1 はじめに

多段結合網 (Multistage Interconnection Network: MIN) は, 2×2 から 8×8 程度の小規模なクロスバスイッチから成るスイッチングエレメントを複数ステージ接続した結合網で, 比較的少量のハードウェアで大きな規模のシステムを実現できる. プロセッサ-メモリ間接続に MIN を用いたマルチプロセッサは, 同時に多数のメモリモジュールをアクセスすることができるため, 古くから研究が進められている.

MIN を用いたマルチプロセッサでは, プロセッサ毎にキャッシュを設けた場合, バスにおけるスヌープ機構を用いることができないため, その一致制御が困難になってしまう問題点がある. 従来,MIN を用いたマルチプロセッサのキャッシュ制御としては, コンパイラ等による事前解析の結果を利用する方式が検討されてきた. 最も簡単なのは, 事前解析により一貫性を保持する必要がない変数のみをキャッシュする（あるいはローカルメモリに割り付ける）方式であるが, これではキャッシュの効果が制限される. そこで, 並列 DO 文の境界毎に, 無効化, キャッシュ開始/ 終了などのキャッシュ制御命令をコンパイラが埋め込む方法 [11], 参照マーキング法 [4], 部分的にハードウェアの助けを借りる方法 [3] 等様々な方法が提案されている.

一方, 最近の VLSI 製造技術の発展により,MIN 用のスイッチングエレメントに高い機能を与えることが容易になった. このことを利用し,MIN のスイッチングエレメント自体にキャッシュを組み込んだり, ディレクトリを組み込む方法の検討が行なわれている [8] [9]. しかし, 従来の方法は MIN のスイッチングエレメント中にキャッシュ本体やディレクトリを組み込んだため, 全体の記憶量が膨大になり, アクセス時間の点でも不利であった. そこで, 本報告ではこれらの問題点を解決するために, 超並列マシン JUMP-1[5] 用に開発された階層ディレクトリ縮約方式 [12][7] を導入し, スイッチングエレメント中にディレクトリを持たず, 簡単なディレクトリキャッシュのみを持つ MIN, MINC(MIN with Cache control mechanism) を提案する.

2 階層ビットマップディレクトリ方式

スヌープ機構を用いないでキャッシュの一貫性を維持するためには, キャッシュラインを保持しているプロセッサを記録するディレクトリを設け, 例えば, キャッシュに書き込みが起きた場合, このディレクトリを引いて, ラインを共有するプロセッサ番号を知り, これらに対し無効化メッセージを送る.

キャッシュディレクトリの管理方式はすべてのプロセッサに対応する bit map を保持するフルマップ方式をはじめ,

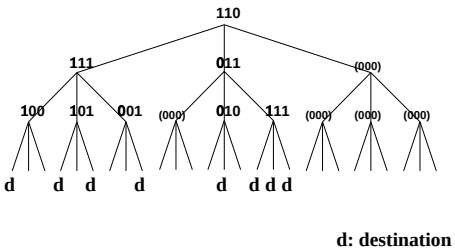

図 1: 階層ビットマップディレクトリ方式

リミテッドディレクトリ方式 [1] チェインドディレクトリ方式 [6] など様々な方式が提案され, 用いられてきた.

これに対して, 結合網が階層構造を持つ場合に有効な方式が階層ビットマップディレクトリ方式である. この方式では, プロセッサが葉に相当する木構造のネットワークを想定する. ここで, 木の根からパケットを供給して, 同一のパケットを複数のクラスタにマルチキャストする. 木構造のネットワークであるから, 各節において送信先の葉が末端にある枝にのみパケットを送れば, 必要なクラスタにのみパケットが届くことになる. 図 1に, 三進木の根から **d** でマークされた葉にパケットを送る場合について示す. この時, 各節に 3bit のビットマップを与えることにより送信先を完全に指定できる.

なお, 単純な木構造では木の根付近のトラフィックが大きくなって隘路となってしまうため, 結合網は Fat-Tree を内包することが望ましい. Omega 網, Generalized-cube(G-Cube) 網を代表とする全ての MIN はメモリをルートとする Fat-Tree 構造を内蔵していると考えることができる. このため,MIN にディレクトリ管理方式を組み込む場合, 階層ビットマップディレクトリ方式は最も自然な方式である. この場合, 中間ノードは各スイッチングエレメントに相当する. 従来のキャッシュ制御機構を組み込んだ MIN では, スイッチングエレメントを階層の中間ノードと考えて, キャッシュ本体またはディレクトリのみを置いて階層ビットマップディレクトリを実現した.

Memory Hierarchy Network(MHN)[8] は, MIN の階層構造を利用し, 各スイッチングエレメント中にそのエレメントを根とするツリーの葉に当たるプロセッサの共有するデータに対するキャッシュを置く方式である. 無駄なコピーとデータ一貫性に関するロスを防ぐため, キャッシュのコピーの数はひとつに制限されている. 共有するプロセッサ数が少ないデータは MHN のプロセッサに近いステージに置かれるため, アクセス遅延は少ない. 多くのプロセッサが共有するキャッシュは MHN のメモリ側のステージに移動していく.

MIND(MIN with Directory)[9] は、各 MIN 上にはキャッシュ自体は持たず, ディレクトリのみを置く方法である.

MIN はそれぞれのメモリモジュールを頂点とするツリー構造として考えることができるので, これを利用して MIN 全体で階層的なディレクトリを実現する. キャッシュの無効化メッセージはディレクトリの内容が 1 の所のみに送られるため, 必要なプロセッサにのみ, マルチキャストの形で無効化メッセージを送ることができる. この方法をより効率化するため, MIN の各スイッチングエレメントをバスで構成する MBN(Multistage Bus Network)[2] も提案されている.

しかし,MHN は各エレメントにキャッシュおよびディレクトリを置くため, 全体として膨大な量のメモリを必要とする. さらに, 各ステージでディレクトリをアクセスするため, キャッシュを共有するプロセッサ数が増加する. MIND は各ステージでキャッシュを持つ必要はないが, やはりエレメント内にディレクトリを持つため, 全体としての記憶容量が大きく, さらに, 各ステージでアクセスするため, 全体として一致制御に要する時間が大きい.

3 MINC(MIN with Cache control mechanism)

3.1 MINC の基本構成

本報告では, 従来のキャッシュ制御機構付きの MIN の問題点を解決する新しい機構 MINC(MIN with Cache control mechanism) を提案する. MINC は図 2に示すように, プロセッサからメモリに対してパケットを送る Forward MIN と, メモリからプロセッサにパケットを送る Backward MIN に分かれている. 従来の方式とは異なり, 階層ビットマップは縮約された形で, 共有メモリ上に置かれる. Forward MIN は MIN で応答パケットに関する制御を除き, 通常の MIN と同じである. Backward MIN は, ビットマップに従ってパケットをマルチキャストする機構を持つと共に, 小規模なディレクトリのキャッシュを持つ. Forward MIN, Backward MIN のトポロジについては特に制限がないが, ここではプロセッサの位置による局所性を利用できるように,Generalized Cube (G-Cube) 接続 [10] を想定する. また, 図 2では 2×2 のスイッチングエレメントを示すが, 実際は LSI チップでの実装を考え,8×8 程度のサイズを想定している.

3.2 ディレクトリの縮約

階層ビットマップ方式はプロセッサ数が多くなると, ディレクトリに必要なメモリ量が膨大になり, 実現が困難になる. また,ディレクトリはかなり大容量になることが予想されるので, スイッチングエレメントを構成するチップの外部に設ける必要がある. この場合, 階層毎にディレクトリを参照するには, 大きな時間を要する. そこで以下に示すビット数を縮約する方法が考えられる. この方法は超並列マシン JUMP-1 のディレクトリ制御用に考案された方式 [12] [13] であり, 以下の原理に基づいている.

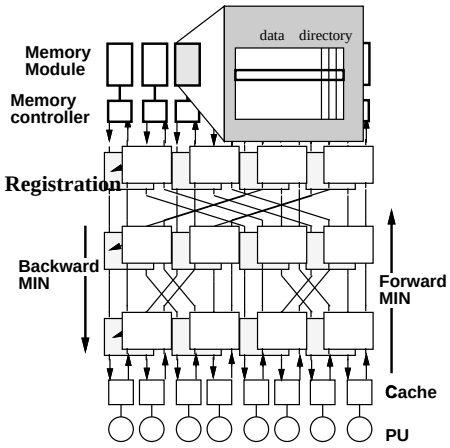

図 2: キャッシュ制御付き MIN の構成

1. いくつかのプロセッサをグループ化して扱うことにより必要なビット数を削減する.
2. 各節においてその節で適用するディレクトリを管理し, パケットがその節に到達した時にディレクトリを引く. こうすれば,n 進木を用いた時, 各節において 1 ディレクトリエントリにつき,$n$bit のビットマップを管理すればよい.

前者について,(1) ある節以下はブロードキャストとする (2) 複数の節で同一のビットマップを用いる のいずれか, もしくは両方の組み合わせによって, 階層毎に n bit(n 進木で) のビットマップを一つだけ持つこととする. このようにすると,ディレクトリエントリ毎に必要な bit 数は,m 階層の n 進木において $m \times n$ bit となる. さらに, ビットマップは, パケットのヘッダ中に持つことができるので, 完全にエレメント内のみでマルチキャストが可能であり, 外部のディレクトリを参照する必要がない. (1) と (2) の組合せにより 3 つのディレクトリ縮約方式が提案されている [7] が, ここでは,SM 法と LARP 法について検討した.

SM 法: 各階層毎に, その階層の全ての節の縮約前のビットマップの論理和をとり, その階層の全ての節で用いる.

LARP 法: 根から送信元に至るパスをその他のパスと区別して扱い, ある節で (1) 送信元を含む枝, (2) 送信元を含まない枝のうちのいずれか, に対し両方共パケットが送られると, 送信元を含む枝の下の部分全体にパケットがブロードキャストされる. それ以外の枝では, 同一階層の全ての節で, それらの節の縮約前のビットマップの論理和を用いる.

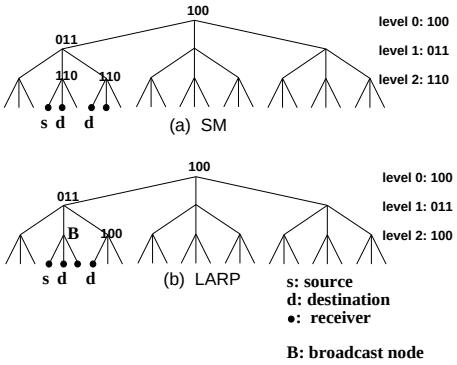

図 3: ディレクトリ縮約方式

図 3に三進木を用いた摸式図を示す．この図で s が送信元のクラスタ,d が本来の送り先,●が結果的にパケットが送り付けられるクラスタである．従って,d の無い●の数が少ないほど無駄にパケットを受けとるクラスタ数が少ないことになる．この縮約方式を採り入れた階層ビットマップディレクトリ方式を縮約階層ビットマップディレクトリ方式と呼ぶ．

縮約方式を用いた場合,MIN のエレメント内にディレクトリを持つ必要はなくなり, 共有メモリ上にのみ, 各ステージ（階層）で用いるビットマップを保持すればよい．Backward MIN を通過するパケットのヘッダにこの各ステージで持ちいるビットマップを入れておき,Backward MIN 内ではこのビットマップに従ってパケットをマルチキャストする．この方式では無駄なパケットがプロセッサに到着してしまうが, 全くスイッチングエレメント内のディレクトリをアクセスする必要がなくなり, 記憶容量が節約されると共に, 無効化の実行速度も向上する．

3.3 キャッシュプロトコル

MINC では, ディレクトリが共有メモリに集中している分プロトコルは簡単である．ただし, ディレクトリを縮約したため, ラインのコピーを保持しているプロセッサ数をカウントするカウンタを共有メモリのディレクトリに設ける必要がある．

各プロセッサに接続されたキャッシュには, そのラインの有効／無効 (Valid:V/Invalid:I) および, コピーが他に存在する／存在しない (Shared:S/Private:P) のタグが設けられている．提案するプロトコルは, これらの組合せにより I,P,S の 3 状態を持つライトスルー型であり, 図 4に示す状態遷移を行なう．有効なキャッシュに対しての読み出しアクセスがヒットすれば, 問題なくキャッシュからデータを読み込む．その他のアクセスは以下のように処理される．

1. 書き込みミス／ヒット: Forward MIN を用いて共有メモリにデータを送り, 共有メモリ上のディレクトリに従って Backward MIN を用いて, 無効化パケットをマルチキャストする．

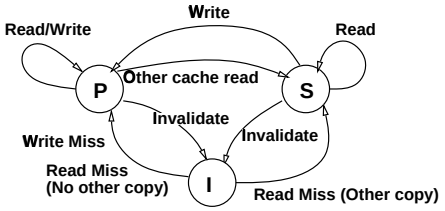

図 4: キャッシュの状態遷移

2. 読み出しミス: ミスしたキャッシュラインが I 状態でなければ,(3) に従って登録消去を行ない読み込み領域を確保する．次に Forward MIN を用いて要求パケットを共有メモリに送る．共有メモリ上のビットマップの必要な場所に 1 をセットし, カウンタをインクリメントする．そして,Backward MIN を用いて要求元のプロセッサのキャッシュに読み出したラインを送る．カウンタが 1, すなわち他に同一ラインを共有するプロセッサが存在しない場合, このパケットのヘッダ中の Private bit をセットする．要求を出したプロセッサのラインはこの bit がセットされていれば P 状態, そうでなければ S 状態になる．ここで, カウンタが 2 になった場合, 共有化パケットを Backward MIN を用いてマルチキャストし, このラインを共有しているもう一つのキャッシュラインの状態を S に変更する．
3. 登録消去: ディレクトリを縮約した場合, ビットマップ中の 1 は複数のキャッシュのコピーを表現している場合がある．したがって、簡単にこのビットをリセットすることはできない．そこで,Forward MIN を用いて登録消去メッセージを共有メモリに送り, ディレクトリ中のカウンタをデクリメントする．カウンタが 0 になった時にのみビットマップ中の全ての bit を消去する．

以上述べたプロトコルは無効化型であるが, 更新型に拡張することも可能である．更新型を用いる場合は, 無効化パケットの代わりに変更データを階層ビットマップを用いてマルチキャストする．データを受け取ったプロセッサのキャッシュラインの状態は変化なしで S のままにしておく．更新型プロトコルを用いる場合は, 初期状態以外に I 状態は用いられない．

3.4 ディレクトリキャッシュの導入

縮約方式は、大きな利点を持つ一方, 無効化パケットが必要以外のプロセッサにも届いてしまう問題がある．この

メッセージは届いたプロセッサで捨ててしまえばよいのでプロトコル上の実害はないが, ラインを共有するプロセッサ数が増えると, 無駄なパケットが増え Backward MIN の混雑が激しくなる. このため, 特定のステージのエレメントに対してのみ, チップ内部に実装できる程度の簡単なフルアソシアティブのディレクトリキャッシュを Backward MIN 上に設ける. ディレクトリキャッシュの実装のためには、Forward MIN と Backward MIN の対応するエレメント間にディレクトリ登録用のデータ線を必要とする. ディレクトリキャッシュは、以下のように動作する.

- Forward MIN を介してライン要求パケットを送る時に, ディレクトリキャッシュ上にラインアドレスを登録し, 対応する入力の bit をセットする. 既にキャッシュ上にエントリが存在する場合は、パケットの入力に相当する bit をセットする.

- Backward MIN を介して無効化パケットを送る時, ステージ上のディレクトリキャッシュを参照し, アドレスがマッチすればそのディレクトリのビットマップを利用し,1 に対応する出力に対してのみパケットを送ると共にエントリから登録を消去する.

頻繁に無効化が行なわれない場合, ディレクトリキャッシュ中に特定のラインアドレスに対応するビットマップが長期間滞在し, ディレクトリキャッシュの登録時に領域がなくなる場合がある. ディレクトリキャッシュの目的は無駄パケットの削減にあるので, このような場合は, 単純にどこかのラインを選んで捨ててしまって領域を確保すればよい. この場合, どのラインを選ぶかは, 以下のアルゴリズムが考えられる.

- 最も 1 の数の多いビットマップを持つものを捨てる.
- 単純な FIFO で先に登録されたものから順番に捨てていく.

4 評価

SM 法,LARP 法の双方のマッピングを用いた MINC を用いて評価を行なった. 評価基準として以下の二つの用語を定義する.

1. シャドープロセッサ : 必要としていないのに無効化メッセージが到達するプロセッサ
2. トラフィック : 無効化メッセージが Backward MIN を通過する際の総リンク数

トラフィックが高いと, 無効化メッセージ同士の衝突確率が高くなり, トラフィックが低いと衝突確率も低くなる. 従って, トラフィックはネットワークの混雑度を示す指標であると考えられる.

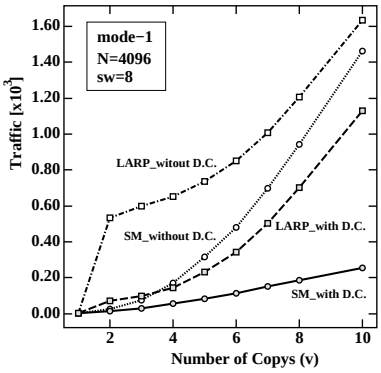

図 5: 条件 (1) でのトラフィック

キャッシュラインが存在する確率が,(1) どのノードも等しい系 (mode-1) と,(2) ライトミスを起こしたノードをピークとした時に, 左右に正規分布に従って分布している系 (mode-2) との両方の場合について, それぞれ, キャッシュラインを持っているノード数とスイッチサイズを変化させて評価した. また, 評価条件として以下のものを定義する.

N: ノードサイズ
sw: スイッチサイズ
v: キャッシュラインを持っているノード数

図 5, 図 6はそれぞれ,N=4096,sw=8 のシステムにおいて, ノード条件を (1),(2) とした時のメッセージ転送ノード数を表したものである. 図中 D.C. はディレクトリキャッシュの略で, 今回の MINC は最下段のスイッチ内部にのみ DC を載せたものと全く載せないものとで比較している. ここで,DC を持っている MINC と持っていない MINC のトラフィックの差がシャドープロセッサ数となることも併せて述べておく.

図 5では, 一様に SM 法が LARP 法よりも良い性能を表している. 更に, v が小さい時,LARP 法が SM 法よりかなりトラフィックが高くなっているが, これはキャッシュラインを持っているプロセッサがシステム全体に均等に存在するため,Tree の根に近いうちにメッセージが二方向以上に飛び, ソースノードの方向に向かってメッセージをブロードキャストしてしまうためである. 一方, キャッシュラインを持っているノード数が多くなってくると,MINC も LARP も変わらず, かなりのノードにメッセージが飛ぶ事になる.

図 6では,DC を搭載した LARP 法 MINC でも,DC を搭載していない SM 法 MINC に及ばないことが分かる. 更に,DC 搭載型 MINC はフルマップとほとんど差がなかった. つまり, データに局所性がある場合は, フルマップよりかなりメモリコストを抑えた MINC でも, 無効化の際の無駄パケットを抑え, 十分に対処できる事を示している.

図 7は,N=4096,v=8 の状態で, スイッチサイズを変化させた時のトラフィックを DC 搭載型 MINC について調べた.

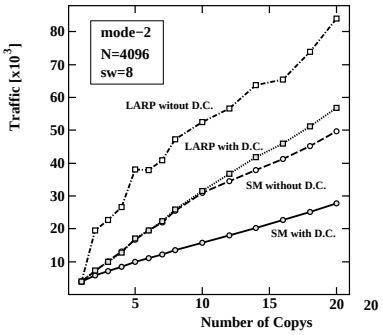

図 6: 条件 (2) でのトラフィック

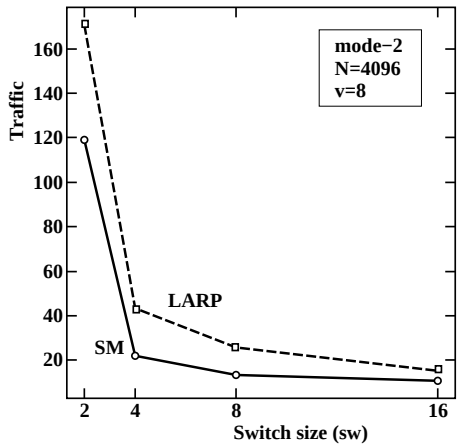

図 7: スイッチサイズを変化させた時のトラフィック

総じて LARP 法よりも SM 法の方がトラフィックを抑えていることが分かる. また,sw $\geq$ v の時はどちらもほぼフルマップと同じトラフィックで済んでいる.

5 結論

階層ディレクトリ方式を導入し, 簡単なディレクトリキャッシュのみを持つ MINC を提案した. データに局所性がある場合のトラフィックを調べた所、フルマップとほぼ同程度である事がわかった.

参考文献

[1] Agarwal, A., Simoni, R., Hennessy, J. and Horowitz, M.: An Evaluation of Directory Schemes for Cache Coherence, *Proc. 15th ISCA*, pp. 280–289 (1988).

[2] Bhuyan, L., Nanda, A. and Askar, T.: Performance and Reliability of the Multistage Bus Network, *Proc. of ICPP*, pp. I–26–I–33 (1994).

[3] Cheong, H. and Veidenbaum, A.: A Cache Coherence Scheme with Fast-Selective Invalidation, *Proc. of 15th ISCA*, pp. 299–307 (1988).

[4] Edler, J. and et al.: Issues Related to MIMD Shared-memory Coputers: the NYU Ultracomputer Approach, *Proc. of 12th ISCA*, pp. 126–135 (1985).

[5] Hiraki, K., Amano, H., Kuga, M., Sueyoshi, T., Kudoh, T., Nakashima, H., Nakajo, H., Matsuda, H., Matsumoto, T. and Mori, S.: Overview of the JUMP-1, an MPP Prototype for General-Purpose Parallel Computations, *Proc. IEEE International Symposium on Parallel Architectures, Algorithms and Networks*, pp. 427–434 (1994).

[6] James, D., Laundrie, A. T., Gjessing, S. and Sohi, G. S.: Distributed-Directory Scheme: Scalable Coherent Interface, *IEEE Computer*, Vol. 23, No. 6, pp. 74–77 (1990).

[7] Kudoh, T., Amano, H., Matsumoto, T., Hiraki, K., Yang, Y., Nishimura, K., Yoshimura, K. and Fukushima, Y.: Hierarchical bit-map directory schemes on the RDT interconnection network for a massively parallel processor JUMP-1, *Proc. International Conference on Parallel Processing* (1995).

[8] Mizrahi, H., Baer, J., Lazowska, E. and J., Z.: Introducing Memory into the Switch Elements of Multiprocesssor Interconnection Networks, *Proc. of 16th ISCA*, pp. 158–166 (1989).

[9] Nanda, A. and Bhuyan, L.: Design and Analysis of Cache Coherent Multistage Interconnection Networks, *IEEE Trans. on Computers*, Vol. 42, No. 4, pp. 458–470 (1993).

[10] Siegel, H. and Smith, S.: Study of multistage SIMD interconnection networks, *Proc. The 5th ISCA* (1978).

[11] Veidenbaum, A.: A Compiler-Assisted Cache Coherence Solution for Multiprocessors, *Proc. ICPP*, pp. 1026–1036 (1986).

[12] 松本尚: 局所処理と非局所処理を分離並列処理するアーキテクチャ, 第 43 回情処全大 (6), pp. 115–116 (1991).

[13] 松本尚, 平木敬: Memory-Based Processor による分散共有メモリ, 並列処理シンポジウム JSPP'93 論文集, pp. 245–252 (1993).