並列応用を指向した分散システムの高速通信機構

伊達新哉[†]　杉野秀和[††]　鶴崎剛大[††]
増田峰義[†]　Damien Le Moal[†]　山添博史[†]
五島正裕[†]　森眞一郎[†]　富田眞治[†]

我々は分散環境において理論上同等の計算能力を持つ並列計算機に匹敵するシステムの実現を目指している。しかし従来のI/Oバスを利用した通信機構を用いる分散システムの場合、通信コストが高すぎるため、ネットワークで接続された計算機資源を有効利用する細粒度並列処理を行なう事は実質的にはほとんど不可能である。そのため我々は高速・低遅延な通信機構として、システムバスを利用したハードウェアによるキャッシュコヒーレントな共有メモリを実現するネットワークインタフェースカードを開発した。従来のI/Oバスを利用した通信機構と比較して、リモートメモリアクセス時の通信コストを減少させることが可能である。

A High Speed Communication Mechanism of a Distributed System for Efficient Execution of Parallel Applications

Shin-ya Date,[†] Hidekazu Sugino,[††] Takehiro Tsurusaki,[††]
Mineyoshi Masuda,[†] Damien Le Moal,[†] Hiroshi Yamazoe,[†]
Masahiro Goshima,[†] Shin-ichiro Mori[†] and Shinji Tomita[†]

We develop a distributed system which can execute parallel applications, but current distributed systems which have an I/O bus based communication mechanism have much communication costs, they cannot support fine-grain parallel processing. So we made a high speed and low latency network interface card that supports hardware cache coherent shared memory communication mechanism. The mechanism can reduce communication costs on the current I/O bus based models.

[†] 京都大学大学院情報学研究科通信情報システム専攻
Division of Communications and Computer Engineering, Graduate School of Informatics, Kyoto Univ.
[††] 京都大学工学部情報学科
The department of Information science, in Faculty of Engineering, Kyoto Univ.

1. はじめに

研究室やオフィスなどのある程度計算機資源がある状況においては、計算機環境として各端末をネットワークで結合した分散環境が一般的である。通常利用に関しては現在のこの環境は快適であるが、全体としての計算機資源は最大限有効に利用されているとは言い難い。

これは現在の分散システムのノードがもともと単体での利用を想定して設計されたものであるため、通信の機能が付加的であり、並列処理を行なうには通信コストが高くなってしまうためである。

このような通信コストは、通信時の資源保護のためのシステムコールや、煩雑な通信プロトコルによるソフトウェアオーバヘッドが大きな割合を占める。

本稿では、並列応用を指向した分散システム「コンピュータ・コロニー」における、上記のソフトウェアオーバヘッドを減少させる通信機構について述べる。

以下、2章で我々が実現を目指しているシステム「コンピュータ・コロニー」の概要を示し、3章でそのためのNIによる高速通信機構について述べる。そして4章で現在開発している分散システムベースのプロトタイプ・ハードウェアへの実装を示し、5章で関連研究とのかかわりを述べる。最後6章においてまとめを行なう。

2. コンピュータ・コロニー

本章では、並列応用を指向した分散システム「コンピュータ・コロニー」の概要とその構成について述べる。

2.1 コンピュータ・コロニーの概要

我々は現在、並列応用を指向した分散システム「コンピュータ・コロニー」の実現を目指している。コロニー —— 群体とは、同種の生物個体が多数集まって共通の体を組織し、相互に連絡して生活するものをいう。すなわち、コンピュータ・コロニーとは、およそ「複数の独立したコンピュータからなり、全体として1つのコンピュータのように動作するシステム」を意味する。コンピュータ・コロニーの目標は、

(1) 現在の分散システムのように、中小規模のワークグループでも容易に導入と維持が可能で、
(2) 現在の分散システムのように、ユーザが個人で気軽に使用することができ、しかも、
(3) 同等の理論最大性能を持つ単一の並列計算機に匹敵する並列処理能力を持つ

計算機システムである。

このためコンピュータ・コロニーでは、従来のシステムをソフトウェア、ハードウェアの双方から再構成を行なう。

2.2 コンピュータ・コロニーの構成

コンピュータ・コロニーでは、従来の分散システムの持つスケーラビリティや低価格性を維持しつつ、通信コストの問題を解決するため、プロセッサ、メモリ、ネットワークインタフェース (NI) を基本単位とするモジュールによるハードウェア構成を採用する。

以下、このモジュールによるハードウェア構成と、コンピュータ・コロニーでサポートされる通信プリミティブについて詳述する。

2.2.1 モジュールによるハードウェア構成

図1は、モジュールによるコンピュータ・コロニーのハードウェア構成のイメージ図である。モジュール内では1個以上のプロセッサと、メモリ、ネットワークインタフェースがそれぞれ密接に結合している。各計算機は、1個以上のモジュールと周辺機器によって構成され、各モジュールがコンピュータ・コロニーにおける一つのノードになる。このモジュールによる構成は以下のようにしてスケーラビリティ、低価格性を実現する。

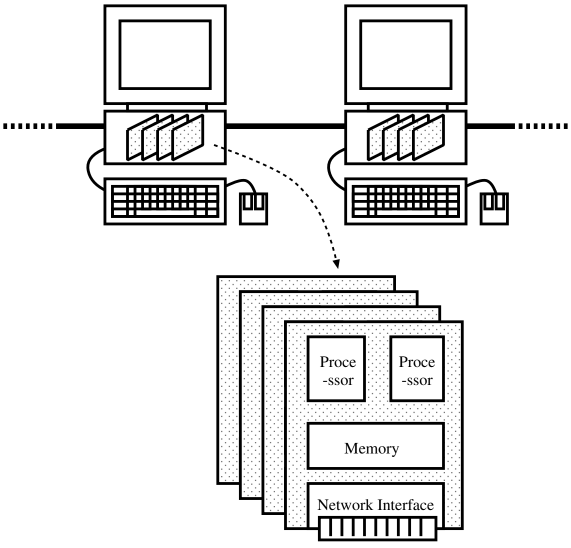

図1 モジュールによるハードウェア構成のイメージ

スケーラブルなシステム規模 システムの規模がモジュールの数によりスケーラブルに変更できる。すなわち通常のWS/PC程度の計算能力を持つものから、中規模サーバ、大規模並列計算機まで、従来のシステムではサポートできなかった広範なシステム規模に対して、基本的に同じアーキテクチャでの実現が可能である。

ネットワーク部分でのモジュール境界 従来にもモジュールとしてプロセッサモジュールやメモリモジュールがあった。しかし、これらのモジュールはモジュール境界がシステム内でクリティカルな要素である内部バスであり、新機種が発売される

とバスの規格の変更等で互換性が保持できないという問題があった。我々は、モジュール境界をシステム内でより制約の緩いネットワーク部分とすることでこの問題を解決でき、システムにおけるモジュールとしての有用性がさらに高まり、低価格性が実現できると考える。

3. 高速通信機構

本章では、コンピュータ・コロニーにおける通信機構と、その評価のためのプロトタイプ・ハードウェアについて述べる。

3.1 通信プリミティブ

モジュールによる構成をとるコンピュータ・コロニーにおいては、プロセッサ、メモリの配置は分散する。このような分散環境におけるプロセッサ間通信としては、ハードウェア・ソフトウェアによる共有メモリと、メッセージ通信があるが、これらの方式では、主に高速性と柔軟性がトレードオフの関係にあるといえる。

コンピュータ・コロニーにおいて多種多様なアプリケーションを実行させる場合、単一の通信プリミティブのみを採用すれば、効率の良い処理が行なえないアプリケーションも存在する。

そこでコンピュータ・コロニーでは、分散したこれらの要素間の通信プリミティブとして、ハードウェア共有メモリと高速メッセージ通信$^{1)}$の両方を採用する。

3.2 高速通信のための要件

分散環境で高速並列処理を行なうため、コンピュータ・コロニーでは従来の分散システムにおける通信コストを削減しなければならない。そのため、通信機構は

(1) システムコールの介在しない通信
(2) プロトコルオーバヘッドの小さい通信

を実現することが必要である。また、通信プリミティブとしてコンピュータ・コロニーではハードウェアによる分散共有メモリをサポートする。このため、

(3) リモートメモリのキャッシング

も行なえなければならない。

3.3 NI の機構

モジュール上の NI では、高速通信を実現するため以下のような機構を実現する。

3.3.1 ユーザレベル通信

システムコールを用いない通信としては、ユーザレベル通信という手法が提案されている。ユーザレベル通信では、高速ネットワークを利用する NI を用い、送信側ではユーザが直接 NI にアクセスし、また受信側では NI が直接ユーザのメモリ空間をアクセスすることにより、OS の介在なしに通信が行なえる。ただし、システムコールによる通信資源の保護を受けられないため、NI において適切なプロテクションを行なう必要がある。

3.3.2 システム規模にあった通信プロトコル

従来の汎用性はあるが煩雑な通信プロトコルではなく、システム規模にあったプロトコルを採用する。具体的には、従来ソフトウェアで行なわれていたフロー制御を NI 上で行なうことにより、プロトコルオーバヘッドの一部を削減する。

3.3.3 ハードウェアキャッシュコヒーレンス制御

共有メモリを利用した通信においては、リモートメモリのキャッシングを行なうことにより無駄な通信回数を減らすことができる。この方法はノード間の通信コストがノード内の通信に比べて非常に高くなる分散システムベースの構成では大変有効である。

しかし、無駄な通信は減少するが、当然ノード間のコヒーレンス制御のための通信が必要になる。このノード間のコヒーレンス制御をハードウェアで実現し、高速化を図る、

3.4 NI の共有メモリ管理機構

NI がサポートするハードウェア共有メモリでは以下のような機構を利用する。

仮想共有メモリ　物理的に共有メモリを持たない分散システムにおいて、共有メモリを実現する手法として、仮想共有メモリ (Shared Virtual Memory : SVM) がある。この手法では、OS の仮想記憶管理機構を利用することにより、通信資源の保護が実現される。実際の通信動作は、メモリアクセスに対しページフォールトを発生させ、通常のディスクアクセスを行なうのではなく、リモートメモリアクセスを行なう、というものである。

ネットワークアドレス　メモリアクセス時にシステムバス上を流れるアドレスは物理アドレスである。リモートメモリアクセスを行なう際にこの物理アドレスをそのまま利用するのはスワップ等が発生した場合の柔軟性に欠ける。よって仮想的なネットワークアドレスが必要となる。この場合、NI で物理アドレスとネットワークアドレスを変換する機構が必要になる。

キャッシングのための物理ページ　本通信機構では、メインメモリに以下の物理ページを用意し、リモートメモリのキャッシングを行なう。

- ホームページ
  物理メモリが実装されている物理アドレスに位置し、メモリ上にデータの実体が存在する。ホームページが存在するノードをホームノードと呼ぶ。
- ダミーページ
  物理メモリが実装されていない物理アドレスに位置し、直接のアクセスはできない。ダミーページに対するアクセスがあると、対応するホームページへリモートメモリアクセスを行なう。得られたデータは当該ノードのメモリにはキャッシングしない。

- コピーページ
  物理メモリが実装されている物理アドレスに位置し、ホームページの内容をキャッシングする。したがって、ホームページとの間で適切な一貫性制御を行なう必要がある。無効であるときにアクセスすると、ダミーページにアクセスした場合と同様、リモートメモリアクセスを行なう。

ノード間でメモリの共有を行なう場合は、この3つのページを利用する。基本的には、ページテーブルにおいて、共有する仮想ページをダミーページにマッピングする。ページ全体に対して頻繁にアクセスされるような場合はコピーページにマッピングし、ノード内のメモリにキャッシングする。

ページとブロックを融合した共有の単位　ノード間でのデータの共有単位としては、ページとブロックが考えられる。SVMのように共有の単位をページ単位とし、データ転送・コヒーレンス制御を行なうとfalse-sharingが発生してしまい、通信が頻発する。しかし共有の単位をキャッシュブロックの単位とすると、データの存在するノード・アドレスを管理するディレクトリの容量が爆発的に大きくなってしまう。

そこで、図2のように、この二つの方式を融合した方式を採用する。すなわちリモートメモリアクセスが発生した場合、アクセスした側のメモリ領域の確保はページ単位で行なうが、実際のデータ転送はブロック単位で行なう。

この方式により、ディレクトリの容量を抑えつつ、false-sharingによる無駄な通信の発生をなくすことができる。

3.5　NIにおける通信機構の実現

以下では、NIにおいて通信機構の要件を実現する方法を述べる。

3.5.1　ユーザレベル通信の実現

通信時にOSが関与せず、NIだけで通信が行なえるためには、NIが自律的に通信の必要性を関知し、通信時に必要な処理を行なわなければならない。本NIでは、以下のようにこの処理を実現する。

物理アドレスによる通信の起動　通信の必要性の関知は、NIがリモートメモリアクセスの必要なダミーページ・コピーページ内のブロックを示す物理アドレスを検出すれば、通信処理を開始する。物理アドレスの構成は、ページテーブルによって規定される。また通信資源のプロテクションは、仮想アドレスを物理アドレスにマッピングする仮想共有メモリ機構を利用することで達成されている。

ネットワークアドレス変換　ネットワークアドレスとしては、仮想空間ID + 仮想アドレスを用いる。

- 送信側
  NIが物理アドレスで逆引きページテーブルを引き、仮想空間ID + 仮想アドレスのネットワークアドレスに変換。
- 受信側
  NIがネットワークアドレスの仮想アドレスでその仮想空間IDのページテーブルを引き、物理アドレスに変換。

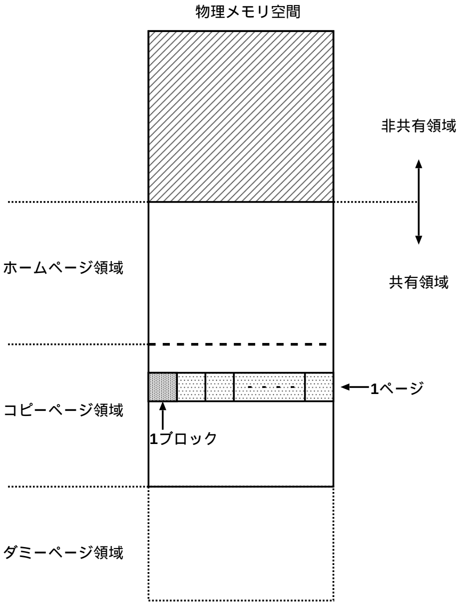

図2　ページとブロックを融合した共有の単位

通信先のノード特定　通信先のノードを特定するために機構が必要。NIでは、ホームノード、コピーノードの位置を特定するために用いる。

実際の処理としては、NIが仮想アドレスでノード情報の入ったディレクトリを引き、通信先のノードIDを得る。

以上の動作をまとめると、図3のようになる。

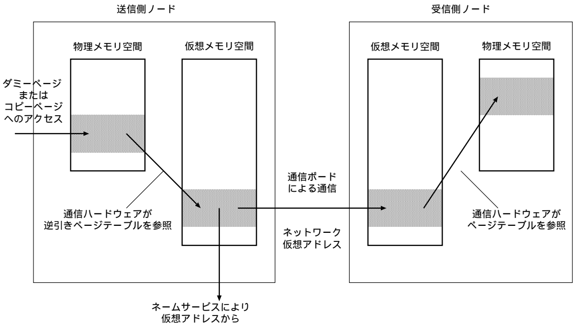

図3　アドレス変換

3.5.2　低レベルのプロトコルの実装

ハードウェアによるフロー制御　ハードウェアのみで高速なフロー制御を行なう。具体的には、送信側

では、送信データを送信完了の通知が来るまで再送バッファに保存する。受信側では、到着したフレームに対してハードウェアがアクノリッジを生成し、送信する。

アクノリッジのたたみこみ　本通信機構では、通常の共有メモリ計算機と異なり、ネットワークの物理層の上に共有メモリを実現している。そのため、例えばリモートリードリクエストのパケットが送出された場合、以下のようなパケットのやり取りが行なわれる。

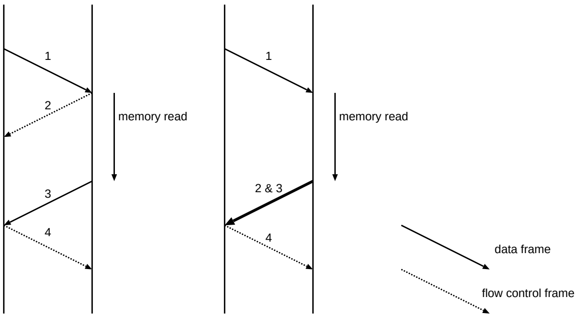

図 4　アクノリッジのたたみこみ

(1)　リードリクエスト
(2)　物理層のアクノリッジ
(3)　共有メモリのデータ
(4)　物理層のアクノリッジ

これら処理のうち、(2) の物理層のアクノリッジと (3) の共有メモリのアクノリッジをまとめて処理することが考えられる。まとめると以下のようなやり取りで済む。

(1)　リードリクエスト
(2)　物理層のアクノリッジと共有メモリのデータ
(3)　物理層のアクノリッジ

3.5.3　ハードウェアキャッシュコヒーレンス制御

ハードウェアによるノード間のキャッシュコヒーレンス制御は以下のものを利用する。

- システムバスのコヒーレンスプロトコル
- NI 上の主記憶タグ
  ブロック毎に Valid,Shared,Inhibit のビットを持つ。

NI は自ノード内のメモリに関して、以下のコヒーレンス操作を行なう。

自ノード内のメモリ無効化　NI が当該ブロックのアドレスに対して無効化要求を発行する。そのブロックがキャッシュに存在しても、メインメモリに存在しても一度の無効化要求で全て無効化が可能である。

自ノード内のメモリライトの認識　NI が当該アドレスに対するライトを関知すると、そのブロックの含まれるページを共有しているコピーノードへ無効化要求を送信する。

自ノード内の無効メモリリードの抑止　NI は無効なブロックに対するアクセスを関知すると、MIH をアサートしそのアクセスを中止させる。そのブロックの有効なデータを NI が責任を持って提供しなければならない。

4. プロトタイプ・ハードウェア

本章では、3 章で述べた高速通信機構を実現するプロトタイプ・ハードウェアについて述べる。

4.1　プロトタイプ・ハードウェアの設計方針

コンピュータ・コロニーは最終的にはモジュールをベースにしたハードウェアにより実現することを目指している。しかし、実際にモジュールによる構成を行なう前に、コンピュータ・コロニーのシステムとしての有効性の評価を行なう環境として、低コストで実現できる分散システムベースのプロトタイプ・ハードウェアを構成する。

本プロトタイプは、低い開発コストでコンピュータ・コロニーのハードウェアを評価する環境を実現するため、分散システムベースの設計方針をとる。すなわち、従来の分散システムのノードである WS/PC に新たに開発する NI を接続し、高速ネットワークでクラスタを構成するという方針をとる。以下、この方針で高速通信を行なうための NI を接続するバスについての考察を行ない、プロトタイプの構成について述べる。

4.2　NI を接続するバスについての考察

上記の高速通信のための要件を満たすため、モジュール上の NI の接続バスについて考察する。

4.2.1　I/O バス利用の問題点

従来の分散システムでは、NI は I/O バスに接続されるものであった。しかし従来通り I/O バスに NI を接続した場合、NI の構成により上記の要件 (1),(2) は実現できるものの、要件 (3) のリモートメモリのキャッシングを高速に行なうためのハードウェアによるコヒーレンス制御が行なえない。その理由は、I/O バスに接続されている NI からは、システムバスに接続されているプロセッサのキャッシュを直接操作できないためである。(図 5)

そのため I/O バスに NI を接続すると、共有メモリによる通信の際ソフトウェアによる処理が発生してしまい、効率の良いリモートメモリのキャッシングが行なえない。すなわち、プロセッサ、メモリ、NI が密に結合せす、コンピュータ・コロニーのモジュール構造の利点が失われる。

4.2.2　システムバスの利用

SMP(対称型マルチプロセッサ) をサポートしている WS/PC では、一つのノード内に複数のプロセッサが存在し、一般にノード内のプロセッサ間ではハード

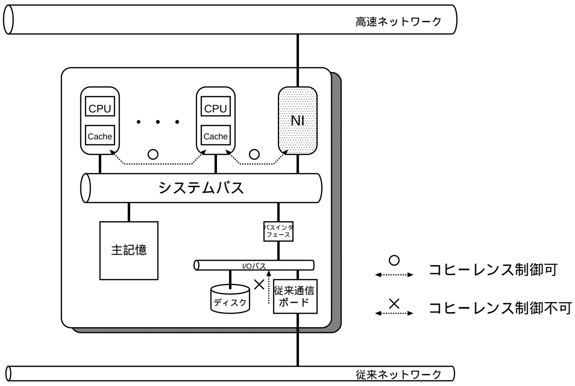

図 5　NI 接続バスとコヒーレンス制御

ウェアによるキャッシュコヒーレンス制御が行なわれる。そこで、コヒーレンス制御を行なうハードウェアとして NI をシステムバスに接続すれば、ノード内のプロセッサのキャッシュに対する操作を NI が行なえる。このため、NI を通して他ノードからのプロセッサのキャッシュに対する操作が行なえるようになり、ノード間のキャッシュコヒーレンス制御が可能になる。

よってモジュール内では図 5のようにシステムバスに NI を接続し、NI が SMP におけるコヒーレンスプロトコルを利用した構成を採用する。

またプロセッサ、メモリが接続されているシステムバスに NI を接続するこの構成により、プロトタイプ・ハードウェアがプロセッサ、メモリ、NI が密に結合したコンピュータ・コロニーのモジュールの構成として有効なものになる。

4.3　プロトタイプの構成

前節の設計方針、考察に基づき、我々が設計したコンピュータ・コロニーのプロトタイプ・ハードウェアの全体像を、図 6に示す。

以下では、設計したプロトタイプ・ハードウェアのノード構成とネットワーク構成について述べる。

4.3.1　システムバスを利用できるノード構成

本プロトタイプ・ハードウェアでは、ノードに Sun SPARCstation20 を利用し、そのシステムバスである MBus に新たに開発する NI を装備する構成をとる。

Sun SPARCstation20 を要素マシンとして採用した理由は以下の通り。

(1)　拡張用 **MBus** スロットを利用できる
Sun SPARCstation20 は、マザーボード上に 2 つの MBus スロットを持つ。1 つの MBus スロットには標準でプロセッサを搭載した MBus モジュールが装着されている。もう 1 つの拡張用 MBus スロットに NI を接続することにより、システムバスを利用した通信が行える。

(2)　**MBus** がキャッシュコヒーレンス制御用信号をサポートしている
Sun SPARCstation20 は、上記の拡張用 MBus スロットにプロセッサを搭載した MBus モジュールを装着することにより、SMP に拡張することができる。そのため、MBus ではキャッシュコヒーレンス制御のための信号をサポートしている。NI がこの信号を利用することにより、キャッシュに対しての完全な操作が可能になる。

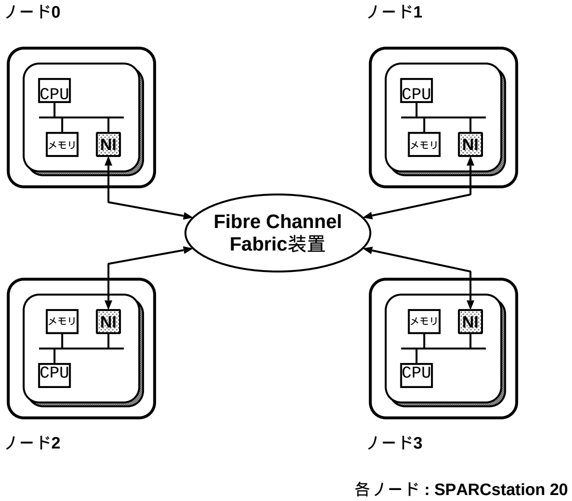

図 6　プロトタイプの構成

4.3.2　軽量プロトコルを実現するネットワーク構成

プロトタイプにおけるノード間ネットワークは、プロトコルオーバヘッドが小さい必要がある。本プロトタイプ・ハードウェアでは、高速シリアル通信である Fibre Channel を利用する。Fibre Channel を採用した理由は以下の通り。

(1)　物理層のみでの実装が容易である
低位から高位までプロトコルがレイヤ毎に規格化されているため、必要な部分のみの実装が可能である。

(2)　部品が比較的手に入りやすい
Gigabit-Ethernet と互換性のある部品が多いため、比較的安価で手に入りやすい。

4.4　NI の構成

NI は MBus,Fibre Channel インタフェースと各種バッファ、プロトコルプロセッサからなる (図 7)。

以下では、通信ボード上の各部分について述べる。

(1)　**MBus** インタフェース部
- **MBus** インタフェース・コントローラ
  MBus 上の物理アドレスを含む MBus トランザクションを常にスヌープし、MBus に対して適切な処理を行なう。
- 主記憶タグ
  自ノードの主記憶の各ブロックについて、共有/無効等の情報を保持する。主にリモートメモリキャッシング時のコヒーレンス制御に利用する。

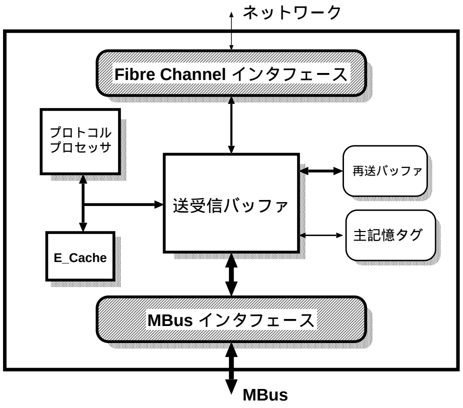

図 7　NI の構成

(2) 送受信用バッファ
MBus, ネットワーク用の送受信バッファ。

(3) プロトコルプロセッサ部
- プロトコルプロセッサ
  主にリモートメモリアクセス時の MBus 物理アドレス ↔ ネットワークアドレスの変換に用いる。そのほか、ハードウェアで実現困難な機能を実現する。
- **E_Cache**
  ボード上のプロトコルプロセッサ用の 2 次キャッシュ。自ノードのメインメモリの内容をキャッシングできる構成にし、プロトコルプロセッサが巨大なメモリ空間を利用できるようにする。

(4) 再送用バッファ
ハードウェアによるフロー制御を行なうため、送信したデータを保存しておくバッファ。

(5) **Fibre Channel 用 Chipset**
Fibre Channel のネットワークインタフェース。

5. 関連研究

I/O バスを利用した NI に関する関連研究としては、Memory Channel6) を利用した DEC の Alpha クラスタがある。この研究では、PCI バスに接続する NI によってページ単位の仮想共有メモリを実現する。NI 上の PCT (page control table) により、共有しているページに対する Memory Channel の動作を規定する。共有ページのキャッシュコヒーレンス制御はソフトウェアによって行なわれる。

また、RWCP の SMP クラスタ COMPaS8) では、PCI バスに接続する Myrinet7) ベースの NICAM という NI を利用する。ノード間の通信は Active Message を利用したメッセージ通信によって行なわれる。

システムバスを利用した関連研究としては、Wisconsin 大学 Madison 校の Typhoon9) がある。Typhoon では、NP(network interface processor) という NI を利用する。NP 内には TLB,RTLB (Reverse TLB) が装備され、アドレス変換が高速に行なわれる。しかし我々の研究のように、プロセッサ、メモリまで含めたモジュール構造までは言及されていない。

また、HAL Computer Systems, Inc. の HAL S1 System10) は、富士通で開発されている Synfinity interconnect を用いた共有メモリ、メッセージパッシングマシンである。主に intel 製プロセッサを 4 つ搭載したサーバ同士を接続する。ホスト側インタフェースは Slot 2 で、ネットワーク側は Synfinity である。NI としてタグ用の SRAM と MCU(Mesh Coherent Unit) を利用し、最大 4 ノードのクラスタ内でハードウェア共有メモリを実現する。クラスタ間のキャッシュコヒーレンス制御はソフトウェアによって行なわれる。

6. おわりに

本稿では、分散環境において低価格性、並列処理、透過性を実現するシステム、コンピュータ・コロニーを提案し、そのための高速通信機構と分散システムベースのプロトタイプにおける実現について述べた。

本通信機構のため、新たにシステムバスを利用した通信を行なうネットワークインタフェースを開発した。このネットワークインタフェースでは、従来の分散システムでは不可能であった細粒度の並列処理を可能にするため、ユーザレベル通信、プロトコルオーバヘッドの小さい通信、リモートメモリのキャッシングを行なう。それぞれの手法で従来の I/O バスを利用した場合に比べ、通信コストの削減が期待できる。

本通信機構で最もレイテンシが大きくなる部分は、ネットワークインタフェース上のプロトコルプロセッサである。このプロトコルプロセッサの処理のレイテンシをどれ程まで許容するか、バス、ネットワークのバンド幅とプロトタイプの通信性能の兼ね合いも含めた評価を行なう必要がある。

このプロトタイプにおける通信機構の評価を行なった後、今後実際にコンピュータ・コロニーのノードとなるモジュールの開発を行なっていく予定である。

謝辞

メンター・グラフィックス・ジャパン株式会社には、Higher Education Program の一環として製品とサービスをご提供いただいたので、ここに感謝する。

本研究の一部は文部省科学研究費補助金（基盤研究 (B)(2) 課題番号 10558045、および、(C) 課題番号 09680334）による。

参考文献

1) 五島 正裕 他 :Virtual Queue:超並列計算機向きメッセージ通信機構, 並列処理シンポジウム

JSPP'95 予稿集, pp.225-232.1995.

2) X3T11 Task Group of ANSI: Fibre Channel Physical and Signaling Interface (FC-PH), Rev. 4.3 June 1,(1994).

3) 山添 博史 他 :並列アプリケーションを指向した分散システム コンピュータ ·コロニーの構想, 情処研報 97-OS-76(SWoPP 阿蘇'97), pp.55-60.1997.

4) SPARC Technology Business: SuperSPARC & MultiCache-Controller User's Manual, 1994 .

5) Ben Catanzaro: SPARC マルチプロセッサアーキテクチャ. 1995.

6) Marco Fillo and Richard B. Gillet, " Architecture and Implementation of MEMORY CHANNEL2 ", *DIGITAL Technical Journal*(on line), 28 Aug. 1997.
(URL http://www.digital.com/info/DTJP03/DTJP03HM.HTM)

7) http://www.myri.com

8) 田中 良男 他 :COMPaS:Pentium Pro を用いた SMP クラスタとその評価, 並列処理シンポジウム JSPP'98 予稿集, pp.343-350.1998.

9) Mark D. Hill, James R. Larus, and David A. Wood, "Tempest: A Substrate for Portable Parallel Programs," Proc. of Joint Symp. on Parallel Processing, pp.123–128, May 1995.

10) Wolf-Dietrich Weber, et al., " The Synfinity Interconnect Architecture : A Cost-Effective Infrastructure for High - Performance Servers, "
(URL http://www.fjst.com/products/synfinitynuma/abstract/)