AP1000+：並列化コンパイラをサポートするアーキテクチャ

林 憲一[†] 土肥実久[†] 堀江健志[††]
小柳洋一[†] 白木長武[†] 今村信貴[†]
清水俊幸[††] 石畑宏明[†] 進藤達也[†]

分散メモリ型並列計算機はそのスケーラビリティから，大規模問題を解くための次世代スーパーコンピュータの有力な候補の一つであり，これらの並列計算機のために，プログラミングの容易化や既存ソフトウェア資産継承を目的としてHPF, Fortran D, VPP-Fortranなどの言語が提案されている．しかしこれまでの分散メモリ型並列計算機は，必ずしもこれらの並列化コンパイラが必要とする機能を十分にサポートしてはいなかった．我々は並列化コンパイラが必要とする機能を検討し，それらをサポートするアーキテクチャを提案した．そしてそのアーキテクチャを実現する分散メモリ型高並列計算機AP 1000+を開発した．本論文ではVPP-Fortran等を用いて書かれた大規模数値計算のアプリケーションによってAP 1000+の性能をシミュレーションによって評価する．

AP1000+ : Architectural Support for Parallelizing Compilers

KENICHI HAYASHI,[†] TSUNEHISA DOI,[†] TAKESHI HORIE,[††]
YOICHI KOYANAGI,[†] OSAMU SHIRAKI,[†] NOBUTAKA IMAMURA,[†]
TOSHIYUKI SHIMIZU,[††] HIROAKI ISHIHATA [†] and TATSUYA SHINDO [†]

The scalability of distributed-memory parallel computers makes them attractive candidates for solving large-scale problems. New languages, such as HPF, Fortran D, and VPP Fortran, have been developed to enable existing software to be easily ported to such machines. Many distributed-memory parallel computers have been built, but none of them support the mechanisms required by such languages. We studied the mechanisms required by parallelizing compilers and proposed a new architecture to support them. Based on this proposed architecture, we developed a new distributed-memory parallel computer, the AP1000+, which is an enhanced version of the AP1000. Using scientific applications in VPP Fortran and C, such as NAS parallel benchmarks, we simulated the performance of the AP1000+.

1. はじめに

分散メモリ型並列計算機はそのスケーラビリティから，大規模問題を解くための次世代スーパコンピュータの有力候補の一つである．このような並列計算機が広く使われるためには，プログラマにとって不連続なアドレス空間やメッセージ通信等，ハードウェアの詳細を意識せずにプログラミングが行えることが重要である．特にプログラマに対してグローバルアドレス空間をサポートすることは，プログラミングの容易化や既存ソフトウェア資産の継承という観点から有用である．こうした要求に応えるためにHPF[6], Fortran D[7], VPP Fortran[16]などの言語が提案されている．

† 富士通(株)
Fujitsu Ltd.
†† (株)富士通研究所
Fujitsu Laboratories Ltd.

しかしこれまでの分散メモリ型並列計算機は，必ずしもこれらの並列化コンパイラが必要とする通信機能を十分にサポートしてはいなかった．HPF等のコンパイラが生成するコードを分散メモリ型の並列計算機の上で実用的に利用するためには，これらのコンパイラの必要とするデータ通信の機能を考慮したアーキテクチャが求められる．

1.1 並列化コンパイラに必要とされる機能

我々はHPFとVPP FortranをAP 1000上に実現した経験[3),20)]と他の並列言語の研究[2),7),11),19)]から，分散メモリ型並列計算機上でこれらの並列化コンパイラによって生成されたコードを効率良く実行するために必要となる通信機能を検討した．その結果，ダイレクトリモートデータアクセス，ブロック一括転送，ストライドデータ転送，バリア同期，グローバル演算が必要である，と結論した．

センド・レシーブを基本としてコード生成を行うも

のとしてはこれまでFortran D[7], Kali[11], Oxygen[19]など様々な方法が提案されている．しかしこれらの方式では，コンパイル時に送受信のペアを求めなければならず，配列の添字に配列が来るような，コンパイル時にペアの求められない不規則なデータアクセスに対しては，放送などを用いる必要があった．これに対してダイレクトリモートデータアクセス方式[20]では，コンパイル時に送受信のペアを求める必要がなく，また不規則なデータアクセスに対する一般的な実現が可能である．また，ダイレクトリモートデータアクセス方式では，バッファを介さずに直接メモリにデータが書き込まれるので，コピー等のオーバヘッドのない低レイテンシ通信が実現できる．さらにダイレクトリモートデータアクセスを用いてコードを生成する方式では，センド・レシーブを用いてコードを生成するよりもコンパイラ自体の作成が容易になるという利点もある．

大規模行列のデータの再分配などに用いられるブロック一括転送は，実行性能を向上させるために不可欠な通信方式である．またこのようなデータ一括転送が扱う多くのデータはストライドデータ転送が可能であり，ストライド転送をサポートすることは，通信オーバヘッドの削減に大きな効果がある．これは，データ転送のオーバヘッドが転送の回数に依存するために，一括転送やストライド転送によって通信の回数を削減することがプログラムの高速化のために重要だからである．

並列化コンパイラの生成するコードでは，プロセッサをいくつかのグループに分けて演算を実行し，各グループ内でバリア同期を取ったり，グローバル演算を実行したりする場合が多い．システムのすべてのノードが参加する場合だけでなく，特定のノードグループの場合でも高速にバリア同期やグローバル演算が実行できる必要がある．

1.2 アーキテクチャによるサポート

分散メモリ型並列計算機で並列化コンパイラの生成するコードを効率的に実行するためにはアーキテクチャによるサポートが重要である．その中で最も基本的なものは，ダイレクトリモートデータアクセス[20]である．

PUT/GET オペレーション

PUT/GETはアクティブメッセージの一例であり[21]，その機能はフラグの更新を除いてダイレクトリモートデータアクセスと同じである．PUT/GETはそれぞれ，以下の操作を行う．

[PUT] ローカルメモリ上のブロックをリモートメモリ上の送り手によって指定されたアドレスに書き込み，送受信完了時にそれを示すためのフラグを更新する．

[GET] リモートメモリ上のブロックを転送して，ローカルコピーを作り，送受信完了時にそれを示すためのフラグを更新する．

PUT/GETはデータをメモリからメモリへデータを直接書き込む方式で，送受信の際にメッセージのコピーを行わない．このため，メッセージ・パッシングのバッファからのコピーのオーバヘッドがなく，低オーバヘッド，高スループットのデータの一括転送を実現できる．PUT/GETはまた，通信と計算がオーバラップできるという性質を持つので，通信時間を隠蔽することができる．

PUT/GETをハードウェアでサポートする上で重要なのは，データの転送の自動化だけでなく，同期を取るためのフラグの更新もデータ転送と一体化して自動化することである．フラグはPUT/GETによる送信の完了を知るための同期手段であり，フラグの更新とデータの転送とは本質的に一体のものだからである．

1.3 高並列計算機 AP1000+

我々は並列化コンパイラが生成するコードから必要な通信機構を検討し，それらをアーキテクチャでサポートする分散メモリ型高並列計算機AP 1000+を開発した．AP 1000+はPUT/GETを通信の基本とし，低オーバヘッド高スループットの通信を実現すると共に，ストライド転送や特定のノードグループにも対応したバリア同期やグローバル演算を高速にサポートし，並列化コンパイラの必要とする通信機構を備えている．

2. 並列化コンパイラの必要とする通信機能

2.1 VPP Fortran と HPF

VPP Fortran[17]は並列プログラミング言語で，分散メモリ型の並列計算機上でハイパフォーマンスコンピューティングを実現するために開発された．

High Performance Fortran (HPF)[6]は分散メモリ型並列計算機の標準言語として提案されており，VPP Fortranと同じプログラミングモデルに基づいている．どちらの言語もグローバルメモリ空間やブロックまたはサイクリック分割のためのディレクティブを持ち，SPMDモデルで実行される．大きな違いはVPP Fortranにはプログラムをチューニングするための多くのディレクティブが用意されていることである．

VPP Fortranはグローバルメモリ空間に加えてロ

ーカルメモリ空間をサポートし，通信のオーバヘッドを最小限に抑えると共に，非同期の一括データ転送のディレクティブを備えている．一方 HPF ではコンパイラがデータの参照関係などを考慮して，一括転送のコードを生成する必要がある．

VPP Fortran では**図1**に示すような階層的メモリモデルを採用している．グローバル空間上のデータは各プロセッサからアクセス可能で，このグローバル空間を用いれば従来のシングルプロセッサ上と同じようにプログラミングが可能である．ローカルメモリ空間は各ローカルプロセッサから通信命令なしでアクセスでき，高速なアクセスが可能となる．このローカルメモリ空間を利用して，プログラムの最適化を図ることができる．

VPP Fortran はプログラムをチューニングするために一括データ転送を指示するディレクティブを用意している．一括転送のディレクティブには SPREAD MOVE と OVERLAP FIX の2つがある．

SPREAD MOVE は配列から配列へデータの塊を転送するディレクティブである．SPREAD MOVE ディレクティブは DO ループでの配列間の代入命令に対して挿入される．リスト1に SPREAD MOVE の例を示す．!XOCL で始まる行は VPP Fortran ディレクティブである．SPREAD MOVE はグローバル配列 B からローカル配列 A へデータを一括転送を行う．

List 1 Example of SPREAD MOVE.

```
1  !XOCL SPREAD MOVE
2              DO 200 J=1, M
3                 A(J)=B(J, K)
4    200       CONTINUE
5  !XOCL END SPREAD (X)
6  !XOCL MOVEWAIT (X)
```

オーバラップエリアは**図2**に示すように，隣合うプロセッサ間で境界のデータを共有する領域で，インデックスパーティションディレクティブによって配列がノード間に跨って分割された場合に利用される．オーバラップエリアを用いれば，隣合うノード間で境界のデータを参照する場合に，通信をすることなく必要なデータにアクセスが可能である．このオーバラップエリアのデータを一括して更新するディレクティブが OVERLAP FIX ディレクティブである．

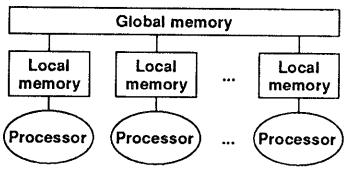


図1 VPP Fortran メモリモデル
Fig. 1 VPP Fortran memory model.

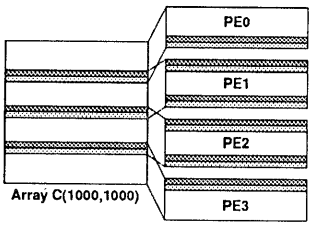


図2 オーバラップエリアの例
Fig. 2 Example of Overlap area.

2.2 PE 間通信

ダイレクトリモートデータアクセス

VPP Fortran のコンパイラは配列の参照関係を基にして，明示的なノンブロッキングダイレクトリモートデータアクセスのオペレーションをプログラムに埋め込み，ランタイムシステムはこれらを PE 間通信のインタフェースとして用いる．

ダイレクトリモートアクセスのインタフェースは以下のようなものである．

readRemote (node_id, raddr, laddr, size)
writeRemote (node_id, raddr, laddr, size)

ここで node_id は宛先のノード ID, raddr はリモートノード上のデータのアドレス，laddr ローカルノード上のデータのアドレス，そして size は転送されるデータの量を示している．

readRemote 関数は size バイトのデータを node_id ノードの raddr 番地からの読み，そのデータをローカルノードの laddr 番地に書き込む．

同様に writeRemote 関数はローカルノードの laddr 番地から size バイトのデータを node_id ノード上の raddr 番地へ書き込む．

ストライドデータ転送

ストライドデータ転送をサポートすることは SPREAD MOVE や OVERLAP FIX によるブロック一括転送を行うために重要である．

リスト1の3行目で，もしループインデックス J が B(K, J)のようにグローバル配列 B の2次元目のインデックスであれば，ストライドデータ転送が必要になる．これはローカル配列 A が連続なのに対し，グロ

ーバル配列Bがストライドだからである．

ストライドデータ転送は図2に示すようにオーバラップエリアが2次元目で行われている場合にも必要となる．

通信終了判定

ダイレクトリモートデータアクセスを用いたコード生成では，リモートノードへのアクセス要求はプロセッサの演算とは非同期に行われる．この場合，同期をとるために各ノードでは通信の完了を検知できなければならない．そうでなければ，データが読まれる前に新たに書き込まれたり，データが書かれる前に古いデータを読んでしまう可能性がある．通信の終了は各ノードでフラグを用い，これをデータの受信と共にフラグの値をインクリメントし，そのあとバリア同期を用いることで可能となる．

readRemoteの完了の判定はデータが帰ってくるので容易であるが，writeRemoteの場合は困難である．これに対する一つの解はアクノレッジを用いることである．この場合，受信ノードではアクノレッジパケットを返し，writeRemoteを行ったノードではこのアクノレッジによってアクノレッジのフラグを更新する．

writeRemoteを行ったノードではこのフラグの値をチェックし，writeRemoteが完了したかどうかを調べ，完了後にバリア同期に入ることで，全ノードでの通信の終了を判定することができる．このアクノレッジ&バリアタイプのオペレーションはデータパラレルプログラミングによく用いられるものである．

2.3 バリア同期とグローバル演算

バリア同期は通信の開始と終了を判定するために重要なオペレーションである．このためアーキテクチャではバリア同期をシステムの全ノード間だけでなく，あるグループ間のノードでも高速に行えるよう考慮する必要がある．

VPP Fortranでグループ間のバリア同期が用いられる例としては，インデックスパーティションディレクティブを用いて配列やDOループを分割する場合がある．この場合，分割された各グループのノードごとにバリア同期を行う必要がある．このインデックスパーティションディレクティブはHPFのALIGN, DISTRIBUTEディレクティブに対応するものである[6]．

バリア同期と同様にグローバル演算もインデックスパーティションディレクティブなどによって分割された場合には各グループごとに実行される．さらにグローバル演算の場合には，演算されるデータがスカラの場合とベクトルの場合があり，この両方を効率良くサポートすることを考慮する必要がある．

3. AP1000+アーキテクチャ

AP 1000+はAP 1000をエンハンスした分散メモリ型高並列計算機で，並列化コンパイラが必要とする機能を備えている．**図3**はAP 1000+のシステム構成，**図4**はセル構成を示す．

プロセッサには50 MHzのSuperSPARCを採用している．MSC+とMCがAP 1000+のために新たに開発したLSIである．メッセージコントローラ(MSC+)はセルとネットワークのインタフェースを行い，PUT/GETをサポートする．メモリコントローラ(MC)はSuperSPARCとメモリの間のV-Busを制御

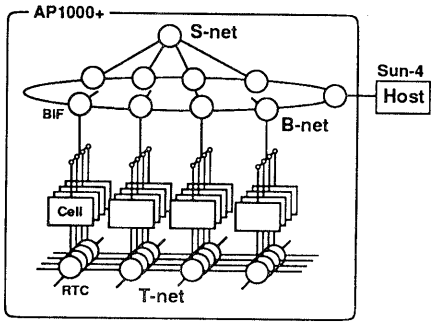


図3 システム構成
Fig. 3 System configuration.

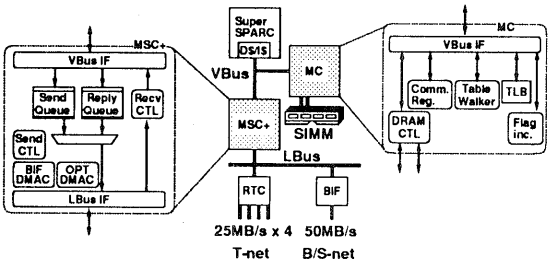


図4 セル構成とMSC+とMCのデータパス
Fig. 4 Cell configuration and data path of MSC+ and MC.

表1 AP 1000+仕様
Table 1 AP 1000+ specifications.

プロセッサ	SuperSPARC (50 MHz)
プロセッサ性能	50 MFLOPS
メモリ	16，64 MB
キャッシュ	36 KB ライトスルー
システム構成	2-1024 セル
システム性能	0.1-51.2 GFLOPS

する．これ以外のRTC, BIF, キャビネット，3つのネットワークはAP 1000と同じものを採用する．**表1**にAP 1000＋の仕様を示す．

3.1 PUT/GETの実装法

ユーザインタフェース

PUT/GETは必要なパラメータをMSC＋内の送信キューに書き込むことで起動される．PUT/GETを利用する場合にはプログラムはパラメータを一つずつ予め指定された特別のアドレスに書き込んでいく．MSC＋はこのアドレスからコマンドの種類と必要なパラメータの数を判定し，最後のパラメータが書き込まれた後自動的にセンドコントローラを起動する．このためプログラムは数個のストア命令を実行するだけで，PUT/GETコマンドを発行することができる．

ストライドデータ転送

AP 1000＋は1次元のストライドデータ転送をサポートしている．これは，高次元のストライドを実現するハードウェアコストと1次元ストライド転送のオーバヘッドとのトレードオフを考慮した結果である．高次元のストライドデータ転送は，1次元ストライド転送のオーバヘッドが小さければ，その繰り返しで実用的に実現できると考えられる．ストライドデータ転送も通常のPUT/GETと同様に実現される．すなわち，ストライドデータ転送のオーバヘッドは数回のストア命令のコストになり，十分小さい．

MMUとプロテクション

PUT/GETのアドレスとしてプログラムは論理アドレスを指定することができる．MSC＋はMC内のMMUを利用して，論理アドレスを物理アドレスに変換する．MCはダイレクトマップのTLB（Translation Lookaside Buffer）を持ち，高速にアドレス変換を行うことができる．

プログラムは不正なアドレスやセルIDをPUT/GETに対し指定する可能性があるので，プロテクションのメカニズムが必要になる．不正なアドレスが指定された場合には，ページフォルトが起こり，割り込みが発生する．またMSC＋はセルのサブグループを指定するレジスターを持っており，プログラムによって指定されたセルIDがそのサブセットの中に入っているかどうかをチェックする．サブセットの中に入っていない場合には，割り込みが発生し，OSが処理を行う．

送受信キュー

MSC＋の中には送信用3つと受信用2つの計5つのキューがある．3つの送信キューはそれぞれ，ユーザによって発行されたPUTとGET要求，システムによって発行されたPUTとGET要求，リモートアクセスである．PUTとGET要求がユーザとシステムに分かれているのは，システムがPUT/GETを利用する時にMSC＋がユーザ発行のキューのエントリーを退避せずに優先的に実行させるためである．リモートアクセスが別のキューを利用するのは，プロセッサがリモートロードをビジーウエイトするからである．

2つの受信キューはそれぞれ，GET返信とリモートロードの返信用であり，この場合もリモートロードの返信はGETの返信より優先されている．

それぞれのキューの最大サイズは64ワードなので，キューは溢れる可能性がある．このために，MSC＋は溢れたデータを自動的に直接DRAM上のバッファに書き込む機能を備えている．

データ転送と一体化したフラグの更新

フラグの更新はDMAによる送受信の完了と一体となって行われる．送信の場合，MSC＋は送信DMAの完了時にキューに書き込まれたフラグアドレスに対してその値をインクリメントするようにMCに要求する．MC内にはインクリメンタがあって，これによってfetch and incrementを行う．受信の場合も同様に，MSC＋はメッセージのヘッダ内に書かれたフラグアドレスに対し，その値をインクリメントするようにMCに要求する．どちらの場合もフラグのアドレスに0が指定された場合，MSC＋はフラグの更新を行わない．

センド・レシーブモデル

AP 1000＋ではPUT/GETモデルに加えてセンド・レシーブモデルをサポートしている．AP 1000＋は主記憶上にセンドされたメッセージをバッファリングするための，リングバッファと呼ばれる受信バッファを備えている．センドはPUTと同じMSC＋の機能を使って実現される．PUTと違うのは目的のアドレスが特定のアドレスではなく，リングバッファを設定することである．センドされたメッセージがリングバッファ上のどのアドレスに書き込まれるかは，MSC＋によって管理されている．受信関数はリングバッファをサーチし，目的のメッセージを発見したらそのメッセージをリングバッファからユーザのデータ領域にコピーする．

3.2 分散共有メモリ

AP 1000＋ではPUT/GET, センド・レシーブモデルの他に分散共有メモリもサポートしている．またAP 1000＋は通信専用のレジスタであるコミュニケーションレジスタを備え，このレジスタを分散共有メモリ空間にマップすることで，高速かつ柔軟なグローバ

ル演算やバリア同期を行うことができる．分散共有メモリについては本論文の範囲を越えるので，ここでは言及しない[12]．

4. シミュレーション

メッセージレベルシミュレータ（MLSim）はアーキテクチャの評価を目的としたシミュレーション・ツールであり，メッセージ・パッシングで記述されたプログラムをシミュレートする[9]．このメッセージレベルシミュレータをAP 1000で割り込みを利用したPUT/GETインタフェース[4]に対応するように改良し，AP 1000+の性能予測を行った．

シミュレータでは**表2**に示すような演算性能などのパラメータを変え，何度でも同じトレース情報を利用してシミュレーションすることが可能であり，短時間に様々なアーキテクチャの評価を行うことができる．この論文ではこれらのパラメータをAP 1000+に合うように設定して評価を行った．表2のパラメータのうち，computation_factorはSPARCに対する比で，その他の通信パラメータはマイクロ秒で与えられる．

表2 MLSimに与えるパラメータの一部
Table 2 Some MLSim parameters.

#		#	
# AP1000 model		# AP1000+ model	
#		#	
# computation SPARC		# computation SuperSPARC	
computation_factor	1.00	computation_factor	0.125
#		#	
# ---- PUT/GET ----		# ---- PUT/GET ----	
#		#	
put_prolog_time	5.0	put_prolog_time	1.0
put_epilog_time	5.0	put_epilog_time	0.0
put_msg_time	0.04	put_msg_time	0.04
put_dma_set_time	15.0	put_dma_set_time	0.4
put_msg_post_time	0.04	put_msg_post_time	0.0
#		#	
check_prolog_time	15.0	check_prolog_time	0.5
intr_rtc_time	20.0	intr_rtc_time	0.0
intr_rtc_recv_time	15.0	intr_rtc_recv_time	0.0
intr_rtc_send_time	15.0	intr_rtc_send_time	0.0
recv_msg_flush_time	0.04	recv_msg_flush_time	0.0
recv_dma_set_time	15.0	recv_dma_set_time	0.4

4.1 PUTの通信モデル

ここでは，MLSimのためのPUT通信モデルについて説明する．**図5**はAP 1000における割り込みを利用したPUT通信モデルを示す．

AP 1000におけるPUTのオーバヘッドには以下のものが含まれる．

- 送信時＝*put_prolog_time*＋*put_enqueue_time*
  ＋*put_msg_post_time*×msg_size
  ＋*put_dma_set_time**
- 受信時＝*intr_rtc_time*
  ＋*recv_msg_invalid_time*
  ×msg_size＋*recv_dma_set_time*

一方AP 1000+では，要求をMSC+のキューに入れるだけで後はハードウェアが自動的に行うため，DMAのセットなどはユーザプログラムの実行を妨げない．またライトスルー・キャッシュを用いているため，キャッシュをポストする必要がない．すなわち，送信時のオーバヘッドは要求をキューに入れる時間*put_enqueue_time*だけである．

受信では，ネットワークからのメッセージのヘッダをMSC+が解析し，自動的に目的のアドレスに書き込みを行う．またキャッシュのインバリデートはメモリへの書き込みと同時に行われるので，受信の前にインバリデートを実行する必要がない．このため受信時にはユーザプログラムの実行を妨げない．

これらAP 1000+のハードウェアの特徴を考慮してパラメータを設定し，受信完了とフラグチェックのための同期等の時間関係を保つようにシミュレーション

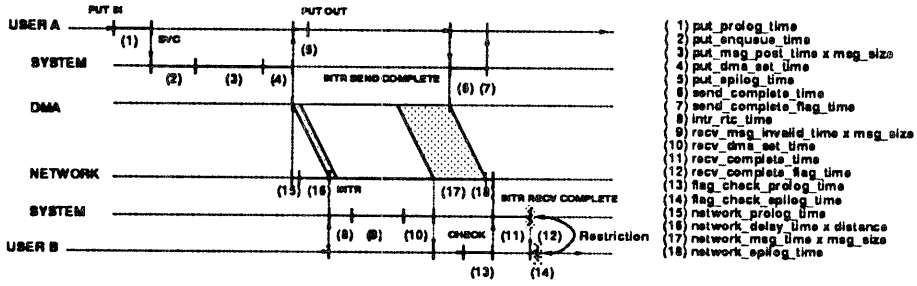


図5 AP 1000上のPUT通信モデル
Fig. 5 PUT communication model on AP1000.

* Postはキャッシュの内容をメモリに反映させる操作．

することで，AP 1000＋での実行時間を予測した．ただし，シミュレーションのモデルには，AP 1000＋のキュー溢れに対応する機構は含んでいない．シミュレータでは，キューは十分長いものとして扱っている．

4.2 アプリケーション

評価に利用したのは**表3**に示す7つのアプリケーションである．VPP Fortranでは，NAS Parallelベンチマーク[1]からEP, SP, CG, FTの4つとSPECベンチマーク[22]からTOMCATVの計5つについて評価を行った[10]．C言語ではPUT/GETをプログラムで直接用いた行列のかけ算とSCG（scaled conjugate gradient）の2つについて評価した．

4.3 シミュレーション結果

シミュレーションはプロセッサの性能をSuperSPARCがSPARCの8倍，その他の通信パラメータはハードウェアの仕様から適切に決定して行った．

表4に各アプリケーションの実行性能がAP 1000に対して何倍になるかを示す．対象としたモデルはAP 1000＋と，AP 1000でプロセッサの性能だけをSPARCからSuperSPARC相当に換えたものの2つである．また**表5**に各アプリケーションの統計情報を，**図6**に2つのモデルについての各アプリケーションにおける演算時間，通信オーバヘッドおよびアイドル時間の割合を示す．VPP Fortranのアプリケーションについては，VPP Fortran/APランタイムシステムの占める割合も示す．区分の意味はそれぞれ以下のとおりである．

[Execution time] 純粋にプログラムが計算を行う時間．

[Run time system] VPP Fortran/APランタイ

表3 アプリケーション
Table 3 Applications.

EP	2^{28}個の疑似乱数を発生し，通信は行わない．
SP	scalar pentadiagonal equationsの解を求める．64×64×64の配列に対し400回の反復を行う．メモリの制限から，最初の10回をトレースした．
CG	共役勾配法による連立方程式の解法．行列サイズは1400で，非零要素数は78184個．
FT	3次元FFT．配列のサイズは256×256×128．
TOMCATV	メッシュ生成プログラム．
MatMul	$A\times B=C$の計算を行う．行列は300×300の密行列．
SCG	2次元ポアソン方程式をSCG法で解く．行列は40000×40000の疎行列．

表4 実行性能予測（対AP 1000比）
Table 4 Performance simulation: compared to AP 1000.

	Application	AP 1000＋	AP 1000*
VPP Fortran	EP	8.00	8.00
	CG	5.62	4.29
	FT	7.01	4.24
	SP	7.62	6.05
	TOMCATV	7.87	6.82
C Language	MatMul	8.16	3.11
	SCG	8.03	5.19

AP 1000*：プロセッサをSuperSPARCにしたAP 1000モデル

表5 統計情報
Table 5 Application statistics.

Application	PE	Send per PE	Gop per PE	V Gop per PE	Sync per PE	PUT per PE	PUTS per PE	GET per PE	GetS per PE	Size of Msg.
EP	64	0	0	0	0	0	0	0	0	0.0
CG	16	366	810	390	3135	0	390	0	0	700.0
FT	128	0	48	0	93	0	17408	0	512	1638.4
SP	64	1	0	1	442	0	11520	0	0	1422.2
TOMCATV	64	0	200	0	600	0	394	0	0	2056.0
MatMul	64	0	0	0	0	0	0	64	0	9600.0
SCG	64	878	893	0	0	878	0	0	0	1600.0

- PE：評価に用いた台数
- Send：センドによる通信回数
- Gop：スカラーデータのグローバル演算回数
- V Gop：ベクトルデータのグローバル演算回数
- Sync：バリア同期の回数
- PUT：PUTによる通信回数
- PutS：ストライド付きPUTによる通信回数
- Get：GETによる通信回数
- GetS：ストライド付きGETによる通信回数
- Size of Msg.：PUT/GET通信の平均メッセージ長（バイト）

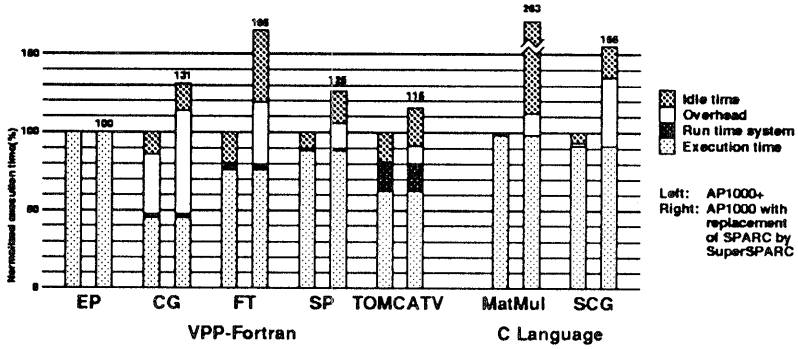


図6 PUT/GET ハードウェアサポートの効果
Fig. 6 Effect of PUT/GET hardware support.

ムシステムが通信を行うために送り先のアドレスの計算や，ストライドの検出およびストライドパラメータの計算等に要する時間で，通信は含まない．

［**Overhead**］ 通信ライブラリ内での処理時間からアイドル時間を除いたもので，この間プロセッサの演算が妨げられる．

［**Idle time**］ メッセージ待ちおよび同期成立待ち時間．

4.4 シミュレーション結果の分析

［**パフォーマンス**］ 表4に示したようにAP 1000＋の性能向上率はプロセッサの性能向上率に近いが，プロセッサの性能を変えただけのモデルはプロセッサの性能向上の約70%程度しか性能が向上していない．

EPは通信がないので，両方のモデルともプロセッサの性能向上と同じだけ性能が上がっている．CGは最も性能向上の小さい場合で，これはバリア同期の回数が非常に多いことと実行時間の大半が大きなベクトルのグローバル演算で占められているためと考えられる．CGでは11200バイト（1400×8）のベクトルの総和を390回計算している．グローバル演算では通信と演算をオーバラップさせることができないので，アイドル時間が大きくなっている．またバリア同期の回数はランタイムシステムの改善とプログラムの最適化を進めることで減らすことができると考えられる．FTとSPは表5に示すように多くの通信を利用しているが，AP 1000＋での通信のオーバヘッドはかなり小さい．TOMCATVはVPP Fortranのアプリケーションの中では最も通信回数が少ないが，多くのバリア同期を利用している．このため，2つのモデルの性能の差が一番小さくなっている．C言語で書かれた2つのアプリケーションはPUT/GETを直接利用し，通信と演算をオーバラップさせているので，プロセッサ性能向上率以上の性能を示している．

［**通信オーバヘッド**］ 図6からCGの場合を除いてAP 1000＋の通信オーバヘッドはAP 1000の約5%になっている．これはAP 1000＋はメッセージハンドリングのためのハードウェアを持っているのに対し，AP 1000では割り込みを利用してソフトウェアで実行しているからである．またユーザインタフェースの改善もオーバヘッド削減の理由の一つである．AP 1000＋ではユーザレベルでコマンドの発行ができるのに対し，AP 1000ではDMAを使うためにシステムコールを利用しなければならないからである．

［**コンパイラサポート**］ 表5はVPP Fortranによって生成されたコードが通信としてsend, put_stride(), get_stride()を利用していることが分かる．sendはベクトルデータのグローバル演算に利用されたものである．

1. ［ストライドデータ転送］ グローバル演算を除いてVPP Fortranの生成したコードが利用する通信はすべてストライドデータ転送である．ハードウェアがストライドデータ転送をサポートしていなければ，代わりにput()やget()が何倍も利用されることになり，性能低下の原因となる．ここでは実験結果を述べていないが，ストライドを使わずに各アイテムごとに別々に送信した場合，大幅に性能が劣化することが分かっている[5]．このことよりストライドデータ転送をサポートすることが，並列化コンパイラにとって必須であることが分かる．
2. ［バリア同期］ 表5に示すように，並列化コンパイラの生成したコードでは多くのバリア同期が利用されている．今回のシミュレーションに用いたVPP Fortranのアプリケーションは1次元分割を用いて並列化しているので，特定グループ内でのバリア同期は利用されていないが，2次元

以上の高次元の分割によってプログラムを並列化する場合には特定グループ内でのバリア同期が利用される．いずれの場合も，高速でかつ柔軟なバリア同期が重要である．

［ランタイムシステム］ ランタイムシステムの占める割合はCG, FT, SPの場合は約5%，TOMCATVの場合は約18%である．割合の違いの原因は通信回数の違いに依っている．TOMCATVでランタイムシステムの割合が大きいのは，通信に対して相対的に演算の割合が小さいためと考えられる．

5. 関連研究

リモートノードのメモリに直接アクセスする方法としては，ソフトウェアによるものとハードウェアによるものなど様々な方法が提案されており，それらと言語との関係について言及しているものもある[13),18),21)]．

ソフトウェアによってダイレクトリモートデータアクセスを実現する方法としては，von Eickenらによるactive message[21)]としてCM-5およびnCUBE/2上での実験結果が報告されている．active messageをベースにしたC言語の拡張であるSplit-CではPUT/GETオペレーションをユーザレベルで利用することで，実行性能が向上するとしている[2)]．

次にハードウェアによってダイレクトリモートデータアクセスを実現する方法としては，メッセージ・パッシングを基本とするものとして，CRAY T 3D, 富士通VPP 500, Meiko CS-2などまたキャッシュコヒーレントを基本とするものとして，Alewife, FLASHなどがある．

T 3D[18)]とVPP 500[17)]は分散メモリ間のデータ転送のための特別なハードウェア機構を備えている．T 3Dは各スイッチノードにメッセージ・パッシングの機構に加えいわゆるBlock-Transfer-Engineを備え，ローカルメモリとリモートメモリの間の大きなデータの非同期転送を可能にしている．そして，これらの機能を直接ソースレベルで利用した場合の性能評価が報告されている[18)]．

さらに最近では，分散共有メモリを効率的に実現するために，キャッシュのコヒーレントのための機構と高いスループットを要求する通信のためにメッセージ・パッシングを組み合わせたマシンが提案されている．

MITのAlewife[13),14)]はキャッシュコヒーレントのハードウェアとメッセージ・パッシングによるブロック転送を統合することを目指している．Alewifeではデータを送信する際にそのアドレスとサイズの組を書くことでストライドデータ転送が可能であるが，数値計算に用いられるような繰り返しの多い転送パターンを意識して作られていないので，そのような転送を行う場合にはオーバヘッドが大きくなる．

スタンフォード大学のFLASH[15)]もまた，キャッシュコヒーレントのためのハードウェアとメッセージ・パッシングを統合することを目指している．FLASHではプログラマブルなプロトコルプロセッサを用いることで，様々なキャッシュコヒーレントプロトコルやメッセージ・パッシングプロトコルを実現できる．しかし，FLASHでのメッセージ・パッシングは転送がキャッシュライン単位に独立のメッセージとして送信されるため，ストライドデータ転送をサポートするのが難しい．

CS-2[8)]は送信コマンドをユーザレベルで通信用コプロセッサ（ELAN）のキューに書き込むことで，低オーバヘッドの送信を実現することができる．ELANは仮想アドレスをサポートしており，アドレス変換もELAN内のTLBによって行われる．しかし，転送の単位は32バイトであり，大きなサイズのデータを転送する場合にはオーバヘッドが大きくなってしまう．

6. おわりに

並列化コンパイラに必要とされる通信機能について議論し，それらをアーキテクチャでサポートする重要性について述べた．またPUT/GETインタフェースをハードウェアで実現するために必要な機能について述べ，並列化コンパイラを効率的にサポートするアーキテクチャを提案した．AP 1000をエンハンスした分散メモリ型並列計算機AP 1000＋はこのアーキテクチャに従い，並列化コンパイラが必要とする機能をサポートしている．このAP 1000＋の性能をメッセージレベルシミュレータを用いて，NAS Parallelベンチマークなどの大規模科学計算のアプリケーションに対する性能を評価した．

PUT/GETはメッセージのバッファリングのオーバヘッドを削減し，また通信と演算をオーバラップさせることでメッセージパッシングに比べて実行性能が向上する．このようなPUT/GETの効果を引き出すためには，そのオーバヘッドを最小限に抑えることが重要であり，PUT/GETをハードウェアでサポートするAP 1000＋の選択は適切である．

今後はキュー溢れの実行性能への影響の研究をAP 1000＋の実機上で行っていく予定である．

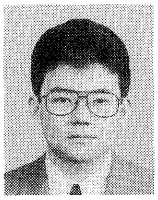
謝辞 日頃御指導，御助言いただく，並列処理研究

センター石井センター長，白石部長，池坂プロジェクト課長，ならびに研究室の同僚諸氏，特に NAS Parallel ベンチマークの VPP-Fortran による並列化を行った金城ショーン研究員に感謝いたします．

参 考 文 献

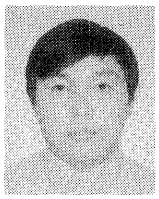
1) Bailey, D., Barton, J., Lasinski, T. and Simon, H.: The NAS Parallel Benchmark, Technical Report RNR-91-002 Revision 2, NASA Ames Research Center, Moffett Field, CA (Aug. 1991).
2) Culler, D. E., Dusseau, A., Goldstein, S., Krishnamurthy, A., Lumetta, S., Eicken, T. and Yelick, K.: Parallel Programming in Split-C, *Supercomputing '93*, pp. 262-273 (Nov. 1993).
3) 萩原純一，金城ショーン，土肥実久，岩下英俊，進藤達也：HPF コンパイラの実装と AP 1000 を用いた評価，SWoPP 琉球 '94 HPC 研究会，pp. 7-12 (July 1994).
4) 林 憲一，堀江健志：アクティブ・メッセージによる並列プログラム実行性能の改善，SWoPP 鞆の浦 '93 プログラミング研究会，Vol. 93-PRG-13-17, pp. 129-136 (1993).
5) 林 憲一，土肥実久，堀江健志，小柳洋一，白木長武，今村信貴，清水俊幸，石畑宏明，進藤達也：AP 1000+：並列化コンパイラをサポートするアーキテクチャ，SWoPP 琉球 '94 HPC 研究会，pp. 21-26 (1994).
6) High Performance Fortran Forum: High Performance Fortran Language Specification Version 1.0 (May 1993).
7) Hiranandani, S., Kennedy, K. and Tseng, C.: Compiler Optimizations for Fortran D on MIMD Distributed-Memory Machines, *Supercomputing '91*, pp. 86-100 (1991).
8) Homewood, M. and McLaren, M.: Meiko CS-2 Interconnect Elan-Elite Design, *Hot Interconnects '93*, pp. 2.1.1-4 (Aug. 1993).
9) Horie, T., Hayashi, K., Shimizu, T. and Ishihata, H.: Improving AP1000 Parallel Computer Performance with Message Communication, *The 20th Annual Int. Symp. on Computer Architecture*, pp. 314-325 (May 1993).
10) 金城ショーン，進藤達也：VPP Fortran を用いた NAS Parallel Benchmark の並列化と AP 1000 を用いた評価，SWoPP 琉球 '94 HPC 研究会，pp. 27-32 (1994).
11) Koelbel, C. and Mehrotra, P.: Compiling Global Name-Space Parallel Loops for Distributed Execution, *IEEE Trans. Parallel and Distributed Systems*, Vol. 2, No. 4, pp. 440-451 (1991).
12) 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：AP 1000+：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), pp. 161-168 (1994).
13) Kranz, D., Johnson, K., Agarwal, A., Kubiatowicz, J. and Lim, B.: Integrating Message-Passing and Shared-Memory: Early Experience, *Fourth ACM SIGPLAN Symp. Principles & Practice of Parallel Programming*, pp. 54-63, ACM (1993).
14) Kubiatowicz, J. and Agarwal, A.: Anatomy of a Message in the Alewife Multiprocessor, *Int. Conf. on Supercomputing*, pp. 195-206 (1993).
15) 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, *22nd Annual Int. Symp. on Computer Architecture*, pp. 302-313 (Apr. 1994).
16) Miura, K., Takamura, M., Sakamoto, Y. and Okada, S.: Overview of the Fujitsu VPP500 Supercomputer, *COMPCON 93*, pp. 128-130 (Feb. 1993).
17) Miura, K., Takamura, M., Sakamoto, Y. and Okada, S.: Overview of the Fujitsu VPP500 Supercomputer, *COMPCON 93*, pp. 128-130 (Feb. 1993).
18) Oed, W.: The Cray Research Massively Parallel Processor System CRAY T3D, Available through ftp from ftp.cray.com (Nov. 1993).
19) Rühl, R. and Annaratone, M.: Parallelization of FORTRAN Code on Distributed-Memory Parallel Processors, *Int. Conf. on Supercomputing*, pp. 342-353 (1990).
20) 進藤達也，岩下英俊，土肥実久，萩原純一：AP 1000 を対象とした VPP Fortran 処理系の実現と評価，SWoPP 鞆の浦 '93 HPC 研究会，Vol. 93-HPC-48-2, pp. 9-16 (1993).
21) von Eicken, T., Culler, D. E. et al.: Active Messages: A Mechanism for Integrated Communication and Computation, *19th Int. Symp. on Computer Architecture*, pp. 256-266 (1992).
22) Waterside Associates: The SPEC Benchmark Report, Fremont, CA (Jan. 1990).

(平成 6 年 9 月 13 日受付)
(平成 7 年 4 月 14 日採録)



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．並列計算機アーキテクチャの研究に従事．現在，富士通株式会社HPC本部に勤務．

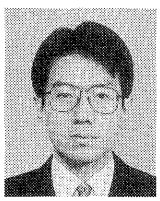


土肥　実久（正会員）

1988年大阪府立大学工学部電気工学科卒業．1990年同大学院工学研究科電気工学専攻博士前期課程修了．同年(株)富士通研究所入社．1994年より富士通(株)．機械翻訳，CADシステム，並列コンパイラの研究に従事

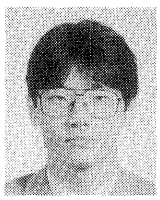
堀江　健志（正会員）

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．



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．現在，富士通株式会社HPC本部に勤務．電子情報通信学会会員．

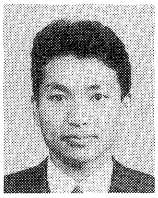


白木　長武（正会員）

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，富士通株式会社HPC本部に勤務．

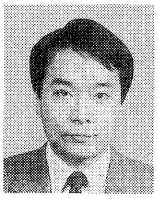
今村　信貴（正会員）

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，富士通株式会社HPC本部に勤務．



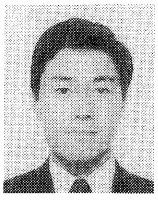
清水　俊幸（正会員）

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．



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，富士通株式会社HPC本部に勤務．元岡賞，電子情報通信学会論文賞受賞．



進藤　達也（正会員）

1983年早稲田大学理工学部電子通信学科卒業．1983年より(株)富士通研究所．1990-1992年スタンフォード大学客員研究員．1994年より富士通(株)．CAD専用マシン，並列処理の研究に従事．1986年篠原記念学術奨励賞．