計算機アーキテクチャ 113－4
（１９９５．８．２３）

スケーラブル並列計算機プロトタイプ：お茶の水５号

対木 潤　田中 清史　松本 尚　平木 敬
Email:{tsuiki,tanaka,tm,hiraki}@is.s.u-tokyo.ac.jp

東京大学大学院理学系研究科情報科学専攻
〒113 東京都文京区本郷 7-3-1

既存の逐次計算機を置き換えるものとしての汎用超並列計算機への要求から、将来の汎用並列計算機にはスケーラビリティと汎用環境での使用を支援するための機構が求められる。本研究では、将来の汎用並列計算機の柔軟かつ強力なプロトタイプとしてハードウェアサポートされた同期機構を持つスケーラブルな並列計算機：お茶の水５号の設計、および実装を行なっている。本稿ではお茶の水５号の分散共有メモリ、プロセッサベース同期機構、メモリベース同期機構について述べる。

Scalable Parallel Processing System Prototype: OCHANOMIZ 5

Jun Tsuiki　Kiyofumi Tanaka　Takashi Matsumoto　Kei Hiraki

Department of Information Science, Faculty of Science, the University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113, Japan.

Mechanisms for supporting efficient use under general environment and Scalability are necessary for future general purpose parallel processing systems, so that they can be exchangeable for existing sequential processing systems. We designed OCHANOMIZ 5: scalable parallel processing system with hardware-supported synchronization mechanisms, as a flexible and powerful prototype of future general purpose parallel processing systems. In this paper, we describe distributed shared memory, processor-based synchronization mechanisms and memory-based synchronization mechanisms being implemented on OCHANOMIZ 5.

1 はじめに

超大型汎用計算機の性能向上が飽和の兆候をみせ始め、それにともなうパーフォーマンス向上にかかるコストの増大が顕著となってきた。逐次または並列度の低い密結合汎用計算機を置き換えるものとしての汎用超並列計算機に対する要求が高まりつつある。こうした背景から、汎用超並列計算機が備えるべき特質として (1) コストを低く抑えられること、(2) 既存の計算機に置き換え可能であることがあげられる。

コストを低く抑えるためにはスケーラブルな構成をとることが必須である（処理システムを同一の構成部品で構築可能）。また既存の計算機に置き換え可能なためにはマルチユーザ、マルチジョブの使用環境を提供する必要があり、そうした汎用環境下で実行効率を保つための機構をもつべきである。本研究では、将来の汎用並列計算機の柔軟かつ強力なプロトタイプとしてハードウェアサポートされた同期機構を持つスケーラブルな並列計算機：お茶の水５号の設計、および実装を行なっている。本稿ではお茶の水５号の分散共有メモリ、プロセッサベース同期機構、メモリベース同期機構について述べる。

2 設計方針

お茶の水５号 (**OCHANOMIZ-5**: Omnipotent Concurrency Handling Architecture with Novel OptiMIZers - 5) は、平木研究室における並列処理プロジェクト (お茶の水計画 [1, 2]) の第５号プロトタイプ計算機である。このお茶の水シリーズでは、実際にプロトタイプハードウェアおよびシステムソフトウェアの製作を通して、並列計算機アーキテクチャ、オペレーティングシステム、プログラミング言語、アルゴリズム、アプリケーションなどの高速化の研究を行っている。実機作製を行っているため、大規模な実用レベルのプログラムを用いた評価が可能であり、ソフトウェアシミュレータでは見逃されるような問題点の発見が期待できる。お茶の水５号はプロトタイプ計算機であり、規模は小規模（最大構成で８プロセッサ）であるが、将来の超並列計算機のプロトタイプマシンとして位置づけ可能なように、階層化構造を取り入れシステムのスケーラビリティを持ち、汎用性を保つために分散共有メモリシステムを採用している。以下、お茶の水５号の設計方針について説明する。

お茶の水５号は将来の汎用超並列計算機のプロトタイプマシンとして位置づけられ、精度の良い性能評価や新機構の評価を行うことを使用目的としている。このために、以下のような設計方針を採用した。

- ハードウェアおよびソフトウェアの両面にわたるアーキテクチャ研究のための多目的な測定ツールとして利用可能な構成を採る。
- FPGA (Field Programmable Gate Array) のコンフィグレーションを変更することにより、各種同期機構、各種通信プロトコル、各種コンシステンシプロトコル、各種ネットワーク演算機能等を実機上で実験評価することができる。
- 大規模システムに対応するためスケーラブルな構成を採る。各クラスタは木構造ネットワークで結合されている。ネットワークノードの段数を増やすことのみで各ノードに変更を加えずにシステム規模を大きくできる。

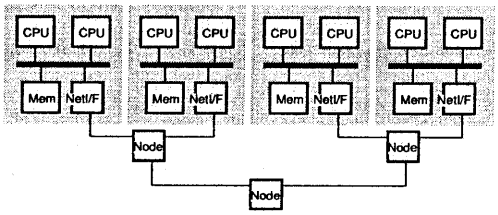

図 1: お茶の水５号の構成

3 お茶の水５号の構成

お茶の水５号は、２個の CPU と分散メモリが共有バスで結ばれたクラスタ４つと、それぞれのクラスタを結合するネットワークで構成される。（図 1） CPU にはマルチキャッシュコントローラ（MCC）つきの SuperSPARC を用いる。お茶の水５号の機能ユニットについて簡単に説明する。（図 2)

- MemoryController
  MBus の仕様をみたすようにラッピング（リクエステッドワードファースト）転送をサポートする。Memory-Based Signal 機構（6節）のワードへの書き込みがあったことを Arbiter に通知する。
- Arbiter
  MBus の調停、CPU への割り込み処理を行なう制御回路からなる。遠隔メモリアクセスを発行した CPU にリトライ信号を送ってバスを解放させ、リモートデータが到着した時に CPU にバス権限を渡す。Memory-Based Signal 機構（6節）のワードへの書き込みを CPU に割り込みで通知する。
- Network Interface
  階層化 Elastic Barrier、分散共有メモリをサポートするための、バスのスヌープ、パケット送受信、ブロックのキャッシュ制御を行なう。特定アドレスへのメモリアクセスをバリア同期命令とみなし、同期カウンタのインクリメント、デクリメントや、ネットワークへの信号伝達をおこない、必要なら CPU を停止させるために Arbiter に通知する。（5節）パケット通信によ

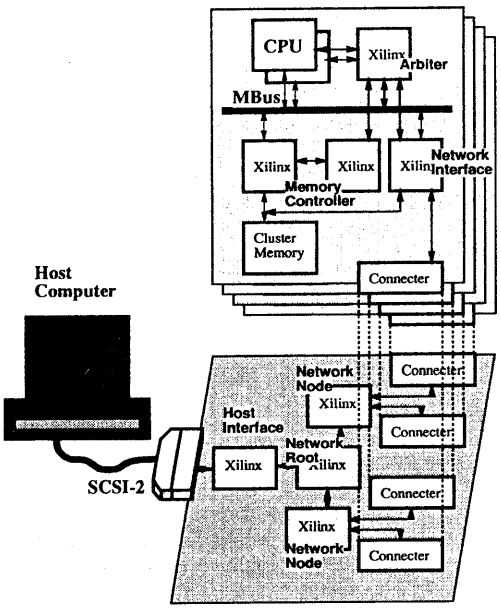

図 2: お茶の水５号の全体図

りネットワークへリモートメモリアクセスを発行する。(4.2節)

- Network Node,Root
  メッセージコンバイニング、ネットワーク上演算、分散共有メモリアクセスのためのパケット通信をサポートする [5]。マルチキャスト時はツリー状ネットワークのメッセージコンバイニングを利用して効率の良い Ack の回収を行なう。各ノードは 256Kbyte の SRAM を持っている。

4 分散共有メモリ

お茶の水５号における分散共有メモリの方針について述べる。分散共有メモリのリモートアクセスはメッセージパッシング&コンテクストスイッチで実現できるがオーバヘッドが大きい。お茶の水５号はハードウェアサポートされたキャッシュを持つ狭義の共有メモリをもつ。(リモートアクセス要求は、CPU に割り込みをかけソフトウェアで処理するのではなく、メモリシステムのハードウェアで処理する。)

分散共有メモリでは、クラスタ間参照(つまり通信または同期)コストを削減するため、リモートのデータのキャッシュを行うことが重要である。分散共有メモリのより容易な実現方法(広義と狭義の中間的存在)として最近よく用いられる方式に IVY[6] 流の仮想共有メモリが挙げられる。この方法は遠隔メモリアクセスをページフォールトで起動されるページ管理機構で処理する。このため、単純にソフトウェアによるキャッシュよりもオーバヘッドが大幅に少ない。しかし、この方法ではデータの転送単位が基本的にページ単位になり、効率が悪くネットワークへの負荷も大きい。お茶の水５号では、プロセッサ内蔵キャッシュのブロック単位でクラスタレベルのキャッシュの管理を行なう。これにより、リモートアクセスのパケットが大きくなり過ぎるのを防ぎ、無駄なネットワークトラフィックの発生を防止する。一方で、大きな配列などを共有するときは、メモリアクセスの転送単位が小さ過ぎるとパケットが多くなりかえってネットワークのトラフィックが大きくなる。このため転送データサイズは１ブロックと８ブロックの２種類用意する。

また、市販のプロセッサモジュールを使用する都合で、実現がクラスタ単位になるが、キャッシュインジェクション機能 [7] の実験的実装を行う(クラスタ間のキャッシュコンシステンシープロトコルにアップデイト方式よりも eager なプロトコルを使い、積極的に共有データをキャッシュすることができる)。

4.1 お茶の水５号での実装

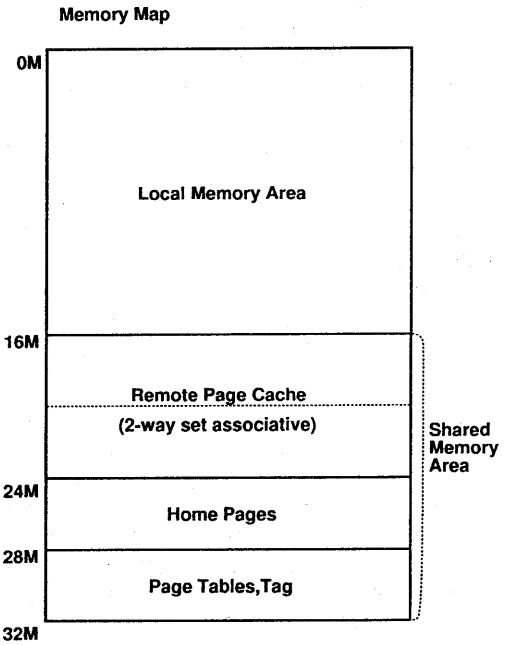

図 3: メモリマップ

各クラスタは 32Mbyte のメインメモリを内蔵し、そのうち 16Mbyte は完全にローカルに使用する(クラスタ間で共有しない)。残りの 16Mbyte は分散共有メモリとして利用し、自分がデフォルトのオーナーとなるホームページとし

て 4Mbyte、リモートページのキャッシュ (2-way set associative) に 8Mbyte、ページテーブルや様々なタグ情報のために残りの 4Mbyte を割り当てる。各クラスタのメモリマップは図 3のようになっている。

各クラスタはホームページのタグ情報として、32byte の各ブロックごとに状態 (Private, Shared, Invalid) 2bit、Owner 2bit、そして共有最大距離（ブロックを共有する最もはなれたクラスタまでの距離。同じクラスタなら 0、隣なら 1 という具合） 2bit を持つ。Owner、共有最大距離ともにプロセッサ台数が N の場合に対して 階層構造を持つクラスタ間ネットワークにより log N のサイズで表現できるので、非常に台数の多い構成でも実現可能でありスケーラビリティは確保されている。また、リモートページキャッシュのタグ情報として、クラスタアドレス 2bit、状態 (Private, Shared, Invalid) 2bit、リプレースメントのための参照ビットを持つ。

共有最大距離は一種のディレクトリ情報であり、疑似フルマップディレクトリ [8] においてエントリ数が多くなったときの方式である。インバリデーションパケットなどは共有最大距離内のノードへのマルチキャスト（共有最大距離分ツリーをさかのぼったノードをルートとするサブツリーへのブロードキャスト）として送られる。この方法は、Stanford 大学の DASH[9] が採用したフルマップディレクトリに比べて大幅にメモリ使用量が少なく、ネットワークのマルチキャスト機能を利用している点でチェインドディレクトリ方式より効率がよい。また、ネットワーク上でのコンバイニングによる acknowledge メッセージの収集 [8] を行なう。

お茶の水 5 号では、データのアクセスタイプに応じてリモートページキャッシュのプロトコル切替えをブロック単位に行なうことができる。各ページのプロトコルの情報はホームのクラスタ、各リモートページキャッシュのタグ部分に置かれる。

クラスタキャッシュの状態（図 4）はクラスタ間のキャッシュコンシステンシーを保つためのものであり、CPU 内蔵キャッシュや MCC の 2 次キャッシュの状態とは異なる。

- Private
  別のクラスタには該当ブロックが全くキャッシュされていない状態（クラスタキャッシュだけでなく別のクラスタの CPU 内蔵キャッシュや MCC の 2 次キャッシュもあわせて）である。クラスタの内部の MCC や CPU 内蔵キャッシュの状態は問わない。これは、後述のメモリトランザクション処理方式（4.2節）のため、片方の MCC が Invalidate を発行してクラスタが Private であるときにもう片方の MCC が Coherent Read を発行してもクラスタの状態が Private のままであるためである。ホームページの初期状態、および Invalidate を発行したクラスタの状態は Private である。

- Shared
  複数のクラスタと該当ブロックを共有している状態。クラスタの内部の MCC や CPU 内蔵キャッシュの状態は Invalid （もしくはエントリがない）または Shared

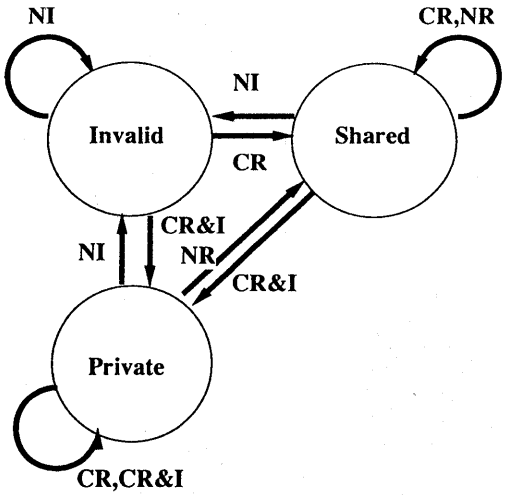

図 4: クラスタキャッシュの状態遷移

である。後述のメモリトランザクション処理方式（4.2節）のため、もともとのオウナーがどのクラスタであろうと Coherent Read 後はホームのクラスタの状態は Shared になる。したがって、Coherent Read を発行したクラスタ、ブロックのホーム、もともとブロックをキャッシュしていたクラスタの状態は Shared になる。

- Invalid
  クラスタキャッシュが該当ブロックをもっていない状態である。クラスタの内部の MCC や CPU 内蔵キャッシュの状態は Invalid である。リモートページキャッシュの初期状態、Invalidate をうけたクラスタの状態は Invalid である。

共有アドレスは、クラスタアドレス＋クラスタ内アドレス 22 ビットで表される。共有アドレスから物理アドレスへの変換は図 5のように上位数ビットの変換のみでおこなう。以下にアドレス変換の詳細を述べる。

1. 参照されるブロックのクラスタアドレスとクラスタ ID を比較して、参照されているブロックがホームページにあるかリモートページにあるかを調べる。

2. ホームページならばクラスタ内アドレスの前に 110 をつけたものが変換後のアドレスになる。

3. リモートページの場合、リモートページキャッシュのタグをクラスタ内アドレスで引く。キャッシュされて

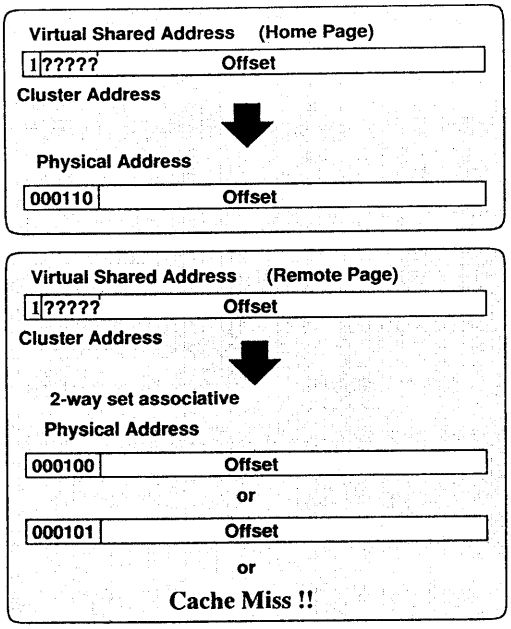

図 5: アドレス変換

いる 2 つのエントリのどちらかのクラスタアドレスが参照されるページのクラスタアドレスと一致しており、さらにそのエントリが Valid ならばクラスタ内アドレスの前に 100 または 101 (クラスタアドレスが一致したエントリにより決まる) をつけたものが変換後のアドレスになる。必要なら参照ビットを更新する。

4. 3 でどちらのエントリのクラスタアドレスも参照されるページのクラスタアドレスと一致しない場合、または一致したエントリが Invalid の場合はリモートページキャッシュミスである。参照ビットから置換すべきエントリをきめ、エントリを吐き出す。クラスタ内アドレスの前に 100 または 101 (吐き出したエントリにより決まる) をつけたものが変換後のアドレスになる。リモートクラスタにリクエストパケットを送り、受け取ったブロックをキャッシュする。

4.2 メモリトランザクション処理方式 (Invalidate 方式の場合)

リモートメモリアクセスにともなうクラスタ間のデータ転送はパケット通信によって行なう。MBus (SPARC の標準バス) 上の SuperSPARC のキャッシュのプロトコルはインバリデイト方式 (固定) であるので、お茶の水 5 号もクラスタ内ではインバリデイト方式を用いる。クラスタ間のプロトコルに関してはインバリデイト方式及びアップデイト方式を用いるが、ここではインバリデイト方式にした場合の例を述べる。

MBus 上のメモリトランザクション (Coherent Invalidate、Coherent Read、Coherent Read & Invalidate) は以下のように扱われる。

- 共通事項

  MCC つきの SuperSPARC の内蔵キャッシュはライトバック方式であるため、ライトが実行された時点では MBus 上では内蔵キャッシュ内のデータに書き込みがあったことを検出できない。このため、ブロック状態が Private なクラスタでは、CPU 内蔵キャッシュだけが正しい値をもっているという状況が頻繁におこる。したがってリモートクラスタからデータのリード要求があった場合には毎回 MBus 上に Coherent Read のトランザクションを出す必要がある。

  クラスタキャッシュの置換は MCC や CPU 内蔵キャッシュの状態によらずおこなわれる (状態が Private なら Coherent Read を発行してホームページにブロックの内容をパケットで送る)。また、書き込みの行なわれない CPU 内蔵キャッシュエントリの置換は MBus 上では検知できないので MCC や CPU 内蔵キャッシュの状態をクラスタキャッシュに反映させるにも限界がある。このため、リモートクラスタからブロックの Invalidate 要求を受けた場合、クラスタキャッシュにブロックがキャッシュされていなくても MBus 上に Coherent Invalidate のトランザクションを出す必要がある。

  リモートアクセスを必要とするメモリトランザクションが MBus 上に発行された場合、Arbiter が MBus 上に R&R(Relinquish and Retry) の信号を出す。これによりトランザクションは引っ込められ、バスは解放される。一連の処理が終るまでトランザクションを発行した MCC にはバスの使用権は割り当てないことにより余計なバストラフィックの増大を防ぐ。

  それぞれの処理でホームクラスタタグのブロックの pending フラグが立っていたらそのブロックは状態遷移中なので、そのブロックへのリクエストパケットは pending フラグがクリアされるまで FIFO キューに入れておく。

- Coherent Invalidate

  CPU 内蔵キャッシュまたは MCC (SuperSPARC のマルチキャッシュコントローラ) がキャッシュしてる exclusive でない (クラスタ内外の他のキャッシュと共有している) ブロックに CPU が書き込みを行なう際に発行される。

  参照先のブロックが他のクラスタと共有されていない場合 (ブロックの状態が Private) は、トランザクションはクラスタ内で処理される。(クラスタ内のトランザクションを発行していない方の MCC がブロックをインバリデイトする。)

ブロックが他のクラスタと共有されているときはトランザクションをR&Rで中断しInvalidate Requestパケットをホームのクラスタに送る。（参照されるブロックのホームがCoherent Invalidateを発行したクラスタである場合は、もちろんこのパケットは送られない。）ホームクラスタブロックのpendingフラグがクリアされているとき、またはキューの順番が回ってきたときに、ホームページタグの共有最大距離の範囲内のノードへInvalidate Requestパケットがマルチキャストされる。マルチキャストしてからAckの回収終了までブロックにpendingのフラグを立てておく。Ackのコンバイニングによる回収が終了したら、pendingのフラグをクリアし、ブロックの状態をInvalidにする。ブロックのオウナーをCoherent Invalidateを発行したクラスタに、共有最大距離を0に変更する。pendingのフラグをクリアし、Coherent Invalidateを発行したクラスタにAckをかえす。（参照されるブロックのホームがCoherent Invalidateを発行したクラスタである場合は、もちろんこのパケットは送られない。以降も同様。この場合ホームクラスタの状態がPrivateになる。）Coherent Invalidateを発行したクラスタはホームのクラスタからAckを受け取ると、ブロックの状態をPrivateに更新する。トランザクションを発行したMCCにバスの使用権をわたしてトランザクションを終了する。

- Coherent Read

  MCCのリードミス時に発行される。参照先のブロックがクラスタ内にキャッシュされている場合は、トランザクションはクラスタ内で処理される。（クラスタ内のトランザクションを発行していない方のMCCがブロックをキャッシュしていればデータを供給する。キャッシュしていなければメモリがデータを供給する。）

  ブロックがクラスタ内にキャッシュされていないときはトランザクションをR&Rで中断しRead Requestパケットをホームのクラスタに送る。

  ホームクラスタブロックのpendingフラグがクリアされているとき、またはキューの順番が回ってきたときに、ホームクラスタタグのブロックの状態を調べる。ブロックがInvalidでなければ共有最大距離を更新し、状態をSharedに変更してデータをリプライパケットで送る。トランザクション発行元のクラスタはリプライパケットを受け取ったら、ブロックの状態をSharedに変更してトランザクションを発行したMCCにバスの使用権をわたしてトランザクションを終了する。

  ホームクラスタのブロックの状態がInvalidのときはオウナークラスタにリクエストパケットをフォワードし、pendingのフラグをセットする。オウナークラスタではリクエストパケットを受けると状態をSharedに変更してデータをトランザクション発行元のクラスタとブロックのホームのクラスタにリプライパケットで送る。（参照されるブロックのホームがCoherent Readを発行したクラスタである場合はパケットは1つだけ送られる。）ホームクラスタはリプライパケットを受け取ったら、pendingのフラグをクリアし、共有最大距離を更新し、ブロックの状態をSharedに、オウナーを自分に変更する。トランザクション発行元のクラスタはリプライパケットを受け取ったら、ブロックの状態をSharedに変更してトランザクションを発行したMCCにバスの使用権をわたしてトランザクションを終了する。

- Coherent Read & Invalidate

  MCCのライトミス時に発行される。参照先のブロックが他のクラスタと共有されていない場合（ブロックの状態がPrivate）は、トランザクションはクラスタ内で処理される。（クラスタ内のトランザクションを発行していない方のMCCがブロックをキャッシュしていればデータを供給し、インバリデイトする。キャッシュしていなければメモリがデータを供給する。）

  ブロックが他のクラスタと共有されているときはトランザクションをR&Rで中断しRead & Invalidate Requestパケットをホームのクラスタに送る。

  ホームクラスタブロックのpendingフラグがクリアされているとき、またはキューの順番が回ってきたときに、ホームページタグのブロックの状態を調べる。ブロックがInvalidでなければホームページタグの共有最大距離の範囲内のノードへInvalidate Requestパケットがマルチキャストされる。マルチキャストしてからAckの回収終了までブロックにpendingのフラグを立てておく。Ackのコンバイニングによる回収が終了したら、pendingのフラグをクリアし、オウナーをセットし、共有最大距離を0に、ブロックの状態をInvalidに更新する。トランザクションを発行したクラスタにリプライパケットでデータを送る。（参照されるブロックのホームがトランザクションを発行したクラスタである場合は、もちろんこのパケットは送られない。また、ブロックの状態はPrivateにする。）トランザクション発行元のクラスタはリプライパケットを受け取ったら、ブロックの状態をPrivateにし、MCCにバスの使用権をわたしてデータを供給しトランザクションを終了する。

  ホームクラスタのブロックの状態がInvalidのときは（オウナークラスタはPrivate）オウナークラスタにリクエストパケットをフォワードし、pendingのフラグをセットする。オウナークラスタではリクエストパケットを受けると状態をInvalidに変更して、データをトランザクション発行元のクラスタにリプライパケットで送り、ホームクラスタにAckをかえす。ホームクラスタはAckを受けるとオウナーを変更し、pendingをクリアする。トランザクション発行元のクラスタはリプライパケットを受け取ったら、ブロックの状態をPrivateにし、MCCにバスの使用権をわたしてデータを供給しトランザクションを終了する。

5 プロセッサベース同期機構

お茶の水5号はプロセッサベースの同期機構として Elastic Barrier[10, 11] の階層化版である階層化 **Elastic Barrier**[12] を実装する。（図6） Elastic Barrier ではプロセッサごとに RREQ、APRV、PREQ の各同期命令（成立、承認、予告）のカウンタを設けることにより、同期期間に重なりのある面バリアの多重発行が可能である。超並

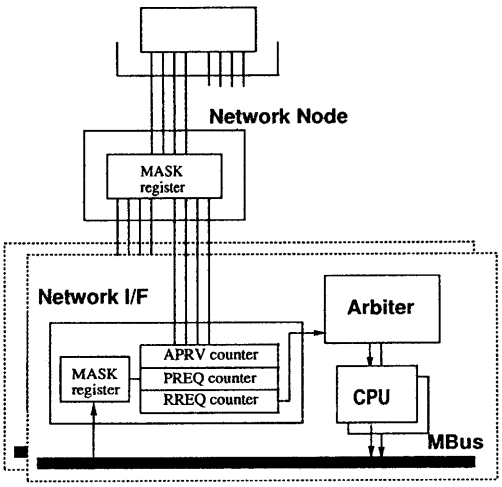

図 6: 階層化 Elastic Barrier

列計算機ではプロセッサ数が多いため、バリアの成立判定をバスや wired-AND 信号線で実現することは現実的ではない。このため、tree 状に combining しながらバリア成立判定を行う barrier combining tree をクラスタ間に用意する。お茶の水5号では汎用並列 OS によるクラスタ単位のパーティショニングをサポートしているため、この barrier combining tree もグループ分け可能である必要がある。

グループ分けは Network Node と Network I/F 内のマスクレジスタを用いて行なう。このマスクレジスタのセットは、特定アドレスへのライト命令によりおこなう。特定アドレスへのライト命令が発行されると Network I/F が特別なパケットを Network Node に送る。これをうけると Network Node はマスクレジスタの内容をスワップし、もとの内容をリプライパケットで返す。Network I/F はこれを受け取り、特定アドレスに書き戻す。

6 メモリベース同期機構

さまざまなメモリベース同期機構 [8] のなかで、お茶の水5号では市販のプロセッサモジュールを使用するという実装上の制約（プロセッサキャッシュ内に同期ビットをもてない）から Memory-Based Signal 機能をサポートする。

Memory-Based Signal

> イベント時に起動したいプログラムのコンテクストをクラスタ内のメモリ上で待機させ、オペレーティングシステムを介さずに起動できる軽いシグナル機構を提供する。該当ブロックにはコンテクストへのポインタ等が格納され、同期ビットがセットされているブロックに書き込み（ただし、データは変更しない）が起こると要素プロセッサへ割り込みが発生するように機構を構成する。

該当ブロックへ書き込みがあると Memory Controller は書き込みのあったアドレスを記録して Arbiter に通知する。Arbiter から CPU が割り込みをうけると、コンテキストを切替えるルーチンへトラップする。コンテクストポインタの書き込みと Memory-Based Signal のトリガとなるライト命令はアドレス上位ビットの 1/0 で区別する。Memory-Based Signal 用のブロックはシグナルが通知されるプロセッサと同じクラスタのホームページに置かれる。

7 おわりに

お茶の水5号の分散共有メモリ、プロセッサベース同期機構、メモリベース同期機構について述べた。本研究では、将来の並列計算機のアーキテクチャ研究のための柔軟かつ強力なプロトタイプとして、ハードウェアサポートされた同期機構をもつスケーラブルな並列計算機：お茶の水5号の設計、実装を行った。

- 今後の課題
  実機上でプログラムを動かし、性能評価を行なう。
  マルチジョブ、マルチユーザをサポートしたオペレーティングシステムを開発する。

謝辞

本研究は通商産業省 RWC プロジェクトの一環として行なわれた。なお、お茶の水5号の作成に当たり、協力していただいた日本 サンマイクロシステムズ 株式会社ならびにデジタルテクノロジー株式会社に深く感謝の意を表します。

参考文献

[1] 平木 敬, 松本 尚, 稲垣 達氏, 大津 金光, 戸塚 米太郎, 中里 学: 細粒度並列計算機お茶の水1号 — 基本構想 —. 第47回情報処理学会全国大会講演論文集 (6), pp.55–56 (October 1993).

[2] 戸塚 米太郎, 大津 金光, 中里 学, 秋葉 智弘, 松本 尚, 平木 敬: 汎用細粒度並列計算機: お茶の水1号 — 構成と性能評価 —. 並列処理シンポジウム JSPP '94 論文集, pp.73–80 (May 1994).

[3] 稲垣 達氏, 松本 尚, 平木 敬: 細粒度並列計算機用最適化コンパイラ: OP.1. 計算機プログラミング研究会報告 No.13-1, 情報処理学会, pp.1-7 (August 1993).

[4] 松本 尚, 古荘 進一, 平木 敬: 汎用並列オペレーティングシステム SSS-CORE の資源管理方式日本ソフトウェア科学会第 11 回大会論文集, pp.13-16 (October 1994).

[5] 田中清史, 対木 潤, 松本 尚, 平木 敬: 汎用並列計算機プロトタイプお茶の水 5 号の再構成可能高機能結合網. 信技報, CPSY 95 (August 1995).

[6] K. Li: IVY: A Shared Virtual Memory System for Parallel Computing. *Proc. 1988 Int. Conf. on Parallel Processing*, St. Charls, IL, pp.94-101 (August 1988).

[7] 松本 尚, 平木 敬: キャッシュインジェクションとメモリベース同期機構の高速化. 計算機アーキテクチャ研究会報告 No.101-15, 情報処理学会, pp.113-120 (August 1993).

[8] 松本 尚, 平木 敬: 超並列計算機上の共有メモリアーキテクチャ. 信技報, CPSY 92-26, pp.47-55 (August 1992).

[9] Lenoski, D., et al.: Design of Stanford DASH Multiprocessor. *Technical Report CSL-TR-89-403*, Stanford Univ. (Dec. 1989).

[10] 松本 尚: 細粒度並列実行支援機構. 計算機アーキテクチャ研究会報告 No.77-12, 情報処理学会, pp.91-98 (July 1989).

[11] 松本 尚: Elastic Barrier: 一般化されたバリア型同期機構. 情報処理学会論文誌 Vol.32 No.7, pp.886-896 (July 1991).

[12] 松本 尚, 平木 敬: 拡張された Snoopy Spin Wait と階層化された Elastic Barrier. 第 47 回情報処理学会全国大会講演論文集 (4), pp.43-44 (October 1993).