システムソフトウェアと
オペレーティング・システム 75－15
（1997．6．5）

RT-Machにおける PC 間高速通信機構の評価*

古屋晋二　森康浩　佐々木真司　金子克幸

松下電器産業（株）　研究本部

OS に RT-mach、PC 間通信に Myricom 社の Myrinet を用いたマルチコンピュータ型アーキテクチャの連続メディアサーバを開発中である。RT-mach 用の Myrinet デバイスドライバを OSF1 用ドライバをベースに開発し、PC 間通信性能の評価を行なった。デバイスドライバを含む通信機構の下位層における処理時間の測定と分析を行ない、システムコールやデータコピー等のオーバーヘッドが性能低下の主要な原因であることがわかった。今回の実装では Myrinet の性能を充分に引き出すことが困難であったので、これらのオーバーヘッドを低減させる方法を検討し、これに基づく効率的な通信機構を提案した。この通信機構の性能を今回の評価から推測し、現在の実装に対して大幅な性能向上が見込めることを示した。

Performance Evaluation of Inter-PC Communication Scheme on RT-Mach

Shinji Furuya　Yasuhiro Mori　Shinji Sasaki　Katsuyuki Kaneko

Corporate Research Division

Matsushita Electric Industrial Co.,Ltd.

Moriguchi-shi,570 Japan

We have been developing the multi-computer-based continuous media server utilizing RT-mach as OS and Myrinet as inter-PC communication. The Myrinet device driver for RT-mach has been developed based on its OSF1 version. We measure the communication performance and analize the processing time in the lower layer of communication scheme. Major factors for communication overhead and several methods to decrease the overhead are shown. New communication scheme employing these methods is proposed and its performance is estimated. The estimation shows that new scheme exceeds current implementation and suits well for continuous media server application.

*この研究は情報処理振興事業協会（IPA）が実施している創造的ソフトウェア育成事業「次世代マイクロカーネル研究（MKng）」プロジェクトのもとに行なわれました。

1 はじめに

近年、映像のデジタル化に伴ない、家庭においてはCATVや電話回線を利用しユーザが自由に映像を選択して視聴できるビデオオンデマンドや映像に対して操作できるインターラクティブテレビなどが実現されつつある。また、放送業界においてはディスク装置に格納された映像情報に対し再生順を操作して送出するデジタル編集機が普及し始めている。連続メディアサーバはこれらのサービスや機器において最も重要な要素であり、読み出し時間と再生時間の差を利用して複数ユーザに対しそれぞれ異なった映像情報を同時に提供する機能をもつ。

我々は、PC AT互換機をベースにマイクロカーネル技術を用いてOSをカスタマイズすることで高コストパフォーマンスのメディアサーバを開発することを目指している。OSには入手の容易性からリアルタイムMach(RT-Mach)[1]を使用しており、シングルPC版[8][9]に続いてマルチコンピュータへの実装を検討している。

マルチコンピュータ型のメディアサーバでは、コンピュータ間で大容量の映像データをリアルタイムに転送する必要があることから、本メディアサーバでは高速LANとしてMyrinet[2]を採用した。Myrinetは米国Myricom社で開発された全二重方式通信ポートを備えたギガビットLANであり、最大1.28Gbps(×双方向)の転送速度をもつ。マルチポート・スイッチを組み合わせることにより自由にネットワークトポロジを構築することができる。Myrinetはインターフェースカード上にDMAエンジンを搭載したLANaiと呼ばれるプログラマブルなネットワークチップと、その制御プログラムであるMCP(Myrinet Control Program)の格納や転送メッセージのバッファリングのためのSRAMを有する。メッセージの送受信は必ずSRAMを通して行なうため、SRAMとホストメモリの間でデータ転送が必要になる(図1)。

我々は、RT-MachにMyrinetを実装するための第一ステップとしてOSF1からデバイスドライバの移植を行い通信処理時間の測定と分析を行なった。このデバイスドライバは、Myricom社から提供される標準的なMCPとMyrinet APIと呼ばれるMyrinet専用プログラミングインターフェースを用いて書かれており、LANaiに対するリクエスト発行や受信バッファの管理はMyrinet APIを利用して行う。すなわち、送信はリクエストをLANaiと共有される送信キューに発行してLANaiに通知することで実行される。受信は割り込みにより処理され、受信バッファの取得や返却などが行なわれる。

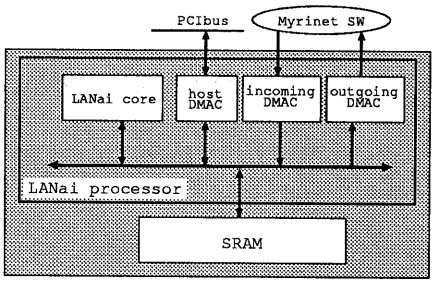

図1: Myrinet I/F Cardの構成

これまでMyrinetにおける高速通信技術の実装例としてはUCBのAM(Active Message)[6]、イリノイ大のFM(Fast Message)[3]、RWCPのPM[4]などがあり、Myrinet APIを使用した場合と比較して高い性能を得ている。

本報告では、RT-Mach(RMK95)+UNIXサーバ(Lites)環境でMyrinetのPC間通信性能を評価して現在の実装の問題点を示す。また、FM、AM、PMでの高速化手法を概観し、本メディアサーバに適した効率的な通信機構の提案を行なう。そして、それをRT-Mach環境において適用した場合の通信性能を今回の評価から推測し、現在の実装の性能と比較する。

2 メディアサーバの構成

マルチコンピュータ型メディアサーバの構成を図2に示す。各PCのPCIバス上にディスクインタフェースカード(DIC)、ビデオインタフェースカード(VIC)およびMyrinetインタフェースカード(MIC)が実装される。DICには連続メディアデータを格納するためのディスクアレイが接続され、VICには外部出力装置が接続される。また、MICは4または8ポート・スイッチを介して他PCと接続される。

連続メディアデータはPC間をまたがず、各々

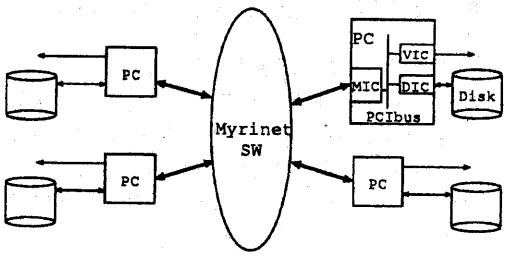

図 2: メディアサーバの構成

の PC に接続されているディスクアレイ内でストライピングされている。これは、PC 内部間と PC 間のデータ転送のリソース消費量を比較すると通常のインプリメントではコスト (=バス使用率またはメモリ使用率) は 2 倍となり、データを分散させて高コストの PC 間通信を使うよりはデータを分散させずに配置して通信コストを下げるような構成にした方がコストパフォーマンスが良く実用的であるという理由からである。

ディスクから読み出されたデータはホストメモリに格納された後に外部出力装置またはネットワークに送出される。また、ネットワークから転送されてきたデータはホストメモリに格納された後に外部出力装置に送出される。このとき各 PC 内のデータ転送能力はメモリを介した転送の場合 200Mbps 程度であることがわかっている [8]。

このような構成において、PC 内のデータ転送能力に対して PC 間のデータ転送能力はどの程度であるべきかという問題がある。具体的にはアプリケーションやシステムの構成によって統計的な解析が必要であるが、直感的には転送コストを 1:2 とするとリソースの配分は $1:\sqrt{2}$ 程度が適当であると思われる。この場合、PC 内のデータ転送能力は 200Mbps 程度なので、配分比は 80Mbps:120Mbps 程度となり、自ディスクからの外部出力が 80Mbps に対して、他 PC へのデータ送出が 60Mbps、他 PC からのデータ受信が 60Mbps となる。つまり、内部結合網の能力としては 60Mbps(7.5MBytes/sec) 程度の送受信が並行して行なえることが望ましい。今回の検討ではこの値の達成を目標とした。

3 通信性能測定

RT-Mach(RMK95)+UNIX サーバ (Lites) 環境において Myrinet による PC 間通信を実行し、通信の下位層の各処理に要する時間を測定した。PC の CPU 動作周波数は 133MHz、転送パケットサイズは 1KBytes であり、PC 同士は 1 つのマルチポート・スイッチを介して接続した。各処理時間はホスト CPU 内のカウンタを利用して求めた。

通信の下位層における処理は次のとおりである。送信時はシステムコールによってカーネルモードに移行した後、ユーザ空間からカーネル空間へデータのコピーを行なう。次にデバイスドライバの Output ルーチンが呼び出され、送信キューが Full 状態でなければ送信キューを介して LANai にリクエストが発行される。受信時は割り込みによりメッセージ到着が通知され、到着したメッセージを格納した受信バッファの取得と受信バッファの返却、IPC(プロセス間通信) による Lites 側への通知などが行なわれる。

PC 間で One-Way 通信を実行したとき、送信側においてカーネルモードへの移行およびカーネル内での各処理にかかる時間を表 1に示す。表において myriApi を冠する関数は Myrinet API の関数である。myriApisend() は送信キューへリクエスト発行して MCP にシグナルを送り終了する。測定結果から、ユーザ空間とカーネル空間との間を跨ぐコストが高いことがわかる。システムコールによるカーネルモードへの移行とユーザ空間からカーネル空間へのデータコピーを合わせて 37usec を要する。

受信側において割り込み処理および IPC によるユーザ側へのメッセージ通信にかかる時間を表 2 に示す。受信時は IPC によるメッセージ通信に数百マイクロ秒の時間を要する。

この実装において、送信時のシステムコール以下の処理に 100usec、受信時の割り込み処理に 50usec、ユーザ側へのメッセージ通信に 350usec 要するとして、上位プロトコル層を除いた通信性能の最大理論値を求めると、

$$\frac{1}{(100+400)\times 10^{-6}\times 1024} = 1.95\text{MBytes/sec}(\times\text{双方向})$$

となり、目標性能には達しない。

パケットサイズを現在の MCP での最大サイズ

表 1: 送信の処理時間

処理内容 (関数名)	処理時間
システムコールによる カーネルモードへの移行	10usec
ユーザ空間からカーネル空間 へのデータコピー (copyin())	27usec
送信キューの状態チェック (myriApisendQueueFull())	14usec
送信キューへのリクエスト発行 (myriApisend())	21usec
ギャザリングリスト生成など	11usec
終了処理 (iodone())	5〜15usec

表 2: 受信の処理時間

処理内容 (関数名)	処理時間
受信バッファの取得 (myriApiGetReceiveBuffer())	22usec
受信バッファの返却 (myriApiAddReceiveBuffer())	13usec
ヘッダ処理など	13usec
IPC によるユーザ側への メッセージ通信	280usec〜

である 8KBytes とした場合は、送信においてユーザ空間からカーネル空間へのデータコピー (copyin()) に約 8 倍、DMA 転送のギャザリングリストの生成に 2〜3 倍の時間がかかる。ギャザリングリストの生成は myriApisend() にも含まれる。受信において myriApiGetReceiveBuffer() におけるスキャタリングリスト生成にも 2〜3 倍の時間を要する。この場合は、送信時のシステムコール以下の処理に 330usec、受信時の割り込み処理に 70usec 要するとして、同様に上位プロトコル層を除いた通信性能の最大理論値を計算すると、10.41MBytes/sec(×双方向) となる。目標値は越えるが目標性能を得るために CPU 資源の大半を消費してしまうことになる。サーバで行なうその他の処理を考えると CPU 資源に対するより大きなマージンが必要である。

次節では、高速化を検討して効率的な通信機構を提案する。また、その通信性能について評価する。

4 PC 間高速通信機構の設計

PC 間の通信を高速化するためには、各処理のオーバヘッドの削減と軽量なプロトコルの採用が必要になる。我々は、既存の Myrinet API および MCP に対して一部の機能追加と変更を行なうことで本メディアサーバに適した効率的な PC 間通信機構を実現する方法を考察した。

4.1 オーバヘッドの削減

大容量の連続メディアデータを通信することから、パケットサイズを大きくして送受信回数を削減することで通信効率を上げることができる。Myrinet 上の SRAM 容量が 512KBytes であることを前提[†] にして、パケットサイズを SRAM 上のバッファエントリ数との関係から 16KBytes 程度とする。

また、PM[4] やミシシッピ大 [7] などの実装のように、MIC の SRAM をユーザ空間にマッピングしてユーザプロセスからアクセスできるようにし、データはユーザ空間から SRAM へ DMA 転送する (Zero Copy 方式)。システムコールを用いた実装では、システムコールおよびユーザ空間からカーネル空間へのデータコピーのオーバヘッドが大きい。とくにデータコピーに要する時間はサイズに比例して大きくなる。ユーザ空間から直接 SRAM へのデータ転送を行なうことによってカーネル空間へのデータコピーを回避できる。

FM[3] の実装のように、小さなメッセージを高速に処理するには CPU によるホスト- SRAM 間転送が有利である。本実装ではホスト-SRAM 間の転送に CPU による転送と DMA よる転送を併用する。すなわち、サイズが小さい他ノードへのリクエストメッセージの転送は CPU で実行し、サイズが大きいデータの転送は DMA で行なう。ヘッダ生成などメッセージの組み立ては CPU により直接 SRAM 上で行ない、メッセージ組み立てのためのデータコピーを回避する。

PM ではホストメモリから SRAM への DMA 転送の開始直後にネットワークへの送信を開始する並行動作によって高スループットと低レイテンシを得ているが、このような MCP の最適化は MCP の大幅な変更が必要となるので今回は採り入れな

[†] 最新版 MIC の SRAM 容量は 512KBytes である。

い。また、受信においてポーリングによってメッセージの到着を待つ方法も提案されているが、複数スレッドがポーリングによってメッセージの到着を待つことは困難であり、CPU に対する処理負荷も大きくなるので受信は割り込みによって処理する。この場合、通信のレイテンシは大きくなるが、メディアサーバではあまり問題にはならないと考える。

4.2 プロトコルの軽量化

FM や AM は低レイテンシを特徴としており、比較的短いメッセージの通信に対応している。さらに、AM は SPMD 型の並列処理でのデータ通信を前提にしたシンプルなプロトコルであり、FM はやや複雑なプロトコルであるが計算と通信のオーバラップが容易であることが特徴となっている。

メディアサーバの場合は、比較的メッセージ長が長い、メッセージ受信後にそのデータを使った計算は少ない、レイテンシについてはあまり重要ではないという性質がある。また、ノード (PC) ごとに処理が異なる。従って、AM や FM のようなプロトコルはメディアサーバには向かない。

これらを考慮して我々が考案した本メディアサーバ用の軽量なプロトコルは、1) メッセージ長が比較的長い転送に向き、2) 送信側が受信側のデスティネーションアドレスを管理することにより受信バッファ溢れによるデータ損失をなくして上位層での再送処理を不必要にするという特徴をもつ。

この通信機構を図 3に示す。ローカルノードとリモートノードは予めメモリ上に物理的に連続した領域をバッファ領域として静的に確保する。リモートノードのディスクにあるデータにローカルノードからアクセスするときは、先ずローカルノードからリクエストメッセージをリモートノードに送る。リクエストメッセージにはデータを受け取るローカルノードのホストメモリ上のバッファ領域のデスティネーションアドレスが含まれる。メッセージを受け取ったリモートノードの LANai はそれがリクエストであることを識別すると upcall して処理を委ねる。upcall を受けると、disk スレッドと myrinet スレッドは同期して、ディスクからのバッファ領域へのデータ読み出しとそのバッファ領域からの Myrinet へのデータ送出を行なう。このとき各パケットの先頭にはローカルノードの最終的なデスティネーションアドレスを示すタグが含まれており、デスティネーションアドレスはパケットが転送されるごとにオフセット値の加算が行なわれる。メッセージを受け取ったローカルノードの LANai はそれがデータであることを識別すると直ちにタグに含まれるデスティネーションアドレスに DMA 転送する。そして割り込みによってデータの到着を video スレッドに通知する。

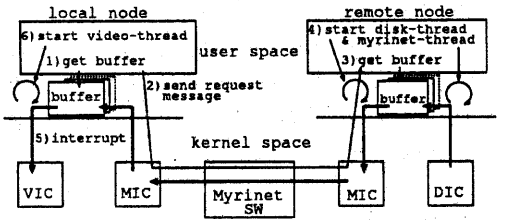

図 3: 通信機構

なお、これらは既存の MCP に対して、SRAM 上のバッファサイズの拡大および到着メッセージの識別とホストメモリの指定アドレスへの転送機能の追加など比較的小規模な変更で実現できる。

4.3 性能評価

パケットサイズが 16KBytes である場合の性能の理論値を求める。ディスクからのデータが転送されるバッファ領域から直接 SRAM にデータを DMA 転送するので、システムコール、カーネル空間へのデータコピー、終了処理のオーバヘッドは回避できる。また、メモリ上のバッファ領域は物理的に連続しているので、パケットサイズが 16KBytes であってもギャザリングリスト生成に要する時間は増加しない。

受信では、受信バッファ領域はユーザ側で用意され、ストリームごとに固定的に使用されるので、受信バッファの取得や返却は必要ない。しかし、割り込みにおけるユーザ側 (video スレッド) へのデータ到着の通知に百マイクロ秒のオーダーの時間を要する。

表 1および表 2からこれらのオーバヘッドを除くと、この実装では送信処理に約 50usec、受信時のヘッダ処理および IPC によるユーザ側へのデータ到着通知に約 370usec 要することになる。上位

層を除いた場合の通信性能の最大理論値を求めると、

$$\frac{16}{(50+370)\times10^{-6}\times1024} = 37.20\text{MBytes/sec}(\times\text{双方向})$$

となる。従って、この Zero Copy 方式では最大理論性能の約 20%で目標性能 (7.5MBytes/sec) を達成することができ、システムコールを使用した現在の実装と比較して高い性能が得られる (図 4)。

なお、受信時にユーザ側へのデータ到着通知に IPC を用いず、データ到着を待つスレッドをブロックし割り込み処理において再起動する方法 [5] もある。この方法を用いると、IPC よりデータ到着通知に要する時間を削減できるのでより高い性能が得られると考えられる。

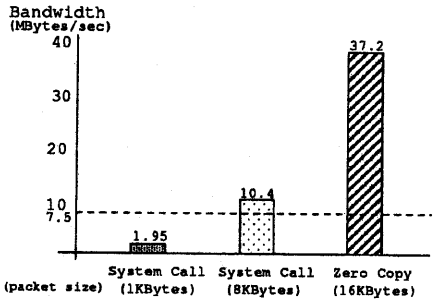

図 4: 通信性能の比較

5 まとめ

Myrinet を用いたマルチコンピュータ型メディアサーバに適した効率的なコンピュータ間通信機構を提案し、その通信性能を推定して現在の実装に対して大幅な性能の向上が見込めることを示した。本通信機構はデータの転送単位を大きくしてデータコピーなどのオーバヘッドを削減するとともに再送処理を回避した軽量なプロトコルであることを特徴としている。今後は、この通信機構を RT-Mach 上に実装して PC 間の通信性能を評価し、実験的な連続メディアサーバの試作を行なう予定である。

謝辞

貴重なご討論を頂いている慶応義塾大学環境情報学部徳田英幸教授をはじめとする MKng プロジェクトの皆様に感謝致します。

参考文献

[1] Hideyuki Tokuda et al., 'Real-Time Mach : Towards a Predictable Real-Time System', In Proceeding of USENIX Mach Workshop, October 1990.

[2] http://www.myri.com/

[3] Scott Pakin et al., 'High Performance Messaging on Workstations: Illinois Fast Messages(FM) for Myrinet' In Supercomputing, December 1995.

[4] 手塚宏史 他, 'ワークステーションクラスタ用通信ライブラリ PM の設計と実装' 情報処理学会並列処理シンポジウム JSPP'96, 1996.

[5] 手塚宏史 他, 'Myrinet 上の連続メディアデータ転送' 情報処理学会コンピュータシステムシンポジウム, 情処シンポジウム論文集 Vol.96,No,.7, Nov 1996.

[6] Thrsten von Eicken et al., 'Acvive Meassage: a Mechanism for Intergruted Communication and Computation' the Proceeding of 19th International Symposium on Computer Architecture, ACM Press. may 1992.

[7] Anthony Skjellum et al., 'A Guide to Writing Myrinet Control Programs for Lanai3.x' http://WWW.ERC.MsState.Edu/labs/icdcrl/learn_mcp/index.html

[8] Junji Nishikawa et al., 'Design and Implementation of Video Server for Mixed-rate Streams' Proceeding of the 7th International Workshop on Network and Operating Systems Support for Digital Audio and Video, may 1997.

[9] 森康浩 他, 'ディスクアレイに対応した連続メディアデータ用ファイルシステム' 信学技報 TECHNICAL REPORT OF IEICE, CPSY95-105(1996-01).