.

今後の課題としては,PGCCS を使用した MPI プログラムの実行状況解析ツールを開発すること, および PGCCS を使用したプログラムの実行制御機構を開発することが挙げられる.

参考文献

[1] Kiyosh Hayakawa, Satoshi Sekiguchi“Design and Implementation of a Synchronization and Communication Controller for Cluster Computing Systems,” Proc. 4th Intel. Conf. High Performance Computing in Asia-Pacific Region, vol.I, pp76-81, May.2000.

[2] Nanette Boden, Danny Cohen, Robert Feldrman, Alan Kulawik, Chrle Seitz, Jakov Seizovic, and Wen-King Su, “Myrinet – A Gigabit per Second Local Area Network”,*IEEE Micro*, vol.15, No.1, pp.29-36, Feb.1995.

[3] H.Tezuka, A.Hori, Y.Ishikawa, and M.Sato, “ PM: An Operating System Coordinated High Performance Communication Library.”,*High-Performance Computing and Networking*, vol.1225, pp.708-717, Apr.1997.

[4] S.Pakin, M.Lauria, and A.Chein, “High Performance Messaging on Workstations: Illinois Fast Messages (FM) for Myrinet”,In Proc. of Supercomputing ’95, 1995.

[5] 田中良夫, 久保田和人, 佐藤三久, 関口智嗣 “並列アルゴリズムにおける Collective 通信の性能比較”, 情報処理学会研究報告,96-HPC-62, pp.19-26, Aug.1996.

[6] 田邊 昇, 濱田 芳博, 須田 均, 山本 淳二, 今城 英樹, 中條拓伯, 工藤 知宏, 天野 英晴”DIMM スロット搭載型ネットワークインターフェース DIMMnet-1 の通信性能評価”, 情報処理学会研究報告,2001-ARC-145,pp.51-56,Nov.2001.