システムソフトウェアと
オペレーティング・システム 70－9
（１９９５．８．２３）

AP1000+におけるMicrokernel IPCの実装と評価

今村 信貴　藤崎直哉‡　石畑 宏明　池坂守夫 *
富士通(株)†　‡ (株)富士通研究所

分散メモリ型並列計算機において高速なユーザレベル通信が必要であるが，ユーザプログラムが直接ハードウェアをアクセスするためプロセス間の保護が難しい．
我々は高並列計算機AP1000+が持つ保護機構を用いて，マルチプロセス，マルチユーザ環境下でユーザレベル通信を実現した．本稿ではマイクロカーネルOSのIPCとユーザレベル通信の実現と評価について述べる．

Implementation and Performance of the Microkernel IPC on AP1000+

Nobutaka Imamura, Naoya Fujisaki‡ , Hiroaki Ishihata, and Morio Ikesaka
FUJITSU LTD. ‡ FUJITSU LABORATORIES LTD.

High speed message communication is important for high performance on distributed memory machines. User-level message passing schemes, as opposed to system-level schemes, are necessary to reduce communication overhead. But, accessing hardware from the user-level makes protection difficult.
Using AP1000+ protection mechanisms, we implement an user user-level message passing scheme on AP1000+. In this paper, we discuss the problems with protection in user-level message passing schemes, and describe an implementation of IPC on Microkernel, and user-level message passing.

*E-mail:{imamura,naoya,hata,ikesan}@flab.fujitsu.co.jp
†211 川崎市中原区上小田中1015

1 はじめに

メッセージパッシングをプロセッサ間の主な通信手段とする分散メモリ型並列計算機において並列プログラムを高速に実行するためには，高速なプロセッサ間通信が必要である．しかし汎用並列オペレーティングシステムが提供するプロセッサ間通信では，コンテキストスイッチ，プロトコルオーバーヘッド，通信バッファ管理などのオーバーヘッドが大きく，並列プログラムが必要とする性能を得ることができない場合がある．

近年，高速なプロセッサ間通信としてユーザプログラムが直接ネットワークハードウェアを操作してプロセッサ間通信をするユーザレベル通信が行なわれるようになった[1, 7, 8]．我々の開発した分散メモリ型高並列計算機AP1000+においても，独自のOSであるCellOS+上でユーザレベル通信を提供しており，高速なプロセッサ間通信が可能である[11]．

しかし，ユーザレベル通信は一般的なオペレーティングシステムの要件であるプロセス間の保護と相反する機能である．ネットワークハードウェアを直接アクセスするためOSによる排他処理ができず，互いに他のプロセスの通信を妨害する可能性がある．また許可されていない他の並列プロセスにメッセージを送ったり，他の並列プロセス宛のメッセージを受けとることは許されない．このためCellOS+ではパーティション毎に異なるユーザに割り当てることはできたが，パーティション内ではシングルユーザ，マルチタスク環境であった．

その一方で，以下の理由からメッセージパッシングを用いる並列計算機においてもUNIX互換のマルチユーザ，マルチタスク環境が要求されている．

- 並列プログラムのTSS
- UNIX互換のシステムコール
- 既存のコマンドやシェルプログラムと並列プログラムの連携
- スループットの向上
- 仮想記憶サポート

以上から，分散メモリ型並列計算機においてもマルチユーザ，マルチタスクのUNIXのプログラミング環境が必要であり，並列プログラミング環境と両立させなければならない．

我々が開発した分散メモリ型並列計算機AP1000+は，マルチユーザ・マルチタスク環境でユーザレベル通信を行なうための保護機構を備えている．この保護機能を用いて，マイクロカーネルベースのUNIX OSであるChorus/MixをAP1000+に移植し，高速なプロセッサ間通信が可能な並列プログラミング環境とUNIX環境の融合の試みとしてユーザレベル通信を行なうCellOS+互換の環境を構築する．

本稿ではまずAP1000+とChorus/Mixの概略について述べる．次にカーネルがサポートするメッセージ通信機構のAP1000+への実装と性能評価について述べる．最後にUNIXのマルチユーザ，マルチタスク環境でユーザレベル通信を実現するための問題点と保護機構について述べ，マイクロカーネル上での性能評価を示す．

2 AP1000+

AP1000+のシステム構成を図 1に示す．

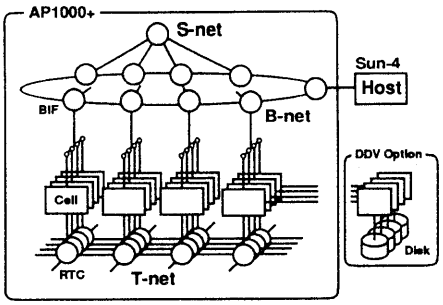

図 1: AP1000+

AP1000+はSuperSPARCプロセッサを用いたセルと呼ぶプロセッシングエレメントを最大1024個持つ．各セルはT-net, B-net, S-netの3つのネットワークで接続される．T-netはセル間の1対1通信に用いられる．トポロジは2次元トーラスとなっており，ワームホールルーティングを行なう．各方向に25MByte/sのバンド幅を持つ．B-netはホストとセル間の放送とデータ集配に使用される．OSのブートもB-netを通じて行なわれる．S-netは各セル上のプログラムの同期に用いられる．AP1000+の仕様を表 1に示す．

T-netの通信では，メッセージ通信のsend,recvの他にリモートセルのメモリ上のデータを直接読み書きするput,getが可能である[10]．これらの操作はCommand Queueと呼ばれるネットワークハードウェアに操作の種別やアドレスなどからなるコマンドワード列を書き込むことで起動される[11]．

また，ネットワークデバイスが仮想アドレスをサポートしている[13]．単に送信側，受信側でネッ

表 1: AP1000の仕様

CPU	SuperSPARC(50MHz)
Cache	20KB(Instruction) 16KB(Data)
Memory	16MB or 64MB
Network Through put	25MB/s(Tnet) 50MB/s(Bnet)
Disk(option)	500MB

トワークデバイスがその時の仮想空間に指定された仮想アドレスでメモリにアクセスするのではなく，送信側と同じContext IDを持つ受信側のプロセスの空間にアクセスできる．このContext IDは，SuperSPARCのMMUがプロセス毎に割り当てる仮想空間のIDであり，アドレス変換の際に最初のテーブルを引くのに用いられる．このため受信側で他のプロセスが動作していてもCPUに割り込みを起こしたりコンテキストスイッチをする事なく，目的の仮想空間にアクセスできる．

AP1000+のChorus/Mixでは各セルに分散ディスクビデオオプション(以下DDV)を接続しChorus/Mixを動作させる(図2)．各セルのChorus/Mix上でCellOS+互換の並列プログラムや通常のUNIXプロセスが動作する．セルに接続されたDDVはufsのディスクとして使用する．

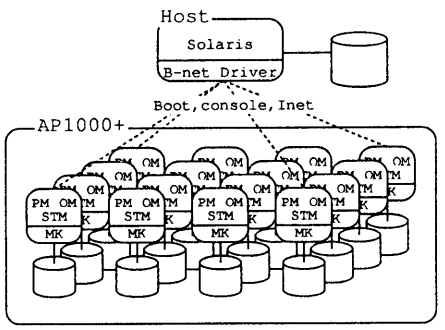

図 2: Chorus/Mix on AP1000+

各セルのChorus/Mix上でCellOS互換の並列プログラムを起動する並列プログラムマネージャを動作させる．並列プログラムマネージャはUNIX上のシステムデーモンとして動作し，並列プログラムの起動要求があると各セルで並列プログラムのプロセスを起動する．

3 Chorus/Mix

Chorus/MixはフランスのChorus Systemes社のマイクロカーネルベースのUNIX OSである．Mixと呼ばれるマイクロカーネル上のサーバプロセス群がマイクロカーネルIPCによって協調動作することでSVR4の機能を実現している．シングルプロセッサの場合のChorus/Mixの構成を図3に示す[9].

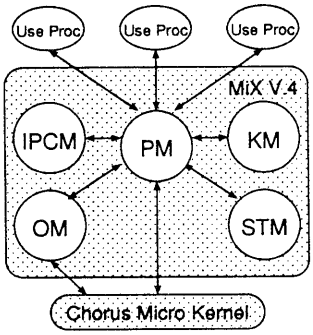

図 3: Chorus/Mix

PMはプロセスマネージャで，UNIXシステムコール全ての入口である．プロセスの実行に関わるシステムコールを処理する．OMはオブジェクトマネージャでファイルシステム関連のシステムコールを処理する．STMはストリームマネージャで，STREAMS関連の処理を行なう．IPCMとKMはSVR4のIPCとそのkeyを担当する．

Chorusマイクロカーネル上では，UNIXのプロセスに相当する概念はアクタとスレッドと呼ばれる．アクタは資源獲得の単位であり，1つの仮想空間を持つ．スレッドはアクタ中の実行主体である．スレッドはポートと呼ぶメッセージキューを介してIPCを行なう．ポートはUnique Identifier(以下UI)によって識別される．ポートはネットワーク透過になっておりポートのあるサイトと呼ぶプロセッシングエレメントがローカルなのかリモートなのかを意識する必要はない．

サイト間のリモートのIPCは，マイクロカーネルの外部のドライバを経由して行なわれる．マイクロカーネルのIPC機構がカーネルのメッセージバッファの管理やコネクション確立を行なうため，このドライバはIPC機構から渡されたメッセージを受信サイトに渡し，受信サイト側ではそれをIPC機構に渡すのが仕事である．

ネットワークにまたがるリモートのマイクロカーネルIPCはシングルシステムイメージを実現する際に，分散されたMixサーバ間で用いられる．小さなレイテンシと高いスループットが重要である．

AP1000+に移植する際にプロセッサ間通信に関わる要件は以下の通りである．

- マイクロカーネル通信の実現と性能の確保
- マイクロカーネル通信とユーザレベル通信の両立
- 並列タスク間の保護の実現

4 AP1000+におけるSystem通信の実装

リモートのマイクロカーネルIPCのドライバとして，sendをデータの転送につかうドライバとgetをデータの転送に使うドライバの2つを検討した．

4.1 sendを用いたドライバ

送信サイトでは，ドライバ層のヘッダとマイクロカーネルのIPC機構から受けとったメッセージをバッファにコピーして送信する．

1. ドライバ内部のバッファにヘッダとメッセージを書く．ヘッダにはメッセージIDやメッセージ長が含まれる．
2. AP1000+のsendでドライバのバッファから送信し，受信側に割り込みを起こす．
3. send終了後メッセージバッファを解放する．

受信サイトでは割り込みハンドラが起動され，ハードウェアのメッセージバッファであるリングバッファから受信を行なう．

1. リングバッファからドライバ層のヘッダを読む．
2. メッセージIDから，あらかじめ確保されたカーネルのメッセージバッファ領域を得る．
3. リングバッファからメッセージバッファへメッセージをコピーする．
4. 上位層を起動する．

AP1000+のネットワークはContext IDによる保護を行なうために，Context IDが同じプロセスどうしで通信しなければならない．ドライバはどのContext IDで実行されるかわからず，相手側にないContext IDを用いる可能性がある．このため，sendを行なう際にネットワークデバイスにContext IDとして0を一時的にセットしてコマンドを投入する．

一回の通信でのメッセージのcopyの回数は，送信側でユーザ空間からカーネルバッファへ，カーネルバッファからドライバのバッファへの2回ある．受信側でリングバッファからカーネルバッファへ，カーネルバッファからユーザ空間への2回あり，合計4回発生する．

割り込みは受信側で発生する1回のみである．

このドライバは次のgetを用いたドライバと比べ，割り込みが1度なのでレイテンシは小さいがスループットで劣ることが予想される．

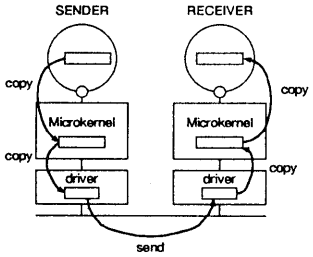

図 4: sendを用いた場合

4.2 getを用いたドライバ

copyの回数を減らすためにAP1000+のgetを用いてデータを転送する．

まず送信サイトがメッセージバッファの領域を受信サイトに知らせる．

1. ドライバのバッファに受信要求のコントロールメッセージを書く．これにはメッセージIDや長さ，カーネルのメッセージバッファの領域が書かれている．
2. 受信サイトを調べsendで送信し，割り込みをかける．
3. 送信ドライバを終了する．

このドライバの中ではカーネルのメッセージバッファを解放しない．

受信サイトでは送信サイトからかけられた割り込みでハンドラが受信を開始する．

1. リングバッファを調べ，ドライバ層のコントロールメッセージを読む．
2. メッセージIDからカーネルのメッセージバッファ領域を得る．
3. 送信元のメッセージバッファの領域からgetでメッセージを読む．
4. sendで送信側に受信完了のコントロールメッセージを送信し，割り込みをかける．
5. 上位層を起動する．

最後に送信サイトの割り込みハンドラがカーネルのメッセージバッファを解放して終了する．

getによる転送では，割り込みがかかった時のContext IDで割り込みハンドラが実行されるため，送信サイトのドライバと受信サイトのハンドラでContext IDが異なることがあり得る．異なるContext ID間でのgetはできないため，送信サイトのドライバでメッセージバッファの仮想アドレスから物理アドレスを求め，送信サイトの物理アドレスから受信サイトの仮想アドレスへのgetを行なっている．

メッセージのcopyの回数はユーザ空間とカーネルバッファの間の合計2回である．割り込みは受信

側で1回，送信側で1回発生する．copyは少ないが割り込みが多いため，レイテンシは大きいがスループットはsendを用いたドライバより良いことが予想される．

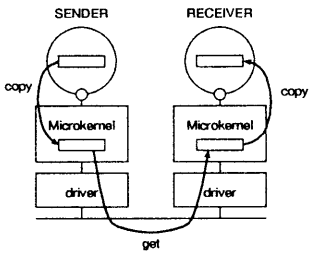

図 5: getを用いた場合

4.3 評価

マイクロカーネルのipcSend()システムコールとipcReceive()システムコールを組み合わせて2つの隣接プロセッサ間でメッセージが往復するのにかかる時間を測定し，その半分を片道に要する時間とした．横軸がメッセージ長で単位はバイトである．縦軸は片道に要する時間で単位はμsである．グラフ中Systemとあるのはシステムアクタ間の通信，Userとあるのはユーザアクタ間の通信である．メッセージ長が長い場合のグラフ(図 7)ではシステムアクタ間通信とユーザアクタ間通信で差が見えないため，システムアクタ間通信の結果のみを示す．

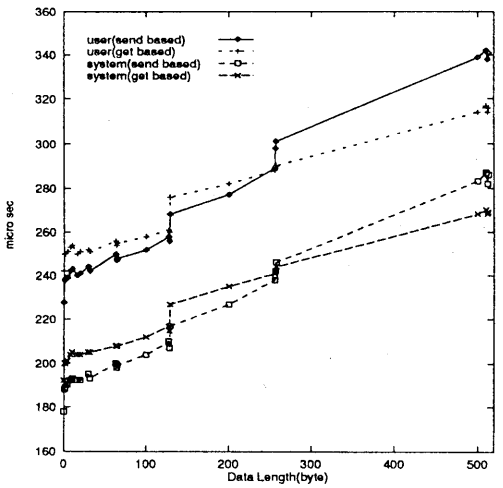

図 6: マイクロカーネル通信(1)

sendによって転送するドライバではメッセージ長が4Byteの場合にシステムアクタで190μs，ユーザアクタで239μsである．また，スループットはどちらも最大約5.3MByte/sである．

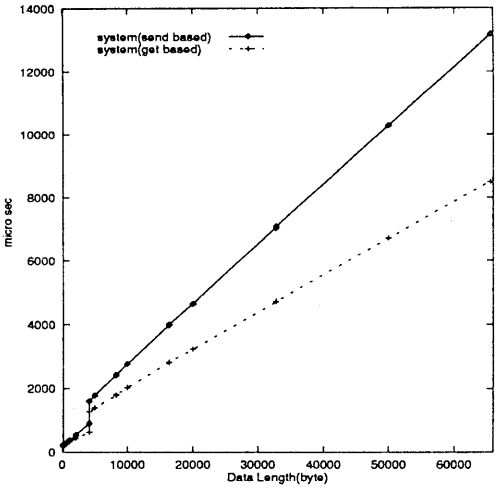

図 7: マイクロカーネル通信(2)

getによって転送するドライバではメッセージ長が4Byteの場合にシステムアクタで201μs，ユーザアクタで251μsである．また，スループットはどちらも最大約8.5MByte/sである．ただし，メッセージを往復させて測定したためバッファ解放のための送信側の割り込みがメッセージの受信待ち状態で発生し，測定時間に含まれていない．

グラフからsendを用いたドライバとgetを用いたドライバでの転送に要する時間はメッセージ長が256バイトに等しくなる．測定時間に含まれていない割り込みのオーバーヘッドも考慮して，AP1000+のリモートIPCのドライバはメッセージ長が短い場合はsendで転送し，512バイト以上になるとgetで転送する．

システムアクタとユーザアクタでレイテンシに差があるのは，システムアクタの場合は形の上ではシステムコールとかわりないが，実際にはマイクロカーネルへの関数呼び出しになるからである．このためトラップ処理や空間切替のオーバーヘッドが発生しない．Mixサーバー群はシステムアクタとして動作するので，低レイテンシの通信が利用できる．

メッセージ長が129,257,4097Byteの所に大きな不連続点があるが，これはマイクロカーネルのIPC機構のメッセージバッファ管理が変わるからである．

4.4 考察

マイクロカーネルIPCのスループットが8.5MByte/sなのは，T-netのハードウェア性能の25MByte/s

と比較して低い．そこでメモリ間copyの時間を測定し，メッセージが長い場合についてスループットの要因を考察した．

AP1000+のセルはデータキャッシュが16KByteと小さいため，マイクロカーネルIPCの性能測定中キャッシュがほぼヒットしなかったと考えられる．そこで32KByteの転送について考える．

グラフから32KByteの片道にかかる時間は4.7msである．また16KByte以上の通信のオーバーヘッドは0.8msである．

bcopyで32KByteをcopyするのにかかる時間は1159μsであった．メッセージ長が大きい場合は，前述したようにcopyが2回あるので，約2.3msかかることになる．これはスループットの約59%を占める．

T-netで32KByteを転送するのに1.25msかかり，スループットの32%を占める．残りの9%程度はカーネルのメッセージ長依存の処理と考えられる．

スループットを支配しているのはcopyであり，copyを速くする，あるいはその回数をさらに減らすと大きな効果を得られることが予想できる．

5 User level通信の問題点

ユーザレベル通信では，OSが介在しないため以下の問題点が発生する．

- 保護
- メッセージ待ちのポーリング
- page fault

以下でAP1000+での解決策を説明する．

5.1 保護

以下の項目の保護が必要になる．

- ネットワークデバイスの排他制御
- メモリ空間の保護
- メッセージバッファの保護

5.1.1 ネットワークデバイスの排他制御

ユーザレベルでネットワークデバイスにアクセスするため，ネットワークデバイスの操作中にコンテキストスイッチを起こし，他のプロセスが別の操作を行なう可能性がある．

これは[12]で提案されたように，コンテキストスイッチ時にデバイスのコマンドキューをチェックし投入途中のコマンドを退避，復帰することで解決する．

5.1.2 メモリ空間の保護

AP1000+ではput,get操作によってリモートのプロセスのメモリ空間にアクセスすることができる．put,get操作で他の並列プログラムのプロセス空間にアクセスすることを禁止する必要がある．

AP1000+のネットワークハードウェアは，コンテキストを意識した仮想アドレスサポートを行なう．AP1000+の概要で述べたように送信側と同じContext IDのプロセスの仮想空間にアクセスすることでプロセスのメモリ空間の保護を行なう．

ただし，Context IDを元に保護を行なうため，並列プログラムのプロセスは全てのセル上で同じContext IDを持たなければならない．そこで，Context IDを指定できるforkシステムコールを独自に追加する．このシステムコールはContext IDを指定する事とスーパーユーザ権限のあるプロセスでしか実行できない事以外は通常のforkシステムコールと同じである．並列プログラムマネージャはこのシステムコールで並列プログラムのプロセスを作る．Context IDが同じであることは並列プログラムマネージャが保証する．

5.1.3 メッセージバッファの保護

AP1000+は，リングバッファをハードウェアの使うメッセージバッファとして用いる．リングバッファを用いることで割り込みの不要な，バッファ効率の良い受信を行なうことができる．しかし，ユーザが直接このリングバッファにアクセスするため，プロセス間の保護が必要になる．

AP1000+ではContext ID毎に異なるリングバッファを用いることができる．一度に3つのリングバッファをContext ID対応にハードウェアで指定でき，テーブルにないIDでメッセージが到着すると割り込みが発生する．[13]で提案されたように割り込みハンドラでそのID用のリングバッファをハードウェアにセットすることでユーザレベル通信を行なう並列プロセスが3つ以上の場合でも対応できる．この機構によりContext ID単位での保護が可能となる．

メモリ空間の保護で述べたのと同様に，並列プログラムマネージャがプロセスに同じContext IDをつけて保護を実現する．

5.2 メッセージ待ちのポーリング

マルチユーザ，マルチタスク環境ではCPUの浪費を防ぐためポーリングによるメッセージ待ちを避けなければならない．

AP1000+のリングバッファは溢れるまで割り込みを発生しない．従ってユーザレベル通信で

のメッセージ待ちはポーリングでの実装となる．CellOS+互換ライブラリでは，ある程度の間ポーリングで待った後マイクロカーネルのthreadDelay()システムコールを行ないsleepさせる．

5.3 page fault処理

仮想記憶上のユーザプロセスではメッセージの領域が物理メモリにあることを保証することができないため，sendやput,get中にネットワークデバイスからメッセージ領域へのアクセスがpage faultを起こす可能性がある．

AP1000+のネットワークハードウェアはpage faultを起こすとそこで状態を保持して停止し，CPUに割り込みをかける．割り込みハンドラはfaultしたページをディスクから読み出しマップする．その後ネットワークハードウェアの再起動を行なう．

6 User level通信の実装と性能評価

マイクロカーネル上で並列プログラムを動かし性能を測定した．UNIXシステムコールを使用せず，デーモンやコマンド，他のプロセスを動作させない状態ではほぼ同等の性能と考えられる．

測定は，CellOS+のsend,recv関数をマイクロカーネル上に移植し，隣接するセル間でメッセージが往復する時間を測定し半分を片道の時間とした．このrecvはポーリングでメッセージ待ちをしている．

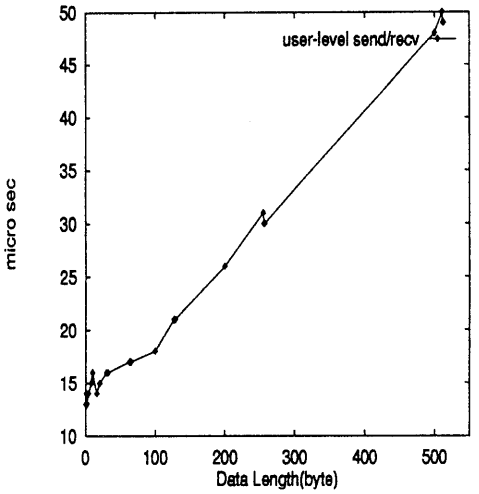

図 8: ユーザレベル通信(1)

4Byteの転送に要する時間は15μsである．またスループットは13.1MByte/sである．ハードウェアのスループットの約半分の性能だが，これは受信側でリングバッファからユーザのメッセージ領域へのコピーが入るためである．

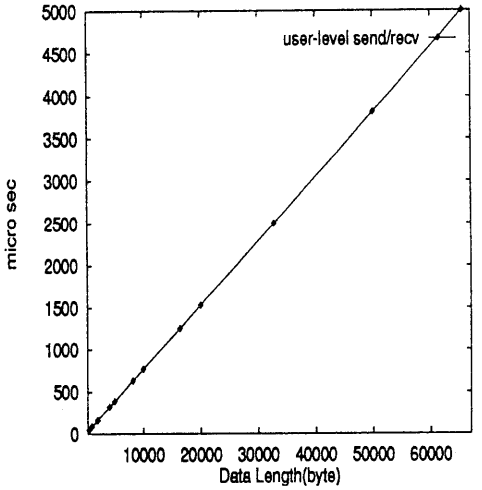

図 9: ユーザレベル通信(2)

7 関連研究

Cenju3のOSはCenju-3/DEと呼ばれる[2]．MACHマイクロカーネル上にオリジナルのサーバーが動作している．カーネル通信はレイテンシが210μs，レイテンシ込みで計算した最大スループットが22MByte/sである[3]．UNIXシステムコールはホスト計算機に依頼する形で実装している．マルチユーザでのユーザレベル通信は，NODOサーバと呼ぶサーバーコールとして実現されている．優先バッファと呼ぶ領域であれば，直接送信バッファに書き込み送信できる[1]．ネットワークハードウェアアクセスはカーネルコールである[4]．通信バッファとして指定された領域上にメッセージを用意しなければならないため，データ領域やヒープ領域のデータをそのまま送信することはできずコピーが必要になる．

SR2001のOSはMACHマイクロカーネルをベースにしたシングルシステムイメージのUNIXである[5]．ユーザレベル通信としては，物理メモリ間でデータを転送する直接メモリ転送機能を提供している[7]．バッファの領域を任意のプロセス間で共有可能である．送信側のバッファと受信側のバッファは，あらかじめコネクションをはってから使用する．Cenju-3/DEと同様にデータ領域やヒープ領域のデータをそのまま送信することはできない．

SP2は各PE上でAIXと呼ぶUNIXが動作してい

る．ユーザレベル通信はシングルユーザのみである[8]．

8 まとめ

マルチタスク，マルチユーザ環境でユーザレベル通信を実現するための問題点を提示し，AP1000+のネットワークハードウェアのマルチプログラミングサポートを利用したユーザレベル通信の実装を示した．

マイクロカーネルIPCでは，短いメッセージのレイテンシはシステムアクタ間で190μs，ユーザアクタ間で239μsであった．長いメッセージにおけるスループットは8.5MByte/sであった．スループットに関してはユーザのメッセージ領域とカーネルのメッセージバッファ間で仮想記憶管理機構のcopy on writeを用い，copyを避けて改善することを考えている．

ユーザレベル通信では，AP1000+のネットワークハードウェアのマルチプログラムサポートを用いて，マルチユーザ，マルチタスク環境においてもOSの介入なしにメッセージ通信が行なえることを示した．短いメッセージのレイテンシとして15μs，スループットとして13.1MByte/sを得た．copyの性能を計算に含めるとほぼハードウェアの性能が得られている．ただしMix上では，実際にはデーモンやスケジューリング，仮想記憶の影響があるはずであり，影響の調査はMix移植終了後の課題となっている．

ユーザレベル通信におけるpage faultの確率を下げる工夫やpageの固定については今後の課題となっている．

AP1000+上のChorus/Mixのシステムイメージは，最終的には全体で1台のUNIXシステムとして見えるシングルシステムイメージを目指している．現在マイクロカーネルの移植を終了し，ネットワークドライバのチューニング，ユーザレベル通信対応が完了した段階である．今後Mix移植を行ない，並列プログラムマネージャの設計を進める予定である．

謝辞

日頃御指導、御助言いただく、HPC本部石井本部長代理、白石部長、GS本部村松主席部長、富士通研究所Pプロジェクト部システム研究部の新開担当部長，岸本主任研，ならびにHPC第2開発統括部，Pプロジェクト部システム研究部の同僚諸氏に感謝いたします。

参考文献

[1] 荒木宏之，高野陽介，小長谷明彦：並列マシンCenju-3上でのユーザレベルIPCに関する考察 -Mach Microkernelをベースとする並列OS DenEnでの実現-，情報処理学会第50回全国大会，4-245,1995.

[2] 高野陽介，Christopher Howson，荒木宏之，菅原智義，小西弘一，小長谷明彦：Machマイクロカーネルをベースとした並列OS DenEnの実現，情報処理学会第50回全国大会，4-243,1995.

[3] 菅原智義，C. Howson，高野陽介，小長谷明彦：並列コンピュータCenju-3用MachにおけるNORMA IPCの実現，情報処理学会研究報告95-HPC-55-14,1995.

[4] 小西弘一，神舘淳，加納健，Christopher Howson，高野陽介：MPI/DE - 並列計算機Cenju-3上のMPIライブラリ- の性能評価，情報処理学会研究報告95-HPC-55-14,1995.

[5] 西門隆，岩嵜正明，藤田不二男，山本徹，柴宮実：SR2001 OSの開発コンセプト，情報処理学会第50回全国大会4-213,1995.

[6] 薗田浩二，西門隆，岩嵜正明，宇都宮直樹，森山健三：SR2001における高速プロセッサ間通信機能，情報処理学会第50回全国大会4-215,1995.

[7] 薗田浩二，西門隆，岩嵜正明，宇都宮直樹，森山健三：SR2001における高速プロセッサ間通信機能，情報処理学会第50回全国大会4-215,1995.

[8] Hubertus Franke, Perter Hochschild, Pratap Pattnaik, Jean-Pierre Prost, Marc Snir: "MPI on IBM SP1/SP2: Current Status and Future Directions", Proc. of the 1994 Scalable Parallel Libraries Conference, IEEE, October 1994.

[9] N. Batlivala, B. Gleeson, J. Hamrick, S. Lurndal, D. Price, J. Soddy, V. Abrossimov : Experience with SVR4 Over CHORUS, Proc. of the Usenix Workshop on Micro-kernels and Other Kernel Architectures, Seattle, WA, April 27-28, 1992.

[10] 石畑宏明，堀江健志，清水俊幸，林憲一，小柳洋一，今村信貴，白木長武：AP1000+: デザインコンセプト，SWoPP 琉球 '94 ARC 研究会 (20), 1994.

[11] 小柳洋一，白木長武，今村信貴，林憲一，清水俊幸，堀江健志，石畑宏明：AP1000+: メッセージハンドリング機構(I) - ユーザーレベルインターフェース -，SWoPP 琉球 '94 ARC 研究会 (21), 1994.

[12] 白木長武，小柳洋一，今村信貴，林憲一，清水俊幸，堀江健志，石畑宏明：AP1000+: メッセージハンドリング機構(II) - システムレベルインターフェース -，SWoPP 琉球 '94 ARC 研究会 (22), 1994.

[13] 白木長武，小柳洋一，今村信貴，林憲一，清水俊幸，堀江健志，石畑宏明：高並列計算機AP1000+のメッセージハンドリング機構，並列処理シンポジウムJSPP'95, 235-240, 1995.