並列コンピュータ Cenju-3 のプロセッサ間通信方式とその評価

広瀬哲也[†] 細見岳生[†]
丸山勉[†] 加納健[†]

Cenju-3では，プロセッサ間通信におけるソフトウェア処理の軽減のため専用のネットワークインタフェースハードウェアを開発し，プロセッサ間通信の高速化を図った．この専用インタフェースは，キューイングされたデータをプロセッサとは独立に送出する機能，データを分割しパケットを自動生成する機能を持つ．また，パケットヘッダに割り込み制御フラグを設けることで受信側での割り込み処理のオーバヘッドを軽減した．この結果，ソフトウェア処理によるオーバヘッドを含めて，ユーザレベルで latency 33 μs throughput 34.7 MB/sec という高いプロセッサ間通信性能を実現した．本論文では，Cenju-3 のプロセッサ間通信方式の概要とその評価について詳細に述べる．

The Inter-Processor Communication of Cenju-3 and Its Evaluation

TETSUYA HIROSE,[†] TAKEO HOSOMI,[†] TSUTOMU MARUYAMA[†] and YASUSHI KANOH[†]

This paper describes an overview of the inter-processor communication of Cenju-3, and its evaluation results. We developed dedicated processor-network interface hardware for high speed inter-processor communication. This interface divide the queuing data into packets, and send to the network automatically, so that software overhead in data sending is very much reduced. Furthermore, to suppress needless interrupts, the Cenju-3 packets has flags to control interrupts. As a result, at user programming level with various software overhead, we get 34.7 MB/sec throughput and 33 μs latency.

1. はじめに

並列コンピュータでは，その性能を十分に発揮するには並列化によって生じる通信のオーバヘッドをいかに少なくできるかが重要なポイントとなる．通信性能としてネットワーク速度が注目されがちであるが，実際はソフトウェアによるオーバヘッドが支配的である．

Cenju-3では，プロセッサ間通信におけるソフトウェア処理の軽減のため，専用のネットワークインタフェースハードウェアを開発しプロセッサ間通信の高速化を図った．この専用インタフェースは，キューイングされたデータをプロセッサとは独立に送出する機能，データを分割しパケットを自動生成する機能を持つ．これによって送信側でのソフトウェアの処理は転送するデータ領域のキャッシュをライトバックし送信インタフェースを起動するだけで済む．また，パケットヘッダに割り込み制御フラグを設けた．これにより大量のデータ転送を行う場合に最後のパケットでのみ割り込みをかけることで，受信側のソフトウェア処理を大幅に軽減した．

このインタフェースはキャッシュを直接制御する機能は持たないが，データの受信とプロセッサによるキャッシュ制御をパイプライン化して実行することで，十分な性能が得られる．

以上によって，ハードウェアレベルで latency 7 μs, throughput 36 MB/sec という高い通信速度を実現した．またソフトウェア処理によるオーバヘッドを含めたユーザレベルでも，latency 33 μs, throughput 34.7 MB/sec（転送サイズ 8 KB 以上の場合）という高いプロセッサ間通信性能を実現した．

他の並列コンピュータでは，IBM の SP2 の場合 latency 39 μs throughput 35.5 MB/sec であると報告されている[1)☆]．

† NEC C&C 研究所
C&C Research Laboratories, NEC Corporation

☆ これらの値はユーザレベルでの測定値であり，throughput は“大きなメッセージ”で測定したと文献に記されている．

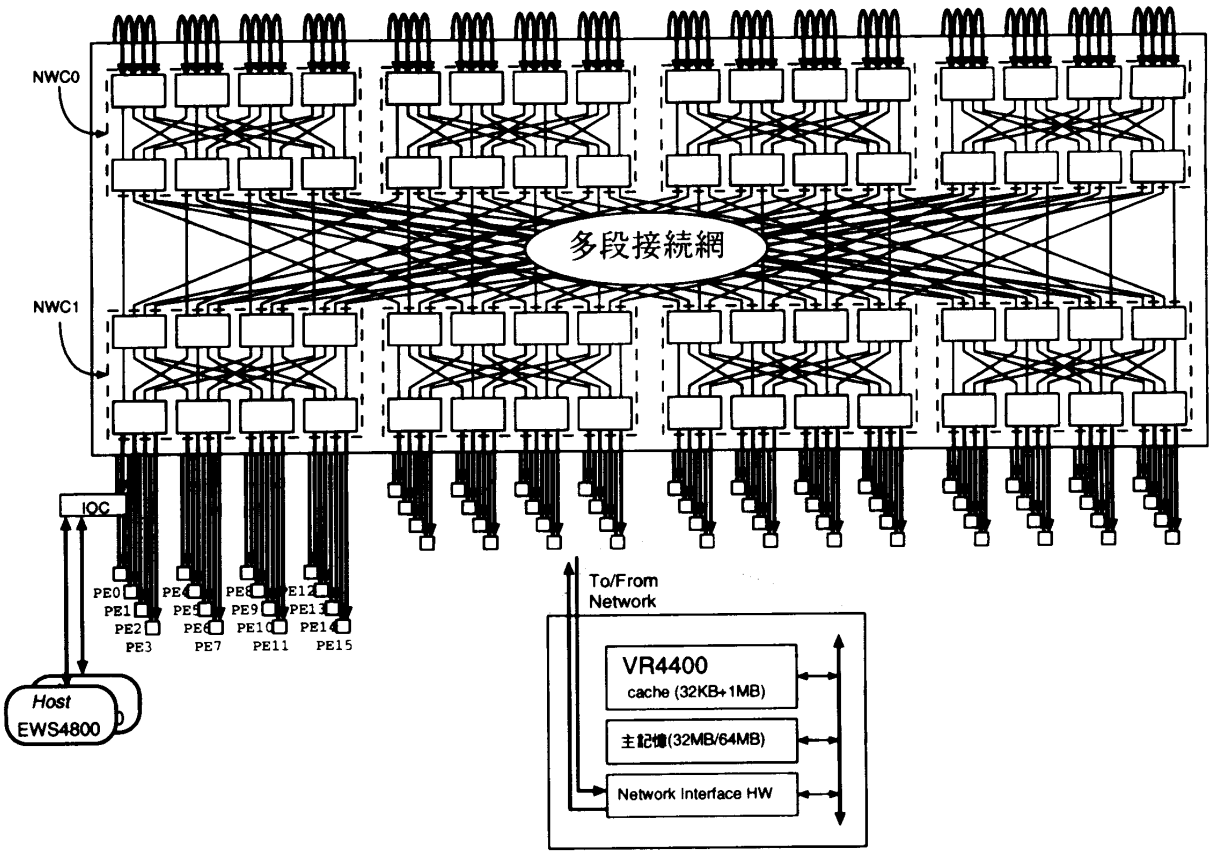

図 1 Cenju-3 (64PE) のハードウェア構成
Fig. 1 Hardware architecture of Cenju-3 (64PE).

本稿では，Cenju-3 のプロセッサ間通信性能についてハードウェアレベル，ユーザレベルでの詳細な評価結果を報告する．以下，2 章で Cenju-3 の概要，3 章でネットワークインタフェースについて述べる．4 章で Cenju-3 OS でのプロセッサ間通信のインプリメントを説明し，次いで 5 章で latency の評価，6 章で throughput の評価を行う．7 章では，より速いプロセッサ間通信を実現するための方針を示す．

2. Cenju-3 の概要[2)~3)]

Cenju-3 は要素プロセッサ（PE）に VR4400 を用い，最大 256PE 構成である．プロセッサ間ネットワークは 4×4 のクロスバスイッチ 4 段による多段接続網であり，40 MB/sec の転送速度を持つ．64PE システムのハードウェア構成を**図 1** に示す．

通信性能としてネットワークのハードウェア転送速度が強調されがちであるが，実際にはメモリとのインタフェース部分とソフトウェアでのオーバヘッドが通信のボトルネックとなっている（6.3 参照）．Cenju-3 では，PE 間通信を司る専用インタフェースハードウェアの開発により通信におけるオーバヘッドを軽減し高速なプロセッサ間通信を実現した．

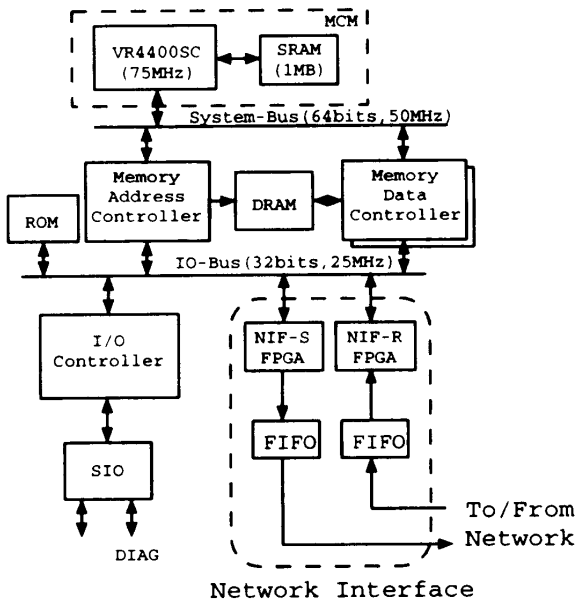

図 2 要素プロセッサのブロック図
Fig. 2 Block diagram of a processing element.

要素プロセッサのブロック図を，**図 2** に示す．専用ネットワークインタフェースは，IO-Bus（32 bit, 25 MHz）に接続される．

3. ネットワークインタフェース

現在の汎用マイクロプロセッサでは，演算処理に比べて割り込み処理には非常に時間がかかる．プロセッサ間で通信を行うためには多くの場合割り込みが必須であるが，このオーバヘッドがボトルネックとなり通信性能を損なっていた．

Cenju-3では，ネットワークのハードウェア性能を十分に活かすため，またパケット送受信時のプロセッサの負荷を軽減するために，プロセッサ間通信用の専用インタフェースハードウェアを開発した．

このインタフェースによって，通信のフロー制御を自動的に行うこと，パケットヘッダに割り込み制御フラグを設け割り込みを必要最小限におさえることを実現し，プロセッサの負荷を大幅に軽減した結果高い通信性能を達成した．

このインタフェースは，送信インタフェースと受信インタフェース（以下，Sender，Receiverと呼ぶ）で構成され，メッセージ通信とDMA通信をサポートする．

- メッセージ通信
  メッセージ通信では，受信側PEあらかじめ指定されたバッファ領域（Message Body Buffer）にデータが格納され，ソフトウェアによって処理される．このメッセージ受信バッファは2組用意されている．
- DMA通信
  DMA通信では，送信側PEで指定したデータ格納アドレスに直接データが書き込まれる．

3.1 送信インタフェースの機能

Cenju-3では，データのコピーを避けるため，宛先PE番号などの情報を持つヘッダに転送するデータ本体へのポインタを持たせ，ヘッダとデータ本体を分離した．送信インタフェース（Sender）はこのポインタをたどりデータ本体をネットワークに送出する．また，Senderは，送信データをパケットに分割してネットワークに送出する機能を持つ（**図3**）．ここで，パケットに分割される前の通信データの単位をParcel，実際にCenju-3ネットワークを通る通信データ単位を，単にPacketと呼ぶ．したがって，ソフトウェアで扱うデータの単位は送信側と受信側で異なる．すなわち，ソフトウェアは送信側ではParcel単位で，受信側ではPacket単位で処理を行う．

- 送信データのキューイングによる自動送信
  プロセッサは，最大255個までのParcel Headerをキューにつなぐことができる．一方，Senderはプロセッサの介在なしにHeaderの指定するデータ領域を送信する．したがって，プロセッサはネットワークの状態に関係なく最大で約64 MB（128 KB × 255個 × 2）のデータをSenderに登録することができ，それらは自動的に送信される．
- 送信データの自動分割
  プロセッサ間ネットワークでは，ハードウェアの制約から一般にパケットサイズは制限される．大量のデータを転送する場合，パケットサイズに分割し，各々にヘッダを付けて送る必要がある．Senderは，最大128 KBのParcelを約512 Bのパケットに自動的に分割して送信する機能を持つ．

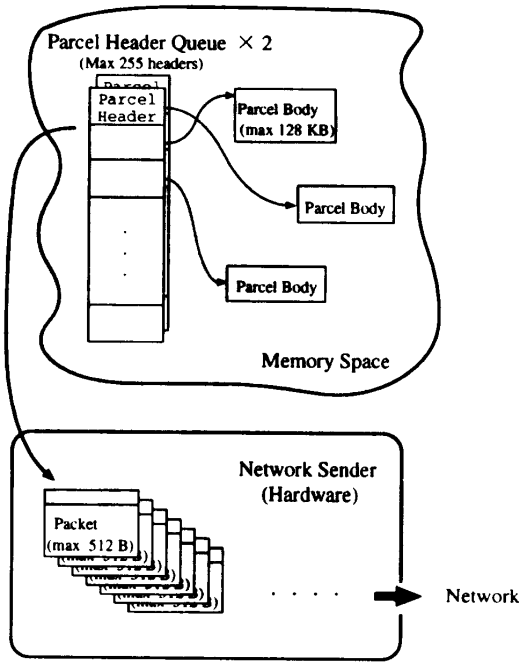

図3 送信インタフェースの機能
Fig. 3 Sender function.

3.2 受信インタフェースの機能

受信インタフェース（Receiver）は，受け取ったPacketのヘッダを解析し指定されたメモリ領域にデータを書き込む（**図4**）．また割り込みフラグに応じて以下のような割り込みを発生する．

- Packet到着割り込み
  この割り込みを指定されたPacketを受信した場合，データをメモリに格納する前に割り込みを発生する．
- Packet受信終了割り込み
  この割り込みを指定されたPacketを受信した場合，データを格納し終った時点で割り込みを発生する．Parcelが複数のPacketに分割されて送られる場合には，最後のPacketのみ割り込みフラ

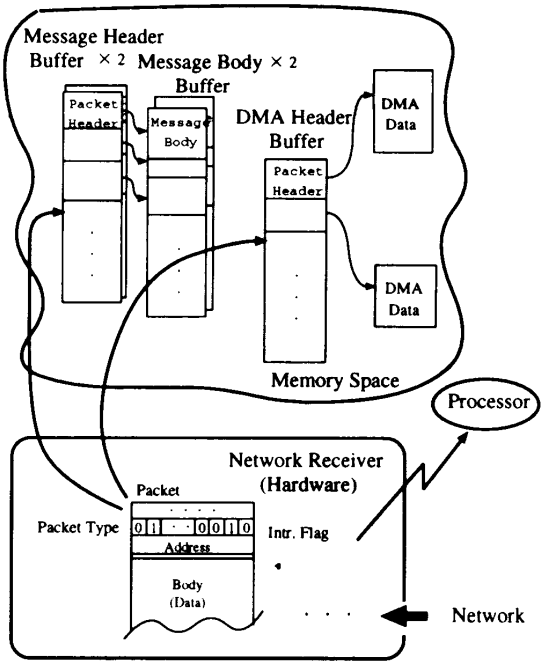

図4 受信インタフェースの機能
Fig. 4 Receiver function.

グが立っているので効率良く処理を行える.

3.3 ネットワークインタフェースの使用法

ソフトウェアから見たネットワークインタフェースの使い方について述べる.

- 送信
  送信側の処理は, メッセージ通信でも DMA 通信でも基本的に同じである. メッセージであるか DMA であるかは, Parcel header の type field の値による. カーネルが, 送信用 Parcel header queue に Parcel header を登録して, Sender の Header Queue Pointer をセットすると, Sender は queue が空になるまで Parcel を送出する.
- 受信
  Receiver は受け取った Packet の type を見て, Message あるいは DMA Header Buffer に, 受け取った Packet Header を格納する. 次に, メッセージであればあらかじめ指定されたバッファ領域にデータを格納する. Cenju-3 OS では, 基本的にメッセージ Packet には受信終了割り込みを立てておき, 受信したメッセージの処理をカーネルが行う. 一方, DMA であれば Receiver が header の転送先アドレスに直接データを書き込む. カーネルは, 最後の Packet 受信後の受信終了割り込みを受けて, 転送領域の cache invalidation を行う.

4. Cenju-3 OS におけるプロセッサ間通信

Cenju-3 OS 上のユーザプログラムでは, 並列処理記述用ライブラリ PARALIB/CJ を用いてプロセッサ間通信などを記述する. PARALIB/CJ では, プロセッサ間のデータ転送のほか, 同期, 排他制御等を実現するため C 言語および Fortran の関数を提供している. また, これとは別に MPI ライブラリも用意されていて MPI による通信も可能である.

4.1 CJrmwrite のインプリメント

本研究では, ユーザレベルでの通信性能評価として CJrmwrite 関数によるデータ転送の評価を行った. CJrmwrite は, 実行プロセッサのメモリ内容を指定プロセッサのメモリに書き込む機能を持つ, リモート DMA 関数である. ここでは評価に先だって, CJrmwrite のカーネル内でのインプリメントについて説明する.

CJrmwrite では, 速度向上のため転送量などに応じてメッセージと DMA を組み合わせて実際の転送を行う. 基本的にはキャッシュラインサイズ以下の転送の場合はメッセージによって, それ以上の場合は DMA によってデータ転送を行う. ただし, キャッシュライン境界の端数部分は, その部分をメッセージで送り, 残りを DMA で送る. また, Sender が送れる Parcel サイズは最大で 128 KB なので, それを越えるサイズの転送を行う場合は, 128 KB ごとに分割して DMA を行う.

- メッセージによるデータ転送手順 (**図 5**)
  メッセージ通信の場合, 転送データあらかじめ用意されたメッセージバッファにコピーされ転送される. この Parcel には, 受信終了割り込みフラグを立てておく.
  受信側では, メッセージ受信終了割り込みを受けると受信メッセージバッファを読んでメッセージタイプに応じて処理を行う. データ転送メッセージの場合は, データをメッセージバッファから転送先アドレスにコピーする.
- DMA によるデータ転送手順 (**図 6**)
  DMA によってデータ転送を行う場合は, DMA に先だって転送先メモリ領域のキャッシュコヒーレンシ処理 (invalidation) を行うようリクエストメッセージを送る. 受信側では, 転送領域の invalidation を終えるまで次のパケットを受け取らない.
  その後 Receiver を起動しデータがメモリに格納された後, 受信終了割り込みを受けると再び cache

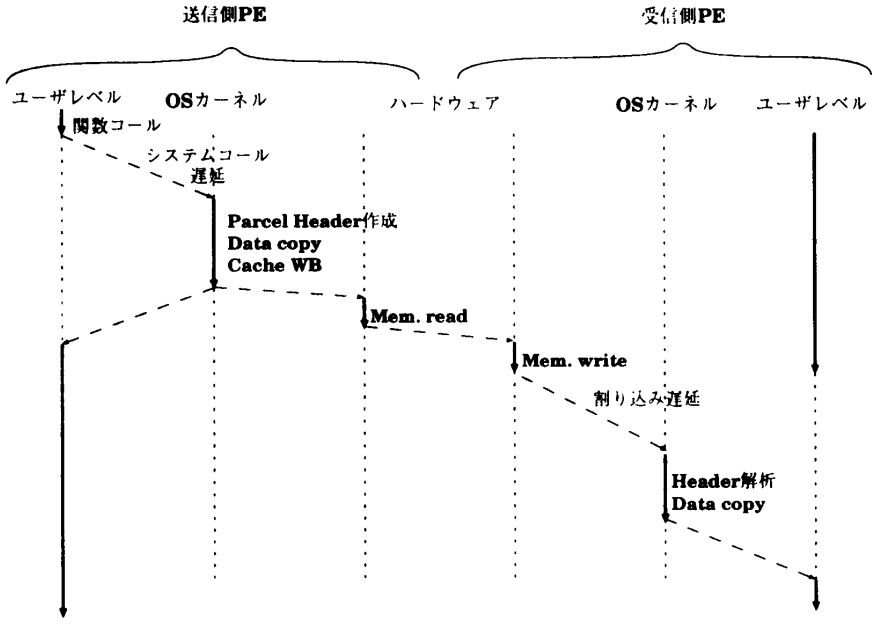

図5 Message 通信手順
Fig. 5 Message communication process.

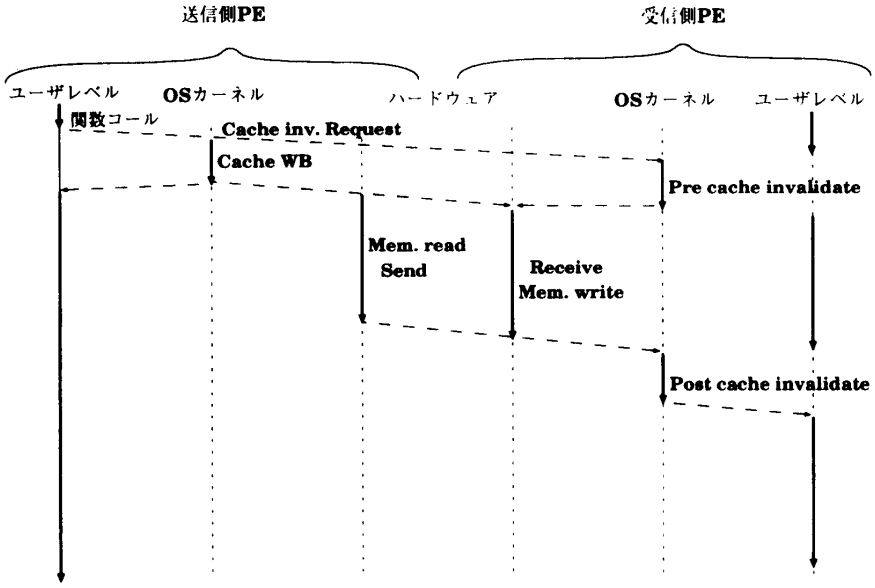

図6 DMA 通信手順
Fig. 6 DMA communication process.

invalidation を行う.

DMA 転送中でも CPU は処理を止めないので, キャッシュの置き換えが生じてキャッシュ上のデータがメモリへ書き戻され, DMA 転送されたデータを上書きしてしまう可能性がある. このような状況を避けるため, DMA 転送前あらかじめキャッシュを invalidate しておく必要がある.

また, DMA 転送中に CPU が転送領域を読むことがありうることを考慮して, DMA 転送後に再度 invalidation を行っている.

5. Latency の評価

5.1 ハードウェアの latency

ハードウェアの latency 評価として, 送信側で Sender を起動してから受信側でメモリ (Packet-HeaderBuffer) にデータが書かれるまでの時間を測定した (図 7). 結果は約 7 μs である.

この内訳は, Sender の Parcel Header Queue

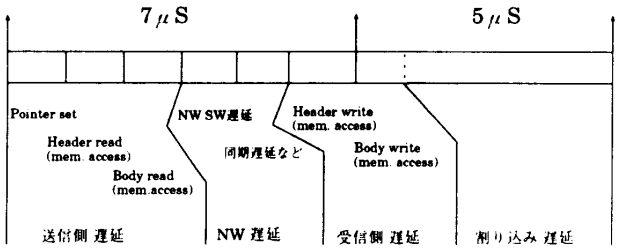

図7 ハードウェア latency
Fig. 7 Hardware latency.

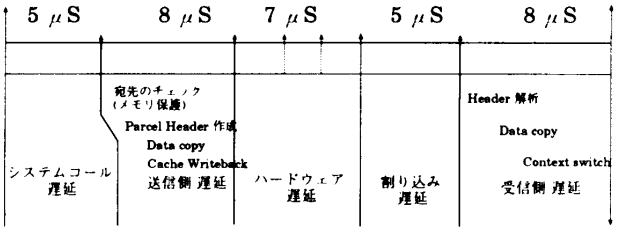

図8 CJrmwrite latency
Fig. 8 CJrmwrite latency.

Pointer のセット（外部アクセス，約 1 μs），Sender での Parcel Header の読み込み（メモリアクセス，約 1 μs），Parcel Body の読み込み（メモリアクセス，約 1 μs），ネットワークの同期遅延（約 1 μs)☆，ネットワークのスイッチング遅延（約 1 μs），受信側での Receiver による Packet Header の書き込み（メモリアクセス，約 1 μs）の処理時間の合計である．受信終了割り込みを立てた場合，カーネルで割り込みを認識するまでには，さらに約 5 μs の時間がかかる．

ここではネットワークでの遅延と比較して，CPU から Sender への外部アクセス，Sender によるメモリアクセス遅延が無視できない値となっている．

5.2 ユーザレベルの latency

ユーザプログラムで，CJrmwrite を用いて 2PE 間で 1 ワードのデータを往復してその時間を測定した．結果は約 65 μs，片道の latency は約 33 μs である（**図 8**）．

この内訳は，ユーザが CJrmwrite を実行しシステムコールでカーネルに制御が移るまで約 5 μs（逆算による），カーネルに制御が移ってから Sender を起動するまでの処理が約 8 μs（実測），Sender を起動してから Receiver による割り込みが入るまでが約 12 μs（実測），受信側でのカーネルによる処理が約 8 μs（実測）である．

今回の測定では簡単なプログラムで行ったのでこのような値が得られたが，キャッシュの当たり具合などによりシステムコールのオーバヘッドやカーネルの処理速度は変動する．そのため，ソフトウェアの処理時間の詳細な評価は難しい．

ハードウェアによるオーバヘッドは，バス速度やメモリアクセスのオーバヘッドによるもので現状では縮めることは困難である．システムコール（コンテクストスイッチ）のオーバヘッドは 5～8 μs 程度かかっていて，全体の latency に占める割合は無視できない．

6. Throughput の評価

6.1 ユーザレベルでの throughput

CJrmwrite によるユーザレベルの throughput 評価を行った．結果を**図 9** に示す．転送サイズ 8 KB 以上で 34.7 MB/sec の throughput が得られた．

Cenju-3 のネットワークは，16 bit × 20 MHz で単純計算では 40 MB/sec の throughput を持つ．しかし，パケットごとにパケットヘッダが付加されているのでその分を差し引くと 38.6 MB/sec である．さらにスイッチ間の同期遅延を考慮すると，ネットワーク自体の最大 throughput は 38 MB/sec 弱であると考えられる．送信側でのメモリ読み出しと受信側でのメモリ書き込みを含めた，実測によるハードウェア throughput は約 36 MB/sec であった（**図 10**）．

したがって，ユーザレベルにおいてもネットワークの性能にほぼ近い性能を実現しているといえる．

6.2 処理の高速化

Cenju-3 では，ソフトウェアによってキャッシュのコヒーレンシを保つ処理を行う．

送信側では，転送データ領域のキャッシュの write-back を行う．受信側では，DMA 前に転送データ領域のキャッシュの invalidation を行い，一連の DMA 転送が終った後の割り込みを受けて，再び転送データ領域のキャッシュの invalidation を行う．

DMA 通信を連続して行う場合，送信側ではキャッシュ処理（CPU）とデータ転送（Sender）をパイプライン化して同時に実行できる．一方，受信側での DMA 転送前の invalidation（Pre-invalidation）は，DMA 転送前に終了していることを保証しなければならないため，両者を同時に実行することはできない．図 9 の “Without Rec PE” のグラフは，受信側の PE を外して測定した値で，これと比較するとやはり受信側での処理がボトルネックとなっていることが分かる．

CJrmwrite では転送後の invalidation（Post-invalidation）をデータのメモリへの格納と並行に行うことで高速化を図った．Post-invalidation については，転送領域を分割して複数の DMA 転送に分けて送るこ

☆ Cenju-3 のネットワークは非同期で動作するため，同期遅延が生じる．

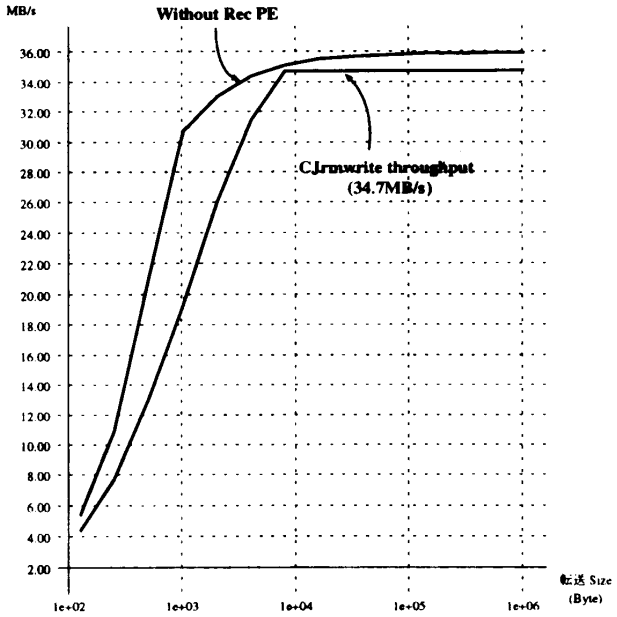

図 9 CJrmwrite の throughput
Fig. 9 CJrmwrite throughput.

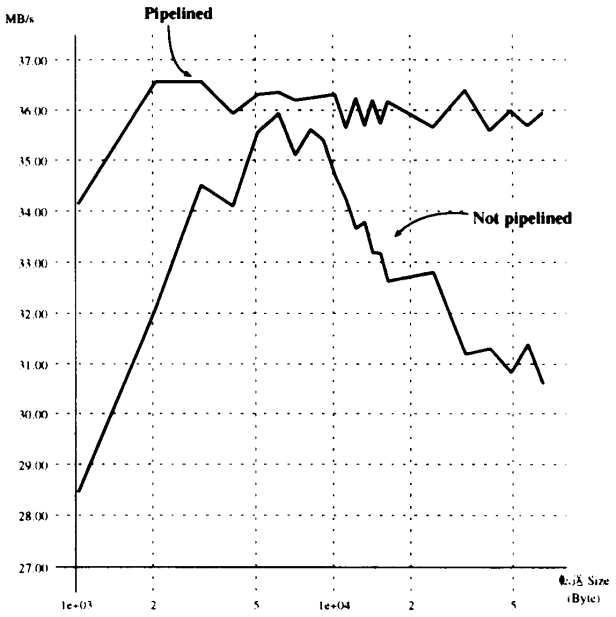

図 11 Post-invalidation のパイプライン化
Fig. 11 Pipelined post-invalidation.

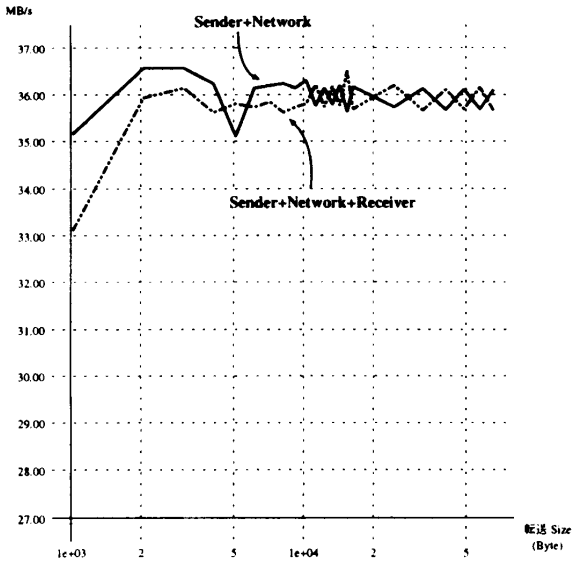

図 10 ハードウェアの throughput
Fig. 10 Hardware throughput.

とで，Packet 受信（メモリ書き込み）とその前の転送領域の invalidation をパイプライン化して処理を行える．

その結果，Pre-invalidation を行わない場合で，約 36 MB/sec の throughput が得られた（図 11 “Pipelined”）．この値は，ハードウェア自体の throughput（図 10）と一致する．Post-invalidation については，throughput を損なうことなく実行できる．

6.3 ボトルネックの解析

図 11 では，受信側での Packet 受信（メモリ書き込み）と，cache invalidation をパイプライン化して並行に行うことで，高い性能を実現した．ここでは，DMA 後の invalidation をパイプライン化せずに実行してボトルネックの解析を行う．

図 11 の “Not pipelined” のグラフが，実測値である．このグラフから分かるように転送サイズ 6〜8 KB でピークとなり，それ以上の転送サイズではだんだん減少して約 31 MB/sec に収束する．

先の図 10 の結果から，Receiver は Network に対して十分速い速度でメモリへのデータ格納を行っていることが分かる．一方，Network と Receiver の間には FIFO（バッファメモリ）がある（図 2）．したがって，Receiver が割り込みを上げて停止した時点で FIFO は空であり，cache invalidation の処理が終わり Receiver が再起動されるまでの間，Receiver は止まっているが Network は FIFO にデータを送り込むことができる．その後 Receiver は，FIFO にデータがあるうちは Receiver 自身の処理速度（バスとメモリの throughput に依存する）でデータを読み出すことができる．FIFO が空になったあとは Network の速度でデータを読み出すことになる．このように Network と Receiver の間に十分な量の FIFO を置くことで，invalidation の遅延を解消することができる．全体の throughput は，キャッシュ処理の時間と Network が FIFO をいっぱいにする時間がちょうど一致するサイズの場合に最大となる．

以上の考察をもとに，受信側での Packet の受信とメモリへの格納，cache invalidation までのシミュレーションを，Network 速度，Receiver 速度，カーネルに

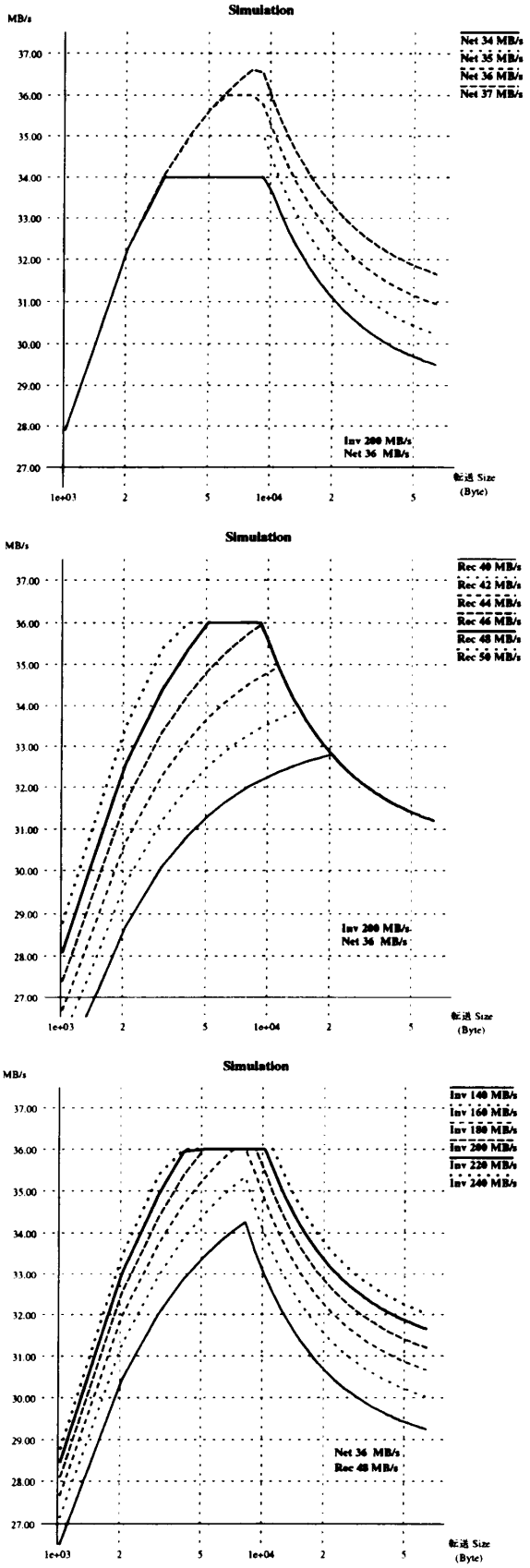

図 12 シミュレーション結果
Fig. 12 Simulation results.

よる cache invalidation 速度を変えて行った（**図 12**）．

この結果から，Network 速度は大量転送時の throughput に影響を与え，Receiver 速度（バスとメモリの throughput）は，サイズの小さな転送 throughput とピーク値を決める（ただし，Network 速度 < Receiver 速度の場合）．一方，cache invalidation 速度は，転送サイズによらず throughput に影響を与えていることが分かる．

図 12 より，Network 速度を 36 MB/sec，Receiver 速度を 48 MB/sec，受信側での cache invalidation 速度（DMA 後受け取った Packet Header を読んで転送された領域を確認しながら invalidation を行う）を 200 MB/sec としたとき，図 11 の実測結果とほぼ一致する．Network 速度の値は，図 10 の実測値と一致し，Receiver 速度も IObus のスピード（100 MB/sec）から妥当な値と考えられる．

次に，データのメモリへの格納が終わってから，割り込みを受け CPU が cache invalidation を始めるまでの，遅延時間をパラメータとしてシミュレーションを行った（**図 13**）．この遅延時間は，小さいサイズの転送 throughput を特に制限している．遅延時間は約 9 μs で実測値と一致し，5 章での latency の評価ともほぼ一致する．

以上で求めたパラメータを用いて，DMA 前の cache invalidation を含めた CJrmwrite のシミュレーションを行ったところ，実測値と一致した（**図 14**）．図 11 と同様 8 KB 付近にピークがあるが，これは受信側の FIFO の効果によるものである．なお，図 9 はユーザレベルのライブラリ関数において 8 KB ごとに区切っ

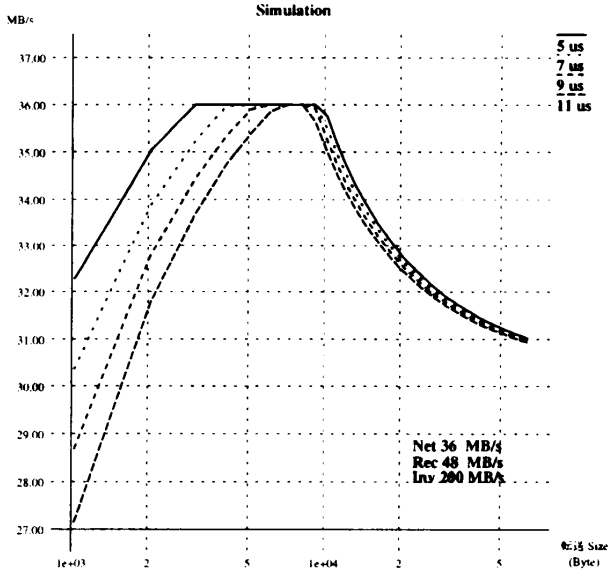

図 13 割り込み遅延の影響
Fig. 13 Interrupt delay simulation.

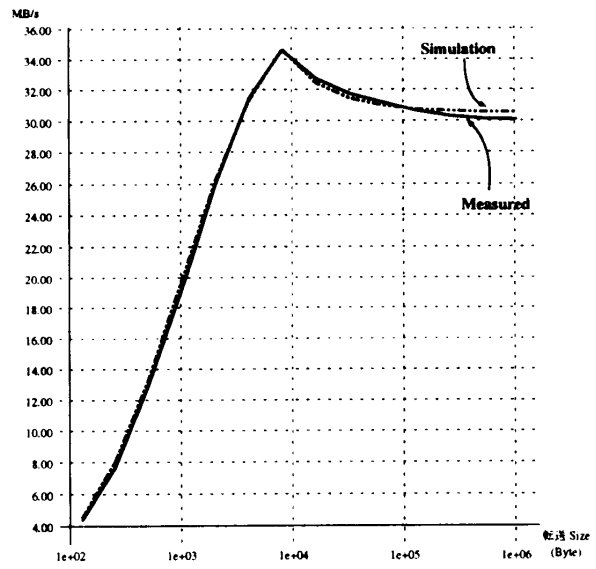

図 14 CJrmwrite—シミュレーションと実測結果
Fig. 14 CJrmwrite - Simulation & measured data.

て送った場合の結果である.

7. 今後の課題

7.1 latency 向上

図 8 から分かるように，現状では latency の大部分がソフトウェアによる遅延である．Cenju-3 では，システムの保護のための引数チェック，論理/物理アドレス変換，cache 処理を行うためシステムコールを用いている．このコンテクストスイッチのオーバヘッドは 5～8 μs 程度かかっていて，全体の latency に占める割合は無視できない.

外部からの cache 処理が可能なインタフェースを用い，ユーザが直接アクセスできるように保護機能を持ったネットワークインタフェースを実現すれば，この部分のオーバヘッドは解消することが可能である.

Cenju-3 OS での予備的な実験では，システムコールによるものに比べ，約 9 μs の latency 向上が確認された．また，Mach ベースの OS である Cenju-3/DE での実験では，システムコールによる場合で 27 μs であったところ，ユーザアクセスでは 11.5 μs の値が得られている[4]．Cenju-3/DE では，cache line 単位の転送のみを扱い，境界のチェックやデータのコピーなどを大幅に省いているため，CJrmwrite に比べて速い結果が得られている.

富士通の AP1000+でも，ユーザレベルで 5.1 + 0.040/B μs，システムコールで 8.6 + 0.040/B μs であると報告されている[5]．これは，Cenju-3 でのハードウェア latency より速いが，CPU や cache システムの違いによるものと考える.

このようにユーザレベル通信が有効であることがすでに示されているが，これらはまだ実験的な段階で実装上の制約が多い．今後は，メモリ/キャッシュ制御部分を含めて実装方法の検討を行い，latency の向上を図る必要がある.

7.2 Throughput 向上

6.3 節で示したように，throughput は Network 速度，Receiver 速度，cache 処理速度により制限される.

現状では，Cenju-3 ではバランスのとれた性能を実現している．現在のメモリアクセスは 128～160 MB/sec であるが，仮に 200 MB/sec の throughput を実現しようとすると，まずメモリがボトルネックとなる.

次に，ネットワークインタフェースが外部から cache 処理を行う場合は，ソフトウェアで直接処理するのに比べ処理が遅いという問題がある．したがって転送サイズによって，システムコール/割り込み遅延とのトレードオフとなる．すなわち，小さいサイズの転送の場合ネットワークインタフェースが直接 cache 処理をする方が速く，大きいサイズになると割り込みを上げてソフトウェアによってキャッシュ処理を行うほうが性能が高くなる．また，キャッシュ/メモリシステムで，外部からのリクエストを優先するか，CPU からのリクエストを優先するかのトレードオフも存在する．転送速度のみに注目するあまり，CPU の演算性能を制限したのでは，全体として高い性能を実現することはできない.

今後，throughput の向上を図るには，高速なネットワークを作ると同時に，高速のメモリを用い，CPU，キャッシュ，メモリ，ネットワーク間のアクセス遅延を極力少なくするようなバランスの良い設計を行う必要がある.

8. おわりに

本研究では，Cenju-3 のプロセッサ間通信の性能についてハードウェアレベル，ユーザレベルで詳細に評価した.

Cenju-3 では，プロセッサ間通信におけるソフトウェア処理の軽減のため専用のネットワークインタフェースハードウェアを開発しプロセッサ間通信の高速化を図った．この専用インタフェースによって，ハードウェアレベルでは，latency 7 μs，throughput 36 MB/sec という通信速度を実現した．また，ユーザレベルにおいても，latency 33 μs，throughput 34.7 MB/sec（転送サイズ 8 KB 以上の場合）という高いプロセッサ間通信性能を実現した.

この数値は，ネットワーク速度だけでなく，PE に

おけるバスやメモリ速度の制約からくるもので，現時点では Cenju-3 のネットワークとネットワークインタフェースは非常によくバランスしていることを実証した．より高い性能を実現するにはネットワーク自体の高速化とともに，PE のメモリ・キャッシュシステム，プロセッサとネットワークのインタフェース部分の高速化高機能化を図る必要がある．

謝辞 本研究を進めるにあたって大変お世話になった，C&C 研究所の中田氏，浅野，稲村両氏をはじめとする NEC 情報システムズの皆様に感謝いたします．

参 考 文 献

1) Snir, M., et al.: The Communication Software and Parallel Environment of the IBM SP2, *IBM Systems Journal*, Vol.34, No.2, pp.205–221 (1995).
2) Koike, N.: NEC Cenju-3: A Microprocessor-Based Parallel Computer, *Proc. IPPS*, Cancun Mexco, IEEE, pp.396–401, IEEE Computer Sciety Press (1994).
3) 広瀬，加納，丸山，浅野，稲村：並列コンピュータ Cenju-3 のアーキテクチャ，情報処理学会研究報告，Vol.94, No.66, pp.121–128 (1994).
4) Kanoh, Y., et al.: User Level Communication on Cenju-3 (1995). Hot Interconnects III において配布．
5) 白木，長武ほか：高並列計算機 AP1000+ のメッセージハンドリング機構，JSPP '95 論文集，Vol.95, No.2, pp.233–240 (1995).

(平成 7 年 9 月 1 日受付)
(平成 7 年 12 月 8 日採録)

広瀬 哲也（正会員）

1990 年筑波大学第 3 学郡情報学類卒業．1992 年同大学院修士課程理工学研究科修了．同年 NEC に入社．以来，遺伝的アルゴリズム，並列処理システムの研究に従事．

細見 岳生（正会員）

1992 年京都大学工学部情報工学科卒業．1994 年同大学院工学研究科修士課程情報工学専攻修了．同年 NEC に入社．以来，並列アルゴリズム，並列計算機システムの研究に従事．

丸山 勉（正会員）

1982 年東京大学工学部電子工学科卒業．1987 同大学院情報工学専門課程博士課程修了．同年 NEC に入社．以来，並列処理，分散処理のためのプログラミング言語および処理系，並列アルゴリズムの研究に従事．

加納 健（正会員）

1987 年京都大学工学部情報工学科卒業．1989 年同大学院工学研究科修士課程情報工学専攻修了．同年 NEC に入社．以来，並列計算機システム，並列処理の研究に従事．