5K－2

MBPコアのファームウェア設計 －疑似フルマップ方式のサポート－

三吉 貴史　松本 尚　平木 敬

東京大学理学部情報科学科

1 はじめに

我々は、分散共有メモリ実装方式としてディレクトリベースキャッシュ方式の一つである疑似フルマップ方式[1]を提案してきた。MBP (Memory Based Processor) [2]は、疑似フルマップ方式の基本機能を高速にサポートする専用ハードウェア回路と、頻度の少ない例外処理や高い自由度の必要な処理を行う汎用細粒度プログラマブル処理機構（MBPコア）を内蔵している。すなわち、ハードウェアによる実現ではコストの見合わない処理に対して、MBPコアのプログラム（ファームウェア）が起動されて処理が行なわれる。

本稿では専用ハードウェアによる実現部分とファームウェアによる実現部分のトレードオフに対して議論する。

2 疑似フルマップ方式

疑似フルマップ方式は、ディレクトリ量に関するスケーラビリティを持ち、データを共有する要素プロセッサ間通信の最適化とキャッシュプロトコルの最適化をサポートするディレクトリ方式である。疑似フルマップ方式は階層化放送機構[1]を備えるネットワークの使用を前提とする。階層化放送機構は、階層性をもつネットワークを使用してハードウェアによるマルチキャスト機能およびACK (acknowledge) のコンバイニングを行なって回収する機能を持つ。

ディレクトリは、(1)近傍フルマップ (2)階層化マップ (3)遠距離直接指定の3種類の表現形式からなり、これらを切替えて使用する。

また疑似フルマップ方式が用いる主要な高性能化機構・機能として、

- 階層化放送機構の使用
- update系プロトコルが使用可能
- ページ単位ディレクトリ／ブロック単位転送
- ディレクトリ情報のキャッシング

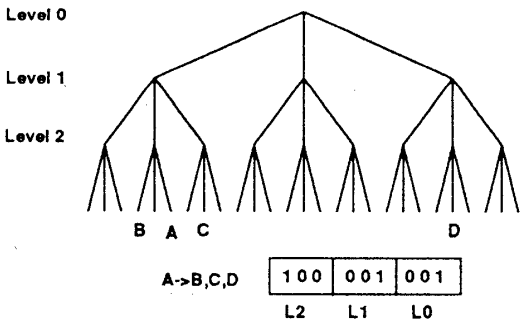

図1: 疑似フルマップ(階層化マップ)

が挙げられる。

疑似フルマップ方式では、階層化放送機構を用いた同一のマルチキャストパタンに対してのみネットワークのFIFO性が仮定されており、1対1通信や1対1通信とマルチキャストの間ではパケットの発信順序と受信順序の対応は保証されない。

各ノード（クラスタ）にはクラスタメモリがMBPを介してrouterに接続して、分散共有メモリ空間を構成する。router間は図1のような階層化放送可能なネットワークで結合する。図の中継ノードの部分にはACKの回収等の機能を実現するために、routerのみではなくMBPとメモリも存在する。論理的にはMBPとrouterは一体であると考えて支障がないため、以下の議論では合わせてMBPとして議論する。

3 疑似フルマップ方式のサポート

疑似フルマップ方式で、ファームウェアによる補助が必要となる部分としては、ハードウェアテーブル／バッファの枯渇やアクセス競合時に対するパケットのキューイング動作が挙げられる。以下の節では、デッドロックフリーでコンシステントな通信プロトコルを設計する際に必要なHOMEノードにおけるシリアライズ方式と、ネットワークの負荷を大幅に軽減するACKコンバイニング方式について説明を行ない、それらの中でファームウェアが果たす役割について議論する。

4 シリアライズ方式

同一メモリブロックに対するアクセスの競合が発生した場合、データを共有するノード間の一貫性を保つため、HOMEノードでシリアライズを行なわなければな

Firmware Design on MBP-Core – Support of Pseudo Fullmap Directory –
Takashi MIYOSHI, Takashi MATSUMOTO, Kei HIRAKI
Department of Information Science, Faculty of Science, the University of Tokyo
{miyoshi,tm,hiraki}@is.s.u-tokyo.ac.jp

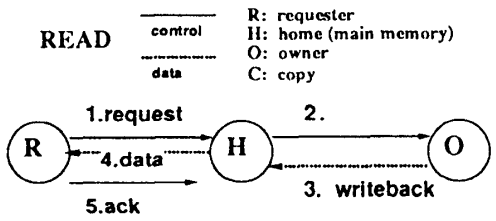

図 2: read プロトコル

らない。例えば、図2 は、read request に対する動作（HOME ノードにデータが存在しない場合）を示しているが、2～5 の間では他の update や invalidate 等の要求を受けつけることが出来ない。したがって、これらの要求は待ち合わせリストにつながれる。一方、read の要求が重なった場合では、多重化出来ることが望ましい。これについては、各ブロック毎にカウンタを用意し、待つべき ACK の数を管理することにより処理することが可能である。ACK が戻ってきた時点で、待ち合わせリスト中の要求を活性化させる。このように、多重化のためのカウンタと待ち合わせリストの操作によりシリアライズを行なう。このうち、待ち合わせリストへの追加とリストからの要求の活性化は、同一ブロックへのアクセス競合時のみ必要であるため、MBP コア を起動してファームウェアで処理する。

要求の待ち合わせリストへの追加は、リストの末尾に追加し該当メモリブロックの状態タグを待ち合わせバケットが存在する状態に変更する。一方、要求の活性化は、リストの先頭から要求を取り出し、先頭の要素が多重化可能な要求ならば、先頭から同じ種類の要求を同時に処理する。また、全ての待ち合わせバケットが存在しなくなれば、該当ブロックの状態タグを更新する（図3）。

```
invoke_next_transaction()
{
    p = serializer[addr]->list_head;
    current_mode = p->contents.mode;
    send(p->contents);
    addr->count ++;
    if (current_mode == read ||
        current mode == update)
        for (q = p->next; q ; q = q->next)
            if (q->contents.mode == current_mode) {
                send(q->contents);
                addr->count ++;
                delete_from_list(q);
            }
            else
                break;
    delete(p);
    if (waiting_list_is_empty())
        wait_flag = 0;
}
```

図 3: 待ち合わせリストの活性化

5 ACK コンバイニング方式

ACK のコンバイニングの操作は、基本的に、中継ノードでマルチキャストをする場合にカウンタをセットし、カウンタの数だけ ACK/DACK を受けとった時点で、1 個の ACK を要求元に向かって返送する一連の動作である。この処理を行なうには、処理する ACK を登録しておくテーブルが必要である。このテーブルは、性能上の理由から MBP 内部に専用ハードウェアによって実装される。しかし、ハードウェアテーブルが溢れた場合には、クラスタメモリ上に拡張領域を確保する。

この拡張領域に関する操作は、(1) エントリ追加時のテーブル溢れ (2) テーブル参照時のミス (3) テーブル上のエントリ削除に伴う操作の 3 種類である。これらの操作は、専用ハードウエアによって処理しても、外部メモリアクセスがボトルネックとなって高速化が難しく、処理内容も複雑なので、MBP コアを起動してソフトウェアによって処理すべき事柄である。

HOME ノードと中継ノードでは、行なう処理が異なるので、次の 5 つの場合の処理をファームウェアとして実装する。

- HOME ノードでのテーブルへの登録時の溢れ
- 中継ノードでのテーブルへの登録時の溢れ
- 中継ノードに ACK が返ってきた時のテーブル参照ミス
- HOME ノードに ACK が返ってきた時のテーブル参照ミス
- ハードウェアテーブル上のエントリ削除時の拡張領域からのエントリ移動

6 おわりに

専用ハードウェアによる実現部分とファームウェアによる実現部分の適切な切り分けにより疑似フルマップ方式の効率の良い実現が可能であることを示した。本稿では現在開発中の MBP を用いた分散共有メモリを例に挙げて、ファームウェアの設計指針について述べた。

なお、本研究の一部は文部省科学研究費（重点領域研究 (1) 課題番号 04235130「超並列原理に基づく情報処理基本体系」）による。本研究の遂行にあたり重点領域研究研究代表者の東京大学工学部田中英彦教授、ハードウェア班班長の京都大学工学部富田真治教授および、重点領域研究に参加している研究者の諸氏に感謝致します。

参考文献

[1] 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," 電子情報通信学会技術研究報告, CPSY92-26, vol. 92, pp. 47–55, Nov. 1992.

[2] 松本尚, "局所処理と非局所処理を分離並列実行するアーキテクチャ," 第 43 回情報処理学会全国大会論文集 (6), pp. 115–116, Oct. 1991.