高並列計算機 AP1000+のメッセージハンドリング機構

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[†]

分散メモリ型並列計算機において高いスケーラビリティを得るためには，計算と通信をオーバラップさせ，通信にかかる時間を隠蔽する必要がある．アクティブメッセージのひとつであり，リモートノードのメモリを直接操作するPUTおよびGET操作の効率的な実現は，並列計算機の実行性能向上に貢献する．我々は効率の良いメッセージハンドリング機構に焦点をあて，AP1000の後継機である高並列計算機AP1000+を開発した．AP1000+はPUT/GET操作，リモートload/storeおよびメッセージパッシングを統合的に扱うハードウェアを備えている．さらに低オーバヘッドかつノンブロッキングのユーザインタフェースを備え，同時にマルチユーザ，マルチプロセス環境に必要な保護機構を実現している．本稿では，AP1000+のアーキテクチャを述べ，実機を使用した性能評価を示す．

Message Handling on the AP1000+ Highly Parallel Computer

Osamu Shiraki,[†] Yoichi Koyanagi,[††] Nobutaka Imamura,[†]
Kenichi Hayashi,[†] Toshiyuki Shimizu,[††] Takeshi Horie[†]
and Hiroaki Ishihata [†]

For distributed-memory parallel computers to have high scalability, it is necessary to conceal communication time by overlapping communication and computation. PUT and GET, which operate directly on memory in remote nodes, are examples of active messages, and the efficient implementation of PUT/GET operations contributes to increasing the speed of parallel computers. We developed AP1000+ highly parallel computer, which is an enhancement of the AP1000, focusing on efficient message handling mechanisms. The AP1000+ has an integrated message handling mechanism which supports PUT/GET operations, remote load/store, and message passing in hardware. Furthermore, it provides low-overhead and nonblocking user interface while achieving protection mechanisms for multi-user and multi-process environment. In this paper, we show the architecture of the AP1000+ and basic performance evaluations using the real machine.

1. はじめに

分散メモリ型並列計算機において，計算と通信をオーバラップさせ，通信にかかる時間を隠蔽することは，高いスケーラビリティを得るために欠かせない．この目的のためアクティブメッセージ[17]が提案されている．アクティブメッセージの適用例であるPUT/GETインタフェースは，リモートノードのメモリを直接操作し，その高速な実現は，並列計算機における演算の高速化に大きなインパクトを与える．PUT/GETインタフェースを用いることによって，プロセッサによる計算とデータ通信のオーバラップで演算実行効率を高めることが可能である．

我々は，SEND/RECEIVE型のメッセージパッシングを基本とする高並列計算機AP1000[7),9),15)]の経験から，どのような機能が並列計算機に必要であるかを検討した．その結果，

- 効率の良い通信機構
- マルチ・プログラミングモデル
- マルチユーザ，マルチプロセス

の実現が重要であると認識し，これらの機能をサポートするハードウェアを備えた高並列計算機AP1000+を開発した[8]．本稿では，効率的な通信機構の要求と実現に論点を絞り，AP1000+の設計方針とそのアーキテクチャについて述べ，実機を使用したAP1000+のアーキテクチャサポートの評価を示す．

† 富士通株式会社
Fujitsu Ltd.
†† 株式会社富士通研究所
Fujitsu Laboratories Ltd.

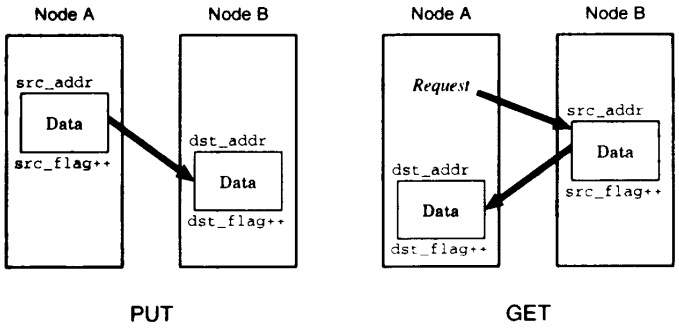


図1 PUT および GET 操作
Fig. 1 PUT and GET operations.

2. 効率的通信へのアーキテクチャサポート

AP1000+では，前身である AP1000 より高速な CPU を採用することにより，基本的な計算能力の向上を図っている．ただし，CPU の性能を向上させるだけで，通信処理の効率が変化しない場合は，システム全体の大きな性能向上は得られないことが，シミュレーションによって予測された[4]．我々は，CPU 速度の向上に見合うだけの通信機構の改善が必須であり，そのために以下の要求に応えるアーキテクチャが必要であると結論した．

2.1 ハードウェアによる PUT/GET の実装

基本的な PUT および GET の動作は以下のとおりである（図 1）．

PUT 送信側が指定したリモートメモリのアドレスに，ローカルメモリブロックをコピーし，データ転送の終了を知らせるために，送信側および受信側のフラグを更新する．

GET 送信側が指定したリモートメモリのアドレスから，リモートメモリブロックの内容を取り出し，ローカルコピーを作成する．データ転送の終了を知らせるために，送信側および受信側のフラグを更新する．

PUT/GET はアクティブメッセージのひとつとして提案された．それをソフトウェアのハンドラを使用して実装すると，アクティブメッセージ・ハンドラを実行している間，ユーザプログラムの実行がブロックされてしまうため，アクティブメッセージの効果が十分に発揮されない．したがって，ユーザプログラムへの割り込みを必要としない PUT/GET の機構が必要となる．

通信を扱うコプロセッサを用意するのもひとつの方法である．その場合，PUT/GET 専用ハードウェアと比較すると，メイン CPU を使用することなく多くの種類のアクティブメッセージを扱うことができるという利点があるが，ハードウェアのコストは高くなり，PUT/GET 操作の性能は専用ハードウェアより落ちる．また我々は，アクティブメッセージは主に PUT/GET 操作に使用されることを考慮し，PUT/GET 専用ハードウェアの方が，リーズナブルであり，かつ有効であると判断した．

2.2 低オーバヘッド通信インタフェース

メッセージ送信の起動オーバヘッドは極力小さいことが望ましい．パラメータの正当性を検査したり，システムデバイスを保護するために，スーパバイザ・コールを必要とするシステムでは，通信コマンドの発行のたびに無視できないオーバヘッドがかかることになる．また，通信を起動できるかどうか検査したり，あるいは通信起動のために特殊な操作が必要になるシステムでは，余分なコードが実行されることになり，それがオーバヘッドとなる．よって，低オーバヘッドな通信を実現するために，ユーザレベルで数ワードのコマンドパラメータの書き込み命令によって通信の起動が可能な機構を用意する．

2.3 ノンブロッキングコマンド発行

メッセージ送信の要求が連続して大量に発生したとき，プロセッサの動作をブロックするのは望ましくない．CPU をブロックすると，計算に使用できる時間をアイドル時間として消費することになり，システム性能を低下させることになる．よって，コマンドを保存して順次発行する，コマンドキューが必要である．

また，連続してキューにデータを書き込んでキューの容量を越えた時に，発生するキューのオーバフローを処理する機構が必要である．キューがフルになったときに CPU をブロックすると，コマンドキューの効果が十分に得られないことが，シミュレーションで観察された[5]．我々は以下の 4 つのモデルを比較した．

(1) キューなし．コマンド発行はブロッキング．
(2) 8 個のコマンドを格納できるハードウェアキュー．
(3) ハードウェアキューと，メモリ上のソフトウェ

アキュー.

(4) 無限の容量を持つ理想的なハードウェアキュー.

モデル(4)が最も良い性能を示し，モデル(1)が最悪であった．直観的には，(2)は(1)より良い性能を示しそうに見えるが，実際にはその性能にほとんど差はなかった．一方，(3)は(4)とほぼ等しい性能を示した．

2.4 転送終了検出

PUT/GETは，リモートノードのメモリを直接操作するため，転送終了を知る手段が必要となる．インタラプトを用いて検出機構を実現した場合，アプリケーションの実行が妨害され，オーバヘッドとなる．オーバヘッドを最小限にするためには，必要なときにメモリ上のフラグを検査する方式が望ましい．したがって，データ転送が終了したときにメモリ上にあるフラグの値を更新する機構が求められる．

2.5 ストライドデータ転送

数値計算アプリケーションの多くには，規則的で反復的な通信パターンが多く現れる．この性質は，配列操作を行い，データ構造の再構成をするようなアプリケーションに共通であり，効率的に通信処理を行うために，数値計算や並列化コンパイラでは，ストライドデータ転送が必要になることが多い[3),4)]．ストライドデータ転送のサポートにより，通信コマンド発行のオーバヘッドを削減し，ネットワークを通過するメッセージ数を減少することができ，またネットワークバンド幅を十分に活用することができる．

2.6 保護機構

計算機に与えた問題が，必ずしもシステム全体の資源を必要とするとは限らないため，高並列計算機のような大規模なシステムを有効に利用するためには，マルチユーザやマルチプロセスをサポートする必要がある．それらのサポートは，システムのスループットの向上に貢献する．

マルチユーザ，マルチプロセスの環境下では，各プロセスの資源を他プロセスから保護する機構が必要となる．ただし，その保護機構のために，性能を落とすことのないように考慮しなくてはならない．そこで，保護のためのハードウェアサポートが必要になる．

オーバヘッド削減のために，ユーザレベルでの通信コマンド発行を許しているシステムでは，ユーザが不正なパラメータを指定する可能性がある．システムはパラメータを検査し，それが不正な場合はエラーを通知しなくてはならない．

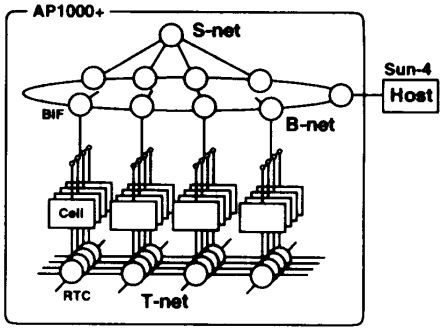


図2 AP1000+システム構成
Fig. 2 AP1000+ system configuration.

3. AP1000+のアーキテクチャ

3.1 概 要

AP1000+のシステム構成を図2に示す．T-net，B-net，S-netの3種のネットワーク，およびこれらのネットワークインタフェースLSIであるRTC，BIFは，AP1000[9)]と同じものを使用し，プロセシングノードであるセルが，AP1000+のために新たに開発された．T-netはセル間の1対1通信に使用される．T-netは2次元トーラストポロジを持ち，ワームホールルーティングを行う．各4方向について，25 MB/sのバンド幅を持つ．S-netはセル間の同期に使用される．ツリートポロジを持ち，高速なバリア同期の実現が可能である．B-netは放送とデータの集配に使用される．ホストコンピュータからのプログラムやデータのダウンロード，結果の収集は，B-netを使用して行われる．

AP1000+のセル構成を図3に示す．各セルは，50 MHzのSuperSPARCプロセッサ，メッセージコントローラ（MSC+），メモリコントローラ（MC），DRAM，およびネットワークLSI（RTC，BIF）で構成される．SuperSPARCとMSC+とMCは，SuperSPARCのバスであるV-Busで相互に接続されている．

MSC+とMCが，メッセージハンドリングのインタフェースおよび機構を提供している．主記憶として用いるセルメモリは，最大64 Mバイトを実装できる．キャッシュはSuperSPARCの内部キャッシュをwrite-throughで動作させ，2次キャッシュは持たない．

3.2 PUT/GETの動作

AP1000+では，PUT/GET，SEND/RECEIVE，リモートload/store☆を統一したメッセージハンドリ

☆ リモートload/storeは，CPUが特殊な操作をすることなく，リモートノードのメモリに対して通常のloadおよびstore操作を行う機能である．現在，機能が限定されており評価も十分になされていないため，本論文では詳細に説明しない．

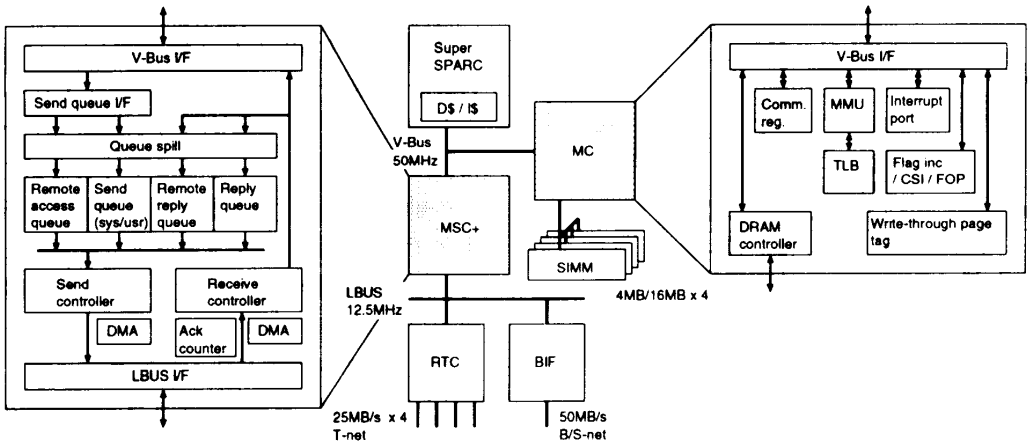


図3 AP1000+セル構成
Fig. 3 AP1000+ cell configuration.

ング機構で扱うために，PUT をベースにし，そのバリエーションとして複数のインタフェースを実現している[8]．これにより，シンプルなハードウェアで柔軟なメッセージハンドリングを可能としている．AP1000+に求められる各種機能が MSC+，MC でどのように実現されているかを以下に示す．

3.2.1 PUT

PUT/GET 起動のオーバヘッドを極力少なくするために，プロセッサからの PUT/GET の要求は，MSC+のセンドキューと呼ぶメモリマップされたあるアドレスに対してコマンドパラメータ列を書き込むだけで起動できる機構として実現する．**図4** に PUT コマンドの動作を示す．

コマンドには，パラメータとして宛先セル ID と書き込みアドレス，および送信すべきデータのアドレスが含まれる．MSC+の送信コントローラがコマンドを解釈し，論理アドレスから物理アドレスへの変換を行い，DMA を起動し，メモリからネットワークにデータを送出する．データ転送がすべて終了したら，MSC+は PUT パラメータ中に指定されたアドレスのフラグ値を更新する．

PUT メッセージが宛先セルに到着すると，MSC+内の受信コントローラがメッセージの処理を開始する．受信コントローラはメッセージヘッダを解釈し，PUT パラメータを得る．そして宛先論理アドレスを物理アドレスへと変換し，後続のデータを宛先領域へと格納する．データ転送が終了すると，MSC+は受信フラグの値を更新する．

データの格納およびフラグの更新時には，対応するキャッシュラインは V-Bus により自動的にインバリデートされる．したがって，メモリとキャッシュのコンシステンシを維持するためのソフトウェアは不要で

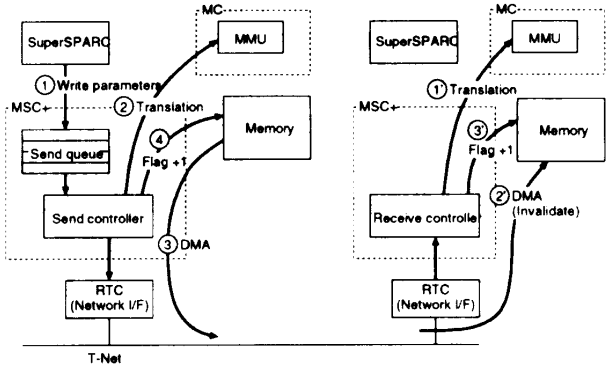


図4 PUT の動作
Fig. 4 PUT operation.

あり，高スループットの通信が実現される．

ユーザレベルで通信コマンドを発行できるため，アドレス・パラメータとしてどんな値でも指定することが可能である．したがって，保護のために，アドレスは論理アドレスで指定されなくてはならない．転送データのメモリアクセスの際に，MC 内にある MMU が起動され，SuperSPARC が用いるものと同じ形式のページテーブルを参照し，論理-物理アドレスの変換が行われる．MMU による変換は，通信レイテンシに直接影響するために，高速化が必須である．そのために MC は，256 K ページに対して 64 エントリ，4 K ページに対して 256 エントリの translation lookaside buffer（TLB）を持つ．

3.2.2 GET

図5 に GET コマンドの動作を示す．GET の要求は PUT と同様 MSC+のセンドキューにコマンドワード列を書き込むことで起動される．コマンドには，パラメータとして宛先セル ID と読み出しアドレス，および受信データの格納アドレスが含まれる．送信コントローラはそのコマンドをネットワークに送出する．

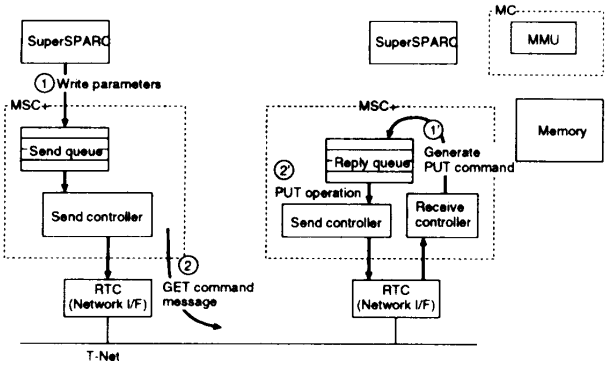


図5 GET の動作
Fig. 5 GET operation.

コマンドを受信したセルの受信コントローラは，そのコマンドを送信元への PUT コマンドに再構成し，リプライキューへ書き込むことで GET を実現する．換言すると，受信側セルでは GET コマンドを PUT コマンドとして扱う．このように，受信側でプロセッサを介在させることなく GET 操作を実現している．

以上に述べたように，通信コマンドの発行以外は，CPU が介在せずに PUT/GET 通信処理を行えることが，AP1000+の特徴である．しかしその実現のために，キャッシュを write-through 動作にする必要があった．キャッシュを write-back にすると，通信処理におけるデータ転送の前に，キャッシュと主記憶との一貫性をとるために，CPU によるキャッシュの操作が必要になるためである．

しかし，キャッシュを write-through にすると，CPU の write 操作のたびに主記憶へのアクセスが生じるため，演算性能の低下が問題となる．ただし，AP1000+で採用した SuperSPARC は，8 段の write バッファを持ち，8 個のストア命令は，後続の命令をブロックすることなく実行できるため，演算性能への影響は少ない．

一方，write-through にしない場合は，送信処理起動時のキャッシュ操作だけでなく，受信時のキャッシュ操作が必要になる．たとえば PUT/GET 要求を受け取った場合，受信した PE では CPU に割込みを要求し，キャッシュの内容を主記憶へと反映させた後に，データを転送しなくてはならない．また，キャッシュ処理に加えて，SuperSPARC では割込みのオーバヘッドが大きく，演算処理への影響が無視できない．

以上に述べた理由により，AP1000+では，write-through 動作にすることによる演算性能の低下は，write-back にした場合の影響よりも軽微であると判断し，write-through を採用した．

3.2.3 フラグ更新機能

PUT/GET のメモリ操作の完了を知るために，DMA が終了するとコマンドパラメータとして指定されたメモリアドレスのフラグ値を +1 する機構を持つ．フラグ更新をハードウェアで行うため，プロセッサの動作を妨げない．またフラグ値の更新は，値のストアではなく，インクリメントで行われるため，複数の PUT/GET の完了を，1 つのフラグの検査により検出できる．

3.2.4 ストライドデータ転送

AP1000+は，1 次元のストライドデータ転送をハードウェアでサポートしている．1 次元までのサポートとしたのは，ソフトウェアで高次元のストライドデータ転送を実現するためのオーバヘッドと，ハードウェアコストとのトレードオフによる．AP1000+では，1 次元のストライド転送がストライドなしの PUT/GET 操作と同様の方法で発行でき，ストライド転送発行のオーバヘッドは数ワードのストア命令のコストですむため，高次元のストライド転送は 1 次元のストライド転送を連発することで実現できる．

もし，システムがストライド転送をハードウェアでサポートしない場合は，ソフトウェアで実現する必要がある．ひとつの方法は，ストライドデータを分割して，複数の PUT コマンドを発行することである．しかし，この方法ではメッセージの数とコマンド発行のオーバヘッドが増大する．また，高いスループットを得るためには，メッセージサイズはある程度大きい必要があるが，この方法ではメッセージサイズが小さいため，ネットワークバンド幅を活用することができない．もうひとつの方法は，ストライドデータを 1 つのブロックにパックして，受信側でそれをアンパックすることである．この方法ではネットワークのバンド幅を活用することが可能だが，ソフトウェアのオーバヘッドが送信側および受信側の双方で非常に大きいという欠点がある．

3.2.5 メッセージパッシング —SEND/RECEIVE

SEND/RECEIVE は，宛先がアドレスを持たないバッファ領域となっている特殊な PUT として実現されている．受信したセルでは，指定されたメモリアドレスに直接受信データを格納する代わりに，専用のバッファ領域に書き込むことで実現する（**図 6**）．各セルはメッセージパッシング用の 3 つのリングバッファを持ち，3 つの異なるプロセスがそれらを使用する．プロセスが 3 つ以上存在し，到着したメッセージに対応するバッファが有効になっていない場合は，MSC+が

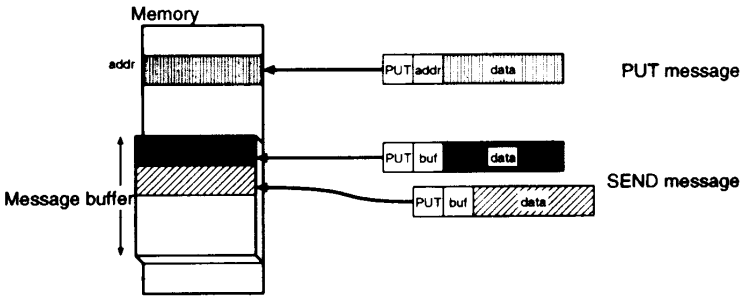


図6 PUT と SEND の違い
Fig. 6 Difference between PUT and SEND.

オペレーティングシステムに割り込み，プロセスに対応したバッファの用意を要求する．

メッセージバッファのリードポインタを示す MSC+内のレジスタアドレスは，3チャンネルが別のページに割り当てられており，そのアクセス権をプロセスごとに設定可能である．このようにレジスタの保護が実現されるため，メッセージバッファのアクセスに関してスーパバイザ・コールの必要はなく，ユーザレベルで処理が可能である．

3.3 通信コマンドキュー

3.3.1 センドキューおよびリプライキュー

MSC+は内部の RAM に，5本の通信コマンドキューを持つ．

送信用キューとして，ユーザの PUT/GET 用，システムの PUT/GET 用，およびリモート load/store 用の3本がある．PUT/GET のセンドキューはシステムとユーザの2つがあり，ユーザのキューに書き込み途中でまだ完結していないコマンドがあっても，それを退避することなくシステムモードで通信コマンドを発行できる．また，リモート load/store を PUT/GET のリクエストより優先させて処理させるために，キューが独立している．リモートメモリからの load は，データが返ってくるまでの間，プロセッサの動作をブロックさせてしまうため，その処理を優先しないとアイドル時間の増加を招くためである．

リプライキューとして，PUT/GET 用とリモート load/store 用の2本のキューを持つ．センドキューと同様，リモート load/store のリプライは，PUT/GET よりも先行して処理される．

3.3.2 キュー・オーバフロー・ハンドリング機構

センドキュー，リプライキューはともに MSC+内部の RAM で実現されているが，その容量には限界（合計 256 ワード）があるため，オーバフローが発生する可能性がある．このときに，プロセッサの動作をブロックさせることは，大きな性能低下を招く[5]．そこで MSC+は，オーバフローしたデータを主メモリに確保したバッファ領域に自動的に退避し，オーバフロー時においてもプロセッサをブロックしない機構を備えている．キューが空になったとき，MSC はオペレーティングシステムに割り込み，退避バッファからキューへのリストアを要求する．

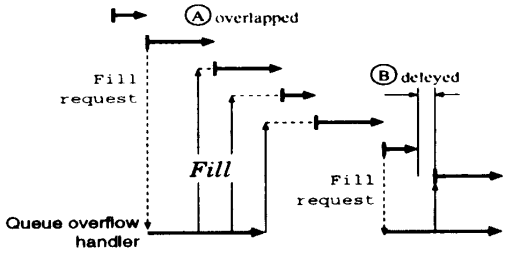


図7 キューオーバフローハンドラの動作
Fig. 7 Action of queue overflow handler.

図7に，キューオーバフローハンドラの動作を示す．キューから最後のコマンドが出力されたら，MSC+はオペレーティングシステムに割り込み，オーバフロー・データをキューへ書き戻すように要求する．その最後のデータ転送に長い時間がかかる場合は，それがオーバフローハンドラの操作を隠蔽する（図中A）．しかし，時間が短い場合は，リストアする処理がネットワーク上で隙間となって見えてくる（図中B）．しかし，分散メモリマシンにおいて使用されるメッセージサイズはハンドリング時間の隠蔽に十分なほど大きく，またリストア機構のハードウェアによる実現は非常にコストが高いために，ハードウェアであふれさせ，ソフトウェアで詰める機構がリーズナブルであると判断した．

3.4 ユーザインタフェース

3.4.1 PUT/GET ユーザインタフェース—センドキュー・エントリ

PUT/GET 要求の発行は，プロセッサが，あるユーザ領域のメモリアドレスに対して割り付けられているセンドキューにコマンドワード列を書き込んでいくことで実現される．

コマンド発行の基本的な形式を図8に示す．コマンドの種類や動作の意味は書き込んだセンドキューのア

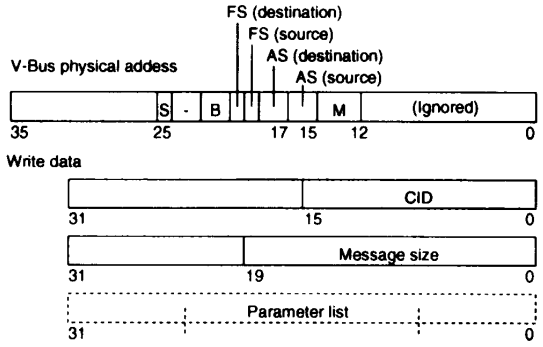


図8 センドキュー・エントリアドレスおよびコマンド基本フォーマット

Fig. 8 Send queue entry address and basic of command format.

ドレスによって決まる．図の最上段は，センドキューの物理アドレスを示す．このアドレスに対し，最初に相手先セルID，次にPUT/GETする領域の長さ（ワード数で指定）を書き込み，以降はコマンドに対応したパラメータワードが続く．アドレスのフィールドにはコマンドの意味が与えられている．

3.4.2 コマンドの分類

センドキューに書き込むコマンドの種類は，キューエントリの物理アドレスによって決まる．しかしその中には，ユーザレベルでの発行を禁止したい操作も含まれているので，システムレベル，ユーザレベルで発行を区別して制限できる機構が必要となる．

AP1000+は，MMUの保護機能を利用してこの機構を実現している．センドキューへのコマンドの書き込みアドレスは，図8のようにビット12から上位のビットを使用している．よって，4Kバイト単位のページテーブルを適切に設定することで，ユーザレベルで発行できるコマンドの種類を制限することができる．この機構により，スーパバイザコールを必要とせずに，ユーザがコマンドを発行することができ，通信オーバヘッドの削減が可能になっている．

4. 性能評価

我々は，これまでに述べたアーキテクチャを持つ並列計算機AP1000+を開発し，その上でいくつかの性能評価を行った．この章では，その結果の分析を行う．なお，この章で述べられるすべての実験は，オペレーティングシステムの下で行われた．

4.1 基本性能 — ピンポン性能

【実験】 セルは，指定されたサイズのデータを，隣接するセルにPUTする．宛先となったセルは，メッセージの受信を確認したら，返信のPUTをする．セルの組はこのピンポン操作を100回行い，平均レイテンシを求める．

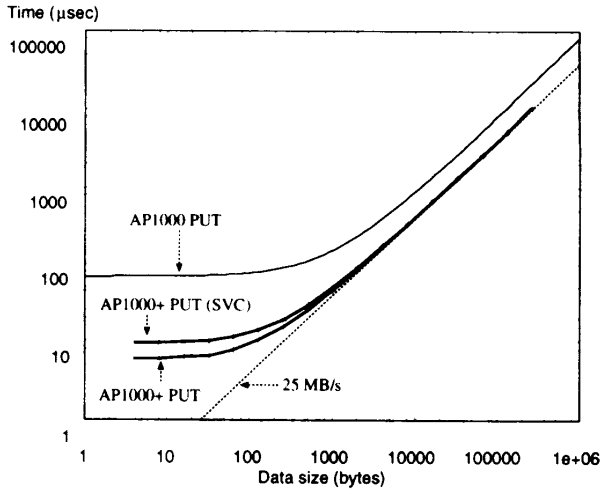


図9 PUTの性能

Fig. 9 Performance of PUT.

表1 PUTの性能

Table 1 Performance of PUT.

Type		Time (μs)
AP1000+	PUT (USR)	$5.1 + 0.040n$
	PUT (SVC)	$8.6 + 0.040n$
AP1000	PUT	$68 + 0.081n$

(n は転送バイト数)

比較のために，スーパバイザコールを使用したPUTの性能も測定した．スーパバイザプログラムはパラメータの範囲を検査し，それらをセンドキューに書き込む．このパラメータ検査は不十分なものなので☆，ここで示される性能は実際より楽観的なものといえる．

【結果および考察】 **図9** にPUTの性能を示す．性能を示す曲線は，転送データサイズの増大とともに，ネットワークバンド幅から求められる理想的なスループットを示す“25 MB/s”に近付く．AP1000はネットワークバンド幅の半分しか性能を引き出せなかったが，AP1000+はネットワークを十分に活用することができる．

表1 に，PUTを発行してから，受信側でデータの到着を確認するまでににかかる時間を示す．スタートアップオーバヘッドは，スーパバイザ・コールを使用したものと比較し，3.5μ秒小さくなっている．

AP1000においてネットワークバンド幅を十分に活用できない理由は，データ転送の前にキャッシュメモリの操作をしなくてはならないためである．送信側セル

☆ パラメータの値が指定した範囲であるかどうかだけを検査している．現実には，通信を要求したプロセスが確保しているメモリ領域との比較や，論理アドレスをサポートしていない場合は物理アドレスへの変換などが必要になる．

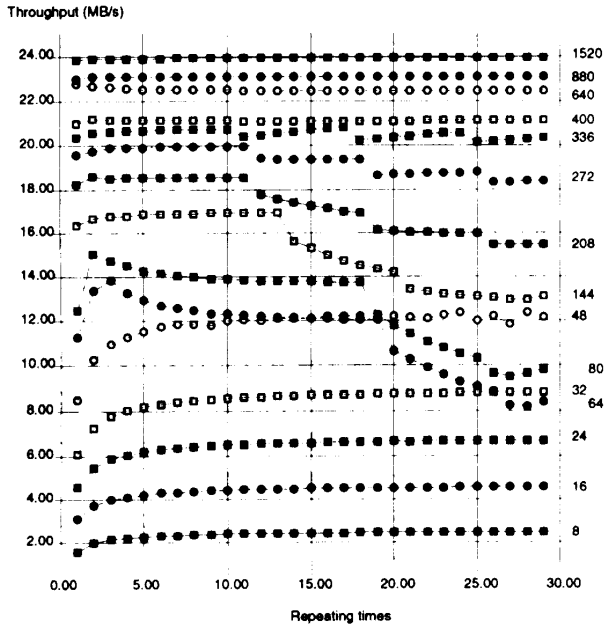


図10 連続 PUT の性能
Fig. 10 Performance of repeating PUT.

においてはキャッシュのポスト☆が必要になり，受信側セルにおいては受信領域に対応するキャッシュラインをインバリデートしなくてはならない．一方 AP1000+においては，通信のときに明示的なキャッシュ操作を必要としないために，理論的なネットワークバンド幅にきわめて近い性能が達成できた．

4.2 キューオーバフローハンドリング

【実験】 セルは，指定されたサイズのデータを隣接するセルに PUT し，それを指定された回数繰り返す．そして，MSC+内部の送信コントローラが転送を終了し，送信フラグを更新するのを待つ．データサイズを 8～1,520 バイトに変化させ，繰返し回数を 1～29 回に変化させた．

【結果および考察】 図10 に結果を示す．X 軸は繰返し回数を表し，Y 軸は転送時間から計算したスループットを表す．各ラインはデータサイズごとの性能を示す．データサイズが小さいとき (8～48 バイト)，繰返し回数の増加とともにスループットが増加し，低いレベルで飽和する．データサイズが大きいとき (400～1,520 バイト)，繰返し回数にあまり関係なく，ほぼ一定のスループットを示す．

しかし，データサイズが中程度のとき (64～336 バイト)，繰返し回数がある値に達したときに，性能が突然低下する．たとえば，“サイズ = 64 バイト”の性能は繰返し回数が 19 の位置で低下する．この，グラフが折れる位置は，キューオーバフローが起きるかどうかの閾値を示しており，いったんキューオーバフローが生じると，性能は低下する．データサイズが小さい場合には，PUT の発行速度がキューから出ていく速度より遅いために，オーバフローは起きていない．

データサイズが大きいときにもキューオーバフローは生じているが，その影響は性能に現れていない．その理由は，オーバフローデータのハンドリング時間が，データ転送時間に隠蔽されているためである (図7参照)．換言すると，サイズが 400 バイトの転送があれば，オーバフローハンドリングの時間を隠蔽することができる．

【ストライドデータ転送の実装に関する考察】 この実験結果によると，小さなデータサイズのメッセージはバンド幅を活かすことができず，またキューオーバフローハンドリングの影響を大きく受ける．

本稿の中で述べたように，ストライドデータ転送を PUT 操作の繰返しによって実現することは可能である．しかし，数値演算においては，非常に小さなアイテムサイズのストライド転送が要求されることが多い．小さなデータサイズの PUT を連発することは，キューオーバフローを多く生じさせ，そのハンドリング時間を増大させる．したがって，最低でも 1 次元のストライドデータ転送は，ハードウェアで実装する必要がある．

いったん 1 次元ストライドデータ転送が適用されると，メッセージサイズが十分大きくなり☆☆，ネットワークバンド幅を活用し，オーバフローハンドリングのオーバヘッドを隠蔽することができる．したがって，1 次元データ転送を使用して，さらに高次元のストライドデータ転送を効率的に実現することができる．この実験の結果は，1 次元ストライドデータ転送をハードウェアで実現するという判断を裏付けるものといえる．

4.3 ストライドデータ転送

【実験】 サイズが 256，1K，4K または 16KB の配列を与える．セルは与えられた配列の内容を，以下の方式で実装されたストライドデータ転送を用いて，隣接するセルに転送する．

☆ ライトバック方式のキャッシュにおいて，キャッシュに書かれている新しいデータを主メモリに反映させる操作．

☆☆ たとえば，ストライドデータ転送がサポートされないとき，並列化コンパイラ VPP Fortran でコンパイルした，SPEC ベンチマークの TOMCATV は，1 セルあたり約 9,600 個の 8 バイト PUT を生成することが観察された．1 次元ストライド転送を適用した後は，38 個の約 2,000 バイトのストライド PUT を生成した．

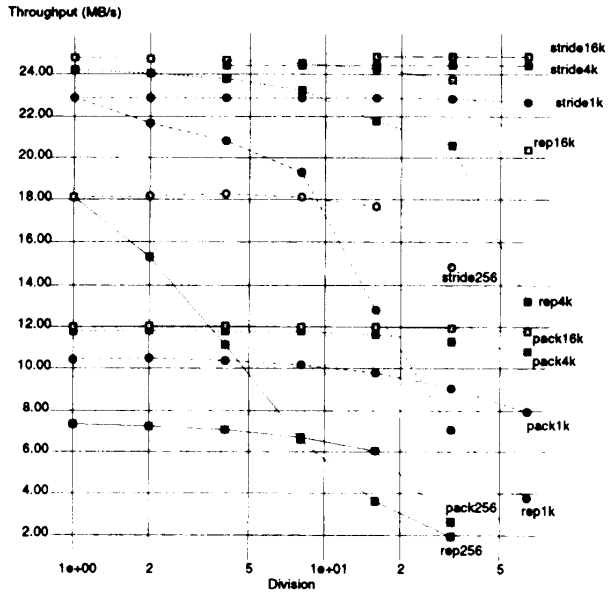


図11 ストライドデータ転送の性能
Fig. 11 Performance of stride data transfer.

STRIDE：ハードウェア．1次元ストライドデータ転送がハードウェアでサポートされている．

REP：**複数PUT**．送信セルは，各データ要素ごとにPUTを発行する．

PACK：ソフトウェア．送信セルにおいて，ストライドデータを1つのブロックにしてPUTする．受信セルにおいて，ソフトウェアを用いて，受信したデータを適切な位置に書き込む．

配列の分割数を1～64に変化させて実験を行った．分割数が増えるとともに，アイテムサイズは小さくなる．

【結果および考察】 図11に結果を示す．X軸は分割数を表し，Y軸は転送時間から計算されたスループットを表す．各ラインは，実装方式と配列のサイズを変化させたときの性能を示している．たとえばstride16kは，STRIDE方式で配列サイズ16 KBのデータを転送したときの性能を表す．

PACKは，送信側および受信側セルにおけるコピーオーバヘッドにより，性能が低く抑えられるために，分割数が小さいときに，REPよりも低い性能を示す．しかし，分割数が大きくなりデータサイズが小さくなると，REPは命令発行オーバヘッドが大きくなり，ネットワークスループットも得られないことから，REPはPACKよりも性能が低くなる．

STRIDEの性能を示すラインは，ほぼ水平である．これは，分割数が大きくなったときでも，STRIDEがストライドなしのPUTとほぼ同じスループットを得ていることを意味している．ただし，STRIDE256の分割数32においては，性能が低くなっている．これは，その条件でのアイテムサイズが8バイトになっており，V-Busのバンド幅が活かせなくなっているためである☆．

5. 関連研究

リモートノードメモリの直接操作をソフトウエアで実現したものとして，von Eickenらによるactive message[17]，これを言語レベルで用いたSplit-C[2]がある．

ハードウェアで実現するものとして，メッセージパッシングを基本とするCRAY T3D[13]，富士通VPP500[16]，Meiko CS-2[6]など，またキャッシュコヒーレントを基本とするMITのAlewife[10),11)]，スタンフォード大学のFLASH[12]，ウィスコンシン大学のTyphoon[14]などがある．

VPP500[16]は，ノード間のメモリ転送ハードウェアを備えているが，起動のためのオーバヘッドが大きい．Meiko CS-2は，送信コマンドをユーザレベルで通信用コプロセッサのキューに書き込んで起動するという点で，AP1000+に似ているが，データ転送はすべて32バイトのパケットを単位として行われており，大規模データ転送時のオーバヘッドが大きい．

プリンストン大学のSHRIMP[1]は，ローカルな書き込みをリモートのアドレスにマッピングすることにより，通信を行う．Bulkデータ転送の機構も備えているが，ページサイズを越えるデータを効率的に転送することが困難である．また，リモートのデータをフェッチすることはできない．

Alewife[10),11)]，FLASH[12]，Typhoon[14]は，キャッシュコヒーレントのアーキテクチャをベースに，その上で高スループットのbulkデータ転送の統合を追求している．AP1000+はこれらのアーキテクチャとは異なり，PUT/GETを基本とする高スループットな通信をベースにし，低オーバヘッドのユーザインタフェースを提供することで，細粒度データ通信の効率化も同時に目指している．さらに，実アプリケーションにおいてPUT/GETインタフェースをより効率的に適用できるように，ストライドデータ転送やフラグ更新の機構を備えている．

6. おわりに

効率的なメッセージハンドリング機構を提供する，高並列計算機AP1000+のアーキテクチャおよびユー

☆ V-Busには1つのトランザクションで64バイトの転送を行えるバースト転送機能があり，データサイズが大きいときには高いスループットが得られる．

ザインタフェースを示し，実機を用いた実験によりその基本的な性能を示した．

我々は，PUT/GET 通信インタフェースは分散メモリ並列計算機では最も有効な機構であると判断し，そのハードウェアサポートを AP1000+に持たせた．

PUT/GET の効果を十分引き出すために，AP1000+は低オーバヘッドのユーザインタフェースを提供する．ユーザレベルで通信コマンドを発行することができ，通信のためにスーパバイザ・コールを必要としない．また，ユーザレベルコマンド発行を実現するための保護機構を持つ．

AP1000+は通信コマンドキューを備え，コマンドをバッファリングして，ノンブロッキングなユーザインタフェースを実現している．この機構により，PUT/GET の利点である，通信と計算のオーバラップが可能になる．さらに，この特長を引き出すことができるように，キューオーバフローハンドリング機構を持たせている．

数値演算に有効であり，メッセージ数を減らすことのできる，ストライドデータ転送をサポートしている．AP1000+は 1 次元データ転送をハードウェアで実装し，高次元のストライドデータ転送は 1 次元のものを繰り返すことで実現する．低オーバヘッドかつノンブロッキングのユーザインタフェースが，この実装をリーズナブルなものにしている．

今後は，AP1000+の提供する機構がアプリケーションの実行に対してどのような効果があるかを評価し，また，AP1000+の機構をより効果的に活用できるアプリケーションの記述方式や，並列化コンパイラの研究をすすめていく．

参 考 文 献

1) Blumrich, M.A., Li, K., Alpert, R., Dubnicki, C., Felten, E.W. and Sandberg, J.: Virtual Memory Mapped Network Interface for the SHRIMP Multicomputer, *The 21st Annual International Symposium on Computer Architecture* (1994).
2) Culler, D.E., Dusseau, A., Goldstein, S., Krishnamurthy, A., Lumetta, S., Eicken, T. and Yelick, K.: Parallel Programming in Split-C, *Supercomputing '93* (1993).
3) 土肥実久，林 憲一，進藤達也：ストライドデータ転送機構を用いたコード生成，SWoPP 琉球 '94，HPC 研究会 (1994).
4) Hayashi, K., Doi, T., Horie, T., Koyanagi, Y., Shiraki, O., Imamura, N., Shimizu, T., Ishihata, H. and Shindo, T.: AP1000+: Architectural Support of PUT/GET Interface for Parallelizing Compiler, *Sixth International Conference on Architectural Support for Programming Languages and Operating Systems*, ACM, pp.196–207 (1994).
5) Hayashi, K., Doi, T., Horie, T., Koyanagi, Y., Shiraki, O., Imamura, N., Shimizu, T., Ishihata, H. and Shindo, T.: AP1000+: Architectural Support for Parallelizing Compilers and Parallel Programs, *International Symposium on Parallel Architectures, Algorithms, and Networks (ISPAN '94)*, IEEE (1994).
6) Homewood, M. and McLaren, M.: Meiko CS-2 Interconnect Elan-Elite Design, *Hot Interconnects '93*, pp.2.1.1–4 (1993).
7) Horie, T., Hayashi, K., Shimizu, T. and Ishihata, H.: Improving AP1000 Parallel Computer Performance with Message Communication, *The 20th Annual International Symposium on Computer Architecture*, pp.314–325 (1993).
8) 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).
9) Ishihata, H., Horie, T., Inano, S., Shimizu, T. and Kato, S.: An Architecture of Highly Parallel Computer AP1000, *IEEE Pacific Rim Conf. on Communications, Computers and Signal Processing*, pp.13–16 (1991).
10) Kranz, D., Johnson, K., Agarwal, A., Kubiatowicz, J. and Lim, B.: Integrating Message-Passing and Shared-Memory: Early Experience, *Fourth ACM SIGPLAN Symposium on Principles & Practice of Parallel Programming*, ACM, pp.54–63 (1993).
11) Kubiatowicz, J. and Agarwal, A.: Anatomy of a Message in the Alewife Multiprocessor, *International Conference on Supercomputing*, pp.195–206 (1993).
12) Kuskin, J., Ofelt, D., Heinrich, M., Heinlein, J., Simoni, R., Gharachorloo, K., Chapin, J., Nakahira, D., Baxter, J., Horowitz, M., Gupta, A., Rosenblum, M. and Hennessy, J.: The Stanford FLASH Multiprocessor, *The 21st Annual International Symposium on Computer Architecture* (1994).
13) Oed, W.: The Cray Research Massively Parallel Processor System CRAY T3D, available through ftp from ftp.cray.com (1993).
14) Reinhardt, S.K., Larus, J.R. and Wood, D.A.: Tempest and Typhoon: User-Level Shared Memory, *The 21st Annual International Symposium on Computer Architecture* (1994).

15) Shimizu, T., Horie, T. and Ishihata, H.: Low-Latency Message Communication Support for the AP1000, *The 19th Annual International Symposium on Computer Architecture*, pp.288–297 (1992).
16) Utsumi, T., Ikeda, M. and Takamura, M.: Architecture of the VPP500 Parallel Supercomputer, *Supercomputing '94*, pp.478–487 (1994).
17) von Eicken, T., Culler, D.E., et al.: Active Messages: A Mechanism for Integrated Communication and Computation, *19th International Symposium on Computer Architecture*, pp.256–266 (1992).

(平成7年8月31日受付)
(平成8年3月12日採録)

白木 長武（正会員）

1964年生．1991年東京大学工学部計数工学科卒業．1993年同大学院工学系研究科情報工学修士課程修了．同年（株）富士通研究所入社，並列計算機に関する研究開発に従事．現在，富士通（株）HPC本部に勤務．

小柳 洋一

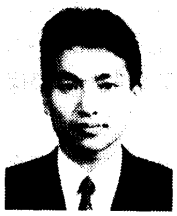
1966年生．1990年東京工業大学工学部情報工学科卒業．1992年同大学院修士課程修了．同年（株）富士通研究所入社．並列計算機に関する研究開発に従事．電子情報通信学会会員．

今村 信貴（正会員）

1967年生．1990年九州大学工学部電子工学科卒業．1992年同大学院総合理工学研究科修士課程修了．同年（株）富士通研究所入社．並列計算機に関する研究開発に従事．現在，富士通（株）HPC本部に勤務．

林 憲一（正会員）

1967年生．1991年東京大学工学部計数工学科卒業．同年（株）富士通研究所入社．並列計算機に関する研究開発に従事．現在，富士通（株）HPC本部に勤務．



清水 俊幸（正会員）

1964年生．1986年東京工業大学工学部電子物理工学科卒業．1988年同大学院理工学研究科情報工学修士課程修了．同年（株）富士通研究所入社，現在に至る．並列計算機に関する研究開発に従事．1992年電子情報通信学会論文賞受賞．電子情報通信学会会員．

堀江 健志（正会員）

1962年生．1984年東京大学工学部電気工学科卒業．1986年同大学院修士課程修了．同年（株）富士通研究所入社，現在に至る．並列計算機に関する研究開発に従事．1992年電子情報通信学会論文賞受賞．

石畑 宏明

1957年生．1980年，早稲田大学理工学部電子通信学科卒業．同年（株）富士通研究所入社．画像処理システムの研究，並列コンピュータアーキテクチャの研究に従事．現在，富士通（株）HPC本部に勤務．工学博士．元岡賞，電子情報通信学会論文賞受賞．