Virtual Queue：超並列計算機向きメッセージ通信機構

五島 正裕† 松本 重光 森 眞一郎 中島 浩 富田 眞治

京都大学 工学部

†：日本学術振興会特別研究員

概要

Virtual Queue システムは，通信機構の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のための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でき，通常のキャッシュと同様システム・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．物理的通信路はベクトル・データの転送にも十分なスループットを持つ．

Abstract

The **Virtual Queue** system is a message communication mechanism, marked by virtualization of message communication hardware and the caching of message streams. The virtualization provides arbitrary number of virtual communication channels, and enables a user to access the hardware without system call just like a usual cache memory. It drastically reduce the message handling overhead of software, the system withstands finer grain communications. And the system makes use of the locality of reference, not to messages themselves but to the stream of the messages, and it caches frequently accessed message streams to physical channels, which provide 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．しかし，従来のメッセージ通信の方式は ① メッセージ処理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対しては効果的ではない などの問題点がある．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，2.1 では メッセージ送受信のオーバーヘッドについて，続いて 2.2 では 通常のキャッシュのメッセージに対する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

共有メモリ/メッセージ，両方式において，メッセージ送受信のオーバーヘッドは数百サイクル程もある．このオーバーヘッドは，細粒度のメッセージを頻繁に交換する応用において，性能に対する影響が大きい．

共有メモリ方式のオーバーヘッド

一般に，共有メモリ・システム上でメッセージ通信を行う場合には，遠隔メモリ上のバッファに対するロック操作が必要となる．超並列規模の計算機では，遠隔ノードに対するロック操作のレイテンシは数百サイクルにものぼる．単なる読み出し操作の場合などとは異なり，更新型のキャッシュ一貫性制御やプリフェッチなどでは，ロック操作のレイテンシを隠蔽することはできない．

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，メッセージ処理の高速化のためにハードウェアを改良することが一般的となっているが，これらの計算機におけるオーバーヘッドはむしろメッセージ処理のソフトウェア部分で生じている．一部のソフトウェア処理は，ソフトウェア自体の最適化やハードウェアの工夫によって省略可能であるが，資源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存在する．これらのソフトウェア処理のため，500 ~ 1000 サイクル程のオーバーヘッドが生じている [1]．

2.2 キャッシュのメッセージに対する不適合

2 つ目の問題点は，キャッシュのメッセージ・アクセスに対する不適合である．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．

参照の時間的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ングする意味は薄い．それにもかかわらずメッセージをキャッシングすると，本来キャッシングしておくべき他のデータを追い出してしまい，かえって性能を低下させかねない．

Virtual Queue：A Message Communication Mechanism for Massively Parallel Computers

M.Goshima† S.Matsumoto S.Mori H.Nakashima S.Tomita

Faculty of Engineering, Kyoto University

†：Research Fellow of the Japan Society for the Promotion of Science

キャッシュを利用したバッファリング

一方，1 つのメッセージ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が有効に働く可能性が高い．

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][2][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][3][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. メッセージが 2 度ずつキャッシュ・メモリのポートを通過するため，スループットが不足する．
2. プリフェッチ・ポストストアの処理量の増加が性能に影響を及ぼす．
3. また，前節で述べた他のデータの追い出しの問題も解決されない．

3 *VQ* システムの設計方針

VQ システムは，前章で述べたの 2 つの問題点を以下のような方針で解決する．

3.1 仮想化によるオーバーヘッドの低減

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しているために，バッファに対するロック操作を避けることはできない．一方 メッセージ方式では，受信ノード上のメッセージ処理用ハードウェアがバッファを集中管理し，処理を不可分に実行するために明示的なロック操作を必要としない．しかし メッセージ方式は一方で，このメッセージ処理用ハードウェアの獲得とアクセス保護のためのシステム・コールが不可欠となってしまう．

VQ システムは，メッセージ方式と同様に，バッファ管理用のハードウェアを導入して明示的ロック操作を避けるが，そのハードウェアを仮想化することで OS の介入を不要とするアプローチを採る．

仮想化

さて，仮想記憶方式の記憶保護機能によって，メモリ・マップ I/O をユーザに解放することは一般に行われている．しかし，それ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ず，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全く不十分である．

一方 通常の (物理アドレス) キャッシュは，OS をまったく介することなく利用されている．それは資源 ——— キャッシュ・ブロックがメモリにマップされていて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，その所有者の交替をハードウェアで実現しているためである．これと同様に，VQ システムは以下のようにして OS 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

ハードウェア・ユニットの処理中の状態を定義するのに十分な情報をそのユニットの コンテクスト と呼ぶ．例えば，リング・バッファの制御回路は，バッファの書き込み/読み出し位置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制御回路のコンテクストとなる．

VQ システムでは，通信機構のコンテクストをユーザ・メモリ上の退避領域に置き，コンテクストを適宜切替えることによって，通信機構を時分割多重利用する．コンテクストをキャッシュする専用のキャッシュ ——— コンテクスト・キャッシュ を用意し，ユーザ・プログラムの退避領域へのアクセスに対して，システムのコントローラがこのキャッシュを使用する形でこれを実現する．こうすることによって，仮想記憶方式の記憶保護機能を応用することができるうえに，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的に実現できるので，通常のキャッシュと同様にシステム・コールを介さない利用が可能になる．

また，通信機構の個々のコンテクストは主記憶上におかれるので，実際上任意の数の仮想的な通信機構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仮想通信機構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．

このように，仮想化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ほとんどを省略することができる．

3.2 バッファリング方式の改良

キャッシュのように，高速なメモリにデータをバッファリングすることによってレイテンシを短縮することは重要であるが，通常のキャッシュは主に参照の時間的局所性を利用して高速化を図るものであるため，メッセージのようにその性質の弱いデータに対しては効率的に働かず，前述のような問題を生じる．

VQ システムでは，通常のキャッシュと同様にメッセージを高速メモリにバッファリングするのだが，メッセージに対するアクセスの性質を利用してバッファリングのために必要な容量を大幅に削減し，このメモリをデュアルポート化/専用化することによってこれらの問題に対応する．

ストリームによる通信モデル

まず第 1 に VQ システムでは，通信モデルをストリームによるもののみに制限する．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

クに対する参照を再び行うことはできないものとする．こうすることによって，古いメッセージの占めていた高速メモリ上の領域を直ちに再利用することができ，高速メモリに必要な容量を削減することができる．

ストリームによる通信のみを用いる言語も多い，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と考えられる．

また ストリームによる通信を仮定すれば，プリフェッチ/ポストストアを自動的に行うことができる．

ストリームのキャッシング

第 2 に 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．さて，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．前述のように，VQ システムはハードウェアのコンテクストのキャッシングを行うのだが，これ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．

バッファリング用の高速メモリもコンテクスト・キャッシュの一部として実装される．このメモリも含めて，コンテクスト・キャッシュは少ない数 ——— 例えば，プロセッサごとに 4 ~ 16 エントリ程度しか用意しないが，そのプロセッサで走行中のスレッドがそれ以下のストリームしか使用しないならば VQ へのアクセスはヒットし続ける．

以上の 2 点によって，バッファリングのために必要なメモリの容量は大幅に削減される．バッファ用メモリとして小容量のもの ——— 256B ~ 1KB 程度を仮定すれば，デュアルポート・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ることは十分に実現可能である．

専用のデュアルポート・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，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．

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 節では 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 節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 1 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．プログラムは任意の数の 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．

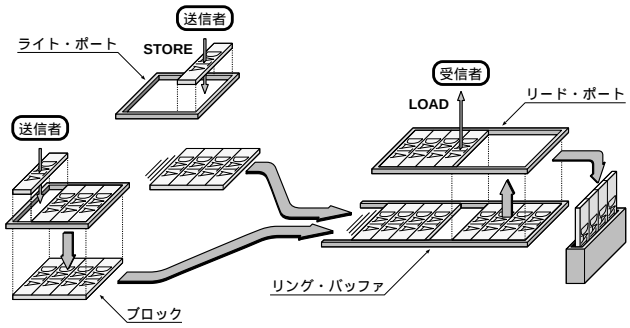


図 1: プログラミング・モデル

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 つ以上のライト・ポートと，1 組のリング・バッファとリード・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．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

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 システムのハードウェアが VQ のコンテクストの位置を特定するため，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 1 つ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アクセスが行える必要はない．ノード間にまたがる大域的仮想アドレス空間の構成方法は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，VQ アクセスのためのアドレス空間 ——— ポート・アドレス空間 を設ける (図 2 左)[1]．ポート・アドレスは，主記憶の物理アドレスに対する一種の別名である．ポート・アドレス空間は，主記憶の物理アドレス空間と等しい大きさを持つ．また，ポート・アドレス空間上のページ ——— ポート・ページ は，そ

[1] その他にメモリ・マップ I/O のアドレス空間があってもよい．

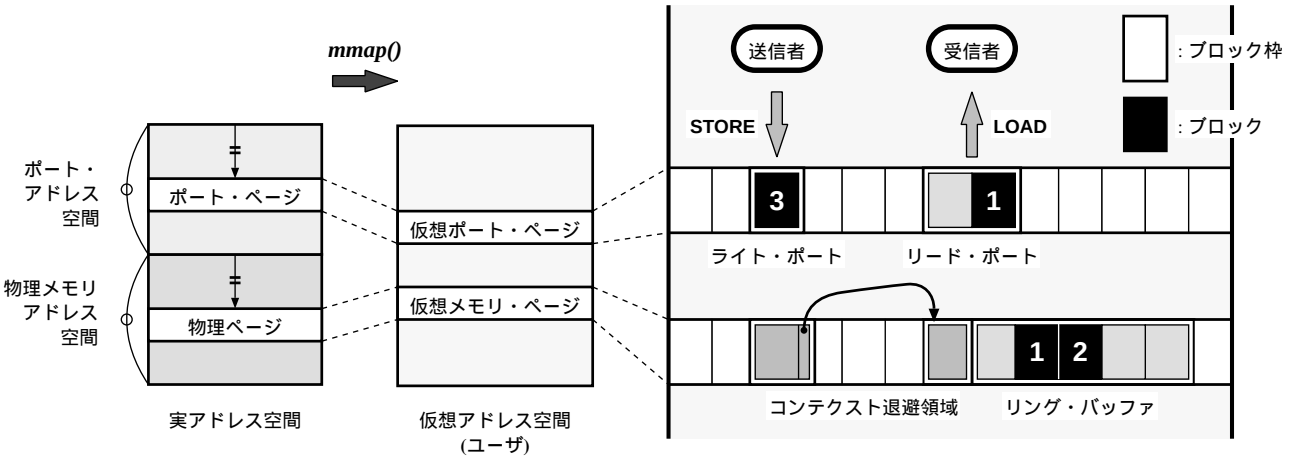


図 2: VQ のデータ構造

れぞれのアドレス空間内での変位が互いに等しい物理ページと対応する．

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，物理ページに対応するポート ·ページを獲得し，仮想ページに写像する (図 2 左から中)．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 (理由は後述)．

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 と呼ぶ．

これらのページの上に 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，一旦ページを割り付けられた後には VQ に関するすべてのアクセスはユーザ ·モードで行うことができる．

***VQ* のデータ構造**

プログラムは，獲得したページ上に図 2 右 に示すように，VQ のデータ構造を設定する．

VQ の各要素は，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 2 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 2 右下の矢印)．図 2 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．

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 2 右上)．

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 2 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ポート ·アドレス空間内の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

通信開始前には，上記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る必要がある．1 つめの VQ に対しては，VQ 以外の同期手段を用いる．

以上の処理の後に，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リング ·バッファの末尾に転送/追加される．転送すべきデータが 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．

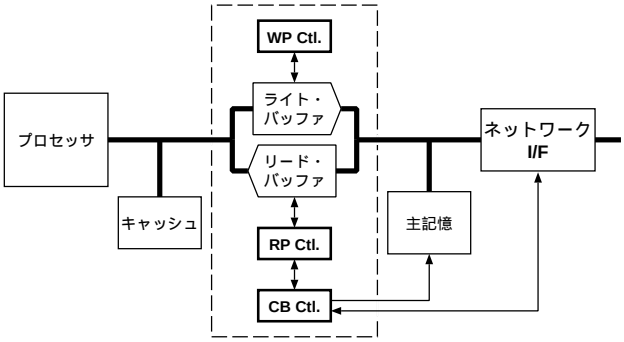


図 3: 処理ノードの構成

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．

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3に，対象計算機の処理ノードの構成を示す[2]．システムはデュアルポート・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．プロセッサがデュアルポートの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，そのコンテクストを主記憶上の退避領域からコンテクスト・キャッシュにキャッシュし，それ上のコンテクストを参照しながら動作する．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3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証する必要はない．

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

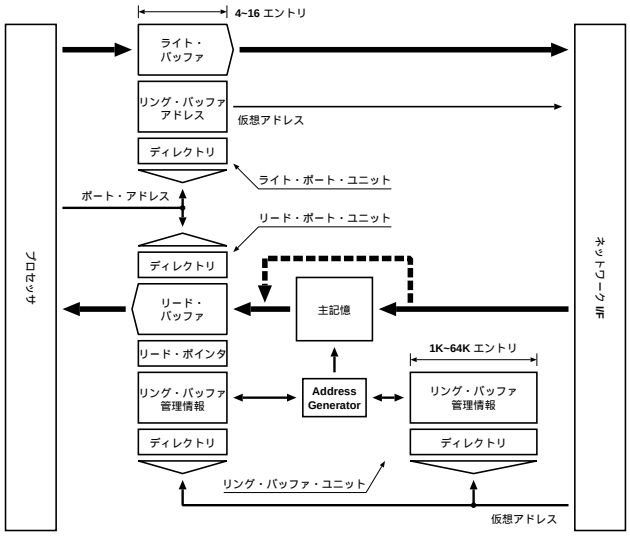
VQシステムの各ユニットは，それぞれのコンテクスト・キャッシュを持ち，コンテクストはそれぞれ独立にキャッシングされる．

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4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．

[2] 図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 4: コンテクスト ・キャッシュ構成

(1) ライト ・ポート ・ユニット

コンテクスト

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. また, 送信先のリング ・バッファの仮想アドレスもコンテクスト ・キャッシュにキャッシュし, 送信処理を高速化する.

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
- 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とすれば, 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, ハードウェア量としては外部 2 次キャッシュのディレクトリなどと同程度である.

(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 4 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上側は, 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スピン ・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するので，ブロックは主記憶をバイパスしてリード ・バッファに送ることができる (図 3 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 アンダーフロー/オーバーフローの検査

本節では，VQ のアンダーフロー/オーバーフローの検査の方法についてまとめる．以下で述べる方法の中には割り込みを用いるものがあるが，紙面の都合により割り込みからの回復の方法に関しては割愛する．

4.4.1 アンダーフローの検査

VQ のアンダーフローは，以下の 3 種のアクセスによって検査される ：

1. 検査用コマンドを使用　コマンドにより，リング ・バッファ中のブロックの数を検査する．
2. 使用しないパタンを応答　アンダーフローを起こした読み出しに対して，ヌル ・ポインタ値 0，あるいは，NotANumber など，アプリケーションが使用しないパタンをブロックにつめて応答する．

   例えば，次のようにするとよい :送信者はブロックの最終バイトへ 0 以外の値を書き込むことによって送信を行い，受信者はその位置が 0 以外であることを確認する．
3. 割り込みを発生する　アンダーフロー時に (しばらく待って) 割り込みを発生する．この方法では，プログラムは VQ のアンダーフローを明示的に検査する必要がないので，VQ が空にならない可能性が高い場合に有効である．

4.4.2 オーバーフローの検査

プログラムは，原則として，以下のいずれかの方法によってオーバーフローの発生以前に送受信するスレッド間で速度を調整するものとする ：

- コマンドにより定期的にブロック数を検査する．
- ブロック数がユーザが設定した値になった時点で割り込みを発生する．

それでもオーバーフローを起こした時には，しばらく受信処理を中断した後に，プロセッサに割り込みをかけて知らせる．エンキューできるようになるまで受信処理を中断する方法も考えられるが，デッドロックを起こす可能性があるので採用しない．

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[a]	10cycles

[a] ノードからノードまで

表 1: シミュレーション ・パラメータ

5 性能評価

256×256 行列 (倍精度) の行列積 $A \cdot B \rightarrow C$ 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それぞれ説明する．

5.1 評価モデル

モデルは，基本的に図 3 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の程度の，性能への影響を取り除くことに特に考慮した ：

プロセッサ　プロセッサ ・アーキテクチャやソフトウェアの最適化などの影響を取り除くため，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スーパースケーラとする．内積型の行列積では 1 積和演算 (2FLOP) ごとに A，B の計 2 要素をロードするので，理論最大性能は 1FLOPC となる．最近のプロセッサ技術と最適化技術によれば，これに近い性能を出すことができるので，この仮定は妥当であると考えられる．

キャッシュ　競合性のミスの影響を回避するため，完全連想とする．置換アルゴリズムは LRU とした．

メモリ　SDRAM をモデル化した．

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．

その他のパラメータを，表 1 にまとめる

上述の基本モデルに対して，VQ の他に以下の 2 種の方式を実装し，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 3 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，主記憶に対してバッファリングを行わないため，理想的な通信機構であると考えられる．

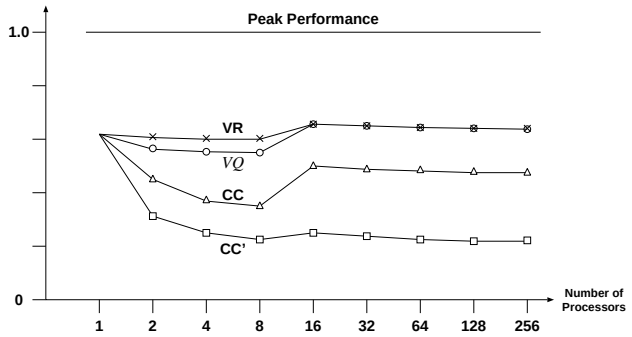


図 5: シミュレーション結果

5.2 評価プログラム

今回用いた行列積プログラムでは，アクセス ——— 特にロードのレイテンシを軽減するため，データをレジスタ上に保持してい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 ・バッファをとって通信を行う．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 ・レジスタを用いて通信を行う．各方式において，バッファ ・サイズは行列の 1 行分とした．

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．

5.3 実行結果

結果を図 5 に示す．横軸にはプロセッサ数を，縦軸には理論最大性能を 1 とした場合の 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．このプログラムでは明示的通信/同期が不要であるため，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．

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以上の効率を示している．

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，シミュレーションの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出ていない．

6 おわりに

VQ システムは，仮想化により任意の数の仮想的通信機構を提供し，システム ・コールを不要とするなど，メッセージ送受信時にオーバーヘッドをほとんど生じない．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をキャッシングすることによって，ハードウェアを小量化/高速化する．

シミュレーションにより，性能上 理想的な通信機構である VR 方式に匹敵する動作が可能であることが示された．ところで VR 方式は，ベクトル ・レジスタの内容はプロセッサのコンテクストとなるため，実際上 マルチ ・ユーザ環境で用いることができない．したがってその可用性を考慮すれば，VQ はほぼ理想適なメッセージ通信機構であるといえる．

ただし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の効果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，固定長メッセージ方式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する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，特別研究員奨励費 「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).