ハイパフォーマンス
コンピューティング 55−14
（1995. 3. 9）

並列コンピュータ Cenju-3 用 Mach における NORMA IPC の実現

菅原 智義　C.Howson　高野 陽介　小長谷 明彦

NEC C&C 研究所

我々は分散メモリ型並列コンピュータ Cenju-3 をターゲットに、Mach マイクロカーネルをベースにした並列 OS DenEn を開発している。DenEn においては、Mach-IPC と互換性を持ちつつ、高トラフィックに対応できるネットワーク IPC が重要である。Mach では Mach-IPC をネットワーク拡張した NORMA IPC が用意されているが、DenEn の要求を満たすものではない。そこで、我々はオリジナルの NORMA IPC をベースに、新たに NORMA IPC/DE を構築した。本稿ではこの NORMA IPC/DE の実装と評価について述べる。NORMA IPC/DE は、前述の要求を満たしつつ、小量メッセージでは低レイテンシ、大量メッセージでは高スループットを実現している。現時点での性能はレイテンシ 210μs、スループット 22M バイト /s である。

An Implementation of NORMA IPC on Cenju-3

Tomoyoshi Sugawara　Christopher Howson
Yosuke Takano　Akihiko Konagaya

C&C Research Laboratories NEC Corporation
1-1, Miyazaki 4-Chome Miyamae-ku Kawasaki Kanagawa 216 Japan

In this paper, we describe an implementation and its evaluation of NORMA IPC/DE. We have modified original NORMA IPC to take advantage of Cenju-3's high-speed and reliable network. Our goals is to provide the same local Mach-IPC semantics over the network, while supporting high message traffic with low latency and high throughput. We have achieved a latency of 210μs for short messages, and throughput of 22Mbytes/s for large messages.

1 はじめに

分散メモリ型、すなわち、多数の要素プロセッサを共有メモリを使わずに高速なネットワークで結合した並列コンピュータが多くのベンダーで開発され、商用化され始めている。このような並列コンピュータのためのOSは、科学技術計算などのために高度な計算能力を提供するだけでなく、データベースやマルチメディアなどの非数値を扱うような並列アプリケーションも効率良く実行できることを望まれている。これを実現するために、我々は分散メモリ型の並列コンピュータCenju-3[1]をターゲットにして、Machマイクロカーネル[3]をベースにした並列OS DenEn[2]を開発している。

Machではタスク間の通信にMach-IPCを用いるが、これとほぼ同じセマンティクスを持った、NORMA(NO Remote Memory Access) IPC[4]と呼ばれるネットワークIPCがカーネルレベルで提供されている。MachではOSサーバに対するサービス要求は元より、マイクロカーネルに対するシステムコールも大半はMach-IPCを使用しているため、分散メモリ型の並列コンピュータにMachを載せた場合、別のノードのOSサーバやマイクロカーネルのサービスを利用することができる。

NORMA IPCは分散メモリ型の並列コンピュータを意識して設計されてはいるものの、同時に複数のノード間通信を確立できない、Mach-IPCのフロー制御のセマンティクスを反映していないなど、Cenju-3プラットフォームに適合しない点が多い。そこで、我々はNORMA IPCをベースに、分散メモリ型並列コンピュータの通信性能を十分に引き出すNORMA IPC/DEを新たに構築し、Cenju-3上に実装した。本稿では、NORMA IPC/DEの設計思想、実装方式、評価結果および高速化の検討結果について報告する。

2 ネットワークIPCに対する要求事項

我々のネットワークIPCに対する要求事項は次の3点である。

- 高スループット、低レイテンシ

  スループットを高くすることは、画像処理などのデータ転送量の多いアプリケーションの性能向上に大きく貢献する。また、分散共有メモリやタスクマイグレーションの実現のためには、OSレベルでも高スループットが要求される。

  また、OSサーバを分散化した場合、リモートのサーバに対するIPCとローカルのサーバに対するIPCとの間に大きな性能差があると使い物にならない。そのため、レイテンシも小さくする必要がある。

- 複数のノード間通信の同時確立

  分散メモリ型の並列コンピュータ上で並列アプリケーションを実行する場合、ノード間の通信は必然的に多くなる。また、OSサーバを分散化した場合、1つのOSサーバに複数のノードのユーザタスクからサービス要求が届くことは頻繁に起こりうる。このような高トラフィックの状況でも、すべてのIPCを処理できることが必要である。

- Mach-IPCとの互換性

  シングルプロセッサシステム用のOSサーバをそのまま分散化し、安全に動作させるためには、インタフェースだけでなく、セマンティクスもMach-IPCと互換にする必要がある。

3 NORMA IPC

3.1 Mach-IPCとの関係

NORMA IPCはMach-IPCのネットワーク拡張であり、セマンティクスはほぼ同じである。Mach-IPCでは、メッセージはポートと呼ばれる通信チャネルを介して受け渡される。実際には、ポートはメッセージのキューであり、一旦、キューにつながれたメッセージはカーネルにより保護される。また、ポートにはポートライトと呼ばれる送受信の権利があり、ポートに対してアクセスできるタスクはポートライトを持つタスクに限定される。これらにより、Mach-IPCはセキュリティの高い通信を実現している。

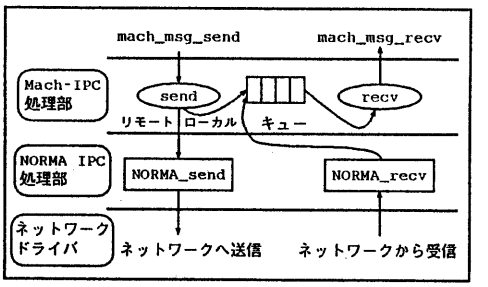

図1: Mach-IPCとNORMA IPCの関係

NORMA IPCを使う場合、カーネルインタフェースはMach-IPCのものをそのまま用いる。図1に示すように、NORMA IPCの処理はMach-IPCのキュー操作の代わりに行なわれる。Mach-IPCの処理ルーチンは、ポートがリモートなものである場合は、ポートのキューに入れる代わりにNORMA IPCの送信処理ルーチン(NORMA_send)にメッセージを渡す。リモー

トノードに送られたメッセージは、NORMA IPC受信処理ルーチン(NORMA_recv)によりMach-IPCのキューにつなげられる。

また、Mach-IPCで扱われるメッセージは、図2に示すようにヘッダとデータタイプ、データから構成される。データはデータ本体を含むinlineデータと、データへのポインタのみを含むout-of-line(OOL)データに2種類に分けられる。OOLデータの転送では仮想メモリの操作でコピーが行われるため、大量データの転送を比較的小さいオーバヘッドで行なうことができる。さらに、ポートライトもデータの1つとして扱うことができ、ポートライトの生成、複製、移動はIPCとして実現されている。

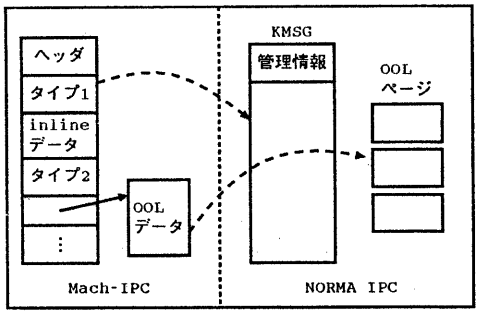

図2: メッセージの関係

NORMA IPCではこれらのデータをすべて扱うことができる。ただし、ポインタをそのまま扱うことはできないので、OOLデータとそれ以外は別々に扱われる。ヘッダとデータタイプ、inlineデータはひとまとめにされ**KMSG**として、OOLデータはページ毎に区切られて**OOLページ**として扱われる。

3.2 仕様上の問題点

Mach-IPCの機能の中で、NORMA IPCではサポートされていないものがある。

1つはMessage Acceptの通知(Notification)で、送信側でメッセージの受信がすでに済んだかどうかを知るための機能であり、ノンブロッキングのRPCを実現するために使われる。

もう1つは、ポートのキューレベルでのフロー制御で、送信側のポートのキューにつながるメッセージの数を制限することによりフロー制御を行なう機能であり、受信側が処理できなくてメッセージが溜った時に送信側をブロックさせる働きをする。低レベルなフロー制御では、不当なユーザタスクが一方的にOSサーバにメッセージを送りつけた場合に、OSサーバの要求受付けのキューが占有されることを防ぐことができないので、OSサーバを安全に動かすためにはこの機能が必要不可欠である。

3.3 プロトコルの問題点

NORMA IPCのプロトコルでは、受信側はメモリが不足した時、メッセージを受けとらずに捨てる。また、送信側はタイムアウトすると該当するメッセージを再送する。このプロトコルでは複数のノードから1つのノードに通信要求が集中した場合、メッセージを途中まで受信したところで受信側の空きメモリが不足すると、受信側はメッセージを捨て続ける。一方、送信側はタイムアウトと再送を繰り返す。結果的に、この受信側ノードに関わっているすべての通信が停止する。

この状態を回避するために、NORMA IPCではある時点で通信できるノードを1つだけに限定している。1つのノードと通信している間に他のノードからのメッセージが届いても、単に棄却するだけである。このため、同時に複数のノード間通信を確立することはできない。

また、信頼性の高いネットワークを仮定しているにも関わらず、プロトコルは、低い信頼性を仮定している点も問題である。

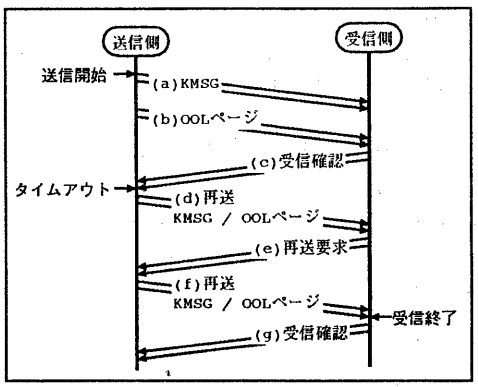

図3: NORMA IPCのプロトコル

以下にNORMA IPCの具体的なプロトコルを示す(図3参照)。以下の説明でメッセージとはすべてネットワークを流れるメッセージを指す。

1. 送信側は、最初にKMSGメッセージを送る(a)。ページサイズを越える 場合はページサイズに分割する。KMSGメッセージの送信が終了すると、

次にOOLページメッセージを送る(b)。これらのメッセージの送信時にはタイムアウトを設定する。

2. 受信側は、KMSGまたはOOLページメッセージを受けとる度に受信確認メッセージを返す(c)。十分な空きメモリがない場合は、メッセージを捨てる。途中のメッセージが失われてしまった場合は、メッセージを再送してもらうために再送要求メッセージを返送する(e)。すべてのメッセージを受信し、すべての確認メッセージを返送した時、受信は完了する(g)。

3. 送信側は、受信確認メッセージが届かずにタイムアウトした場合、該当するメッセージを再送する(d)。再送要求メッセージを受けとった場合も、同様に該当するメッセージを再送する(f)。送信したすべてのメッセージに対する受信確認メッセージを受けとった時、送信は完了する。

3.4 実装上の問題点

オリジナルのNORMA IPCの実装では、ほとんどの処理が割り込みレベルで行なわれるため、Mach標準のメモリ割り当て機能を使えず、メッセージをそのまま上のレベルに渡すことができない。このため、ネットワークドライバのレベルで受信したメッセージをすべて別のバッファにコピーしている。

コピーのオーバヘッドはメッセージサイズが小さい時は無視できるが、サイズが大きくなると処理時間の大半を占めることになるため、この実装ではスループットを高くすることは難しい。

4 NORMA IPC/DEの設計方針

2節で列挙したネットワークIPCに対する要求と、3節で述べたNORMA IPCの問題点を踏まえて、NORMA IPC/DEを設計する。設計の要点をまとめると次のようになる。

プロトコルの変更

同時に複数のノードからの通信を受け付けられるようにプロトコルを変更する。NORMA IPCのプロトコルの問題点である、通信の途中で受信側のメモリが不足する状態を回避するために、通信の始めに必要なメモリを予約して、予約が成功した時だけKMSGやOOLページを送ることにする。このために、メモリ予約用のメッセージを追加する。これにより、通信の途中で受信側がブロックすることはなくなる。また、受けとられることのないメッセージが送られなくなるので、ネットワークのトラフィックを無駄に増やすこともなくなる。

これにより、タイムアウトや再送をプロトコルに組み込む必要がなくなるので、これらを取り除き、プロトコルを簡単化する。

また、サーバに対するネットワークIPCのレイテンシを小さくするために、数百バイトまでの小量のinlineデータ転送のためのプロトコルを簡単なものにする。サイズを数百バイトに限定した理由は、サーバに対する要求、応答のメッセージの大半は数十〜百数十バイト程度に収まるからである。

プロトコルメッセージ受信用バッファの確保

変更したプロトコルでは、最初のメモリ予約のためのメッセージだけは、常に受信できなければならない。しかし、受信用のバッファを動的に割り当てる方法では、メモリが不足した時、メッセージを受信できなくなる。そこで、全ノード数分のプロトコルメッセージ（KMSGとOOLページ以外のメッセージ）受信用のバッファを固定的に確保する。これにより、空きメモリの有無に関係なくプロトコルメッセージだけは受信できることが保証される。

受信データの直接利用

大量データを効率良く転送するために、ネットワークドライバで扱うバッファをそのままMach-IPCのレベルまで渡せるようにする。ドライバレベルでは、カーネルが提供する一般的なメモリ割り当て手段を使うことはできないので、AST(Asynchronous Software Trap)レベルでこれを行なうことにする。ASTレベルとは、カーネルレベルからユーザレベルに戻る時のことを指し、すべてのカーネルスレッドが休止した後にASTレベルのハンドラが動作する。

Message Accept 通知機能のサポート

Message Accept通知機能をサポートするためにプロトコルを追加する。これにより、ポートキューレベルでのフロー制御も可能になる。

5 NORMA IPC/DEの設計および実装

前章の設計方針に基づいて、NORMA IPC/DEを設計し、実装した。ここでは、ターゲットであるCenju-3のハードウェアについて簡単に説明し、その後で、NORMA IPC/DEの設計、実装について述べる。

5.1 Cenju-3

Cenju-3は我々が開発した分散メモリ型並列コンピュータである。要素プロセッサにはR4400 75Mzを用いており、256台の最大構成時の性能は38.4GIPS、12.8 GFLOPSである。

NORMA IPC を実現するにあたり最も関係が深いのは通信機能である。ネットワークのハードウェアは4段の多段接続網で、スループットは40Mバイト/s、1段当たりの遅延時間は250nsである。Cenju-3では、ネットワークのハードウェアに見合う十分な性能を実現するための専用のインタフェースハードウェア（以下インタフェースと略す）を持つ。このインタフェースの特徴を以下に示す。

- 通信の種類は、メッセージ通信とリモートDMAが提供されている。
- プロセッサとは独立に、キューイングされたデータを自動的にパケットに分割してネットワークに送り出すことができる。パケットに分割される前のデータの最大長は128Kバイト、パケットの最大長は508バイトである。
- インタフェースからメモリへのアクセスは、物理アドレスに限られる。また、インタフェースに渡すアドレスもすべて物理アドレスを指定する必要がある。
- インタフェースが使用するメモリ領域は、ソフトウェアで明示的にキャッシュ制御を行なう必要がある。

5.2 NORMA IPC/DE のプロトコル

オリジナルのNORMA IPCに対して行なったプロトコルの変更を以下に示す。

第1に、メモリ予約のための送信要求メッセージと、メモリ割り当てができたことを知らせる送信許可メッセージを加えた。また、不要になった再送機能をプロトコルから取り除いた。

第2に、MachのIO処理に使われるiodoneスレッド（後述）を送信側のページマッピングに流用したかったので、一度に送ることができるOOLページの数を8ページに制限した。そして、8ページ単位でOOLページを送信するために、ページ送信許可とページコピー終了のメッセージを加えた。

第3に、小量のinlineデータ、すなわちKMSGのサイズが小さいメッセージに用いる簡単なプロトコルでは、送信要求メッセージにKMSGを含めて送ることにした。これにより、通常のプロトコルに比べ、送信許可とKMSGメッセージの2つを減らすことができる。Cenju-3の1パケットのサイズが508バイトなので、これから送信要求の情報などを減じた440バイトまでのKMSGを持つメッセージをこのプロトコルの対象にした。

第4に、大量データ転送時のコピーを減らすために、OOLページの転送にはリモートDMAを利用することにし、送信許可とページ送信許可メッセージの中には書き込み先を示す物理アドレスのリストを持たせた。

以下に、実装したNORMA IPC/DEのプロトコルを示す。（図4参照）

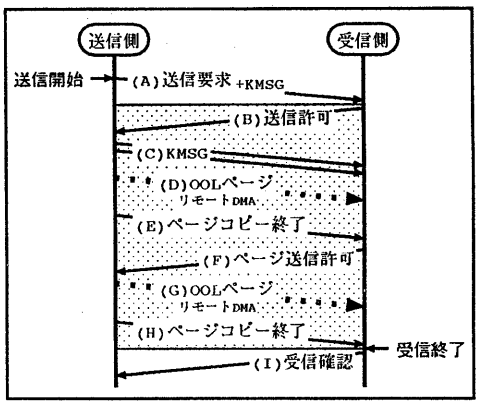

図4: NORMA IPC/DE のプロトコル

1. 送信側でKMSGのサイズとOOLページの数を計算し、それらを送信要求メッセージとして、受信側に送る。この時、KMSGのサイズが440バイト以下ならば、KMSGも付けて一緒に送る(A)。
2. 受信側は送信要求メッセージを受けとると、KMSGとOOLページを受信するのに必要なメモリをすべて割り当てる。KMSGが含まれている時は、割り当てたメモリにKMSGの内容をコピーし、6の処理へ移る。KMSGが含まれていない時は、送信許可メッセージを返送する(B)。

   受信すべきOOLページがある場合は、割り当てたページの先頭物理アドレスをリストにして、送信許可メッセージに含める。この時、OOLページが8ページ以上ならば、8ページ分だけのアドレスリストを含める。
3. 送信側は送信許可メッセージを受けとると、まずKMSGメッセージを送る(C)。128KBを越えるKMSGは分割される。

   OOLページは、KMSGメッセージの送信終了後、受信許可メッセージに含まれるアドレスリストを基に、OOLページをページ単位でリモートDMAにより転送する(D)。OOLページ(最大8ページ)の転送終了後、ページコピー終了メッセージを送る(E)。

4. 受信側は、KMSG メッセージを受けとると、その内容を割り当てたメモリにコピーする。

   ページコピー終了メッセージを受けとると、まだ、受けとっていない OOL ページがある場合、次の 8 ページまでの物理アドレスのリストを含んだページ送信許可メッセージを送信側に返す (F)。

5. 送信側は、ページ送信許可メッセージを受けとると、3と同様に OOL ページを転送し (G)、ページコピー終了メッセージを送る (H)。すべての OOL ページの転送が終るまで、4、5を繰り返す。

6. 受信側はすべてのデータを受信し終ったら、受信終了メッセージを送信側に送る (I)。

このプロトコルでは、440 バイト以下の KMSG だけを含む場合は、2 メッセージ、440 バイトを越える KMSG だけを含む場合は、最低 4 メッセージが必要である。OOL ページを含む場合は、8 ページ毎に 2 メッセージずつ必要になる。

5.3 送受信バッファの管理

複数のノードからのメッセージを同時に受け付けるために、ノード毎に 256 個 (256PE 構成の場合) のプロトコルメッセージ受信用バッファを固定的に確保した。メッセージ受信バッファの他にもノード毎に必要な情報があるので、1 ノード当たり 716 バイト必要になった。256 台分だと 182K バイト程度になる。無視できるほどに小さいとは言えないが、全体のメモリサイズからすれば、この程度ならば問題ない。

また、KMSG や OOL ページを格納したバッファを Mach-IPC のレベルに直接渡せるように、Mach 標準のメモリ割り当て機能 (kalloc と vm_page_grab) を使った。これらのメモリ割り当て機能を使えば、カーネルレベルでもユーザレベルでもメモリをフリーすることができる。また、OOL ページ用のバッファは、ネットワークインタフェースハードウェアにより直接、書き込まれるため、すべてのページをページアウトされないように固定し、さらに、受信許可メッセージを返す前にすべてのページについてキャッシュフラッシュしている。

送信バッファに関しては、KMSG も OOL ページも Mach-IPC の処理部分ですでに割り当てられているので、新たな割り当ては必要ない。ただし、リモート DMA を使うためには OOL ページに物理メモリをマッピングしなければならない。この処理はブロッキングの恐れがあるので、スレッドレベルで行なう必要がある。Mach では、ディスクからの DMA でマッピングの処理を行なう iodone スレッドが用意されているので、これを OOL ページのマッピングの処理にも流用した。iodone スレッドによって行なわれるマッピングの処理は、8 ページに限られている。

さらに、OOL ページをリモート DMA する前にキャッシュインバリデイトする必要がある。この処理には、時間がかかることが予想されていたので、1 ページ毎にキャッシュインバリデイトとリモート DMA を繰り返し、複数ページの送信の際はこれらをオーバラップできるようにした。

5.4 Message Accept 通知プロトコル

Mach-IPC の Message Accept 通知機能を加えるために、新たに Message Accept 通知プロトコルを加えた。以下にプロトコルを示す。

1. 送信側では、シーケンス番号を KMSG に書き込んで送信する。

2. 受信側で、受信タスクがポートからメッセージを受信する時に、KMSG の中のシーケンス番号と送信側のポート番号を Message Accept リストに登録する。このリストは後で、AST レベルの Message Accept ハンドラが Message Accept メッセージとして送信側に送る。

3. 送信側では、Message Accept メッセージを受けとると、リストのポート番号に対応するポートに対して Message Accept 通知の処理を行ない、ポートのキューから Mach メッセージを取り除く。

このように、このプロトコルは送信側のポートのキューにつながるメッセージの管理も一緒に行なうので、ポートのキューレベルでのフロー制御もできるようになる。

6 評価

ここでは Cenju-3 上に実装した NORMA IPC/DE の定量的評価を示し、その考察を述べる。

6.1 小量データ転送のレイテンシ

小量データ転送のレイテンシを調べるために、4 バイトから 2K バイトまでの inline データについて、メッセージの往復にかかる時間を測定した。メッセージの送信には mach_msg_send 、受信には mach_msg_recv を用い、mach_msg_send の前と mach_msg_recv の後で時間を計り、その差の半分を片道の所要時間とした。また、比較のために Mach-IPC でも同様の測定を行なった。これらの結果を inline データのサイズに対する所要時間のグラフ (図 5) に示す。ここでの測定結果の精度は 0.1μs である。

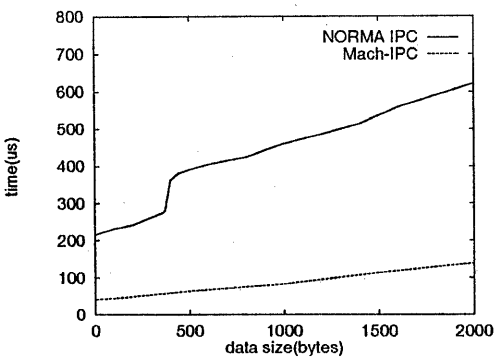

図 5: inline データのレイテンシ

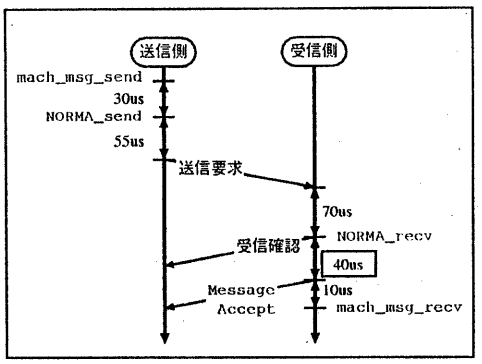

図 6: レイテンシの内訳

このグラフを見ると、440 バイト付近でのレイテンシの跳ね上がりから、5.2節で述べた簡単なプロトコルと通常のプロトコルの差が読みとれる。その差は 100μs 程度である。この差は 2 つのメッセージ（送信許可と KMSG）を減らしたことによるものである。したがって、各プロトコル処理にかかる時間に大きな差はないと仮定すると、1 つのメッセージの送信から受信までの時間、すなわち、メッセージの生成、キャッシュ操作、送信ハードウェアインタフェースの操作、ネットワークでの遅延、受信側プロセッサの割り込み処理、メッセージ受信後のプロトコル処理にかかる時間の合計は、およそ 50μs であると考えられる。

ここで、4 バイトデータの場合についてレイテンシの内訳を考えてみると Mach IPC の処理時間はグラフから約 40μs であり、やりとりされるネットワークメッセージは送信要求と受信確認の 2 つであるから、計算上のレイテンシは、

ネットワークメッセージの送信から受信までの時間 $\times 2$
　　+ Mach-IPC の処理時間
$=$ 50(μs) $\times$ 2 + 40(μs)
$=$ 140(μs)

程度になるはずである。実際にかかっている時間は約 210μs であるので、70μs 程余計にかかっていることになる。いかにプロトコル処理に時間がかかったとしても、70μs は大き過ぎるので、何か他にレイテンシを増やす原因があると考えられる。

そこで、4 バイトデータの場合について、プロトコル処理にかかる時間の内訳を調べた。その結果を図 6 に示す。この図は送信側で mach_msg_send を呼んでから、受信側で mach_msg_recv が戻るまでの時間を、プロトコルの処理毎に示したものである。ここでの測定時間の精度は 1μs 程度である。

これを見ると前の計算では入っていなかった Message Accept 通知プロトコルの処理がレイテンシに含まれていることがわかる (図中四角枠の中の 40μs)。Message Accept 通知プロトコルの処理は、受信タスクがポートからメッセージを受けとった後に行なわれるものなので、このレイテンシの計算には関係ないと考えていたが、実際にはこのような形で関わっていた。

6.2 大量データ転送のスループット

大量データ転送のスループットを調べるために、1 ページから 25 ページまでの OOL データについて、メッセージの往復にかかる時間を測定した。ページサイズは 16K バイトである。測定の方法は inline データのレイテンシ測定の場合と同じである。ここでの測定結果の精度は 0.1μs である。スループットは、測定結果の半分を片道の所要時間とし、送信したデータサイズをその時間で割って求めた。この結果を OOL データのページ数に対するスループットのグラフ (図 7) に示す。ここでは、平均のスループットとともに 1000 回測定した中での最大のスループットも併記する。

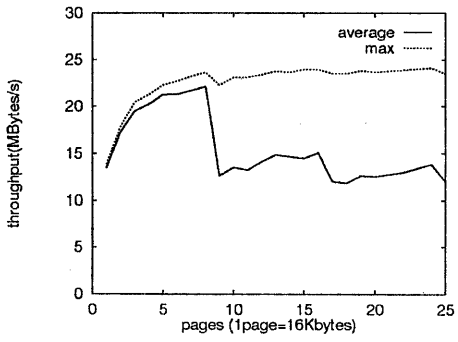

図 7: OOL データのスループット

最初に、平均のスループットを見ると、8 ページまではページ数が増えるほど高くなっており、8 ページ

の時で平均22Mバイト/s程度である。これは、送信要求、受信許可、受信確認のメッセージによるオーバヘッドが、ページ数が増えるほど小さくなるためであると考えられる。

また、8ページを越えると平均のスループットが激減し、16ページ、24ページを越えたところでも減少が見られるが、これには次の原因が考えられる。

1. 1度のリクエストで送信できる量を8ページに制限しているので、8ページを越えるとページ送信許可メッセージが必要になる。

2. 送信側では、8ページの送信が終了した後で、次に送る8ページに物理メモリをマッピングするためにiodoneスレッドを呼んでいる。このスレッド内での処理がブロックすることがあり、最悪の場合、数十msの間、再スケジューリングされないことがある。

原因1は、8ページ毎に1メッセージ分、すなわち、数十〜数百μsの処理時間を増大させるだけであり、これは全体の所要時間（8ページで6ms程度）の数%にに過ぎず、スループットに与える影響は小さい。一方、原因2は最小の処理時間の数倍に相当する処理時間を増大させることになるため、影響は極めて大きい。したがって、9ページ以上の転送でのスループット減少の主な原因は、この予測できないiodoneスレッドの待ち時間にあると考えられる。

また、9ページ以上のスループットが8ページ単位で単調増加せず、上下しているのも、原因1によるものと考えられる。

一方、最大のスループットは、iodoneスレッドによる処理時間が最小である場合と見なすことができる。このグラフから、8、16、24ページのところに山があり、24ページの時には24Mバイト/s程度のスループットになることがわかる。したがって、iodoneスレッドの影響がなければ、ページ数が増えるほどスループットが上がるものと予測できる。

7 高速化の検討

前節の評価を踏まえて、今後の高速化のための検討を行なう。

まず、レイテンシに関しては、Message Accept通知のプロトコルメッセージを他のメッセージの一部に含めて送ることで、50μs程度減少できると考えられる。Message Accept通知のための情報は1つのポートにつきたかだか2ワードしか必要でないので、他のメッセージに含めて送ることができる。

また、スループットに関しては、iodoneスレッドの代わりにページ送信専用のスレッドを用意することにより、必要以上に長い時間ブロックしないようにすることが可能である。これにより、9ページ以上の転送の平均のスループットを最大のスループットにかなり近づけることができると考えられる。

さらに、最大のスループットをあげるためには、ページマッピングの単位を大きくする方法がある。しかし、これは一時的とはいえユーザタスクが利用できる物理メモリを減少させるので、メモリサイズとの兼ね合いでマッピングの単位を決めなければならない。

8 まとめ

MachのネットワークIPCであるNORMA IPCをベースに、新たにNORMA IPC/DEを構築し、分散メモリ型並列コンピュータCenju-3上に実装した。NORMA IPC/DEはMach IPCと同じセマンティクスを持ち、複数のノード間通信の同時確立をサポートし、さらに、大量メッセージについては高スループット、小量メッセージについては低レイテンシを実現している。

まだ性能面では改良の余地はあるものの、現時点で、小量メッセージ転送ではレイテンシ210μs、大量メッセージ転送では最大スループット22Mバイト/sの性能を出している。

今後はさらに7節で述べた高速化を行ない、再評価を行なう予定である。

参考文献

[1] 広瀬, 加納, 丸山, 中田, 浅野, 稲村, "並列コンピュータCenju-3のアーキテクチャ", 情報処理学会 計算機アーキテクチャ研究会報告 Vol.107-16, pp.121-128, Jul. 1994.

[2] 高野, 小西, C. Howson, 荒木, 菅原, 小長谷, "Machマイクロカーネルをベースとした並列OS DenEnの実現", 情報処理学会 '95 全国大会 予稿集.

[3] Joseph S. Berra III, "A fast Mach network IPC implementation", *In Proceedings of the Usenix Mach Symposoium*, pp.1-12 Nov. 1991.

[4] David B. Golub, Randall Dean, Alessandro Forin and Richard Rasid, "Unix as an Application Program", *USENIX Summer Symposium*, Jun, 1990.