並列計算機 Cenju-4 のユーザレベルネットワークインタフェース

加納 健[†] 中村真章[†] 広瀬哲也[†]
細見岳生[†] 中田登志之[†]

並列計算機 Cenju-4 は，ユーザレベルのメッセージ通信機能を持った分散メモリとキャッシュ一貫性制御付き分散共有メモリの 2 つのメモリアーキテクチャを持つ．Cenju-4 の構成は，8〜1024 個のノードが，マルチキャストと同期の機能を持つ多段ネットワークで接続されている．各ノードは，MIPS の R10000 と 512 Mbyte までのメモリからなる．本論文では，Cenju-4 のユーザレベルメッセージ通信を司るネットワークインタフェースのアーキテクチャとその実現方法に関して述べる．さらに，メッセージ通信の性能評価結果についても述べる．

User-level Network Interface for a Parallel Computer Cenju-4

Yasushi Kanoh,[†] Masaaki Nakamura,[†] Tetsuya Hirose,[†]
Takeo Hosomi[†] and Toshiyuki Nakata[†]

Cenju-4 is a parallel computer which supports two memory architectures, a distributed memory with user-level message passing communication and a distributed shared memory with cache-coherent non-uniform memory access (cc-NUMA) feature. The Cenju-4 system consists of up to 1024 nodes connected by a multistage network which has multicast and synchronization functions. Each node has the MIPS R10000 processor with up to 512 Mbyte memory. This paper describes architecture and implementation of the Cenju-4 user-level network interface. In addition, performance results are presented for message passing communication.

1. はじめに

並列計算機 Cenju-4 のネットワークインタフェースの設計時には次のような課題があった．

- 低レイテンシ，高スループットの実現
- ユーザレベル通信
- マルチタスク環境の支援
- ネットワークによるマルチキャストと同期の支援

マイクロプロセッサにはスーパスカラが採用され，計算性能は飛躍的に向上した．その計算性能とバランスをとるためには通信性能の向上が必須となった．通信性能を表す指標には，レイテンシとスループットがあるが，メッセージ通信にはこの両方の性能が必要となる．たとえば MPI では，プロトコル部分には低レイテンシが，実際のデータ転送には高スループットが要求される．

また，低レイテンシを実現するためには，システムコールを使わずユーザレベルから直接起動できるユーザレベル通信が必須である．これを実現するためには，OS が行っていた情報の保護とイベントのユーザへの通知をハードウェアが行う必要がある．Cenju-4 では論理値から物理値に変換するテーブルを使って情報の保護を行っている．また，イベントの通知方法としては，割込みに加えて，指定されたアドレスに完了通知を書き込む方法をとっている．

さらに，通常の OS を動かすには，ネットワークインタフェースをマルチタスク環境でも使えるように設計する必要がある．そのためには，複数のユーザタスクへのパケットの到着への対応や，OS にユーザタスクの切替えを行えるタイミングを伝える必要もある．

Cenju-4 のネットワークはマルチキャストと同期の機能を持っている．しかしながら，これらの機能はパケットヘッダに正しいルーティング情報が設定されていることを前提にしている．Cenju-4 では，システムコールによるルーティング情報のテーブルへの登録と，ユーザプログラムからのそのテーブルのオフセットの指定という方法で，これらの機能を使えるようにしている．

本論文では，Cenju-4 のユーザレベルネットワーク

† NEC C&C メディア研究所
C&C Media Research Laboratories, NEC Corporation

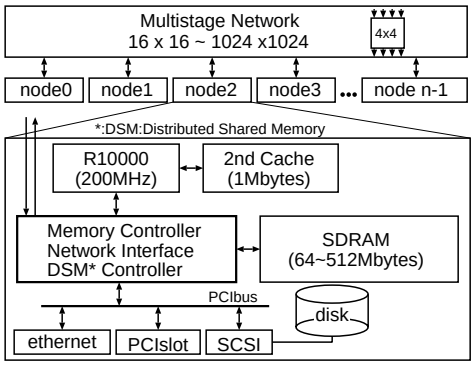


図 1 Cenju-4 のシステム構成
Fig. 1 Cenju-4 block diagram.

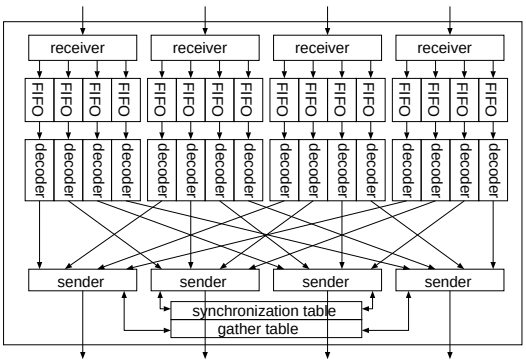


図 2 Cenju-4 のネットワークスイッチ
Fig. 2 Cenju-4 network switch.

インタフェースがどのような機能を持ち，どのように実現されているかについて述べる．さらに，実現方法を評価するための実験結果を示す．

2. Cenju-4 の概要

Cenju-4 は，1024 個までのノードが 4×4 のネットワークスイッチで構成される多段ネットワークによって接続された構成をとる（**図 1**）．

各ノードは CPU として MIPS の R10000 を搭載している．R10000 は各 32 Kbyte の命令キャッシュとデータキャッシュを内蔵している．Cenju-4 では 1 Mbyte の SSRAM を 2 次キャッシュとしている．主記憶は 512 Mbyte までの容量の SDRAM で構成されており，8 byte × 80 MHz のピークスループットを持つ．システムの鍵となる LSI は，メモリの制御（メモリコントローラ）と，プロセッサ間通信の機能である，分散共有メモリ（DSM コントローラ）[4]とメッセージ通信（ネットワークインタフェース）の機能を持つ．

OS として，MACH マイクロカーネルベースの並列 OS である DE4 を開発した．マイクロカーネルベースの OS では，それぞれの機能がモジュールごとに分かれているため，機能の拡張や追加に適している．DE4 では並列処理機能としてユーザレベル通信サーバや分散共有メモリサーバなどを追加した．

3. Cenju-4 の多段ネットワーク

並列プログラムでよく使われる通信パターンとして，マルチキャスト通信や同期などがある．Cenju-4 のネットワークには，これらの集団通信をサポートする高機能なネットワークを採用することとした．

まず，マルチキャスト機能を実現するためには，2 つのスイッチが同時に同じスイッチにマルチキャストパケットを書き込む場合の調停で発生するデッドロックを回避する必要がある．デッドロックを回避するために，書き込み時に調停の不要なクロスポイントバッファ方式を選択した．4 × 4 のクロスポイントバッファのスイッチには 16 個の FIFO メモリが必要である（**図 2**）．また，デッドロックを回避するもう 1 つの手段としてヴァーチャルカットスルー方式をとった．

次に，ノードの任意の集合間の同期機能を実現するために，各同期処理の状態を記憶する同期テーブルをスイッチに持たせた．

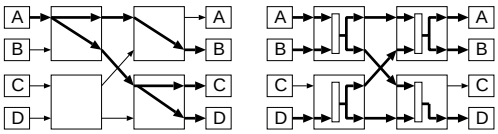


図 3 Cenju-4 ネットワークの機能
Fig. 3 Cenju-4 network functions.

図 3 に Cenju-4 のネットワークの機能を概念的に説明する．マルチキャストのルーティング情報は，64 個のノードが構成するグループ内のそれぞれのノードを 1 bit で示す 64 bit と，どのノードグループかを指定する bit からなる．この情報を使って，ノードグループ内の 64 個のうち任意の集合に対するマルチキャストが可能である．同期のルーティング情報も 64 個のノードを 64 bit で表す方法を使う．スイッチ内のヘッダレジスタの容量の制限から，任意のノード集合間の同期機能は 64 ノード以下のシステムでしか動作しない．

これらの機能を持つ 4 × 4 のネットワークスイッチを設計した．このスイッチは，3 byte × 80 MHz のハードウェアスループットと，シングルキャスト，マルチキャスト，同期に対して，それぞれ，125，187.5，200+ α☆ nsec のレイテンシを持つ．ネットワークボー

☆ 同期テーブルのアクセスで待つ場合があるため．

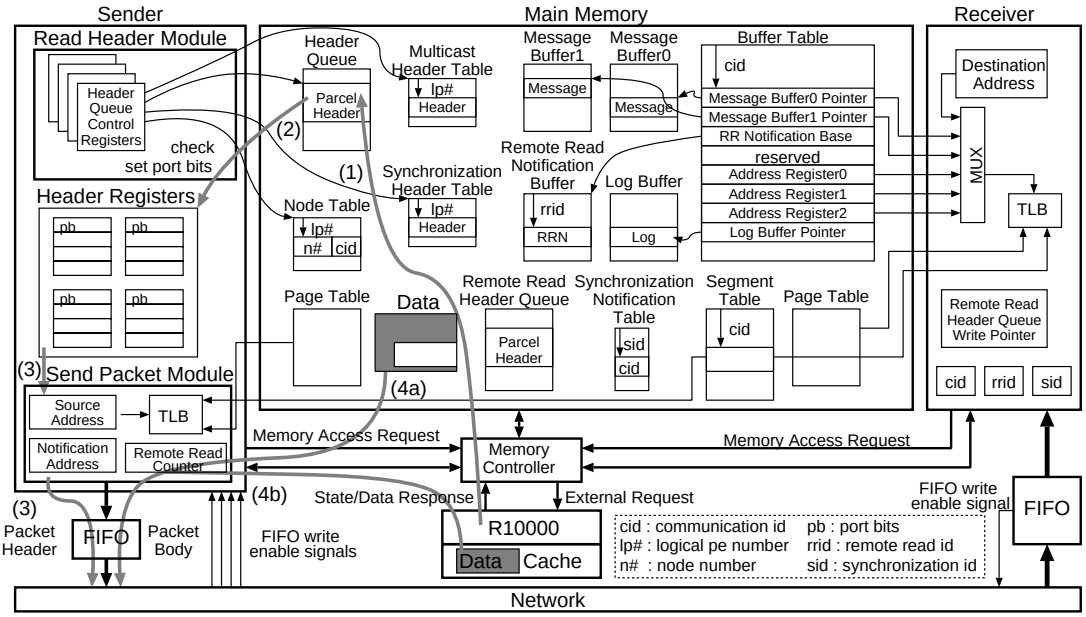


図 4 Cenju-4 のネットワークインタフェース
Fig. 4 Cenju-4 network interface.

ドは，このスイッチを 8 個搭載し，16 × 16 の多段ネットワークを構成する．ネットワークボードをケーブルで接続して，各構成の多段ネットワークを構成する．

各フリット（3 byte）には 1 bit のパリティビットがついている．また，3 フリットで，64 bit のデータと 8 bit の ECC チェックビットになっている．さらに，フリット落ちを検出するために，各フリットには 3 bit のポジションビットがついており，奇数フリット，偶数フリット，ラストフリットのどれかを表している．1 つのパケットで転送できる最大データ長は 256 byte である．これは 2 次キャッシュの 2 ライン分にあたる．パケットの最初のフリットには，パケットの種類とポートビットと呼ばれる 4 bit の情報がある．ポートビットは次のスイッチの 4 個の FIFO メモリのどれにこのパケットを書き込むかを示している．

4. ネットワークインタフェースの機能と構成

Cenju-4 のネットワークインタフェースは，パケットをネットワークに送り出す Sender と，ネットワークからパケットを受け取る Receiver の独立した 2 つのモジュールから構成される．また，2 つのモジュールとネットワークの間には 2 つの FIFO メモリがあり，データ幅の変更やパリティビット，ECC チェックビットの生成とチェックを行っている（**図 4**）．

4.1 ネットワークインタフェースの機能

Cenju-4 では，パーセルと呼ばれる，ヘッダとデータからなる構造体を通信の単位として扱う．送信を行うには，パーセルのヘッダを主記憶上のヘッダキューに書き込み，Sender の書き込みポインタを更新する．パーセルは Sender によっていくつかのパケットに分割され，ネットワークに送られる．

表 1 ネットワークインタフェースのプロトコル
Table 1 Cenju-4 network interface protocols.

protocol	2nd	3rd	4th	description
remote DMA	src	dst	(ntf)	remote DMA from src to dst
r DMA with addr reg0/1/2	src		(ntf)	remote DMA from src to v(address register 0/1/2)
message0/1	src		(ntf)	remote DMA from src to message buffer 0/1
atomic add	dst	im	(ntf)	adds im to v(dst)
remote read	src	dst	(ntf)	remote DMA from src to dst on the remote node
swap	dst1	im	dst2	writes im into dst1 and writes v(dst1) into dst2
comp&swap	dst1	im1	im2	if v(dst1) = im1 then writes im2 into dst1, and returns v(dst1) and result of comparison

src : source address, dst : destination address,
ntf : notification address, im : immediate, v() : value

1 つのパーセルは最大 512 Kbyte のデータを送ることができる．パーセルヘッダは 32 byte で構成される．最初の 8 byte には，パケットの種類，プロトコルの種類，データ長，宛先論理プロセッサ番号，その他制御情報が格納されている．残りの 24 byte はプロトコルによって異なっており，ソースアドレス，デスティネーションアドレス，送信完了の通知先アドレス，などから構成される．**表 1** に Cenju-4 のネットワークインタフェースで使えるプロトコルとパーセルヘッダの構成を示す．

ネットワークインタフェースでのマルチタスクを実現するため，タスクを区別する 8 bit のコミュニケーショ

ン ID（cid）を導入した．ネットワークインタフェースの資源は cid によって多重化される．10 bit の物理ノード番号と cid の組合せがパケットの宛先となる．

4.2 Sender

各 cid に対して，パーセルヘッダキュー，論理アドレスを変換するためのページテーブル，宛先論理プロセッサ番号を変換するためのノードテーブル，マルチキャストヘッダテーブル，同期ヘッダテーブルが主記憶上に設けられる．

Sender は 2 つのサブモジュールからなる．RHM（Read Header Module）はパーセルヘッダをパーセルヘッダキューから読み出し，ヘッダレジスタに格納する．SPM（Send Packet Module）は，ヘッダレジスタ内のパーセルヘッダとメモリ内のデータを使って，パケットを作って送り出す．

Sender は次の手順でパーセルを送信する（図 4 内の矢印の数字に対応）．

(1) Sender を割り付けられたタスクが，パーセルヘッダキューにパーセルヘッダを書き込み，その書き込みポインタを更新し，Sender に対してパーセルの送信を依頼する．

(2) RHM がメモリからパーセルヘッダを読み出し，パーセルヘッダの正しさを調べ，パケットの種類によって，ノードテーブル，マルチキャストヘッダテーブル，同期テーブルをアクセスして，ポートビットを付け加えて，4 個あるヘッダレジスタの 1 つに格納する．

(3) SPM は，ヘッダレジスタ内のパーセルヘッダのポートビットと次段のスイッチの FIFO メモリの状態とから，パケットを送ることができるパーセルヘッダを 1 つ選び，読み出して，TLB でソースアドレスを変換し，パケットヘッダを作って FIFO メモリに書き込む．

(4) SPM は，メモリコントローラに対してパケットの本体にあたるデータの読み出し要求を出し，主記憶（4a）または，キャッシュ（4b）からのデータを FIFO に書き込む．

各タスクは 4 ～ 64 Kbyte の大きさのパーセルヘッダキューを主記憶上に持つ．RHM はパーセルヘッダキューを制御する制御レジスタセットを 4 個持つ．1 つの制御レジスタセットは OS 用で，そのヘッダキュー内のパーセルはスーパバイザモードで処理でき，アドレスなどに物理値を直接指定できる．もう 1 つは，受信側から返事を返すプロトコル remote read，swap，comp&swap 用である（以下，リモートリード用）．残りの 2 つはユーザタスク用である．各タスクはヘッダキューを制御するための構造体を主記憶上に持っており，Sender が割り付けられたときに OS によって構造体の値がこれらの制御レジスタセットに設定される．

Sender には次のような理由から 4 個のヘッダレジスタがある．クロスポイントバッファ方式のスイッチでは，次の段のスイッチのどの FIFO メモリにパケットを書き込むかでルーティングが行われる．そして，スイッチの送信部は，別々の入力ポートに接続された 4 個の FIFO メモリに接続されており，その中の 4 個のパケットのヘッダのポートビットと，次の段のスイッチの 4 個の FIFO メモリの状態とから，送信可能な 1 個のパケットを選んで送信する．したがって，Sender は最初のルーティングを行うことになり，クロスポイントバッファのメリットを活かすには複数のヘッダレジスタ内のパーセルが送信の候補となるような構成が必要となる．

SPM はパケットのデスティネーションアドレスがキャッシュライン境界になるようにパケット長を決定する．なぜなら，受信側での主記憶への書き込みが 1 キャッシュライン全体への書き込みになるからである．キャッシュライン全体の書き込みは，主記憶への書き込みと R10000 へのそのキャッシュラインの無効化要求だけで処理されるため高速に行える．また，SPM は，1 つのパケットのデータの読み出しと書き込みでページ境界を越えないようにパケット長を決定する．

SPM からのデータの読み出し要求に対して，メモリコントローラは R10000 にそのアドレスのキャッシュラインの状態を問い合わせる．そして，キャッシュの状態が dirty の場合には，キャッシュから出てきたデータをメモリコントローラが受け取る必要がある．Cenju-4 では，キャッシュの状態の問合せに対する応答の標準的なレイテンシは 160 nsec である．また，キャッシュラインが dirty で，キャッシュからデータが出てくるまでの標準的なレイテンシは 410 nsec である．このため，R10000 のキャッシュからのデータの読み出しのスループットは約 200 Mbytes/sec になる（128 byte ÷ (410 nsec + 12.5 nsec × 16)）．

送るべきデータは少し前に R10000 によって作成されているので，そのキャッシュラインの状態は，ほとんどの場合 dirty である．したがって，もし外部にタグメモリを付けて dirty かどうかを R10000 に問い合わせることなく判断できたとしても，ほとんどの場合が dirty なので，R10000 に問合せをして data response を受ける必要があり，タグメモリは有効に働かないと考える．

R10000 へのキャッシュの状態の問合せの回数を少なくするために，大きい方（128 byte）の 2 次キャッ

シュラインサイズを採用した．さらに，メモリコントローラは，R10000 がキャッシュから返してきたデータを主記憶に書く前に，直接 SPM に渡すことが可能である．これにより，メモリコントローラが主記憶にデータを書いた後，それを読み出して SPM に渡す分のレイテンシを削減している．

4.3 Receiver

Sender とは異なり，Receiver は一時にどの cid のパケットも受信する可能性がある．したがって，Receiver の資源はすべて主記憶上に実現し，必要に応じて受信したパケットの cid の資源を Receiver にキャッシュする．

バッファテーブルは，各 cid に対する 64 byte の資源の構造体を 256 個並べたテーブルである．バッファテーブルのエントリは Receiver 内にキャッシュされて使われるが，値の更新は主記憶上のバッファテーブルにもすぐに反映されるので，タスクはバッファテーブル（キャッシュ上）をポーリングすることになる．資源の構造体は以下のものからなる（図 4）．

- 2 つのメッセージバッファポインタは，message のパケットを書き込むメッセージバッファの書き込みポインタである．タスクはこのポインタをポーリングすることでメッセージの到着を知ることができる．
- 3 つのアドレスレジスタは，remote DMA with address register のパケットの書き込みアドレスを格納している．対応するタスクが CPU に割り付けられている間は，タスクがアドレスレジスタの値を更新できる．
- リモートリード到着通知ベースアドレスレジスタは，リモートリード到着通知バッファの先頭を指している．このバッファは remote read と comp&swap の返事の到着をタスクに知らせるために用いる．これらのプロトコルのパーセルには 6 bit のリモートリード ID（rrid）が設定されており，Receiver がこれらの返事の最後のパケットを受信したときに，この rrid をオフセットとするリモートリード到着通知バッファに，到着通知や比較結果を書き込む．ユーザタスクは rrid に対応したリモートリード到着通知バッファのアドレスをポーリングすることにより，返事の到着を知ることができる．
- ログバッファポインタは，ログバッファの書き込みポインタである．ログバッファには到着したパケットのログが Receiver によって書き込まれる．ログには，到着したパケットの種類，送り元の論理プロセッサ番号，データの書き込みアドレスなどの情報が含まれる．

Receiver は remote read，swap，comp&swap に対して返事を送り返す．これらのパケットを受信した場合，Receiver はその返事のパーセルヘッダをリモートリード用ヘッダキューに書き込み，書き込みポインタを進める．このヘッダキューが溢れるのを防ぐため，Sender はこれらのパケットの発行個数を制限する．

5. Cenju-4 での課題の実現方法

この章では，1 章であげたネットワークインタフェースの課題をどのように実現しているかについて述べる．

5.1 通信性能の向上

Cenju-4 では次に述べる転送方法により通信の高速化を図っている．

5.1.1 インラインモード

remote DMA などのプロトコルでは，パーセルヘッダに 8 byte のソースアドレスを指定している．しかしながら，もし送るデータが小さな場合には，この 8 byte に直接データを書き込めば，データを読み出すためのメモリコントローラへのアクセスを行わなくてもパケットを送ることができる．インラインモードとは，ソースアドレスによって送るデータを指定するのではなく，パーセルヘッダに送信するデータを直接書き込んで送信するモードである．パーセルヘッダにインラインモードを示す bit を指定することによって，Sender はパーセルヘッダにデータが書いてあるものとしてパケットを送信する．remote DMA では，1 ダブルワード（64 bit；以下，DW）までのデータをインラインモードで送ることができる．

5.1.2 データ読み出しの最適化

4.2 節で述べたように，Sender の SPM は，パケットのデスティネーションアドレスがキャッシュライン境界になるようにパケットを送る．SPM の送信データの読み出しはキャッシュライン単位で行われるので，SPM が 1 パケットを送るのに，メモリコントローラに対して何回読み出しリクエストを出すかは，ソースアドレスのアラインメントによる．もし，ソースアドレスがキャッシュライン境界なら，SPM は，1 つのパケット（2 キャッシュライン分）を送るのに 2 回のコヒーレント要求（R10000 にキャッシュの状態を問い合わせ，dirty な場合はデータをキャッシュからフラッシュさせる）を出す．そうでない場合には，SPM は 3 回のコヒーレント要求を出す．

しかしながら，そのパケットがそのパーセルの最初のパケットでない場合，3 回のコヒーレント要求のうちの最初の要求はコヒーレントである必要がない．なぜなら，最初の要求で読み出すデータは，その直前の

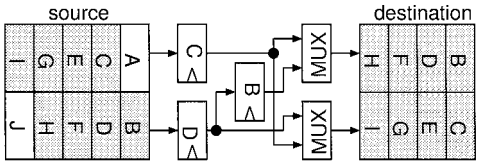


図 5 奇数ワード境界データの並べ換え
Fig. 5 Rearrangement of odd word aligned data.

パケットを送ったときの 3 回目のコヒーレント要求で読み出されたのと同じラインで，すでにキャッシュから主記憶にライトバックされているはずだからである．SPM では，そのパケットが最初のパケットでない場合，3 回の要求のうち最初のものをノンコヒーレント（R10000 にキャッシュの状態を問い合わせずに直接主記憶から読み出す）の要求にして，データの読み出しの最適化を行っている．

5.1.3 奇数ワード境界のデータ転送

Cenju-4 では，32 bit を 1 ワードとしてアクセスするプログラムもある．そのため，メッセージ通信のソースアドレスやデスティネーションアドレスが DW 境界にない場合もある．アドレスが DW 境界にあるかないかで以下の組合せがある．

A：ソース，デスティネーションとも DW 境界．
B：ソースは DW，デスティネーションは奇数ワード境界．
C：ソースは奇数ワード，デスティネーションは DW 境界．
D：ソース，デスティネーションとも奇数ワード境界．

B と D の場合は，1 ワードのデータを別のパーセルで送ることで，それぞれ，C と A となる．Cenju-4 では，C もハードウェアでサポートする．

図 5 に，C の場合の処理（B〜I を送る）を示す．ソースアドレスが奇数ワード境界になっている場合，SPM はパケットサイズよりも 1DW 多いデータを読み出し，FIFO メモリに書き込むところで 32 bit のレジスタを使ってワードの並べ換えを行う．

奇数ワード境界のデータの転送では，ソースアドレスのページ境界の先頭の 1 ワードに対し，前のページと物理アドレスで連続しているページのデータを間違えて送ってしまう．そのため，この 1 ワードを別のパーセルで送り直す必要がある．このような処理が必要なのは，DW 境界のデータ転送の構成に必要最低限のハードウェアを付加することにより奇数ワード境界のデータ転送を実現しているからである．このようなオーバヘッドはあるものの，ソフトウェアで DW 境界にコピーしてから送るよりも高速に転送が行える．

5.2 情報の保護

ユーザタスクがアクセスできるネットワークインタフェース内のレジスタの物理アドレスは，別のタスクが使うものどうしは 4 Kbyte 離れているので，タスク間の保護はページのマップによって行うことができる．

一方，Receiver のバッファテーブルは，異なるタスクの構造体が主記憶上に連続して並んでおり，ページのマップでは保護できないが，read only でマップされて各タスクから読み出し専用である．ただし，CPU が割り付けられたタスクのアドレスレジスタは，Receiver のレジスタとして見え更新が可能である．

Cenju-4 ではソースアドレス，デスティネーションアドレスなどに論理アドレスを使う．論理アドレスは，Sender と Receiver の各 TLB で物理アドレスに変換される．TLB はそれぞれ 4 つのページ情報をキャッシュできる．主記憶上には，セグメントの情報を持つセグメントテーブルと，各セグメントごとのページテーブルがあり，TLB は必要に応じて主記憶上のこれらのテーブルにアクセスしてアドレス変換を行う．セグメントテーブルには，各 cid に 4 個ずつのセグメント情報が並んでいる．セグメント情報はページテーブルへのポインタとページ番号の最小値，最大値からなる．Sender と Receiver は，cid をもとにセグメントテーブルを引き，そこからページテーブルをたどって，主記憶から TLB へページ情報の読み出しを行う．また，アクセスしたテーブルに情報がない場合などは，OS への割込みを発生して OS の助けを借りる．

宛先の保護は，Sender がノードテーブル，マルチキャストヘッダテーブル，同期ヘッダテーブルを使って，パーセルヘッダに指定された 12 bit の論理プロセッサ番号をオフセットとして，物理値（物理ノード番号+cid やマルチキャストヘッダ，同期ヘッダ）に変換することで行われる．各テーブルはオフセットの最大値を持ち，オフセットが範囲外のときにはエラーとなる．

5.3 イベントの通知

Sender は 2 つの方法によってタスクにイベントを伝えることができる．1 つは送信完了通知である．SPM は，そのパーセルの最後のパケットを送り終えたところで，パーセルヘッダの 4 個目の 8 byte に書かれたアドレスに，特別なパターンのデータを書き込み，送信が完了したことをタスクに伝える．もう 1 つは，パーセルヘッダキューが空で，かつ，最後のパーセルヘッダの最後のパケットを送り終えた場合に発生する割込みである．この割込みは，OS が Sender を割り付けるタスクを切り替えるタイミングを知るのに使用される．

Receiver では受信終了割込みとバッファテーブルのログバッファなどを使った受信完了の通知が行える．受信終了割込みは受信側で割込み禁止に設定できる．受信通知は主記憶に対してキャッシュ制御付きで行われるので，キャッシュをポーリングすることになる．remote read と comp&swap のプロトコルでは，パケットに付加された rrid をオフセットとしたリモートリード到着通知バッファの位置に返事の受信通知が書き込まれる．また，メッセージバッファやログバッファには，バッファの 4 分の 1 単位でオーバフローによる割込みを設定できる．これにより，バッファの溢れを起こす前に割込み処理を引き起こすことが可能である．

5.4 マルチキャスト，同期のインタフェース

マルチキャストや同期を行うには，それに関わる論理プロセッサの集合を指定する必要がある．Cenju-4 では，あらかじめシステムコールによりヘッダをテーブルに登録し，そのオフセットを論理プロセッサ番号として指定する方法をとった．この方法では，実際にヘッダを作るのはソフトウェアであるが，Cenju-4 のネットワークのマルチキャストと同期のパケットヘッダは物理プロセッサ番号が分かれば簡単に作れるフォーマットである．

ユーザプログラムがマルチキャストを使うには，まず，ヘッダを登録するためのシステムコールを行う必要がある．このシステムコールは，引数となっている論理プロセッサ番号のリストから，物理ノード番号の集合を計算し，その集合に対するマルチキャストのヘッダを作成し，マルチキャストヘッダテーブルに登録する．返り値には，そのヘッダのテーブルでのオフセットが返される．そして，マルチキャストを実行するときに，宛先論理プロセッサ番号の 12 bit に，そのオフセットを指定する．Sender はオフセットで示されたマルチキャストヘッダを使ってマルチキャストパケットを作成しネットワークに送り出す．

同期パケットを使うにも，同期パケットのヘッダを登録するためのシステムコールが必要である．さらに，1 組の同期には，複数の同期を区別するための 7 bit の同期 ID（sid）が必要である．同期パケットのヘッダには sid が含まれていて，スイッチは同期テーブルの sid に対応した場所をアクセスして，待合せの状態を知り同期処理を行う．

デスティネーションアドレスが異なる場合にもマルチキャストが使えるように remote DMA with address register のプロトコルを用意した．Receiver のアドレスレジスタにデスティネーションアドレスを指定することで，プロセッサごとにデータの書き込みアドレスが異なる場合でもマルチキャストを使うことが可能となる．

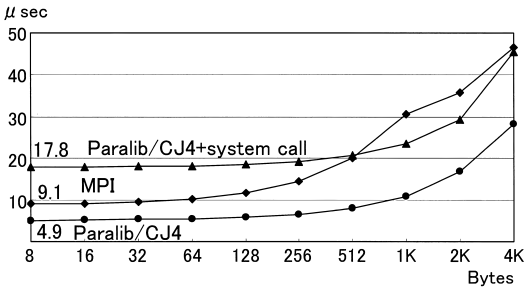


図 6 1 対 1 通信のレイテンシ
Fig. 6 1 to 1 communication latency.

6. 性能評価

メッセージ通信を使った場合の並列計算機の重要な通信性能指標は，1 対 1 通信のレイテンシとスループット，N 対 N 通信でのスループット，バリア同期のレイテンシである．そこで，本章ではこれらの通信性能について評価結果を示す．

6.1 1 対 1 通信

Cenju-4 独自の並列ライブラリ Paralib/CJ4 と，MPI の 2 つの通信ライブラリで，1 対 1 の通信レイテンシ（往復遅延 ÷ 2）を 32 ノードのシステムで測定した（**図 6**）．それぞれ，8 byte 送ったとき，4.9 μsec，9.1 μsec のレイテンシが得られた．Paralib/CJ4 の 8 byte 転送時にはインラインモードを使っているので 16 byte に比べ約 0.3 μsec 少なくなっている．

また，Paralib/CJ4 で送信時にシステムコールを行って送信データのキャッシュフラッシュを行った場合のレイテンシも測定した（図 6 の Paralib/CJ4+system call）．Cenju-4 でのシステムコールのオーバヘッドは約 13 μsec であり，ユーザレベル通信の効果が大きいことが分かる．

1 対 1 通信のスループットを，上の 2 つのライブラリを使って，次の 3 つの場合について測定した（**図 7**）．

A：ソースアドレスがキャッシュライン境界の場合（2 回のコヒーレントリクエスト）．

B：ソースアドレスがキャッシュライン境界でなく，データ読み出しの最適化を行わなかった場合（3 回のコヒーレント要求）．

C：ソースアドレスが，キャッシュライン境界ではなく，データ読み出しの最適化を行った場合（1 回のノンコヒーレント要求と 2 回のコヒーレント要求）．

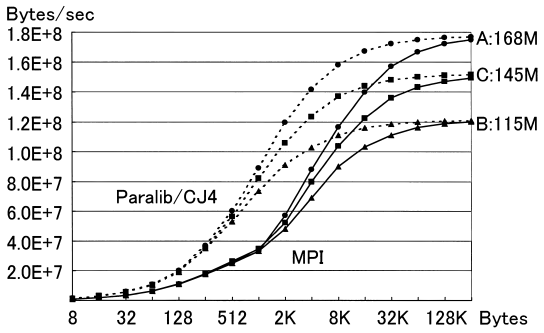


図 7 1 対 1 通信のスループット
Fig. 7 1 to 1 communication throughput.

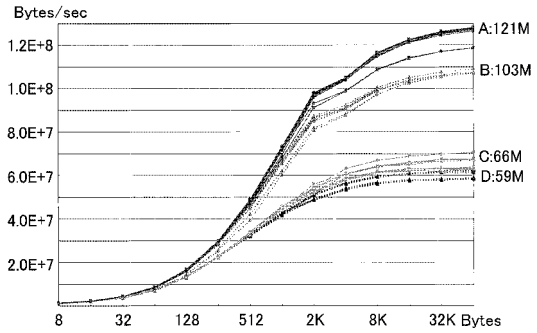


図 8 奇数ワード境界データ転送のスループット
Fig. 8 Odd word aligned data transfer throughput.

B と C を比べるとデータ読み出しの最適化を行うことにより，約 30 Mbytes/sec スループットが向上している．3 つの場合で，どちらのライブラリのグラフも同じところに収束している．これは，大きなデータを通信する場合には，MPI のプロトコルで使われる送信と受信の待合せのための短い通信の時間が無視できるようになり，さらに，MPI のデータ転送が Paralib/CJ4 を使って書かれているためである．

次に奇数ワード境界のデータ転送機能を評価するために，ソフトウェアで DW 境界にコピーしてから送った場合と，ハードウェアでデータの入換えを行った場合の Paralib/CJ4 でのスループットを，ソースアドレスとデスティネーションアドレスの差が 1 ワード, 3 ワード, $\cdots$, 31 ワードの場合で測定した（**図 8**）．

グラフの 4 つのグループは以下の場合である．

A：ハードウェアによるワードの入換えと，読み出しの最適化を行った場合．
B：ハードウェアによるワードの入換えを行い，読み出しの最適化を行わなかった場合．
C：ソフトウェアによる DW 境界へのデータのコピーと，読み出しの最適化を行った場合．
D：ソフトウェアによる DW 境界へのデータのコピーを行い，読み出しの最適化を行わなかった場合．

ハードウェアによるワードの入換えを行うことにより，ソフトウェアで DW 境界にコピーする場合に比べて約 2 倍のスループットが得られた．ソフトウェアでのコピーではデスティネーションアドレスと同じキャッシュライン境界にコピーすることで最適化できるが，ここではこの最適化を行っていない．

グループ A に入るべきグラフが A と B の中間にあるのは，アドレスの差が 1 ワードのときに読み出しの最適化を行えないためである．スループットは往復の時間の半分で測定しているので，この場合 1 ワード差の転送と 31 ワード差の転送が行われて，1 ワード差の方が読み出しの最適化が行われないので，グラフが A と B の中間に位置する．差が 1 ワードの場合に読み出しの最適化を行えないのは，ページ境界の先頭で違うページのデータを読み出してしまうため，ページ境界の先頭のラインではコヒーレント要求が必要となるからである．そのため，ページの先頭かどうかにかかわらず，1 ワード差の場合には読み出しの最適化を行っていない．

データ読み出しの最適化や奇数ワード境界データの転送でのデータの並べ換えは少しのハードウェア追加によって実現しているが，十分に大きな効果が得られた．

6.2 N 対 N 通信

ネットワークのマルチキャスト機能の効果を評価するため，N 対 N 通信（$N = 32$）のスループットを次の転送方法で測定した（**図 9**：Y 軸は 1 ノードあたりのスループット）．

A：シングルキャストを使って，各ノードが ノード 0, ノード 1, $\ldots$, ノード $N-1$ の順に送った場合．
B：シングルキャストを使って，ノード i がノード $i+1$, ノード $i+2, \ldots,$ ノード i の順に送った場合．
C：N 個のノード宛のマルチキャストを使った場合．

A の場合には，同じ宛先に通信が集中するためにスループットが低くなっている．

B の場合は，同じ宛先に通信が集中するのを避けるために，それぞれ異なる宛先に送るようにしている．B の場合にデータ量が多くなってスループットが落ちているのは，i 回目の通信と $i+1$ 回目の通信の間でネットワークの競合が起こるためと考える．

C の場合は，マルチキャスト機能を使うことで，最終段のリンクの利用率が 100%に近くなり，1 対 1 通

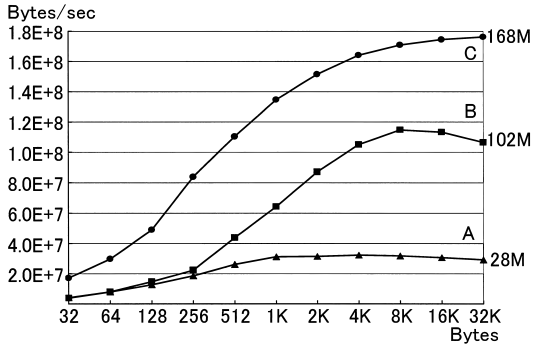


図 9 N 対 N 通信のスループット（$N = 32$）
Fig. 9 N to N throughput ($N = 32$).

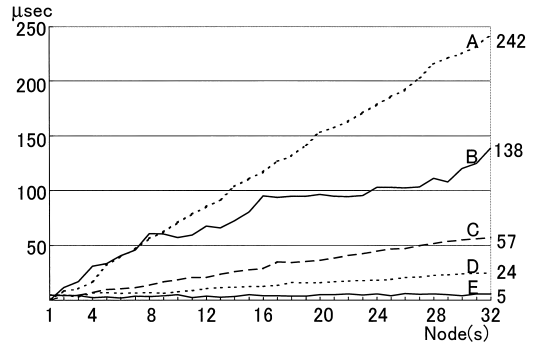


図 10 バリア同期のレイテンシ
Fig. 10 Barrier synchronization latency.

信のときと同じスループットを得られた．ネットワークでマルチキャストを実現するにはハードウェアが複雑になるが，N が大きい場合の N 対 N 通信では，シングルキャストでは実現できない性能を得ることができる．

6.3 バリア同期

ネットワークの同期機能の評価のため，N 個のノード（$N = 1, 2, \ldots, 32$）での，次のような方法によるバリア同期の実行時間を測定した（**図 10**）．

A：N 個のノードがシングルキャストで N 個ノードにパケットを送る．

B：2 分木構造によってバリア要求を集め，逆方向に同期完了を伝える．

C：N 個のノードがシングルキャストの “+1” の atomic add パケットを N 個のノードに送る．各ノードでは，atomic add の結果が N になれば，同期が完了したことになる．

D：N 個のノードが，N 個のノード宛のマルチキャストの “+1” の atomic add パケットを 1 個ずつ送る．

E：N 個のノードが，同期パケットを 1 つずつ送る．ネットワークで同期が行われ，各ノードに，同期パケットが 1 個到着すれば，同期が完了したことになる．

E の同期パケットを使った場合のレイテンシは N に関係なく約 5 μsec となった．これは，ネットワークでの同期パケットの処理が同期テーブルのアクセスの部分では逐次化されるが，その他の部分はポートごとに並列で動作するため，処理時間は同期パケットが通るスイッチの段数によって決まるからである．また，マルチキャスト機能と atomic add を使った D のレイテンシも N に対する増加量が少ない．Cenju-4 の同期パケットは 64 ノード以下のシステムでしか動作しないので，64 ノードを超えるシステムでは，マルチキャストと atomic add を組み合わせてバリア同期を実現している．

7. 関連する研究

ユーザレベル通信は，Meiko の CS-2[1] などの並列計算機から始まった．その後，SHRIMP[2] などの専用のネットワークインタフェースを PC に接続するシステムが現れる．最近では VIA（Virtual Interface Architecture）[3] の提案があり，並列計算機の並列プログラム用から始まった技術は，PC クラスタの一般のアプリケーション用に移ってきている．

CS-2 では，Elan という通信用コプロセッサがネットワークとのインタフェースとなっている．Elan は SPARC の Mbus に接続されており，Mbus のキャッシュコヒーレントアクセスを使って SPARC のキャッシュの操作を行う．論理アドレスから物理アドレスへの変換はメモリ上の Elan 用のページテーブルを使って行われる．ユーザレベル通信の保護機能は Elan が行っており，エラーや割込みなどを処理する．また，リモートからのアクセスに対する処理や通信プロトコルの処理も Elan が行う．コンテクスト番号によってマルチプロセスにも対応している．リモート DMA での他のプロセッサへの書き込みのレイテンシは 9 μsec が報告されているが，PVM のレイテンシは 78 μsec と非常に大きくなっている．これは Elan 自体のソフトウェアオーバヘッドが削減できていないためであると考える．Cenju-4 ではネットワークインタフェースは基本的な通信機能しかサポートしておらず，通信のプロトコルの部分は CPU が行っている．また，Cenju-4 ではマルチタスク機能としてメッセージ用バッファなどの受信側の資源もタスクごとに多重化している．

SHRIMP は PC に通信用のカードを差すことにより，並列計算機と同じ通信性能を実現することを目的

としている．データ転送はEISAバスを通して行われるが，メモリアクセスを監視するためにExpressバスと呼ばれる内部バスをプローブしている．ユーザレベル通信はVirtual Memory Mapという方法で実現している．ユーザがあらかじめシステムコールでローカルの物理ページとリモートの物理ページをマップしておき，ユーザがローカルの論理ページに書き込むことにより，プロセッサ間通信を行うものである．アドレスの論理から物理への変換はCPUのTLBで行う．この方法では，物理ページにマップされた宛先プロセッサが決まっているので，宛先プロセッサごとに送信バッファと受信バッファを設けることになる．そして，ソフトウェアによる送信データの送信バッファへのコピー，受信バッファから宛先アドレスへの受信データのコピーを行う必要がある．また，SHRIMPではデータの書き込みによってデータ転送を行うが，Cenju-4のremote readのようにリモートのデータを読み出す機能はない．

CrayのT3E[6]は共有メモリ型の並列計算機である．T3Eでは，DEC Alpha 21164のメモリインタフェースをEレジスタと呼ばれる機構により拡張し，他のプロセッサのメモリをキャッシュを通さずにアクセスできるようにしている．グローバルアドレスを導入することにより，宛先プロセッサとメモリアドレスが保護される．通信の最大スループットは約350 Mbytes/秒と非常に高い．また，ストライド転送もサポートしている．Cenju-4でもストライド転送をハードウェアでサポートすることを検討したが，キャッシュとの一貫性を保持するにはキャッシュライン単位でのデータの読み出しとなり，性能が出せないのでサポートしなかった．Eレジスタの1回の通信の単位は最大8ワードと非常に小さく，データ転送のためのEレジスタへのコマンドの発行のオーバヘッドが大きいと考える．また，高いスループットを得るには多く（128個）のEレジスタを使用する必要がある．Cenju-4では1回の転送要求で転送できるデータ量を512 Kbyteとし，CPUのオーバヘッドを軽減して計算と通信のオーバラップを図っている．

FLASH[5]はMAGICという通信プロセッサによりメッセージ通信とDSMを両方サポートしている．しかしながら，MAGICにはアドレス変換の機構がなく，論理アドレスから物理アドレスへの変換はMAGICのソフトウェアによって行う．Cenju-4でも通信処理の柔軟性を確保するためにネットワークインタフェースを通信プロセッサで実現することも考えたが，命令供給のオーバヘッドや並列に行える処理が逐次化されるなどの弊害があると判断し，専用ハードウェアで実現した．

8. おわりに

本論文では，Cenju-4のユーザレベルメッセージ通信を司るネットワークインタフェースのアーキテクチャと実現方法について述べ，メッセージ通信の性能について報告した．

Cenju-4ではメッセージ通信とDSMを両方使うことが可能である．通信されるデータが事前に決定している場合やひと固まりの場合には，メッセージ通信が向いているが，通信されるデータが動的に決まる場合やデータが分散している場合には，DSMを使う方が有利と思われる．今後，これら2つの通信方法を組み合わすことにより，うまく並列処理が記述できたり，効率良く並列処理を行えるような問題を見つけ，Cenju-4のアーキテクチャの有効性を示していきたい．

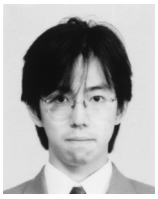
参考文献

1) Beecroft, J., et al.: Meiko CS-2 interconnect Elan-Elite design, *Parallel Computing*, Vol.20, pp.1627–1638 (1994).
2) Blumrich, M.A., et al.: Design Choice in the SHRIMP Syetem: An Empirical Study,*Proc. 25th Ann. Int'l. Symp. on Comp. Arch.*, pp.330–341 (1998).
3) Dunning, D., el al.: The Virtual Interface Architecture, *IEEE Micro*, Vol.18, No.2, pp.66–76 (1998).
4) 細見岳生ほか：並列計算機Cenju-4の分散共有メモリ機構，並列シンポジウムJSPP'99論文集, pp.15–22 (1999).
5) Kuskin, J., et al.: The Stanford FLASH Multiprocessor, *Proc. 21st Ann. Int'l. Symp. Comp. Arch.*, pp.302–313 (1994).
6) Scott, S.L.: Synchronization and Communication in the T3E Multiprocessor, *ASPLOS-VII*, pp.26–36 (1996).

(平成11年8月30日受付)
(平成12年3月 2 日採録)

加納 健（正会員）

1962年生．1989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NEC入社．以来Cenju-3，Cenju-4等の並列計算機の研究開発に従事．



中村　真章（正会員）

1969 年生．1993 年東京大学工学部電気工学科卒業．同年 NEC 入社．並列計算機アーキテクチャに関する研究に従事．

広瀬　哲也（正会員）

1967 年生．1992 年筑波大学大学院修士課程理工学研究科修了．同年 NEC 入社．並列処理アーキテクチャの研究に従事．

細見　岳生（正会員）

1969 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 NEC 入社．並列計算機アーキテクチャに関する研究に従事．

中田登志之（正会員）

1957 年生．1985 年京都大学大学院工学研究科情報工学専攻博士後期課程単位取得退学．同年 NEC 入社．工学博士．並列処理アーキテクチャ/ライブラリ/応用の研究に従事．