Virtual Queue：超並列計算機向きメッセージ通信機構

五島 正裕[†,††] 森 眞一郎[†]
中島 浩[†] 富田 眞治[†]

Virtual Queue システムは，仮想化とストリームのキャッシングを特長とするメッセージ通信用ハードウェアである．仮想化によって，任意の数の仮想的通信機構を提供し，システム・コールなしで利用できるなど，メッセージ送受信時のソフトウェア・オーバーヘッドを大幅に削減するので，細粒度の通信に耐える．また，メッセージそれ自体ではなく，メッセージのストリームへの参照の局所性を利用し，よくアクセスされるストリームを物理的通信路にキャッシングすることによって高速化を図る．シミュレーションにより，ベクトル・データの転送にも十分なスループットを持つことが確認された．

Virtual Queue : A Message Communication Mechanism for Massively Parallel Computers

Masahiro Goshima,[†,††] Shin-ichiro Mori,[†] Hiroshi Nakashima[†]
and Shinji Tomita[†]

The **Virtual Queue** system is a message communication mechanism, marked by virtualization and caching of message streams. The virtualization provides arbitrary number of virtual communication hardware, and enables a user to use the hardware without system calls. The system withstands finer grained communications, because the virtualization drastically reduces the software overhead in handling messages. And the system makes use of locality of reference, not to messages themselves but to the stream of the messages, and caches frequently accessed message streams to the fast physical channels. The simulation result showes that the system provides enough throughput for transmission of vector data.

1. はじめに

ノード間の距離が大きくなる超並列規模の計算機では，メッセージ処理の高速化がますます重要になるが，従来のメッセージ通信の方式は ① メッセージ送受信のオーバーヘッドが大きい，② 通常のキャッシュはメッセージには適合しない などの問題点がある．前者は細粒度のメッセージを頻繁に交換する記号処理などの応用において，後者はベクトル処理のようにプロセッサの稼働率とメモリ・アクセスの頻度が高い場合に，性能に対する影響が大きい．本稿では，この2つの問題点を解決するメッセージ通信用ハードウェア，**Virtual Queue** システムを提案する．Virtual Queue は，以降 ***VQ*** と略す．

2. 従来のメッセージ通信の問題点

本章では，従来のメッセージ通信方式における問題点として，メッセージ送受信のオーバーヘッド とメッセージに対するキャッシュの不適合 について述べる．

2.1 メッセージ送受信のオーバーヘッド

並列計算機は，通信方式によって，共有メモリ方式とメッセージ方式に大別することができるが，どちらの方式も，メッセージ送受信の度に数百サイクルものオーバーヘッドを生じる．

共有メモリ方式のオーバーヘッド

共有メモリ・システムでは，メッセージのバッファに対する排他制御のために生じるオーバーヘッドを本質的に回避することができない．共有メモリ上でメッセージ通信を行う場合には一般に，遠隔メモリ上のメッセージ・バッファに対してロック操作を行う必要があり，ロック操作のレイテンシがメッセージ送受信の度にオーバーヘッドとして現れる．超並列規模の計算機では，遠隔ノードに対するアクセスのレイテンシは数

[†] 京都大学工学部
Faculty of Engineering, Kyoto University

[††] 日本学術振興会特別研究員
Research Fellow of the Japan Society for the Promotion of Science

百サイクルにものぼる．更新型のキャッシュ一貫性制御やプリフェッチなどのレイテンシ隠蔽手法はロック操作には応用できない．

メッセージ方式のオーバーヘッド

メッセージ方式の計算機では，メッセージ処理用のハードウェアがバッファを集中管理するため，共有メモリ方式のようなオーバーヘッドは生じない．受信ノード上のメッセージ処理用ハードウェアは，バッファを集中管理し，バッファに対する処理を不可分に実行する．したがって，メッセージを送受信するプロセッサ間で明示的なロック操作を行う必要はない．

しかし一方で，このメッセージ処理用ハードウェアに関して，ソフトウェア処理のオーバーヘッドが生じてしまう．一部の処理は，ソフトウェア/ハードウェアの工夫によって省略可能であるが，ハードウェアの獲得とアクセス保護のためのシステム・コールは省略することができない．また，受信時に必要なメッセージを検索したり，システム領域とユーザ領域間でメッセージをコピーする必要のあるシステムもある．これらのソフトウェア処理のオーバーヘッドは，数百から数千サイクルにものぼる1)．

2.2 メッセージに対するキャッシュの不適合

2つ目の問題点は，メッセージに対するキャッシュの不適合である．共有メモリ/メッセージの両方式においてキャッシュは，通常のデータに対するアクセスの高速化には大きな役割を果たすが，メッセージに対しては効果的でない．

参照の局所性とキャッシング

メッセージを構成する個々のデータに対する参照はたかだか1回程度であることが多く，時間的局所性の観点からは，メッセージをキャッシュする意味は薄い．それにもかかわらずメッセージをキャッシュすると，本来キャッシュしておくべき他のデータをキャッシュから追い出してしまい，かえって性能を悪化させかねない．

一方，1つのメッセージを構成する複数のデータに対する参照は時間的に集中して行われることが多く，バッファリングによるレイテンシの隠蔽手法が有効に働く可能性が高い．

キャッシュを利用したバッファリング

実際，キャッシュをバッファとして利用する方式がいくつか提案されおり，これらの方式では以下のようにしてメッセージ・アクセスのレイテンシを隠蔽する：送信側ではキャッシュ上で生成したメッセージをキャッシュから直接送信し1)〜3)，受信側ではプロセッサが参照する前にメッセージをキャッシュに移しておく2)〜4)．

しかしキャッシュをバッファとして利用する方式では，ベクトル処理のようにプロセッサの稼働率とメモリ・アクセスの頻度が高い場合，以下の問題が生じる：

(1) スループットの不足　メッセージがキャッシュ・メモリのポートを2度ずつ通過するため，スループットが不足する．

(2) バッファリングのオーバーヘッド　プリフェッチ/ポストストア処理の増加が性能に影響を及ぼす．

(3) 他のデータの追い出し　前述した他のデータの追い出しの問題は解決されない．

3. *VQ* システムの設計方針

VQ システムは，前章で述べた ① メッセージ送受信のオーバーヘッド ，② メッセージに対するキャッシュの不適合 の2つの問題点を以下の方針で解決する．

3.1 通信機構の仮想化

共有メモリ方式では，各プロセッサがメッセージのバッファを分散管理するために，バッファに対する排他制御のオーバーヘッドを避けることができない．メッセージ方式では，メッセージ処理用ハードウェアがバッファを集中管理するため，排他制御のオーバーヘッドは生じないが，一方で，このハードウェアを獲得するためにシステム・コールが必要となるなど，ソフトウェア処理のオーバーヘッドを生じてしまう．

VQ システムは，メッセージ方式と同様 メッセージ処理用のハードウェアを導入して排他制御のオーバーヘッドを避け，そのハードウェアを仮想化することでソフトウェア・オーバーヘッドを避けるアプローチを採る．

仮　想　化

一部のハードウェア資源は，その物理アドレスの，ユーザの仮想空間に対する写像を許可することによって，そのユーザに対して解放することができる．このように仮想記憶方式の記憶保護機能を利用してハードウェア資源をユーザに解放することは一般的に行われているが，これを指して 「そのハードウェアは仮想化されている」とは言わない．

本稿では 仮想化 を以下のように定義する :すなわち仮想化とは，そのコンテクストを適宜切り替えることによってハードウェア資源を時分割多重利用し，利用者に対してあたかもその資源の物理的制限が取り払われたかのように見せる手法である．コンテクストとは，物理的資源の処理中の状態を定義するのに十分な情報である．コンテクストの切り替えのコストは，切り替えの生じる間隔に対して十分に小さくなければならない．

仮想記憶方式の記憶保護を利用してユーザに解放す

るだけでは，所有者の交替時のシステム・コールを省略することはできない．したがって，メッセージ処理用のハードウェアのようにその所有者が頻繁に交替する資源に対しては記憶保護だけでは全く不十分である．

さて，通常のキャッシュは，システム・コールをまったく介することなく利用されている．それは仮想記憶方式によって記憶保護が実現されているだけでなく，その所有者の交替，すなわち，キャッシュ・ブロックのリプレースが，ハードウェアで安全に実現されているためである．

通常のキャッシュと同様，VQ システムは次のようにしてシステム・コールを介さないハードウェアの利用を可能にする．

VQ システムにおける仮想化

VQ システムは，ユーザに仮想的な通信機構を提供する．この仮想通信機構を ***VQ*** と呼ぶ．システムのコントローラは，個々の VQ のコンテクストを適宜切り替えることによって，物理的な通信機構を時分割多重利用する．物理的通信機構のコンテクストとは以下のようなものである；例えば，リング・バッファの制御ユニットは，バッファの書き込み/読み出しアドレスやエントリ数などの管理情報を保持するレジスタを持つが，それらのレジスタの内容はそのユニットにおけるコンテクストである．

VQ システムでは，コンテクスト・キャッシュと呼ぶ特殊なキャッシュを用いてコンテクストの切り替えを実現する．個々の VQ のコンテクストはユーザ・メモリ上の退避領域に置き，VQ システムのコントローラは，必要なコンテクストをコンテクスト・キャッシュにキャッシュして動作する．こうすることによって，通常のキャッシュと同様 仮想記憶によって記憶保護が実現できるうえに，所有者の交替をコンテクスト・キャッシュのミスとしてハードウェアで安全かつ高速に実現できるので，ハードウェアを獲得するために OS を呼び出す必要がなくなる．

仮想化には，その他に，以下の 2 つの効果がある：

仮想化の効果**(1)** ソフトウェア処理の省略

ハードウェア資源の物理的制限が緩和されるとことによって，従来のメッセージ方式で必要であったソフトウェア処理の多くを省略することができる．個々の VQ のコンテクストはメモリ上におかれるので，実際上 任意の数の VQ をユーザに提供できる．ユーザは，メッセージのストリームごとに VQ を割り付けることによって，メッセージの検索やコピーを避けることができる．

仮想化の効果**(2)** ストリームのキャッシング

VQ システムは，メッセージそれ自体ではなく，メッセージのストリームに対する参照の局所性を仮定し，これを利用する．一度アクセスされたメッセージは 2 度とアクセスされないことが多いと述べたが，一度アクセスされたメッセージのストリームは近い将来再びアクセスされる可能性が高いと考えられる．VQ は普通 メッセージのストリームごとに割り付けられるため，VQ のコンテクストのキャッシングは，よくアクセスされるストリームを高速な通信路にキャッシュすることと捉えることができる．

したがってコンテクスト・キャッシュは，プロセッサで走行中のスレッドが同時に使用するストリームの数程度のエントリを用意すればよい．具体的には，プロセッサごとに $4 \sim 16$ エントリ程度を想定する．

3.2　ストリームによる通信

高速なメモリにメッセージをバッファリングすることによってアクセスの高速化を図ることは重要であるが，通常のキャッシュは主に時間的局所性を利用するものであるためメッセージに対しては効率的に働かず，スループットの不足，バッファリングのオーバーヘッド，他のデータの追い出しのような問題を生じる．

VQ システムは，通常のキャッシュと同様に，バッファリング用高速メモリ（以下，単にバッファとする）を用いて高速化を図るのだが，以下のようにしてこれらの問題に対応する．

まず，VQ システムでは，通信をストリームによるものに制限する．すなわち，送信者が一旦 送信したブロック，受信者が一旦 参照を終えたブロックに対する再参照を禁止する．ストリームによる通信のみを用いる言語も多く，プロセッサ間チェイニングによるベクトル処理などにも対応可能であるなど，この制限は決して厳し過ぎるものではない．

ストリームによる通信を仮定することによって，以下の効果が得られる：

(1) バッファの再利用性の向上　古いメッセージに対するアクセスを禁止することによって，そのメッセージが占めていたバッファ上の領域を，同じストリームの新しいメッセージのために直ちに再利用することができる．

(2) バッファリングの自動化　プリフェッチ/ポストストアのタイミングを量ることが容易になり，ハードウェアによって自動的に行うことができる．

バッファの小量化による **2** ポート化/専用化

バッファの再利用性の向上とストリームのキャッシングによって，以下の効果が得られる：

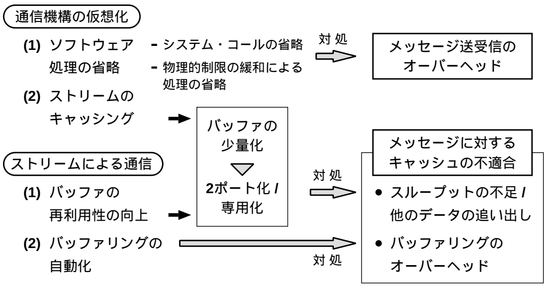


図 1 *VQ* システムの設計方針
Fig. 1 Design Policy of *VQ*System

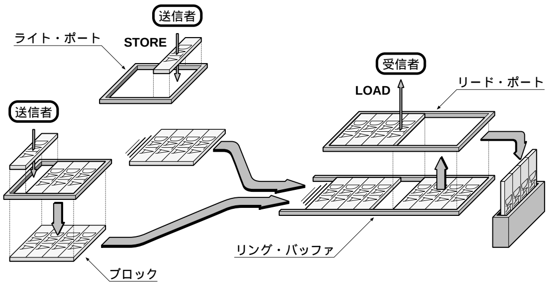


図 2 プログラミング ·モデル
Fig. 2 Programming Model

(1) バッファの再利用性の向上による効果　古いメッセージが占めていたバッファ上の領域をちに再利用するので，バッファの容量は，ストリームを流れるメッセージの長さとは無関係に，1 つのストリームに対して一定量の領域を用意すればよい．

(2) ストリームのキャッシングによる効果　バッファ上にバッファリングされているメッセージは *VQ* のコンテクストの一部と捉えることができるので，バッファはコンテクスト ·キャッシュの一部として実装できる．したがって，ストリームのキャッシングの効果により，バッファのエントリ数は，プロセッサで走行中のスレッドが同時に使用するストリームの数程度でよい．

以上の 2 点により，バッファの容量を大幅に削減することができる．具体的には，1 プロセッサあたり 256B ~ 1KB 程度を想定する．

この程度の容量であれば，2 ポート ·メモリを用いてこのバッファを構成し，メッセージのバッファリング専用に用いたとしても十分に実現可能である．2 ポート ·メモリを用いてバッファリングを行えば，バッファリングの実施によってシステムのスループットが低下することはなく，メッセージ専用に用いれば通常のキャッシュ上の他のデータを追い出すこともない．

本章の内容をまとめると，図 **1** のようになる．

4. *VQ* システムの構成

本章では，*VQ* システムについて詳細に説明する．まず 4.1 では *VQ* システムがプログラムからどのように見えるか示し，4.2 以降でその動作を実現するハードウェアについて述べる．

4.1 プログラムに対するインターフェイス

図 **2** に，*VQ* システムにおける通信の様子を示す．

VQ は，送信スレッドと受信スレッドの間の共有仮想空間上に設定された仮想的な通信機構である．ユーザは任意の数の *VQ* を設定できるので，通常 メッセージのストリームごとに *VQ* を設定する．

VQ は，3 種の仮想空間上の領域 —— ライト ·ポート，リング ·バッファ，および，リード ·ポート を構成要素とする．1 つの *VQ* は，1 つ または 複数のライト ·ポートと，1 対のリング ·バッファ/リード ·ポートからなる．

VQ は，数十バイトの ブロック を通信の単位とする．可変長のメッセージは扱わない．ブロックは不可分に扱われるので，転送経路各部の実効バンド幅を向上させ，管理情報を小量化するほか，遠隔手続き呼び出しなどの応用において有用である．

ブロックの送信はライト ·ポートへのストアによって，ブロックの読み出しはリード ·ポートへのロードによって行う．1 つのライト ·ポートから送信されたブロックは，送信された順序でリード ·ポートから読み出されるが，複数のライト ·ポートから送信されたブロック間の順序は非決定的となる．

4.1.1 *VQ* のデータ構造

では，*VQ* がプログラムからどう見えるか，より詳細に説明する．ユーザは，*VQ* の使用に先だって *VQ* のデータ構造を設定する必要がある．このデータ構造は，*VQ* システムが *VQ* のコンテクストを特定するため，アクセスされたアドレスとコンテクストの退避領域のアドレスの間の簡単な写像を定義する．

システムのアドレス空間の構成

1 つのプロセス内のスレッドは単一の仮想アドレス空間を共有するものとする．ただし，遠隔ノードのメモリに対して直接ロード/ストアなどのアクセスが行える必要はない．ノード間にまたがる仮想アドレス空間にはいくつかの構成方法が考えられるが，本稿の範囲を越えるので，ここでは触れない．

VQ システムの実アドレス空間中には，主記憶の物理アドレス空間とは別に，特殊なアクセスのためのアドレス空間 —— コマンド ·アドレス空間 をいくつか設ける☆．コマンド ·アドレス空間は，主記憶の物理

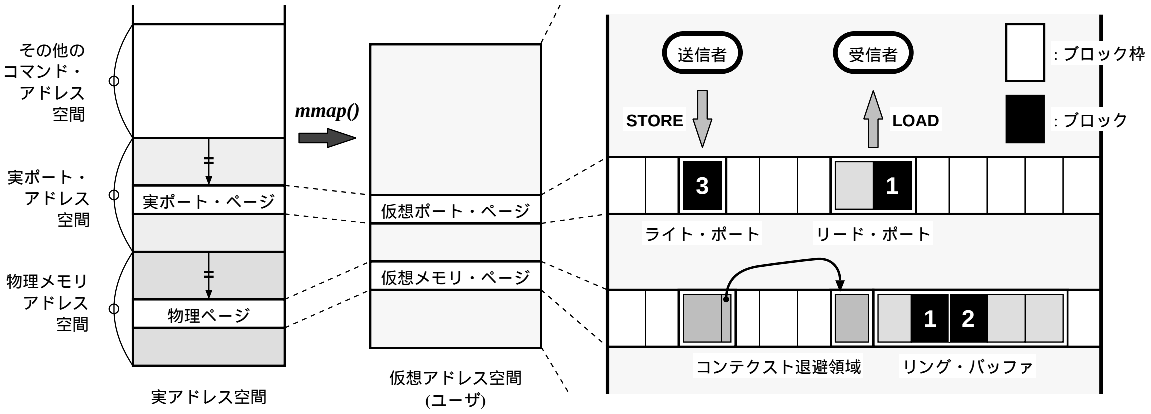


図 3　VQ のデータ構造
Fig. 3　Data Structure of VQ

アドレス空間に対する別名であり，各空間の大きさは互いに等しい．コマンド・アドレスへのアクセスは，空間内での変位が等しい主記憶に対するアクセスを修飾する．すなわち，各空間内の変位を表す実アドレスの下位部分は主記憶の物理アドレスに対応し，残りの上位部分はコマンドの識別子となる．

コマンド・アドレス空間のうちの1つを，VQ アクセスのためのアドレス空間 ——— 実ポート・アドレス空間 とする（図 3 左）．VQ を使用するユーザ・プロセスは，UNIX では $mmap()$ に相当するシステム・コールにより，物理ページと対応する実ポート・アドレス空間上のページ ——— 実ポート・ページ を獲得し，仮想ページに写像する（図3 左から中）．物理アドレス空間に写像された仮想ページを 仮想メモリ・ページ，ポート・アドレス空間に写像された仮想ページを 仮想ポート・ページ と呼んで区別する．理由は後述するが，OS は，ユーザが対応する物理ページを既に獲得している場合に限り，実ポート・ページへの写像を許可する．

ユーザは，これらの仮想ページの上に VQ のデータ構造を設定するのだが，仮想記憶によってアクセス保護が実現されるので，各ページを獲得した後は，VQ に関するすべてのアクセスはユーザ権限で行うことができる．

VQ のデータ構造

プログラムは，獲得した仮想ページ上に図3 右に示すように，VQ のデータ構造を設定する．

VQ を構成する各要素は，アラインされた1ブロック・サイズの領域 ——— ブロック枠 を単位とする．

プログラムはまず，仮想メモリ・ページ上に各要素のコンテクスト退避領域とリング・バッファを設定する．ただし，リード・ポート・ユニットとリング・バッファ・ユニットのコンテクストは1つのブロック枠に詰め，リング・バッファはその次のブロック枠からはじまる1ページ内の連続領域に割り付ける（図3 右下）．

ライト・ポートとリング・バッファ/リード・ポートは，送信先のリング・バッファ/リード・ポートの仮想アドレスをライト・ポートのコンテクストとして指定することで結び付けられる（図3 右下の矢印）[☆]．

仮想ポート・ページ上の，退避領域とページ内変位が互いに等しいブロック枠がそれぞれのポートとなる．ただし，リード・ポートは2ブロック枠を使用する（図3 右上）．

実ポート・ページと物理ページは，各アドレス空間内での変位が互いに等しい位置にとった（図3 左）ので，ポートと退避領域の実アドレスは各アドレス空間内での変位が互いに等しい．したがって，VQ システムのハードウェアは，ポートへのアクセスがミスを起こした時，ポートの実アドレスの空間内変位から対応する退避領域の物理アドレスを知ることができる．

4.1.2　通信手順

上述のデータ構造の設定の完了を確認するため，送受信者間で同期を取った後に VQ による通信が可能となる．通信は以下のように行う：

送信　送信者は，ライト・ポートに対するストアにより，ライト・ポート上にブロックを形成する．ライト・ポートの最後のバイトが変更を受けた時，その時点でのライト・ポート上のブロックが転送され，リング・バッファの末尾に追加される．転送すべきデー

[☆] メモリ・マップI/O のアドレス空間はさらに別に設ける．

[☆] 図3 では，説明の都合上ライト・ポートとリング・バッファ/リード・ポートを同一ページに設定していているが，実際には，相互の位置関係に制限はない．

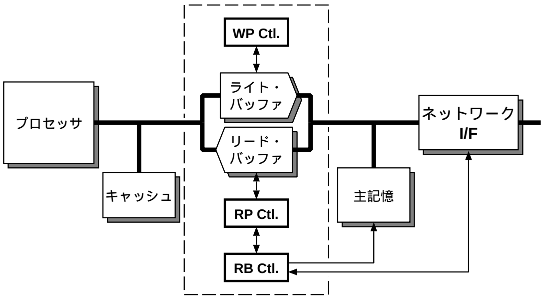


図4　処理ノードの構成
Fig. 4　Processing Node

タが1ブロックに満たない場合には，最後のバイトへのダミー・ライトを行う．

最後のバイト以外の部分に対する書き込みには，順序，回数などの制限はなく，また送信処理の完了を確認する必要はない．

受信　リード・ポートの2つのブロック枠を用いて2面バッファリングを行う；2つのブロック枠のどちらか一方にはリング・バッファの先頭のブロックが，他方には2番目のブロックが写像されている．先頭のブロックへの参照が終り，2番目のブロックへの参照を開始すると，参照の終ったブロックは自動的にリング・バッファからデキューされ，空いたブロック枠には新たに2番目となったブロックが写像される．

1つのブロック枠内のデータに対する参照には，順序，回数などの制限はない．

以上のように，通信の実行に際しては，通信のための余分な処理はほとんど必要ない．

*VQ*のアンダー/オーバーフローについては，4.4でまとめる．

4.2　ハードウェアの構成

*VQ*システムは，ライト・ポート，リング・バッファ，およびリード・ポートそれぞれに対応する3種のユニットからなり，これらのユニットの協調動作によって上述の通信処理を実現する．

図4に，対象計算機の処理ノードの構成を示すシステムは2ポート・メモリで構成されたライト・バッファ/リード・バッファを備え，バッファリングにより*VQ*に対するアクセス・レイテンシを隠蔽する[☆]．プロセッサが1次キャッシュを内蔵する場合にそれをライト/リード・バッファとして利用するなど，いくつかの実装が可能であるが，以降では簡単のためプロセッサが1次キャッシュを内蔵していない場合について説明する5)．

ポートへのアクセスに対して各ユニットは，それぞれのコンテクストを主記憶上の退避領域からそれぞれのコンテクスト・キャッシュにキャッシュして動作する．OSはユーザが対応する物理ページを既に獲得している場合に限り，ポート・ページへの写像を許可すると述べた．したがって，ポートへのアクセスが許可されていれば，退避領域へのアクセスの正当性を改めて検査する必要はない．

以下，3種のユニットがどのように動作して前述の通信処理を実現するか説明する．同時に，記憶保護の実現について述べる．コンテクスト・キャッシュについては，次節で改めて詳述する．

(1)　ライト・ポート・ユニット

ライト・ポートに対してストアされるデータは，実際にはライト・バッファに書き込まれ，ライト・バッファ上でブロックにまとめられる．

ライト・バッファの最後のバイトへのストアをトリガとして，保持するブロックを送出する．ブロックは，主記憶をバイパスして，ライト・バッファから直接ネットワークI/F部に送られる（図4を参照）．

ネットワークI/F部では，リング・バッファの仮想アドレスから宛先ノードの物理アドレスへの変換が行われ，転送されるブロックにはリング・バッファの仮想アドレスが付加される．

(2)　リング・バッファ・ユニット

リング・バッファ・ユニットは，ネットワークからブロックが届いた時，ブロックに付加されたリング・バッファの仮想アドレスでリング・バッファ・ユニットのコンテクスト・キャッシュにアクセスし，主記憶上のリング・バッファの書き込み位置の物理アドレスを求める．したがって，OSが仮想アドレスと物理アドレスの写像を制御することによって，記憶保護が実現される．

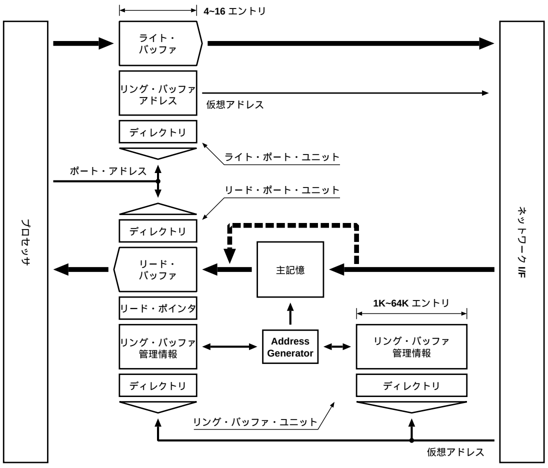
書き込みに関するリング・バッファの管理情報の更新はリング・バッファ・ユニットで行われる．

(3)　リード・ポート・ユニット

リード・ポート・ユニットのコントローラは，先頭と2番目のブロックがリード・バッファ上に存在するように制御する．リード・ポートに対するロードでは，実際にはリード・バッファから読み出しが行われる．

リード・バッファ上の2ブロック枠を用いて2面バッファリングを行い，主記憶上のリング・バッファに対するアクセス・レイテンシを隠蔽する．アクセスが先頭のブロックから次のブロックに移った時に，リード・ポート・ユニットのコントローラはリード・ポートの実アドレスでコンテクスト・キャッシュにアクセスし，主記憶上のリング・バッファの読み出し位置を求める．リ

[☆] 図4中には示していないが，通常のメモリ・アクセスのためのバイパスが，ライト/リード・バッファに並置される．



ユニット	エントリ数	連想度	アドレス
(1) ライト・ポート	4 ~ 16	完全	実
(2) リング・バッファ	1K ~ 64K	1 ~ 4	仮想
(3) リード・ポート	4 ~ 16	完全	実/仮想

図 5　コンテクスト・キャッシュ構成
Fig. 5　Context Caches

ング・バッファはリード・ポートのコンテクスト退避領域と同一ページ内に割り付けられるので，リード・ポートへのアクセスが許可されていれば，リング・バッファへのアクセスの正当性を改めて検査する必要はない．

読み出しに関するリング・バッファの管理情報の更新は，リード・ポート・ユニットで行われる．

4.3　コンテクスト・キャッシュ

VQ システムの各ユニットは，それぞれのコンテクスト・キャッシュを持ち，コンテクストはそれぞれ独立にキャッシングされる．

前述のライト/リード・バッファは，実際には，コンテクスト・キャッシュの一部とみなすことができる．

図 5 にコンテクスト・キャッシュの構成と論理的なデータのながれを示す．また，各コンテクスト・キャッシュのエントリ数，連想度，アクセスするアドレスを，同図中の表に示す．

以下，各ユニットのコンテクストと，コンテクスト・キャッシュについて詳細に説明する．

(1)　ライト・ポート・ユニット

コンテクスト　送信先のリング・バッファの仮想アドレスの他，ライト・バッファ上の生成途中のブロックは，ライト・ポートのコンテクストとなる．

コンテクスト・キャッシュ　ライト・ポート・ユニットのコンテクスト・キャッシュは，通常のプロセッサが持つストア・バッファなどとほぼ同じものと考えてよい．ライト・ポートの実アドレスがディレクトリの保持するアドレスと比較され，ヒット/ミスが判断される．

(2)　リング・バッファ・ユニット

コンテクスト　以下のリング・バッファの管理情報がコンテクストとなる：

- サイズ
- 読み出し/書き込み位置を示すポインタ
- 現在のブロック数

コンテクスト・キャッシュ　ライト・ポート，リード・ポートへはプロセッサ上で走行中のスレッドのみがアクセスするのに対して，リング・バッファ・ユニットへは遠隔の多くのノードから非同期にブロックが届くので，1 つのリング・バッファ・ユニットは性能上同時に多数のコンテクストをキャッシングする必要がある．したがって，リング・バッファ・ユニットのコンテクスト・キャッシュは，RAM チップを用いて構成する．理由は後述するが，リング・バッファに対する書き込みはクリティカルではない．通常のメモリ・アクセスが *VQ* アクセスをバイパスできるようにすれば，このキャッシュに対するアクセスのレイテンシを短縮する必要はない．スループットに関して，(このキャッシュのアクセス・サイクル数) = (主記憶のブロック・アクセス・サイクル数) ÷ (主記憶アクセスに占める *VQ* アクセスの割合) 程度であればよい．

複数サイクルをかけてアクセスすることで線幅を縮小できるので，このキャッシュは主記憶と同等のサイクル・タイムを持つ SRAM チップ 1 個程度を用いて構成できる．これは，ゲート量としては外部 2 次キャッシュのディレクトリと同程度である．

(3)　リード・ポート・ユニット

コンテクスト　ライト・バッファとは異なり，リード・バッファ上のブロックはリード・ポートのコンテクストとする必要はない．これらのブロックはリング・バッファのコピーであり，リング・バッファのコンテクストがリード・ポートの状態を完全に指定する．

一方 リード・ポートの 2 ブロック枠のうちどちらに先頭のブロックが写像されているかを示すビットが必要であるが，このビットはリード・ポートのコンテクストとなる．

コンテクスト・キャッシュ　図 5 の，リード・ポート・ユニットのコンテクスト・キャッシュのリードバッファからプロセッサ側は，通常のキャッシュと同じものと考えてよい．

リード・ポート・ユニットのコンテクスト・キャッシュはリング・バッファ・ユニットとは独立にリング・バッファの管理情報をキャッシュし，リード・バッファへのプリフェッチ時にはここの情報を用いて高速に主記憶

上のブロックにアクセスする．2面バッファリングを行っているとはいえ，プリフェッチは比較的高速に行う必要があるので，大容量で低速なリング・バッファ・ユニットのコンテクスト・キャッシュとの共用はしない．リング・バッファの管理情報を2箇所にキャッシュすることには，プロセッサからのアクセスとネットワークからのアクセスを並列に処理できるというメリットもある．

このように，リング・バッファの管理情報はリード・ポート，リング・バッファ各・ユニットの2箇所にキャッシュされるため，それらの一貫性を保つ必要がある．リード・ポート・ユニットのコンテクスト・キャッシュの図中での下の部分は仮想アドレスでアクセスできるようになっており，外部からブロックが到着した時には，同時に2つのコンテクスト・キャッシュにアクセスする．双方に同一のエントリが存在した時には，リード・ポート側を優先して用い，リング・バッファ側のエントリは無効化する．

また，あるVQに対してプロセッサがビジー・ウェイトしている最中にブロックが到着した場合には，リード・ポート側のコンテクスト・キャッシュがヒットし，主記憶をバイパスしてブロックをリード・バッファに送ることができる（図4参照）．前項でリング・バッファのコンテクスト・キャッシュに対するアクセスはクリティカルでないと述べたが，それはこの理由による．

4.4 アンダー/オーバーフロー

プログラムがアンダー/オーバーフローの発生を知る方法としては，コマンドによる検査 と 例外/割り込みによる検出 の2種がある．どちらの方法を選択するかは，応用プログラムの戦略による．個々のVQのアンダー/オーバーフローの生起確率が目安となる．

コマンドによる検査

4.1.1において，実アドレス空間中にはコマンド・アドレス空間を設けると述べたが，このこの空間のうちのいくつかをアンダー/オーバーフローの検査のコマンドのために用いる．したがってコマンドは，今まで述べてきたVQへのアクセスと同様，OSを介さずに通常のロード程度のコストで実現できる．

もっとも基本的なコマンドは，リング・バッファ中のブロック数を返す．

また，受信側では以下のコマンドが使用できる：VQが空であるとき，ヌル・ポインタ値0など，応用プログラムが使用しない値を応答して空であることを伝える．空でない場合には正しい値が応答されるので，検査のためのロードを省略することができる．

例外/割り込みよる検出

コマンドによってアンダー/オーバーフローが発生しないように検査する他に，例外/割り込みによってアンダー/オーバーフローの発生を知ることができる．

例外/割り込みを発生する前には，状態が自然に回復することを期待して，しばらくの間 処理を中断する．中断の期間は，例外/割り込みに対する処理のコストの数分の1程度を想定する．

オーバーフローに対する対処

アンダーフローとは異なり，オーバーフローは，原因となるプロセッサとは異なるノードにおいて，原因となるアクセスとは非同期に発生するため，その対処は一般に難しい．

VQシステムでは，読み出しポインタがある値になったときに割り込みを発生する機能を用意する．このポインタの値はリード・ポート/リング・バッファのコンテクストとして指定する．オーバーフローの発生時には，応用プログラムは，この機能を用いて以下のように対処するとよい：

(1) リング・バッファ中の後半のブロックを別の領域に待避し，空きを作る．
(2) 待避したブロックのあった位置に読み出しポインタが達したときに割り込みが発生するように設定する．
(3) 割り込みの発生によって，読み出しポインタが設定した値になったことを知る．
(4) 待避しておいたブロックを戻し，ポインタを適当に設定し直す．

読み出しポインタが設定した値になるまでの間は，リード・ポートとライト・ポートのインターフェイスは変わらないので，送受信者ともに，通常どおりに処理を続行できる．

5. 性能評価

256×256行列（倍精度）の行列積$A \cdot B \to C$の実行をシミュレートしてFLOPC (Floating Operation Per Cycle) 値を計測し，他の2種の方式との比較を行った．以下，シミュレーション・モデル，および，対象プログラムについて説明した後，結果を述べる．

5.1 評価モデル

評価モデルは，基本となるモデルに対して，VQと他の2種の通信方式を実装する．

基本モデル

評価モデルの処理ノードは，基本的には，図4に示した構成とする．モデル化を行うにあたっては，通信機構以外の部分 —— 特に，ソフトウェアの最適化

表**1**　シミュレーション・パラメータ
Table 1　Simulation Prameters

パラメータ		値
キャッシュ	ブロック・サイズ	4DW (32B)
	容量	64KB
	バンド幅	1DW/cycle
	レイテンシ	0cycles
メモリ	バンド幅	1DW/cycle
	レイテンシ	5cycles
ネットワーク	バンド幅	1DW/cycle
	レイテンシ	10cycles

の程度の，性能への影響を取り除くことを考慮した：

プロセッサ　プロセッサ・アーキテクチャの影響を排除するため，ロード/ストア・ユニットが1である以外，その他のユニットの数量の制限は設けないようなsuperscalarとする．内積型の行列積では1積和演算(2FLOP)ごとにA，Bから計2要素をロードするので，理論最大性能は1FLOPCとなる．

キャッシュ　競合性のミスの影響を排除するため，完全連想/LRUとする．

ネットワーク　トポロジーの影響を排除するため，クロスバーとする．

以上の構成は確かに非現実的ではあるが，最近のソフトウェアの最適化技術を駆使した場合には，現実的な構成でも同等の性能を引き出せるものと考える．

その他のパラメータを，表**1**にまとめる．

通信方式

上述の基本モデルに対して，*VQ*と，以下の2種の通信方式を実装し，比較する：

CC　アップデート・プロトコルによってキャッシュのコヒーレンスを保つ分散共有メモリ．

VR　図4におけるリード/ライト・バッファの部分に，比較的大容量の通信用のベクトル・レジスタを備える．*VQ*と異なり主記憶に対するバッファリングを行わないため，理想的な通信機構として評価の基準となる．

5.2　評価プログラム

今回用いた行列積プログラムでは，ロードのレイテンシを軽減するため，データを生産するプロセッサが次にそのデータを消費するプロセッサに対して，生産の直後に明示的に送信を行うようにした．CCでは，消費者のノードのメモリ上にダブル・バッファをとり，store barrierと1-writer lockによって同期をとる．*VQ*/VRでは，それぞれ，*VQ*/ベクトル・レジスタを用いる．各方式において，バッファ・サイズは行列の1行分とした．

なお，このプログラムでは，1本のストリームしか使用しないため，*VQ*におけるコンテクスト・キャッシュのヒット率はほぼ100%である．

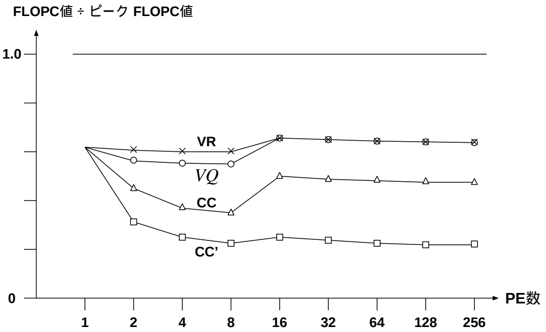


図**6**　シミュレーション結果
Fig. 6　Simulation Result

5.3　実行結果

結果を図**6**に示す．横軸にはプロセッサ数を，縦軸には（FLOPC値）÷（ピークFLOPC値）を表す．

参考のため，CCのハードウェア上で，明示的に通信を行わない単純なプログラムを実行した結果をCC'として示した．この場合には同期が全く不要であるため，性能の悪化は主にリモート・メモリへのロードのレイテンシによる．ただし，ネットワークがクロスバーであり，プロセッサ間の距離がプロセッサ数によらず一定であるために，プロセッサ数の増加による性能の悪化は小さい．

それに対しCCでは，アップデートが成功すればキャッシュから，失敗してもローカル・メモリからデータをロードすることができるので，全域で性能が向上している．基本的に1対1通信であるため，同期処理のオーバーヘッドはほとんど現れていない．

しかし，プロセッサ数が少なく1プロセッサの担当するデータ量が多い場合には，ソフトウェアのバッファがキャッシュから溢れ，消費者のキャッシュに対してアップデートが行われないため，性能が大きく低下している．また，プロセッサ数が多い場合には，アップデート自体がプロセッサをディスターブするために，*VQ*，VR程の性能は出ていない．

一方，*VQ*とVRは，プロセッサ数が少ない場合で1プロセッサの場合と同等，キャッシュが溢れない領域ではそれを超える効率を示している．

*VQ*では，送信されたデータは主記憶上にとられたリング・バッファに一旦書き込まれるために，理論的にVRより良い性能を示すことはない．しかし，結果に現れている性能差は主記憶のバンド幅の問題ではなく，主記憶へのアクセス競合のためにキャッシュ・ミス時のレイテンシが増加しているためである．したがって，キャッシュ・ヒット率が高い領域では有意な性能差

は現れていない．

6. おわりに

VQ システムは，低オーバーヘッドと高バンド幅を両立するメッセージ通信機構である．シミュレーションにより，理想的な通信機構である VR 方式に比肩する高いバンド幅を持つことが示された．

ところで VR 方式では，大容量のベクトル・レジスタの内容がプロセッサ・コンテクストとなるため，マルチ・ユーザ環境で用いることが難しい．したがってその可用性を考慮すれば，VQ は，バンド幅に関して，ほぼ理想適なメッセージ通信機構であるといえる．

一方 今回の性能評価では，細粒度通信における低オーバーヘッド性を示すことはできなかった．また，VQ システムの高い性能はソフトウェアに対して，ストリームによる通信モデル，固定長メッセージなどの制限を与えた結果として得られるものであり，これらの制限の妥当性を検証する必要がある．したがって今後は，細粒度の通信を伴う非定型的なプログラムに対して性能評価を行っていく予定である．

謝辞　本研究の一部は，文部省科学研究費補助金（重点領域研究(1)「超並列ハードウェア・アーキテクチャの研究」，および，奨励研究（特別研究員）「高速メッセージ通信機構に基づく並列計算機システム」）による．

参考文献

1) 清水俊幸, 堀江健志, 石畑宏明：高速メッセージハンドリング機構—AP1000 における実現—, in *JSPP '92*, pp. 195–201 (1992).
2) Byrd, G. T. and Delagi, B. A.: StreamLine: Cache-Based Message Passing in Scalable Multiprocessor, in *ICPP '91*, pp. I–251–I–254 (1991).
3) 五島正裕, 森眞一郎, 富田眞治：プロセッサ間通信をサポートする On-Memory FIFO 機構, 情処研報 92–OS–56, pp. 111–118 (1992).
4) 平木敬他：超並列計算機プロトタイプ JUMP-1 の構想, 情処研報 93–ARC–102 (1993).
5) 五島正裕, 森眞一郎, 富田眞治：Virtual Queue：超並列計算機向きメッセージ通信機構, 情処研報 94–ARC–107, pp. 145–152 (1994).
6) 五島正裕, 松本重光, 森眞一郎, 中島浩, 富田眞治：Virtual Queue：超並列計算機向きメッセージ通信機構, in *JSPP '95*, pp. 225–232 (1995).

(平成？年？月？日受付)
(平成？年？月？日採録)

五島　正裕（学生会員）

1968 年生．1994 年 京都大学大学院工学研究科情報工学専攻修士課程修了．同年より日本学術振興会特別研究員．並列計算機アーキテクチャの研究に従事．1996 年 同大学大学院工学研究科情報工学専攻博士後期課程退学．同年より同大学工学部助手．

森　眞一郎（正会員）

1963 年生．1987 年 熊本大学工学部電子工学科卒業．1989 年 九州大学大学院総合理工学研究科情報システム学専攻修士課程修了．1992 年 九州大学大学院総合理工学研究科情報システム学専攻博士課程単位取得退学．同年京都大学工学部助手．1995 年 同助教授．工学博士．並列/分散処理，計算機アーキテクチャの研究に従事．IEEE-CE, ACM 各会員．

中島　浩（正会員）

1956 年生．1971 年 京都大学大学院工学研究科情報工学専攻修士課程修了．同年 三菱電機（株）入社．推論マシンの研究開発に従事．1992 年より京都大学工学部助教授．並列計算機のアーキテクチャ，プログラミング言語の実装方式に関する研究に従事．工学博士．1988 年 元岡賞，1993 年 坂井記念特別賞受賞．

富田　眞治（正会員）

1945 年生．1973 年 京都大学大学院博士課程修了，工学博士．同年京都大学工学部情報工学教室助手．1978 年 同助教授．1986 年 九州大学大学院総合理工学研究科教授．1981 年 京都大学工学部情報工学科教授．計算機アーキテクチャ，並列計算機システムに興味を持つ．著書「並列計算機構成論」「並列処理マシン」「コンピュータアーキテクチャI」など．電子情報通信学会，IEEE，ACM 各会員．