可変構造型並列計算機におけるキャッシュ・コヒーレンス処理

5X-2

岩田英次　森 眞一郎　村上和彰　福田 晃　富田眞治
(九州大学大学院総合理工学研究科)

1. はじめに

現在我々は，128台のPE（Processing Element）を128×128のクロスバー網で接続するマルチプロセッサ・システム「可変構造型並列計算機」を開発している[1]．本システムでは，128台のPEそれぞれに，容量4MBのローカル・メモリおよび容量128KBの仮想アドレス・キャッシュを備えている．キャッシュの主記憶更新アルゴリズムには，ストア・スルー方式を採用した．また，任意のメモリ・イメージをプロセッサ及びプロセスに対して見せることができる“可変構造メモリ・アーキテクチャ”を採用している．したがって，複数のプロセッサがメモリを共有する形態をとる場合，キャッシュ・コヒーレンス問題が発生する[2]．本稿では，本システムにおけるキャッシュ・コヒーレンス処理について，その制御機構と動作について述べる．

2. キャッシュ・コヒーレンス問題

2. 1 一般的解決策

主記憶を共有する密結合型マルチプロセッサ・システムにおいて，プロセッサがプライベート・キャッシュを有する場合，キャッシュ・コヒーレンス問題が発生する．この問題に対する解決策としては，一般に以下の4つが用いられている．

①ブロードキャスト方式
各々のプロセッサがキャッシュに書き込む度に，その書き込み情報を全てのプロセッサにブロードキャストし，無効化あるいは更新を行う．

②スヌーピング・キャッシュ方式
バス共有型マルチプロセッサ・システムを前提とした対処法である．各々のプロセッサに共有バス上を流れる他プロセッサのトランザクションを監視する機構を置き，自キャッシュに影響を及ぼすトランザクションを認識すると，それに対応する自キャッシュ上のブロックの無効化あるいは更新を行う．

③グローバル・ディレクトリ方式
キャッシュ・ブロックと主記憶ブロックとの対応やそのブロックの現在の状態を表すディレクトリのコピーを主記憶制御装置で集中管理し，無効化されるべきコピーを持っているキャッシュにのみ無効化要求を送る．

④ソフトウェア制御方式
一般には，実行前にユーザあるいはコンパイラが，データの属性を共有あるいは非共有と宣言し，更新可能な共有データについてキャッシングを許さないという手法が採られる．

2. 2本システムにおける対処

これら4つの方式と本システムとの親和性について考察する．

①のブロードキャスト方式では，プロセッサ数に比例して無効化処理のオーバヘッドおよび無効化要求のトラフィックが増大する．特に，動的網という相互結合網の性質上，ブロードキャストに伴う通信オーバヘッドにはとても耐えられない．

②のスヌーピング・キャッシュ方式についても，あるPEに他のPEの全てのアクセスを監視させるためにはブロードキャストが必要となり，ブロードキャスト方式と同様の問題が生じる．

一方，③のグローバル・ディレクトリ方式は，無効化要求のトラフィックが少ないという点で有利である．ただし，最大128台のプロセッサがメモリを共有可能なことから，グローバル・ディレクトリのハードウェア量の増大を招く．これを防ぐために，コンシステンシを維持する主記憶ブロックの単位（コンシステンシ・ブロックと呼ぶ）を大きくする，あるいはコンシステンシ・ブロックあたりのキャッシング可能なプロセッサの数を制限するといった制約が必要となる．

④のソフトウェア制御の場合，実行時のオーバヘッドは生じないという利点があるが，キャッシュの透過性が保てないことや，共有データのキャッシングを制限することによる性能低下といった欠点がある．

上記の考察を鑑み，我々はグローバル・ディレクトリ方式およびソフトウェア制御方式の両方式を採用し，ハードウェアおよびソフトウェアの両レベルでキャッシュ・コヒーレンス問題に対処することにした．

Cache Coherence in the Kyushu University Reconfigurable Parallel Processor
Eiji IWATA, Shin-ichiro MORI, Kazuaki MURAKAMI, Akira FUKUDA, and Shinji TOMITA
Kyushu University

3. コヒーレンス制御機構

3.1 ページ・テーブル

ページ・テーブル・エントリ中にCA（Cacheable）ビットを設けた．このビットにより，仮想記憶空間の各ページ単位にキャッシング可／不可を指定することが可能となる．

3.2 コンシステンシ・ディレクトリ

グローバル・ディレクトリを各PEのメモリ・ユニットに分散し，各々を“コンシステンシ・ディレクトリ”と呼ぶ．コンシステンシ・ディレクトリはコンシステンシ・ブロックがどのPEにキャッシングされているかを記録する．コンシステンシ・ブロックのサイズは，無効化の効率と無効化処理の際のオーバヘッドを考慮してキャッシュのライン・サイズ（32B）と同一にした．コンシステンシ・ディレクトリのフォーマットを以下に示す．

CC	CD	キャッシング権を持っているPEの番号（7bit）

CC : Consistency Check
CD : Cached

CC（Consistency Check）ビットは，後述の無効化処理を行うか否かをコンシステンシ・ブロック単位で指定する．無効化処理を行う場合，コンシステンシ・ブロックは一時には1台のPEにしかキャッシングを許さないという制限（Single Copy）をつけた．

4. コヒーレンス処理

4.1 ソフトウェアによる処理

ページング・スーパーバイザはCAビットを用いて，更新可能な共有データをキャッシング不可のページに割り付けることでコヒーレンス処理を行える．

また，CCビットにより，無効化処理を禁止すること

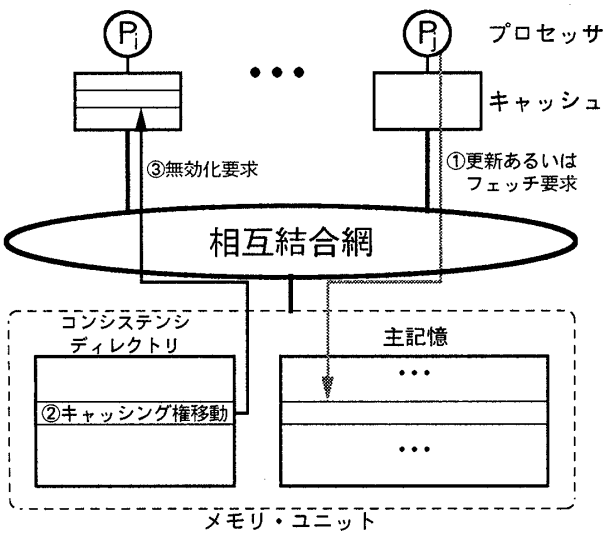

図1 ハードウェア制御による
マルチキャッシュ・コンシステンシ保持

で，read－onlyの共有データに関して，複数のPEにキャッシングを許すこと（Multiple Copy）が可能となる．

これらCAおよびCCビットの設定は，プログラム実行前はもちろん，実行中においても可能であることから，プログラムの実行フェーズに応じて動的に，キャッシュ・コヒーレンス処理にOSが介在できる．

4.2 ハードウェアによる無効化処理

無効化要求を送る必要が生じる状態は以下の2つである．

①PEiのキャッシュにコピーが存在するコンシステンシ・ブロックに対して，リード・ミスに伴うPEjからのブロック・フェッチ要求がきた場合（Single Copyの保証）

②PEiのキャッシュにコピーが存在するコンシステンシ・ブロックに対して，PEjがライト・アクセスを起こした場合（consistencyの保証）

①の場合は，PEiが持っているそのコンシステンシ・ブロックのキャッシング権をPEjに移動させ，PEiに対して無効化要求を送出する必要がある．これに対し，②の場合は，PEjはそのコンシステンシ・ブロックをフェッチしないのでキャッシング権を移動させる必要はないが，PEiのキャッシング権を取り上げ，PEiに対して無効化要求を送出する必要がある．

図1に示すように，メモリ・ユニットでは，上記の手順に従って，コンシステンシ・ディレクトリの操作および無効化要求メッセージの作成・送出を行う．無効化要求メッセージを受け取ったPEのキャッシュは速やかに当該ブロックを無効化する．

5. おわりに

以上，本システムにおけるキャッシュ・コヒーレンス処理について述べた．ソフトウェアの介在のしかたに多様性を持たせているので，コヒーレンス処理として様々な手法をとることが可能である．今後は，多様な主記憶参照パターンとこれらのコヒーレンス処理との親和性を評価し，コヒーレンス処理のソフトウェアによる最適化を図る．

参考文献

[1] K. Murakami, et al. : The Kyushu University Reconfigurable Parallel Processor － Design of Memory and Intercommunication Architectures － , Proc. 1989 Int'l Conf. Supercomputing, pp. 351－360, June 1989.

[2] 蒲池ほか：可変構造型並列計算機のメモリ・アーキテクチャ，情処38全大論文集，3T－2（1989年3月）