計算機アーキテクチャ 101－3
（１９９３．８．１９）

超並列計算機 RWC-1 における記憶構成

松岡 浩司† 岡本 一晃† 廣野 英雄† 横田 隆史*
堀 敦史† 児玉 祐悦‡ 佐藤 三久‡ 坂井 修一†

†(技組) 新情報処理開発機構 つくば研究センタ
*(技組) 新情報処理開発機構 超並列三菱研究室
‡電子技術総合研究所

超並列計算機における記憶階層の要件を整理し、超並列計算機 RWC-1 におけるメモリの大域的仮想化とデータの共有方式について述べる。RWC-1 では、大域仮想アドレスを相対 PE 番号と局所仮想アドレスに分割し、それぞれに変換機構を設けることによって、アドレス変換の柔軟・簡素化を図っている。また、データの局所化に関して、マルチスレッド環境におけるデータの共有がどうあるべきかを論じ、RWC-1 で採用を予定している Shared Virtual Memory(SVM) について述べる。

Memory System for the Massively Parallel Computer RWC-1

Hiroshi MATSUOKA† Kazuaki OKAMOTO† Hideo HIRONO† Takashi YOKOTA*
Atsushi HORI† Yuetsu KODAMA‡ Mitsuhisa SATO‡ Shuichi SAKAI†

†Tsukuba Research Center, Real World Computing Partnership
Tsukuba Mitsui Building 16F, 1-6-1 Takezono,
Tsukuba-shi, Ibaraki 305, Japan
*Massively Parallel Systems Mitsibishi Lab., Real World Computing Partnership
‡Electrotechnical Laboratory

This paper describes the memory system design principles of RWC-1, a massively parallel computer. At first, we discuss the requirements for the memory hierarchy of massively parallel computers. Then the global virtual addressing schemes on such machines are discussed. In RWC-1, we propose the simple address-translation method where each virtual address is expressed by a relative processing element number(RPE) and a local virtual address in a PE. Next, we discuss data sharing methods among PEs. In massively parallel computers, data should be localized for the latency reduction and network traffic reduction. We intensively discuss how data should be localized in the multi-thread execution. Refinement of Shared Virtual Memory(SVM) for data-localization/data-copying is proposed in this paper, which will be implemented in RWC-1.

1 はじめに

デバイスの高速化と直列実行の最適化を中心とする計算機システムの性能向上は限界に達しつつある。より高い性能を提供するためには、システムの並列化が必然的であり、近年では商用機の高並列化が始まった。こうした背景のもと、(技組)新情報処理開発機構(RWCP)では、中長期的な展望に立った超並列計算機の研究開発を行なっている。本研究開発の目的は、(1)リアルワールドコンピュティング(RWC)研究計画における新しい情報処理の実行母体となる計算機システムの開発、(2)汎用超並列計算機のアーキテクチャ技術およびソフトウェア技術の構築、の2点である。RWCPでは、最初に要素プロセッサ(PE)数1000規模のプロトタイプRWC-1を製作し、これを基礎として、さらに大規模なシステムの研究開発を行なう予定である。

RWC-1のような超並列計算機では、メモリシステムの基本的な構成法が機能・性能に決定的な影響を持つと考えられる。メモリシステム構築の観点から超並列計算機の研究開発を観ると、目下、2つのアプローチがある。1つは、DASH[1]やAlewife[2]などのようにコヒーレントキャッシュを擁するもの、いま1つは、EM-4[3]や*T[4]などのようにマルチスレッド実行に基本を置くものである。両者は、それぞれ、latency reduction、latency hidingを主たる目的として、発展したアーキテクチャと捉えることができる。RWC-1では、細粒度の並列処理を重視し、後者の流れに基礎を置き、必要に応じて前者の技術を融合し、さらに新しい実行モデル・要素技術を確立していく、というメモリシステム構築の方法論を採用した。

本稿では、まず、2節において、RWC-1の処理方式に関する紹介を行い、記憶階層の要件を整理する。次に、3節では、メモリの大域的仮想化について述べ、4節では、データの共有管理について述べる。データを局所化するためには、データを共有管理することが必要になるが、マルチスレッド環境におけるデータの共有とはどうあるべきかについて論じる。さらに、5節では、キャッシュについて述べ、最後に、6節で、現状の課題を述べ、今後のRWC-1の記憶階層の研究開発について触れる。

2 RWC-1の基本処理方式と記憶階層

2.1 RWC-1の基本処理方式

超並列計算機では、プロセッサ間通信における通信オーバヘッドの削減が最も重要な課題である。RWC-1では、この通信オーバヘッドを削減するために、通信と実行を融合させたRICA(Reduced Interprocessor Communication Architecture)を採用している[9]。

RICAは通信と演算をプロセッサアーキテクチャ上で融合させ、簡単化したものであり、(1)メッセージの送信・受信のコストの軽減、(2)RISC化による高速化、などを狙いとしている。RICAの概念図を図1に示す。到着したメッセージは結合網から直接RISCの実行部に注入され、オーバヘッド無しに処理が起動される。また、メッセージ生成は他の命令実行とは独立して処理され、高速化されている。さらに、局所的なアクセスに関しては、キャッシュにより、レイテンシィの短縮が図られている。その他、スーパスカラー形式の実行系が採用され、演算とメモリリード/ライトの同時実行が可能である。

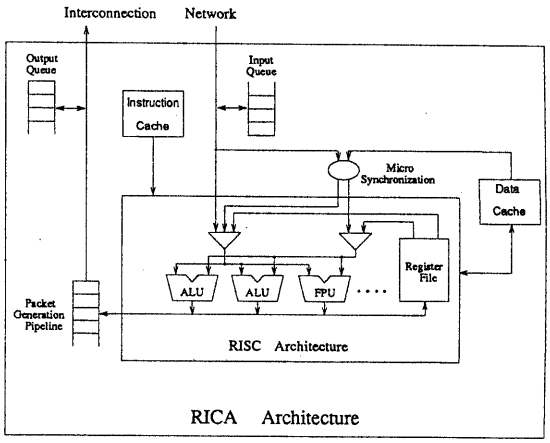

図1: RICA:Reduced Interprocessor Communication Architecture

2.2 記憶階層の要件

大域的仮想化

計算機のスタンドアロン化には仮想記憶の実現が不可欠である。超並列計算機では、特に、大域仮想メモリ(GVM: Global Virtual Memory)の提供がメモリアーキテクチャ上最大の課題である。

メモリの大域化、つまり、システム全体で一意のフラットな論理アドレスを与えることによって、以下のような点でソフトウェアの実行効率を改善できる。

- アドレスの輸出入が可能
  call by nameが可能となる。したがって、データ転送において、整列化/非整列化などの処理が不用となる。
- データ共有の管理が容易
  システムで一意のアドレスが与えられているので、同一のデータかどうかの判定が容易であり、データの共有管理などが簡素化される。

さらに、仮想化によって、以下のようなOSの実現に不可欠の機能を提供できる。

- 広い論理アドレス空間
  実アプリケーションでは広いメモリ空間が必要である。
- メモリ保護機能
  アドレス変換機構にメモリ保護の機能を組み込むことにより、不適切な操作に対するシステムの耐性が高まる。

データの局所化

多数のPEを接続しようとする超並列計算機では、必然的にネットワークの直径が大きくなる。そのため、接続するPE数が増加すると、プロセッサ性能の向上に対するネットワークの相対的な性能が低下する。この性能低下は、(1)データアクセスのレイテンシィの増加、(2)ネットワークの混雑によるポートあたりのスループットの低下、の2点に現れる。これらの性能低下を回避するためには、高性能の相互結合網を構築することが重要であると[8]、同時に、データを局所化する技術、局所化したデータを効率良くアクセスする技術が重要となる。

データの共有

並列計算機では、複数のPE間で単一のデータ(構造)を共有するのが一般的である。データ(構造)を共有する場合、レイテンシィの短縮のためにデータの(複数個の)コピーをPE群が持つことが考えられる。コピーを行なう場合の問題点として、データのコピー、更新、無効化などにともなう通信量の増加があり、データの共有管理が重要な課題となる。

キャッシュ

キャッシュを採用することによって、メモリアクセスの平均的なレイテンシィを短縮できる。キャッシュの問題点は、ミスヒット率が高くなった時の性能低下である。これは、大規模データを扱う際のスラッシングや、多数のリモートデータや多数のコンテクストのデータを入れた時のヒット率の低下が原因となる。

3 大域的仮想化

3.1 アドレス変換

大域的なメモリの仮想化を行なうためには、基本的に、大域仮想アドレス(GVA)から、物理PE番号(PPE)と局所物理アドレス(LPA)を求める変換が必要となる。一般的にこの変換はテーブル引き(およびハッシュなどの組み合わせ論理)を用いて行なわれるが、変換自体は以下のように分類できる。これらの変換の手順を図2に示す。

- 一括完全変換方式: 図2(a)
  GVAからPPEとLPAの組を一回の変換で求める。
  GVA → (PPE,LPA)
- 二段階完全変換方式: 図2(b)
  GVAから、まず、PPEとPE内に閉じた局所仮想アドレス(LVA)を求める。次に、LVAからLPAを求める。
  GVA → (PPE,LVA)
  LVA → LPA
- RPE変換方式: 図2(c)
  GVAがPEを示す部分(GVA.RPE)とPE内に閉じた局所仮想アドレス(GVA.LVA)に分割できる場合に適用可能である。ここで、RPEは相対PE番号(Relative PE number)である。RPE変換方式では、RPEからPPEを、LVAからLPAをそれぞれ求める。
  GVA.RPE → PPE
  GVA.LVA → LPA
  RPEからPPEの変換が固定的である時、メモリシステムは、従来の分散共有メモリとなる。

3.2 アドレス変換機構

各アドレス変換方式における、アドレス変換機構を表1にまとめる。

二段階完全変換方式およびRPE変換方式では、LVAで指定されるメモリ空間はPPEで指定されるPEで管理され、その変換は被アクセス側のPE内に閉じている。その他の変換はアクセス側で行なわれ、変換に要する情報はPE間で共有される。

3.3 方式比較

前述した各方式を比較すると、一括完全変換方式、二段階完全変換方式、RPE変換方式の順で、アドレス変換の自由度が大きい。反面、実装を考えると、前のものほど構造が複雑で、共有を管理しなければならない情報が多いという欠点を持つ。例えば、仮想記憶を導入する場合を考えると、一括完全変換方式では、一般にマッピングの更新の影響をシステム全体に反映させる必要があり、一貫性を維持管理するための通信が発生する。これに対して、二段階完全変換方式およびRPE変換方式では、マッピングの更新の影響がPE内に閉じていて、ページのマッピングが更新されても通信は発生しない。なお、二段階完全変換方式でも、第一の変換に要する情報はPE間で共有されるが、マッピングの更新がプロセスの生成などのタイミングで行なわれるため、一括完全変換方式と比較した場合、一貫性を維持管理するために必要とする通信の量は少ない。

変換方式	変換	場所	変換情報
一括完全変換方式	GVA → (PPE,LPA)	アクセス側	(分散)/ 大規模 / 頻繁
二段完全階変換方式	GVA → (PPE,LVA) LVA → LPA	アクセス側 被アクセス側	分散 / 大規模 / 稀 局所 / 小規模 / 頻繁
RPE 変換方式	GVA.RPE → PPE GVA.LVA → LPA	アクセス側 被アクセス側	分散 / 小規模 / 稀 局所 / 小規模 / 頻繁

表 1: アドレス変換機構

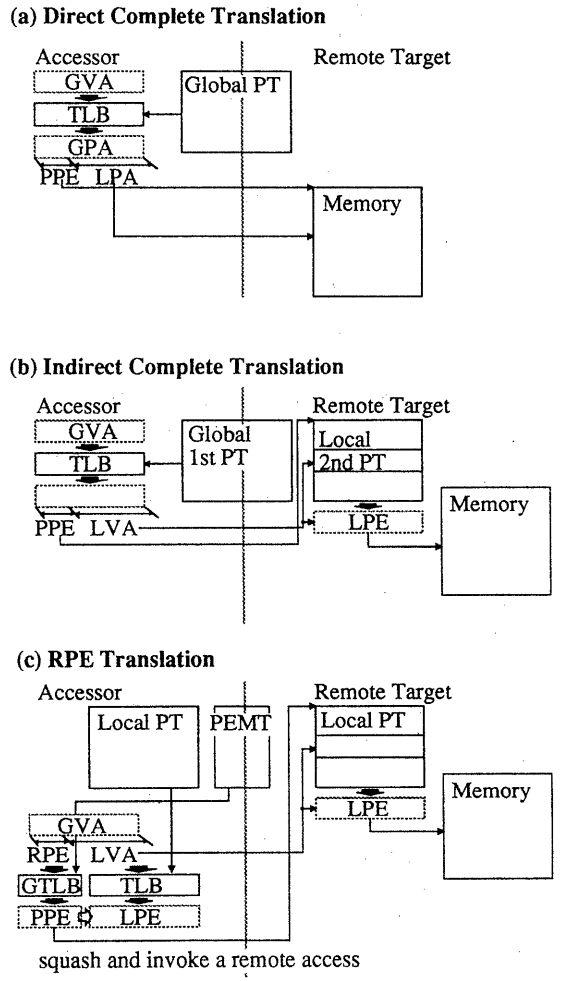

図 2: アドレス変換

リモートデータをキャッシングする場合には、アクセス側でアドレスを物理アドレスに変換しなければならない。したがって、一括完全変換方式のみが適用可能である[1]。

次に、二段階完全変換方式と、RPE 変換方式を比較すると、RPE 変換方式では、PPE を求める変換と LPE を求める変換を並列に行なうことができ、自系に対するアクセスの変換に要する時間が短い。一方、二段階完全変換方式では、第一の変換を適切に定めることによって、大きなデータをインターリーブ化して複数の PE 上にマッピングすることなどが可能である。

以上のような比較検討の結果から、RWC-1 では、RPE 変換方式を採用することにする。これは、以下の理由による。

1. RPE の利用による柔軟性の実現・メモリ保護機能の導入が可能である。
2. ハードウェアが簡単で高速の変換が可能である。
3. 仮想 PE の実現は、ソフトウェアの中に容易に組み込むことが出来る。

なお、最後の項目は、

original GVA → GVA.RPE | GVA.LVA

の変換をソフトウェアが行なうことを意味する。

3.4 RWC-1 の TLB

RWC-1 のアドレス変換は、RPE から PPE への変換と、LVA から LPA への変換からなる。LVA から LPA への変換は、通常プロセッサにおけるアドレス変換ほぼ同じである。以下では、並列計算機として着目すべき RPE から PPE への変換について検討する。

RPE から PPE への変換は、プロセス毎に用意する PEMT(PE Mapping Table) を用いて行われる。実際には、GTLB(Global Translation Look-aside Buffer) という PEMT のキャッシュがこれを行

[1]論理アドレスキャッシュを採用すれば、その他の方式も適用可能であるが、aliasing の問題や、一貫性管理の効率が低下するなどの問題点が多い。

なう。RWC-1では、複数のPEが1つのPEグループを構成するようになっており、グループ内の識別番号がRPEとPPEで一致するようにマッピングの自由度を制限している。これによって、GTLB全体の大きさを削減することが可能であり、フルマップ構成を採ることができる。フルマップ構成のGTLBは通信を行なう可能性のあるすべてのRPEに対する変換情報を保持する。このため、例えば、プロセス切替のタイミングでGTLBの内容を更新すれば、プログラム実行中にGTLBにおける変換例外は発生しない。

図3にRWC-1で採用を予定しているGTLBの構造を示す。図3の例は、1024のPEが8PEからなるグループに分割される場合のGTLBの構成を示している。

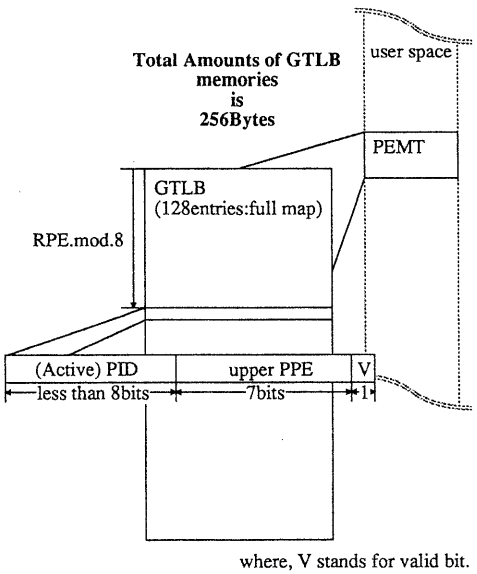

図 3: GTLBの構成

4 データの共有管理

4.1 データの共有とデータのコピー

並列計算機では、複数のPE間でデータを共有するのが一般的である。データを共有する方法は、以下の2つに大別できる。

- データのコピーをとらない方法
  他系のPE上のデータをアクセスする場合には、毎回、リモートアクセスを行なう。リモートアクセスのレイテンシィは大きいため、マルチスレッド実行などによってレイテンシィを隠蔽する必要がある。
- データのコピーをとる方法
  まとまったデータをリモートアクセスする時には、データをローカルにアクセスできる場所にコピーすることにより、平均的なレイテンシィを短縮することができる。同時に、ネットワークのトラフィックを削減する可能性も見込まれる。反面、複数のコピーが作られたデータに書き込みを行なう場合、一貫性の維持管理が必要となる。

データのコピーをとる方式において、複数のコピー間で一貫性を維持するためには、基本的に、他系のPE上の同じデータのコピーを無効化あるいは更新する必要がある。本稿では、一貫性を維持するためのこのような制御をデータの共有管理と呼ぶ。

データの共有管理には、(1) コヒーレントキャッシュ、(2)Shared Virtual Memory(SVM)、(3) 明示的コピー、を用いる方法がある。これらはアプリケーションの性質に応じて併用が可能である。

4.2 コヒーレントキャッシュ

リモートデータをキャッシングすることによって、リモートアクセスの回数を削減することができる。これによって、メモリアクセスの平均的なレイテンシィを削減でき、また、ネットワークのトラフィックを低く抑えることができる。また、共有を管理する単位がキャッシュラインであるから、細かなデータの共有の管理が可能となる。反面、キャッシュに保持されたデータのコピー間で一貫性を維持管理する必要がある。コヒーレントキャッシュでは、これをハードウェアが行なう。例えば、PE間で共有するデータに書き込みが行なわれた場合、書き込みを行なったPE以外のキャッシュに格納されたデータのコピーを無効化する。

バスで結合された小規模なシステムでは、各PE上のキャッシュが個々に一貫性を維持するための通信を監視していて、必要があれば、格納したデータの無効化などを行なうことができる (スヌープキャッシュ [5][6])。ところが、ネットワークで接続されたシステムでは、ブロードキャストのコストが大きいため、データのコピーを保持したPEをdirectoryによって管理し、通信量を削減する必要がある [1]。接続するPEが多くなると、一貫性の維持、つまり、directoryの管理に必要とするハードウェアの負担は非常に大きい。

4.3 SVM

仮想記憶では、アドレスのマッピングをページ単位で行なう。その際に論理ページと物理ページの対応関係を示すのがページテーブルである。SVMでは、図4に示すように、ページテーブルによる間接化を利用して、データのコピーを実現する。

データ共有管理方式	管理主体	共有を管理する単位	発生間隔
コヒーレントキャッシュ	ハードウェア	キャッシュライン	短い
SVM	ソフトウェア (OS)	ページ	中程度
明示的なコピー	ソフトウェア	データ構造	長い

表 2: データ共有管理方式の比較

他系の PE 上のデータをアクセスした場合、まず、アドレス変換例外が発生する。このアドレス変換例外の処理において、アクセスしたページが他系にマッピングされている場合、自系のメモリ上に空きの物理ページを確保し、その確保した物理ページに他系のデータをページ単位でコピーする。この時、ページの状態は clean にしておく。以降のリードアクセスでは、他系 PE 上のデータをあたかも自系にあるかのごとくアクセスすることができる。コピーページへの書き込みが行なわれると、書き込み例外が発生する。この書き込み例外の処理では、他系のコピーページを無効化するか、リモートアクセスを起動し他系のコピーページ内の書き込みが行なわれたデータを更新することにより、コピーページ間の一貫性を維持する。

論理的には、SVM はコヒーレントキャッシュでハードウェアで行なっていたデータの一貫性の維持管理をソフトウェアが行なうものと捉えることができる。したがって、ハードウェアが単純ですむ反面、データの共有を管理するためのソフトウェア的なオーバヘッドがやや大きい。

4.4 明示的なコピー

ある時点まで書き込みの結果を元データに反映させる必要が無い、あるいは、ある時点まで他系の PE 上のコピーを更新する必要が無い場合には、ソフトウェアによるデータの明示的なコピーが有効である。

反復計算のように、分割され局所化されたデータを用いて計算を進め、計算が完了した時点で結果を全体に反映させる場合に前述の条件があてはまる。この場合には、計算を完了した時点で、必要なデータだけを書き戻し、同期を取った後、各 PE が必要とする新しいデータを供給して、計算を進めていくことになる。データの転送はソフトウェアで起動され、転送されるデータは必要最小限である。また、命令実行と独立に、明示的にデータの転送を前もって起動 (プリフェッチ) できれば、大規模なデータの転送に要する時間を隠蔽することも可能である。

4.5 方式比較

データ共有管理方式の比較を表 2に示す。また、各方式の特徴を以下に簡単にまとめる。

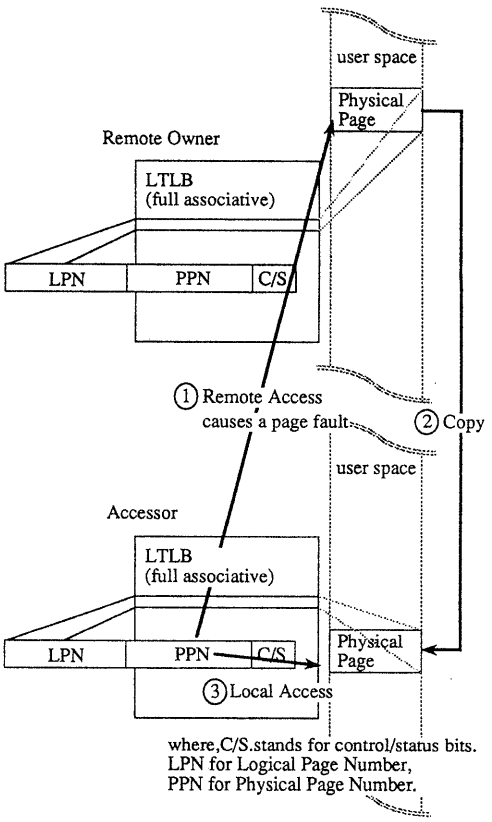

図 4: SVM

- コヒーレントキャッシュ
  細かくデータの共有を管理できるが、データのコピーを保持する PE を directory などを用いて管理する必要があり、ハードウェア的な負担が大きい。また、データを無効化するあるいは更新するためには通信が発生する。共有するデータへの書き込みが多い場合には、リモートアクセスの回数を削減できる反面、一貫性を維持するための通信により性能が低下してしまう [5]。コヒーレントキャッシュの効果は、アプリケーションの性質 (共有データに書き込みが行なわれる頻度) に、強く依存する。

- SVM
  ページ管理機構を利用するので、必要とするハードウェアの量は少ない。反面、データの共有をソフトウェアで管理するため、オーバヘッドがやや大きい。また、データの共有を管理する単位であるページのサイズが大きい場合には、false sharing[7] の問題が生じ、共有管理の効率が低下する。
- 明示的なコピー
  データの共有を管理するためのハードウェアを必要としないが、データ転送をソフトウェアで指定する必要がある。そのためには、データの構造を意識したアルゴリズムレベルの明示的な操作が必要である。

RWC-1では、基本的に、データのコピーをとらないことにする。したがって、共有するデータが他系のPE上にある場合には、毎回、リモートアクセスを行なう。リモートアクセスのレイテンシィはマルチスレッド実行によって隠蔽する。一方、ページ管理機構を用いることにより、ページ単位でデータを(コピーし)共有することができるから、リードオンリーデータに対してはSVMを導入する。これはコードの共有などに有効である。なお、false sharingの問題を解決するために、小さなページを導入を考えている。小さなページの導入は、(1)TLBの使用効率が低下する、(2)cache aliashingの問題が生じる、など影響が大きく、今後の検討課題となっている。さらに、小さなページのSVMでは、今まで以上に共有管理のためのオーバヘッドが大きくなる危険がある。このオーバヘッドは、主に、コンテキストの切替えに起因するものと考えられるから、小さなページのSVMの導入にあたっては、コンテキスト切替の高速化が課題となる。

5 キャッシュとページング

5.1 物理アドレスキャッシュの採用

メモリアクセスのレイテンシィを削減するためには、キャッシュが有効である。キャッシュは論理アドレスキャッシュと物理アドレスキャッシュに大別される。論理アドレスキャッシュは論理アドレスのタグを持ち、物理アドレスキャッシュは物理アドレスのタグを持つ。論理アドレスキャッシュではaliasingの問題が生じるため、RWC-1では、物理アドレスキャッシュの導入を考えている。

5.2 世代管理方式の提案

物理アドレスキャッシュでは、ページインなどのようにページのマッピングが更新される時、いままで割り当てられていた論理ページの変換結果である“古い”物理アドレスを持つキャッシュ上のデータをすべて無効化する必要がある。この無効化には、次の2つの方法がある。

- オールフラッシュ
- アドレスを指定した選択的な無効化

前者では、フラッシュ直後のヒット率の低下により平均的なアクセスのレイテンシィが長くなる。後者では、無効化に要する時間が長いといった問題がある。

通常のシステムではページのマッピングが更新される時間間隔が十分長く、このようなオーバヘッドはほとんど問題とならない。ところがSVMを採用した場合、他系のPEから極めて短い時間間隔でデータが供給されるため、キャッシュの無効化に要する時間をできるだけ短くする必要がある。このような背景から、無効化に要する時間を短縮する方式の検討を行なった。

ページのマッピングが更新される場合の不都合を回避するためには、“古い”物理アドレスを持つデータをキャッシュ上から抹消するか、“新しい”物理アドレスと“古い”物理アドレスを識別すれば良い。今回提案する世代管理方式は後者の方式である。世代管理方式では、物理ページのマッピングを更新した時点で、物理ページの世代を更新し、世代を含めた物理アドレスにより、“新しい”物理アドレスと“古い”物理アドレスを識別する。世代管理方式を採用したキャッシュの構成を図5に示す。

世代管理方式では、世代が尽きない限り、世代までを含めた物理アドレスによって、ページのマッピングの更新前と変更後の変換結果を識別することができる。ある物理ページに対する世代が尽きた場合には、キャッシュ上のデータの無効化が今までと同様必要となるが、無効化の頻度を1/(世代数)に削減できるため、SVMによるデータ共有の効率を改善できものと考えている。

6 まとめ

超並列計算機の記憶階層の要件を示し、メモリの大域的仮想化とデータの共有管理について述べた。RWC-1では、大域仮想アドレス(GVA)をPEを決定する部分と(RPE)とPE内に閉じた局所仮想アドレス(LVA)の組とすることにより、変換機構の柔軟化と簡単化を図っている(RPE変換方式)。また、データの共有は、基本的にはリモートアクセスを行ない、マルチスレッド実行によってレイテンシィを隠蔽する方針である。一方で、ページ管理機構を用いたSVMを導入する予定である。本稿では、さらに、SVMを導入する場合に問題となる、ページング時のキャッシュの無効化に要する時間を短縮する世代管理方式を提案した。小さなページを導入の含め、定量的な評価が今後の課題である。RWC-1は1996年春の完成を目指

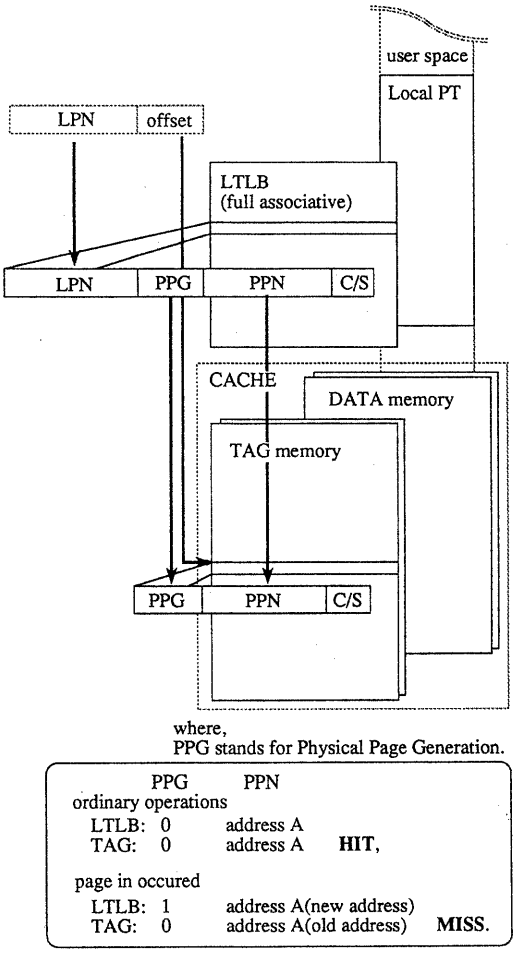

図 5: 世代管理方式

しており、詳細設計を進めながら、シミュレータを用いて他方式との比較を行い、採用した方式の評価を進めていく方針である。

謝辞

本研究を遂行するにあたり、有益な御指導、御討論をいただいた島田つくば研究所長、古谷超並列・ニューロ研究部長、石川超並列ソフトウェア研究室長、超並列ソフトウェア研究室員の諸氏、ならびに超並列アーキテクチャワーキンググループの諸氏に感謝いたします。

参考文献

[1] D. Lenoski, J. Laudon, K. Gharachorloo, W.D. Weber, A. Gupta, J. Hennessy, M. Horowitz, M. S. Lam, The Stanford Dash Multiprocessor, IEEE Computer, Vol.25, No.3, pp.63-79, 1992.

[2] D. Chaiken, J. Kubiatowicz, A. Agarwal, LimitLESS Directories: A Scalable Cache Coherence Scheme, Proc. of ASPLOS91, pp.224-234, 1991.

[3] S. Sakai, Y. Kodama, Y. Yamaguchi, Prototype Implementation of a Highly Parallel Dataflow Machine EM-4, Proc. of IPPS, pp.278-286, 1991.

[4] R. S. Nikhil, G. M. Papadopoulos, Arvind, *T: A Multithreader Massively Parallel Architecture, Proc. of ISCA92, pp.156-167, 1992.

[5] J. Archibald, J. Badr, Cache Coherence Protocols: Evaluation Using a Multiprocessor Simulation Model, ACM Trans. on Computer Systems, Vol.4, No.4, pp.273-298, 1986.

[6] T. Matsumoto, K.Hiraki, Dynamic Switching of Coherent Cache Protocols and its Effects on Doacross Loop, Proc. of ICS93, pp.328-337, 1993.

[7] K. Li, Shared virtual memory on loosely coupled multiprocessor, PhD theis, Yale Univ., 1991.

[8] 横田, 松岡, 岡本, 廣野, 堀, 児玉, 佐藤, 坂井, 超並列計算機 RWC-1 の相互結合網, 情処 ARC (SWoPP93), 発表予定.

[9] 坂井, 岡本, 松岡, 廣野, 児玉, 佐藤, 横田, 超並列計算機 RWC-1 の基本構想, Proc. of JSPP93, pp.87-94, 1993.