計算機アーキテクチャ 143－3
（2001．5．11）

RHiNETネットワークインタフェースプロトタイプの性能評価

大塚 智宏* 横山 知典* 土屋 潤一郎* 宮脇 達朗†
清水 敏行‡ 山本 淳二†† 西 宏章†† 工藤 知宏†† 天野 英晴*

*慶應義塾大学理工学部 †NEC情報システムズ
‡シナジェテック ††新情報処理開発機構

RHiNETは，オフィスのフロア等に分散配置されたPC/WSをノードに用い，光インタコネクションで専用のネットワークスイッチと接続することによって，高性能な並列処理システムを構成するためのネットワークである．ノードに搭載される専用のネットワークインタフェースRHiNET/NIは，低遅延で高バンド幅のユーザレベルゼロコピー通信を実現する．

本論文では，CPLDをコントローラに用いたプロトタイプであるRHiNET/NI0の実装と性能評価について述べる．4ノードシステムでの実アプリケーションによる評価の結果，最大で3.9倍の性能向上が得られた．

Performance Evaluation of RHiNET Network Interface Prototype

Tomohiro Otsuka*, Tomonori Yokoyama*, Junichiro Tsuchiya*, Tatsuaki Miyawaki†,
Toshiyuki Shimizu‡, Junji Yamamoto††, Hiroaki Nishi††, Tomohiro Kudoh††,
and Hideharu Amano*

*Faculty of Science and Technology, Keio University
†NEC Informatic Systems, Ltd. ‡Synergetech
††Real World Computing Partnership

RHiNET is a network which enables high performance parallel computing by connecting PCs or WSs distributed on one or more floors of a building. The RHiNET system consists of network interfaces, network switches, and high speed optical interconnections. The network interface called "RHiNET/NI" is attached to the PCI bus of a PC, and performs low-latency/high-bandwidth user-level "zero-copy" communication by a hardwired logic. Currently, a prototype with a CPLD controller called RHiNET/NI0 is in operation.

Experimental performance evaluation results shows that 3.9 times speedup on a system with four PCs is achieved using RHiNET/NI0 when the load is well balanced.

1. まえがき

RHiNET (RWCP High Performance Network)[1)]は，ビル内，フロア内等に分散配置された数十～数百台のPC/WSを高速な光リンクで相互接続することにより，高性能な並列計算環境を実現することを目的としたネットワークである．物理的に離れて配置されたPC/WS間を通常のクラスタシステムに匹敵する低遅延，高バンド幅のネットワークで接続することにより，PC/WSが使われていない間の余剰計算資源を有効に活用することができる．

RHiNETでは，専用のネットワークインタフェースおよびネットワークスイッチ，そしてそれらを結合するGbpsクラスの転送速度を持つ光インタコネクションによってシステムを構成する．このうちネットワークインタフェースRHiNET/NIは，PCIバスに接続されるカード上に実装され，低遅延かつ高バンド幅のユーザレベルゼロコピー通信を実現し，並列処理を支援する通信プリミティブをハードウェアによって高速に処理する．

現在RHiNET/NIは，CPLDをコントローラに用いたプロトタイプRHiNET/NI0が実装されている．本稿では，このプロトタイプRHiNET/NI0について述べ，実アプリケーションによる評価結果を示す．

2. RHiNET

RHiNETは，PCやWSのI/Oバスに装着されるネットワークインタフェースと，高速なネットワー

クスイッチ，そしてそれらを相互接続するGbpsクラスの転送能力を持つ光インタコネクションによって構成される．図1に示すように，ビル内やフロア内に分散して配置されたPCやWSを接続してシステムを構成し，並列分散処理を行う．

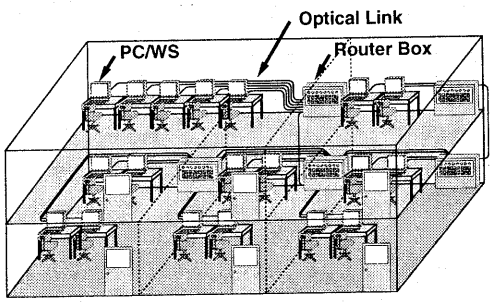

図1　RHiNETシステム例

ネットワークインタフェースRHiNET/NI0は，PCIバスに接続されるカード上に実装され，ノード間通信を高速に処理するために，効率の良いユーザレベルゼロコピー通信[2]をサポートする．そのために必要なアドレス変換などの処理は全てハードウェアで実現し，ソフトウェアオーバヘッドを減らしている．ネットワークスイッチRHiNET/SWは，長距離の接続を可能にするフロー制御と，PC間接続を柔軟に行うためのトポロジフリーでかつデッドロックフリーなルーティングを提供する．現在，最大転送容量が1.33 Gbps/portのRHiNET-1/SW，8 Gbps/portのRHiNET-2/SWが稼働しており，さらに信頼性の高いフロー制御を持つRHiNET-3/SWを開発中である．RHiNETはOSカーネルを改変することなくシステムを構成することを前提としており，カーネルモードでの実行が必要なソフトウェアレベルの機能は全てデバイスドライバにより実現する．

本報告では，評価対象となるRHiNET/NI0について詳しく紹介する．

2.1　RHiNET/NI0

RHiNET/NI0は図2に示すように，ノードのPCIバスに接続されるカード上に実装されている．

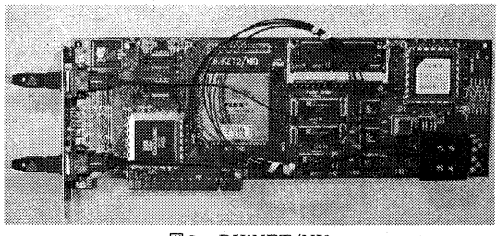

図2　RHiNET/NI0

高速な通信プリミティブを実現するための機構として，主にユーザレベルゼロコピー通信とアドレス変換の2つの機能を提供する．RHiNET/NI0では，プリミティブ処理ユニットとしてこれらの機能をCPLD上ハードウェア論理により実装している．

2.2　ユーザレベルゼロコピー通信

RHiNET/NI0では，システムコールを用いず，主記憶とネットワークインタフェースとの間で直接データを転送するユーザレベルゼロコピー通信を行うことで，高速な通信プリミティブを実現する．

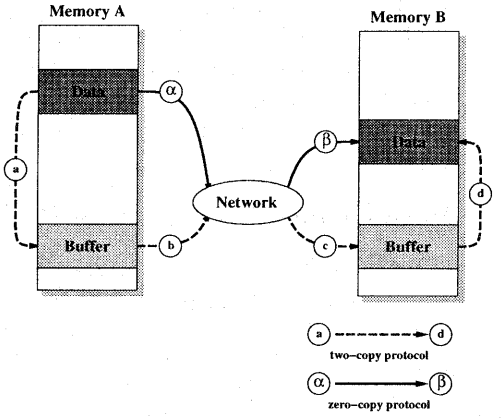

図3　ゼロコピー通信

通常のノード間通信では，図3 a～dに示すように，送信側・受信側ともに通信データをバッファを介して転送するため，主記憶内でデータコピーが発生することになり，通信のバンド幅がメモリコピーのバンド幅に支配される．

これに対して，ユーザレベルゼロコピー通信では，図3 α, βに示すように，I/Oバスを介したDMA転送によって，ネットワークと主記憶の間で直接データを転送する．データ転送が主記憶内でのデータコピーバンド幅に制限されないため，通信ハードウェアのピーク性能に近いバンド幅を実現することができる．

2.3　アドレス変換

RHiNET/NI0では，メモリベースの高速なユーザレベルゼロコピー通信を行うため，物理アドレス，仮想アドレス，グローバルアドレスID＋オフセットの3種類のアドレスを扱い，それらの間の変換を行う．

仮想アドレス／物理アドレス

ユーザプロセスから渡される仮想アドレスをNIがDMAするのに必要な物理アドレスに変換するために，RHiNET/NI0では仮想アドレステーブルを持つ．

グローバルアドレスID (GID)

通信に使用するメモリ領域の仮想アドレスはプロセスごとに異なるため，通信に先だってメモリ領域の先頭にグローバルアドレスID (GID) を付けてNIに登録しておく．通信を開始するノードは，このグローバルアドレスIDとそこからのオフセットにより通信先のアドレスを指定する．

2.4 パケット送受信

パケット送信の際，DMAでデータを全て読み出してからパケットを生成したのでは，レイテンシの増大を招く．パケット受信の際も，パケット全体を受信してからDMAを起動していたのでは，同様の結果となる (図4上)．

RHiNET/NI0では，これらのレイテンシを隠蔽するため，DMAによる主記憶アクセスとパケットの送受信をパイプライン化している．

パケット送信時には，DMAによってデータを読み出しながら，同時に送信FIFOへのデータの書き込みを行う．

パケット受信時には，パケットヘッダを受信してパケットの種別を認識できたら，ヘッダに含まれているデータサイズに従ってDMAを起動する．続けて受信されるデータを受信FIFOから読み出しながら，同時にDMAによって直接主記憶にデータを書き込む (図4下)．

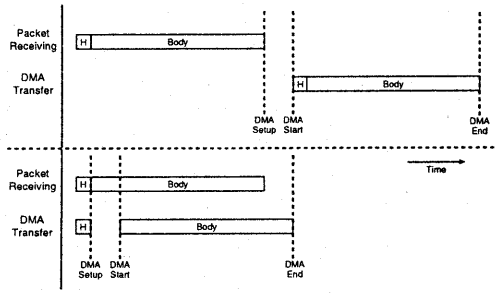

図4 パケットの受信

3. 通信プリミティブ

通信プリミティブは，RHiNET/NI0のプリミティブ処理ユニットによって実行される．プロトタイプのRHiNET/NI0では，CPLD上にプリミティブ処理ユニットを実装した．また，ユーザが通信プリミティブを利用するためのデバイスドライバ，APIを提供している．

3.1 通信プリミティブの実行

プリミティブの起動は，プロセスのアドレス空間上にマップされたNIのレジスタに必要な情報を書き込むことで行なわれる．

プリミティブの完了は，NIがホストに割り込みをかけるか，ホストの主記憶に完了フラグを書き込むことで通知される．したがって，DMAに干渉する可能性があるNIへのポーリングによる完了待ちを行う必要がない．

3.2 PUSH/PULL

PUSHおよびPULLは，ノード間通信を実現する最も基本的な通信プリミティブであり，それぞれリモートメモリライト／リードに相当する．

PUSH

PUSHは，アドレスとサイズで表されるローカルノードのメモリブロックを，リモートノードのGID＋オフセットで表されるアドレスから始まるメモリ領域に書き込むプリミティブである．

PULL

PULLは，リモートノードのGID＋オフセットとサイズで表されるメモリブロックを，ローカルノードのGID＋オフセットで表されるアドレスから始まるメモリ領域に読み込むプリミティブである．

3.3 高機能プリミティブ

RHiNET/NI0は，基本的なPUSH/PULL以外にもいくつかの高機能なプリミティブを実装している．これらのプリミティブはPUSH/PULLとホストのソフトウェアの組み合わせでも実装可能であるが，NI上のハードウェアで実装することで性能向上を目指している．

3.3.1 PUSH_BITMAP

PUSH_BITMAPは，メモリ領域のうち，ビットマップで示された部分 (バイト単位) のみをPUSHするプリミティブである．

PULL_WITH_TWIN/PUSH_DIFF

PULL_WITH_TWINおよびPUSH_DIFFは，ソフトウェア分散共有メモリを実現するためのマルチプルライタプロトコル[3]用のプリミティブである．

PULL_WITH_TWINは，PULLを実行すると同時にそのデータのコピー (TWIN) をNI上のTWINメモリにも書き込む．一方，PUSH_DIFFは，PULL_WITH_TWINで生成されたTWINと主記憶上のデータとを比較して，変更部分のみをPUSHするプリミティブである．両者を組み合わせることにより，特定の主記憶上の変更部分のみを転送することができ，ソフトウェア分散共有メモリの効率的な実装に寄与する．

3.4 BARRIER

BARRIERは，バリア同期を実現する通信プリミ

ティブである．バリアに参加するノードはマスタとなるノードのカウンタをデクリメントし，マスタはカウンタが0になったらブロードキャストによって全ノードに完了通知を行う．

3.5 デバイスドライバとAPI

ユーザプロセスは通信プリミティブ実行のための各種設定のために，デバイスドライバを経由してRHiNET/NI0にアクセスする．デバイスドライバが提供する機能を以下に示す．

- RHiNET/NI0の認識
- NIのアドレスの仮想メモリ空間へのマッピング
- ページテーブル・GIDテーブルのセットアップ
- 通信用のメモリ領域のピンダウン

また，プログラマにノード間通信を行うためのAPIを提供するライブラリが用意されている．主なAPIを表1に示す．

表1 主なAPI

rn_initialize	RHiNET/NI0の初期化
rn_set_gid	GIDの設定
rn_pindown	メモリ領域のピンダウン
rn_push	PUSHの実行
rn_pull	PULLの実行

4. 評　　価

RHiNET/NI0の実効的な通信性能を評価するため，PC4台をRHiNET/SW2に接続した小規模なRHiNETシステムを構成し，プリミティブおよびアプリケーションの実行性能を測定した．

4.1 評価環境

評価の際の環境は，以下の通りである．

- ホスト
  - プロセッサ：Intel Pentium III 400MHz
  - メモリ：256 MB
  - OS：Linux (kernel 2.2.17)
  - PCIバス：32bit/33MHz
- 伝送モジュール
  - 媒体：光インタコネクト
  - ケーブル長：5 m
  - クロック：133 MHz
  - 転送速度：1.46 Gbps
- ノード間遅延(4 byte転送時)：7.15 μsec
- 最大パケットサイズ：512 byte

4.2 プリミティブ実行性能

プリミティブの実行性能を評価するため，APIを用いた簡単なプログラムで実行時間を測定した．

PULL_WITH_TWIN/PUSH_DIFF

図5に，PULL_WITH_TWINと，PULLしたデータを主記憶上でコピーするPULL_WITH_TWINのエミュレーションのそれぞれの実行時間を示す．

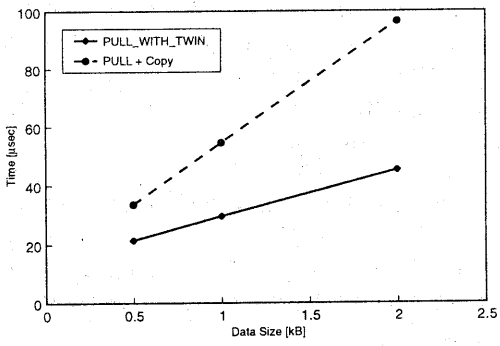

図5 PULL_WITH_TWINの実行時間

PULL_WITH_TWINの実行時間は，PULLとコピーによるエミュレーションの実行時間の53～68%であり，データサイズが大きくなればほぼ半分の時間で実行できることがわかる．

次に図6に，PUSH_DIFFと，ソフトウェアで差分を抽出してPUSH_BITMAPプリミティブで送信するPUSH_DIFFのエミュレーションのそれぞれの実行時間を示す．

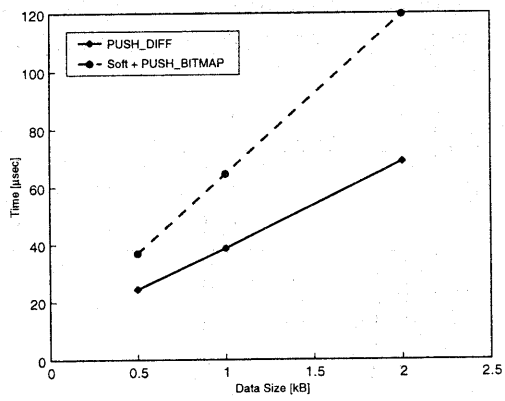

図6 PUSH_DIFFの実行時間

PUSH_DIFFの実行時間は，ソフトウェアでの差分抽出とPUSH_BITMAPによるエミュレーションの実行時間の55～61%であり，ほぼ半分の時間で実行できることがわかる．

以上のことから，PULL_WITH_TWINプリミティブおよびPUSH_DIFFプリミティブは，マルチプルライタプロトコルによって高性能なソフトウェア分散共有メモリを実現するのに十分な性能を持つと言える．

BARRIER

図7に，BARRIERプリミティブの実行時間を，関連する他の研究によるバリアと比較して示す．

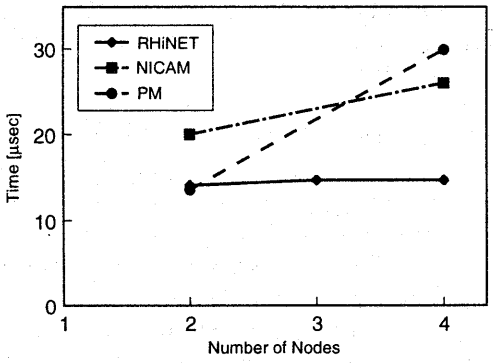

図7　BARRIERの実行時間

2ノードではPM[4]のバリアが速いが，4ノードになるとPMもNICAM[5]もRHiNET/NI0に比べてかなり遅くなっている．一方RHiNET/NI0のBARRIERプリミティブは，ノード数によらず15 *mu*sec以下の時間で実行できており，非常に高性能であることがわかる．これは，プリミティブ全てNI上で処理し，ブロードキャスト機構を備えているためである．

4.3　アプリケーション

評価用のアプリケーションとして，APIを用いて表2に示すベンチマークプログラムを実装した．ノード間の同期には，PUSHプリミティブによるソフトウェアバリアを用いた．

表2　評価用アプリケーション

RT	レイトレーシングによるCGの生成
FFT	複素数データの高速フーリエ変換
LU	密行列のLU分解

使用した問題サイズ，およびその1ノードでの実行時間は，表3に示す通りである．

表3　使用した問題サイズ

	Problem Size	Time [sec]
RT	800 × 600ピクセル	75.8
FFT	2^{20}個の複素数	7.2
LU	1024 × 1024の実数行列	30.1

4.4　台数効果

図8に，各アプリケーションにおいて1ノードでの実行時間を1とした場合の台数効果を示す．

RTでは，4ノードでほぼ4倍と理想に近い性能向上

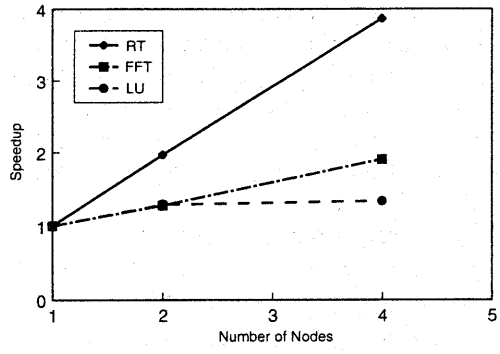

図8　台数効果

下の性が得られているのに対し，FFT，LUでは，その半分以能向上しか見られない．特に，LUでは，2ノードと4ノードでの台数効果にほとんど差がなく，性能向上が頭打ちとなっている．この原因については，次節以降で詳しく議論する．

4.5　演算時間

各アプリケーションの実行時間のうち，演算に要した時間の占める割合を表4に示す．演算時間以外の時間に含まれるのは，通信時間，同期操作による待ち時間，およびそれ以外の初期化などに要した時間である．

表4　実行時間に占める演算時間の割合

	1 Node	2 Nodes	4 Nodes
RT	99.9%	99.4%	95.8%
FFT	99.9%	97.5%	78.5%
LU	99.6%	91.1%	82.7%

RTでは，4ノードでも演算時間が実行時間の95%以上であるのに対し，FFT，LUでは約80%にとどまっている．このことから，FFTおよびLUでは，演算以外の処理に要する時間の増加が性能向上があまり見られなかった原因の一つであるとわかる．

次に，1ノードでの演算時間を1としたときの2ノード，4ノードでの相対的な演算時間を図9に示す．

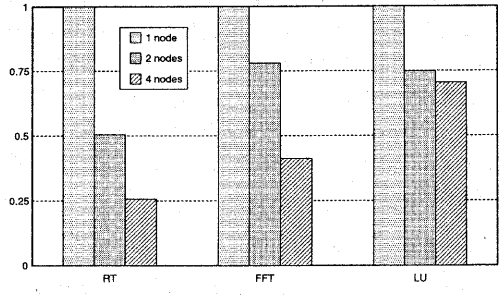

図9　ノード数の変化に対する演算時間の変化

RTでは，演算時間は2ノードでほぼ1/2，4ノードでほぼ1/4となっており，理想的な並列化が行われていることがわかる．一方，FFTやLUでは，演算時間は期待されるほど減っていない．これは，アプリケーションの並列化が十分行われていないことを示しており，これも性能向上があまり見られなかった原因となっている．

4.6 通信時間・同期待ち時間

表5に，各アプリケーションの4ノードでの実行時間のうち，演算時間以外の時間の内訳，すなわち通信時間，同期待ち時間(バリア同期に要した時間＋受信待ち時間)，およびそれ以外の初期化等に要した時間をそれぞれ示す．

表5 処理時間の内訳(単位: msec)

	Total	Com.	Wait	Others
RT	19552	13	778	25
FFT	3358	123	704	3
LU	24050	196	3702	255

実行時間のうち，通信時間の占める割合は各アプリケーションとも極めて小さく，同期待ちの時間が多くを占めている．すなわち，通信時間はアプリケーション全体の性能にほとんど影響を与えておらず，RHiNETは実用的なアプリケーションにおいても非常に低遅延の通信を実現していると言える．

4.7 通信量

表6に，各アプリケーションの4ノードでの実行時の総メッセージ数とデータ量，および通信バンド幅を示す．

表6 通信量・バンド幅

	♯ of msgs.	Data [kB]	BW [MB/s]
RT	3150	1440	27.7
FFT	24578	12583	25.6
LU	32253	12620	16.1

RTおよびFFTでは，通信データのサイズはほとんどが最大パケットサイズの512 byteだが，LUでは最小パケットサイズの4 byteのパケットが2割強あり，バンド幅が低下している．しかし，これらのバンド幅の値は，RHiNETが実用的なアプリケーションを動作させる上で十分な通信性能を提供できることを示していると言える．

5. 結　論

RHiNETのネットワークインタフェースであるRHiNET/NI0は，高速なノード間通信を行うための通信プリミティブをハードウェアによって高速に処理する．プロトタイプの実装では，プリミティブ処理ユニットをCPLD上に実装している．ユーザはAPI関数を呼び出すことでユーザレベルでNIにアクセスし，プリミティブを実行できる．

4ノードシステムでの評価の結果，1つのアプリケーションで3.9倍のほぼ理想的な台数効果を得た．他の2つのアプリケーションでも性能の向上を確認し，RHiNETの通信遅延によるアプリケーションの実行時間に対する影響は非常に小さいこと，実用的なアプリケーションにおいてもRHiNETは非常に高い通信バンド幅を提供できることを示した．

また，同期待ちに必要な時間が長かったことから，並列化のアルゴリズムを最適にすれば，アプリケーションの実効性能向上の余地はまだ十分にあることがわかった．

参考文献

1) 山本 淳二, 建部 修見, 横山 知典, 土屋潤一郎, 宮脇 達朗, 清水 敏行, 天野 英晴, 工藤 知宏, “高性能並列計算用ネットワークRHiNET-1の実装と評価”, 情処研報, 2000-ARC-137, pp.59–64, Mar. 2000.
2) 手塚 宏史, 堀 敦史, Francis O'Carroll, 原田 浩, 石川 裕, “ピンダウンキャッシュを用いたユーザレベルゼロコピー通信”, 情処研報, 97-ARC-125, pp.167–172, Aug. 1997.
3) J. B. Carter, J. K. Bennett, and W. Zwaenepoel, “Techniques for reducing consistency-related information in distributed shared memory systems”, ACM Trans. on Computer Systems, Vol. 13, No. 3, pp.205–243, Aug. 1995.
4) H. Tezuka, A. Hori, and Y. Ishikawa, “Design and Implementation of PM: A Communication Library for Workstation Cluster”, Proc. of JSPP '96, pp.41–48, Jun. 1996.
5) 松田 元彦, 田中 良夫, 久保田 和人, 佐藤 三久, “SMPクラスタ向けネットワーク・インタフェースAM通信”, 情処学論, Vol. 40, No. 5, pp.2225–2234, May. 1999.
6) 手塚 宏史, 堀 敦史, 石川 裕, 曽田 哲之, 原田 浩, 古田 敦, 山田 努, “PCとギガビットLANによるPCクラスタの構築”, 情処研報, 96-ARC-119, pp.37–42, Aug. 1996.
7) 石川 裕, 佐藤 三久, 工藤 知宏, 秋山 泰, 島田 潤一, “分散環境におけるシームレス並列コンピューティングシステムの構想”, 信学技報, CPSY97-62, pp.83–90, Aug. 1997.