計算機アーキテクチャ　107－28
（１９９４．　７．　２３）

疑似フルマップディレクトリキャッシュの実装方式

三吉貴史 *1　　松本尚 *1　　佐藤充 *2
平木敬 *1　　田中英彦 *2

*1 東京大学理学部　　*2 東京大学工学部

大規模並列システムにおいて、キャッシュの使用は通信のレイテンシを削減することに大きな効果がある。同一データのコピーを複数のキャッシュが有することにより引き起こされるコンシステンシ管理の解決を目的として、様々なプロトコルが提案されている。分散共有メモリ上では、主としてディレクトリ方式が用いられている。疑似フルマップディレクトリ方式は、階層性を有する相互結合網を利用する効率的なキャッシュ管理方式である。本稿では、分散共有メモリ計算機システム上で疑似フルマップを実装する際に問題となるポイントについて検討する。まず従来提案されているディレクトリ方式および実際にディレクトリ方式が採用された例について検討し、実装という観点から性能・ハードウェア量・複雑さなどについて論じる。この結果を踏まえて、疑似フルマップ方式の特徴であるキャッシュプロトコル切替、ページ単位共有・ブロック単位転送、デッドロックフリーなプロトコルなどの実装方式について述べる。

Implementation of Pseudo Fullmap Directory Cache

MIYOSHI Takashi*1　MATSUMOTO Takashi*1　SATO Mitsuru*2
HIRAKI Kei*1　TANAKA Hidehiko*2

*1Faculty of Science, University of Tokyo
*2Faculty of Engineering, University of Tokyo

Use of cache can reduce latency of the communication on large scale multiprocessor systems. The problem of cache consistency occurs where multiple cache memories share the same data. In order to solve this problem, number of protocols are suggested in many years. Directory schemes are mainly used in distributed shared memory systems. Pseudo fullmap directory scheme is an effective cache managing method, which uses hierarchical networks and broadcast systems. This paper examines the problems on the implementation of the pseudo fullmap directory on the distributed shared memory system. First, we examine the directory schemes ever suggested and instances of the systems which uses directory schemes, and describe the performance, the amount of hardwares and complexity from the point of implementation. Then we presents the implementation method of the pseudo fullmap directory and its features such as the change of cache protocol, page-unit sharing / block-unit transferring, and the deadlock free protocol.

1 はじめに

大規模分散並列システムでは、通信のレイテンシ削減のためキャッシュの使用が必要不可欠である。しかし、複数のキャッシュで同一データのコピーを有する場合、データの書き込みによって引き起こされるキャッシュ間データのインコンシステンシの問題が起こる。従って、キャッシュコピー間のコンシステンシを保つプロトコルが必要である。バス結合型並列計算機では、キャッシュ操作の情報はバスを通じてブロードキャストされるので、バス上に流れるデータやアドレスなどの情報を常時監視してコンシステンシを保つことが可能である。このようなスヌープキャッシュプロトコルは数多く提案されてきている [6, 3]。一方、一般のネットワークではブロードキャストを行うと通信コストがかかるので、スヌープを用いるのは困難である。従って、どのキャッシュがデータを保持しているのかを示すディレクトリを用いて管理をする [11, 10]。ディレクトリを用いる方式は、ディレクトリを集中管理するか、分散管理するかにより大別される。集中管理型では、データを保持しているキャッシュの位置をビットベクタ、ポインタなどにより管理する [11]。一方、分散管理型のディレクトリ [8, 12] は、同一データ対するキャッシュエントリをリストでつないで管理する。

本稿で取り上げる疑似フルマップ方式 [15, 16] は、集中管理型のディレクトリ方式の一つで、ディレクトリに必要なビット数を節約してスケーラブルなキャッシュ管理を行なう。疑似フルマップは、階層化放送機構をもつネットワーク上で構成され、階層性を利用したデータ共有情報の管理を行なう。また、キャッシュプロトコルおよびディレクトリ形式の切替をサポートし、効率の良いキャッシュディレクトリを提供する。

本稿では、疑似フルマップ方式を従来提案されてきた方式と比較を行いこの方式の性能について定性的な解析を試みる。また、疑似フルマップを分散共有メモリ型計算機システム上に実装する場合には、コンシステンシモデルやデッドロックなどの問題に対してどのように取り組むかが重要になるので、これらの問題に対する機構や通信プロトコルについて論じる。2節で従来のディレクトリ方式について紹介し、ついで3節で疑似フルマップの説明を行なう。4節では、これらのディレクトリの比較を行なう。5節では疑似フルマップを用いたシステム JUMP-1[1] 上での通信プロトコルについて述べる。

2 従来のディレクトリ方式

ディレクトリによるキャッシュコヒーレンス管理は、データのコピーを共有しているプロセッサの位置をビットベクタにより保持するフルマップディレクトリ [11, 4] に端を発している。この方式では、共有情報を完全に保持しているので、コンシステンシ管理のためのメッセージ数を少なく押えることができる。しかし、フルマップディレクトリでは、主記憶のエントリ毎にプロセッサ数分の大きさのビットベクタが必要なのでスケーラビリティに欠けるという重大な欠点がある。従って、性能を出来るだけ低下させないようにこの欠点を克服することが、研究の対象となってきた。以下では、これまで提案されてきた方式各々について概観する。なお、これらの方式は invalidate 系のプロトコルを前提としているため、write は invalidate とする。

Fullmap Directory

特徴 キャッシュブロック毎にプロセッサ数幅のビットベクタを用意して必要なプロセッサにメッセージを送る。

動作 read req. → 要求元のプロセッサと対応するビットをたてる。
write req. → ディレクトリ中のビットの立っているプロセッサに invalidate request を送る。

備考 この方式では各メモリブロックに対してプロセッサ数に比例するディレクトリメモリが必要であり、大規模なシステムにおいては不適当である。

Limited Pointer 系プロトコル

ブロードキャストを用いない場合

特徴 フルマップのビットベクタの代わりに一定個数のポインタを用意しておく。

動作 read req. → ポインタに空きがあれば、要求元のアドレスをセット。ポインタが全て埋まっている場合はエントリを一つ invalidate し、処理を行う。
write req. → ディレクトリ中のポインタのさすプロセッサに invalidate request を送り、ポインタをクリアする。

備考 この方式では共有数が大きい問題に対して、"thrashing" の状態になり、性能低下が著しい。

ブロードキャストを用いる場合

特徴 一定個数のポインタを用意。

動作 read req. → ポインタに空きがあれば、要求元のアドレスをセット。ポインタが全て埋まっている場合はポインタをクリアして、ポインタ溢れを示すビットをたてる。
write req. → ポインタ溢れの場合には、invalidate request をブロードキャストする。

備考 ネットワークに放送機能が求められる。また、ブロードキャストを用いているのでスケーラビリティに欠ける。

LimitLESS[5]

特徴 一定個数のポインタを用意。ポインタ溢れにはソフトウェアによるフルマップのエミュレーションを行う。

動作 read req. → ポインタに空きがあれば、要求元のアドレスをセット。ポインタが全て埋まっている場合は割り込みを発生してソフトウェアで処理を行う。メインメモリにビットベクタを割り当て、現在保持しているポインタに対応するビットをたてる。
write req. → 保持しているポインタおよびメインメモリ中のビットベクタに対して invalidate request を送る。

備考 トラップによりソフトウエアを起動するので、高速な割り込み処理が必要となる。

Superset Scheme[2]

特徴 各エントリに対し2個のポインタを用意。ポインタ溢れの場合には、2個のポインタを組み合わせて広範囲の指定を行う。

動作 read req. → ポインタに空きがあれば、要求元のアドレスをセット。2個のポインタが埋まっている場合は、それらを組み合わせて1つのポインタを構成。このポインタは、各ビットが {0,1,X} の3状態をとる。このポインタに新しいエントリを加えるときには、競合するビットを X の状態にする。
write req. → 保持しているポインタに対して invalidate request を送る。ポインタが複合ポインタの場合は、X を 0,1 に展開し、該当するアドレス全てに対してメッセージを送る。

備考 invalidate 時に発行するメッセージ数は、複合ポインタ中の X の数を d とすると、2^d 個となる。

Coarse Vector[7]

特徴 プロセッサをグループ化し、ビットベクタを各グループに対応させる。

動作 read req. → ポインタに空きがあれば、要求元のアドレスをセット。ポインタが溢れている場合は、グループのビットベクタに切替えて、自分の属するグループのビットを立てる。
write req. → 保持しているポインタに対して invalidate request を送る。ビットベクタの場合には、対応するグループに属する全プロセッサに対して invalidate request を送る。

備考 超並列環境では一つのグループの大きさが巨大になる。この方式では、実際には共有していないノードへのメッセージがどの程度になるかが性能上重要になる。

Chained Directory 系プロトコル

Single Linked List[12]

特徴 ディレクトリ情報をキャッシュ上に分散させ、シングルリストで繋げて管理する。

動作 read req. → 要求元をリストに繋げる。
write req. → リストをたどりながら順に invalidate を行う。

備考 write は、リストを順にたどるので、共有数に比例してレイテンシが増大する。

Double Linked List[8]

特徴 ディレクトリ情報をキャッシュ上に分散させ、ダブルリストで繋げて管理する。

動作 基本動作はシングルの場合と同じ。

備考 シングルの時と比べて、キャッシュリプレイス時の操作が簡単になるが、メモリオーバーヘッドが倍になる。

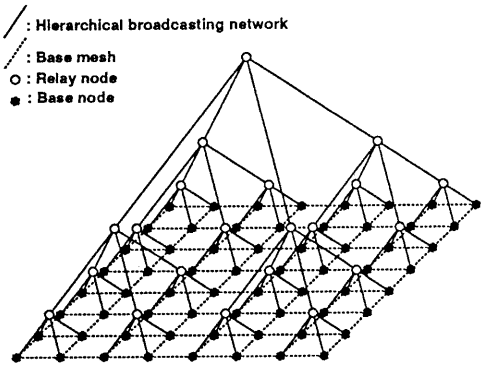

図 1: 4進木ネットワーク＋二次元メッシュ

3 疑似フルマップディレクトリ方式

疑似フルマップディレクトリ方式は、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。疑似フルマップは階層化放送機構を持つネットワークの使用を前提とする。階層化放送機構とは、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りマルチキャストを行なう機能およびack(acknowledge)のコンバイニングを行なって回収する機能を合わせ持つ機構である。このようなネットワークとしては m-進木ネットワークを含むもの（図1）やRDT[17]などがある。

階層化放送機構を前提とすることから判るように疑似フルマップディレクトリ方式はネットワーク等の実装方式と不可分なディレクトリキャッシュ方式である。そこで、文献[15, 16]において提案した時点では、疑似フルマップディレクトリ方式は単にディレクトリの表現形式ではなく、さまざまな高性能化のためのユニークな機構や機能を含んだ呼称であった。しかし、本稿では他のディレクトリスキームと対比する都合上、ディレクトリの表現形式と他の特徴を分離して、表現形式のみを疑似フルマップと呼ぶ。

3.1 疑似フルマップの表現形式

ディレクトリは次の3種類の表現形式（モード）からなり、これらを切替えて使用する。

- 近傍フルマップ
- 階層化マップ
- 遠距離直接指定

データを共有しているノードがホームノードから見て一定の近傍内に収まる時は近傍フルマップによりノードの位置を特定する。階層化マップは、ネットワークの階層性を利用したモードで、ホームページの近傍を詳しく、遠方のコピーページを大まかに記憶する近傍詳細型(LPRA:Local Precise Remote Approximation)、逆に近傍は大まかに、遠方を詳しく記述できる遠方詳細型(LARP:Local Approximation Remote Precise)、同一マップ型(SM:Single Map)がある。なお、LARP型とSM型は最近になって工藤らによって考案されたものである[13]。遠距離直接指定では、遠方に1ノードだけ共有しているノードが存在する時、一意的に定まるような指定を行う。これらのモードを用いることによって、ノード数 N に対し $\log(N)$ のオーダーのオーバヘッドで効率良く共有情報の管理を行なう。

疑似フルマップでは、共有されているデータに対して invalidate または update を行なう場合、マルチキャストを使用する。マルチキャストを行な場合、これらのディレクトリを参照してメッセージを発行する。特に、階層化マップを用いる場合には、階層化放送機構を利用して部分的ブロードキャストを組み合わせてマルチキャストを実現する。

疑似フルマップ方式の中でも一番特徴的な階層化マップを使用した場合のキャッシュ動作を以下に簡単に述べる。ここで、例としてネットワークとして m-進木を考える（図2）。m-進木の葉のレベルを l 、ルートのレベル0とする。階層化マップは、それぞれのレベルについて、m bit のベクタをもつ。このベクタは木の各レベルでどのようにマルチキャストを行うのかを示すものといえる。例えばLPRAでは、メッセージを発信した葉から、ルートまで順にたどりながら、現時点のレベルのマップをみて

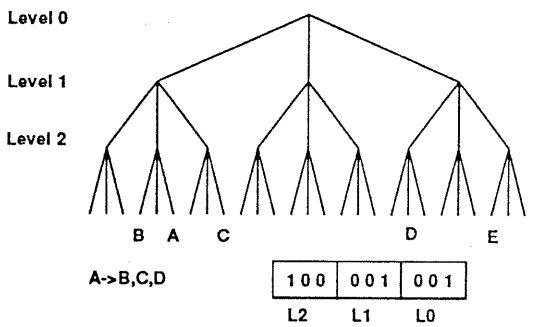

図 2: 疑似フルマップ (LPRA)

立っているビットに対応する部分木に対してブロードキャストを行う。

3.2 その他の特徴

疑似フルマップディレクトリを用いた高性能化のための機構・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

が挙げられる。

階層化放送機構の使用

階層化放送機構は、先に述べたように、マルチキャストを行なうのに用いられる他、効率的な ack の回収に用いられる。ブロードキャストに対する ack は、階層化放送機構を利用して回収することが出来る。レベル i のノードにおいてブロードキャストの要請を受けると、m 個のレベル $i+1$ の部分木に対しブロードキャストを行い、m 個の ack または dack (dummy ack) をレベル $i+1$ のノードから受けとると 1 個の ack/dack を要求元に返答する。マルチキャストに対する ack も同様に元の経路をたどり回収されてゆく。このような ack 回収システムを使用することによって、特に大規模なシステムでは、ネットワークのトラッフィクが大幅に削減される。

update 系プロトコル

update 系プロトコルを使用する場合、invalidate 系プロトコルに比べ通信量が増加する。しかし、ack のコンバイニングにより通信のコストを削減できるので、update 系プロトコルも使用が可能になる。

ページ単位ディレクトリ / ブロック単位転送

キャッシュブロックサイズがページ管理のサイズ（ページサイズ）に比べ十分小さい時には、ディレクトリ管理をページ単位で行うという方法がある。第一の理由は、ディレクトリをキャッシュブロック単位で管理する場合に比べディレクトリによるメモリのオーバヘッドが小さいことがあげられる。また、ディレクトリの操作にはコストがかかるが、ページ単位の管理によって、ディレクトリを操作する回数は減少する。

ところで、データ転送の単位もページ単位で行なうと通信のレイテンシが増加するので、データ転送はキャッシュブロックサイズで行なう。このとき、各キャッシュブロック毎に valid/invalid の情報と、ホームページ上のブロック毎にオーナーの情報を保持する。すると、ページ単位のディレクトリ管理により、メッセージが関係のないノードにまで届くことはあっても、いわゆる false sharing の状態にはならない。

ディレクトリ情報のキャッシング

ディレクトリの大きさが $\log(N)$ のオーダーで抑えられ、かつ、ページ単位ディレクトリとすると、一時に使用するディレクトリは小量で収まる。従って、ディレクトリを高速キャッシュ上にのせることにより、高速なディレクトリ操作が可能となる。

4 ディレクトリ方式の比較

表 1は、様々な視点から各種のディレクトリ方式を比較したものである。表中の記号は、

- N : ノード数
- i : ポインタ数

方式	update	memory	cache	HBF	dir. caching	page	map 切替	局所性
fullmap	○	× (N)	0	(○注)	×	○	×	○
limited(iB)	△	○ ($i \log N$)	0	△	○	○	×	×
limited(iNB)	×	○ ($i \log N$)	0	×	○	○	×	×
superset	△	○ ($2 \log N$)	0	×	○	○	△	×
coarse(i)	△	○ ($i \log N$)	0	×	○	○	△	×
chained (s)	△	○ ($\log N$)	$\log N$	×	○	○	×	×
chained (d)	△	○ ($2 \log N$)	$2 \log N$	×	○	○	×	×
pseudo (m)	○	○ ($m \log_m N$)	0	○	○	○	○	○

表 1: ディレクトリ方式の比較

- m : 木の分岐数
- (B) / (NB) : ブロードキャストを行なう／行なわない
- (s) / (d) : シングルリスト／ダブルリスト

を表す。また、表の各列は左から順に、update 系プロトコルの使用、ディレクトリによる主記憶のオーバヘッド、同じくキャシュのオーバヘッド、階層化放送機構 (HBF: Hierarchical Broadcasting Facility) の適用、ディレクトリのキャッシングの可／不可、ページ単位ディレクトリ管理の可／不可、マップ切替のサポート、局所性の利用／パーティショニングの相性を表す。

まず、update 系プロトコルの使用の欄をみる。本来、表中のディレクトリ方式は疑似フルマップを除いて invalidate を前提にしているが、ここでは無理やり update を用いた時の状況を示す。フルマップは完全な位置情報を保持しているので、update 系でも使用できる。Limited でブロードキャストをするものは、update ではトラフィックが大きくなる。Chained の場合には、update 時のレイテンシが、共有数に比例して増加してしまう。また、superset では、共有数が増えるとディレクトリ中の X の数が増え、級数的にメッセージ量が増加する。Coarse vector については、超並列環境では、プロセッサ数に比べディレクトリのビット数が少ないためブロードキャストの時と同じ問題が発生する。ブロードキャストを用いない limited は、もともと共有数が多い場合を記述できないため、update の使用は無理である。

メモリとキャッシュのオーバヘッドの欄をみると、フルマップがスケーラビリティに欠けていることが確認できる。階層化放送機構の使用の欄では、△はブロードキャストの機能を用いただけである。フルマップが注釈つきなのは、階層構造を記憶しておくために $rN/(1-r)$ ビットが必要だということである。

ディレクトリキャッシングについては、フルマップではディレクトリのサイズが大きくなるのでキャッシングは不適当となるが、他の方式では可能である。ページサイズのディレクトリ管理は、ディレクトリの形式に依存しない問題なので、全ての方式で可能となる。マップの切替を行なっているかという欄では、△は limited pointer との切替を行なっていることを示す。最後に、局所性の利用やパーティショニングを考慮にいれたディレクトリは疑似フルマップだけであることを示している。総合的に見ると、疑似フルマップは他の方式に比べ良い性質を持っていることがわかる。

5 疑似フルマップを用いた JUMP-1 上の通信プロトコル

JUMP-1[1] は、分散共有メモリ型超並列システムのプロトタイプで、疑似フルマップを用いたキャッシュ管理を行っている。JUMP-1 では、次のようなネットワークを使用する。

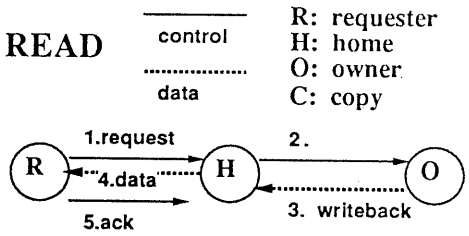

図 3: read request に対する動作

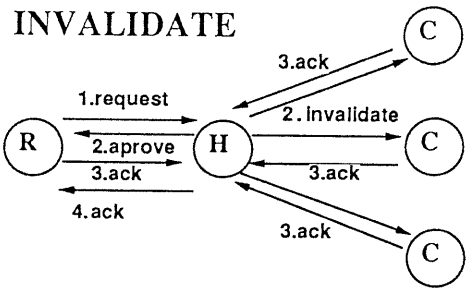

図 4: invalidate request に対する動作

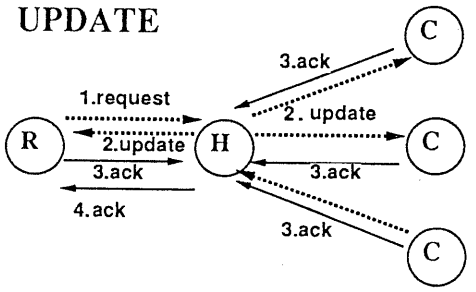

図 5: update request に対する動作

- 一対一通信ではネットワークの FIFO 性を仮定しない。
- 階層化放送機構は FIFO 性をもつ。
- 上記の両者間では、順序は保存されない。

この条件の元でデッドロックフリーとアダプティブルーティングを目指した通信プロトコルについて述べる。

コヒーレンス管理のためにネットワークに出される要求メッセージには read request, invalidate request, update request 等がある。これらのメッセージをどのように扱うかによって、通信のレイテンシや、デッドロックの回避に影響を及ぼす。基本的には、これらのリクエストをホームノードでシリアライズすることにより、デッドロックを回避する。

1. read request (図3)

read request は、まずホームノードへ送られる。該当ブロックが invalid の場合には、そのブロックの所有者のノードに対しライトバックを行うように要求する。ホームノードがデータを受けとると、要求元に転送するとともにホームノードに書き戻す。要求元はデータを受けとると、それに対する ack をホームノードに送り返すことで read の動作が終了する。

2. invalidate request (図4)

invalidate request は、まずホームノードへ送られる。ホームノードではコピーノードに対して invalidate request を送るとともに、要求元に対して read-inv request が承認されたことを示すメッセージを送る。コピーノードと要求元のノードは、それぞれホームノードに ack を返した後、ホームノードはそのブロックを invalidate して、一連の操作が完了したことを示す ack を要求元に返送する。

3. update request (図5)

update request は、まずホームノードへ送られる。ホームノードではメモリの内容を書き換えた後、コピーノードおよび要求元ノードに対して update のメッセージを送る。コピーノードと要求元のノードは、それぞれホームノードに ack を返す。これらの ack は、コンバイニングされて回収される。この後、ホームノードは ack を要求元に返送する。

6 おわりに

本稿では、分散共有メモリ上でのディレクトリ方式について総括し、従来の方式に対する疑似フルマップ方式の特徴を述べた。また、これらの方式のメモリのオーバヘッドやプロトコル切替のサポートなど様々な点から比較を行なった。さらに疑似フルマップ上で、コンシステンシを保ち、デッドロックを回避させるプロトコルについて述べた。

今後は、疑似フルマップをMBP(Memory Based Processor)[14]を利用したシステム上で実装する際の問題点について詳細な検討を行う予定である。

謝辞

本研究の遂行にあたり慶応大学の天野氏、東京工科大学の工藤氏および、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。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。

参考文献

[1] 文部省重点領域研究「超並列原理に基づく情報処理基本体系」第二回シンポジウム予稿集, 1993.

[2] Agarwal, A., R. Simoni, J. Hennessy, and M. Horowitz, "An Evaluation of Directory Schemes for Cache Coherence," in *Proceedings of the 15th Annual International Symposium on Computer Architecture*, pp. 280–289, 1988.

[3] Archibald, J. and J.-L. Baer, "Cache Coherence Protocols: Evaluation Using a Multiprocessor Simulation Model," *ACM Transactions on Computer Systems*, vol. 4, no. 4, Nov. 1986.

[4] Censier, L. M. and P. Feautrier, "A New Solution to Coherence Problems in Multicache Systems," *IEEE Trans. Comp.*, vol. 27, no. 12, pp. 1112–1118, December 1978.

[5] Chaiken, D., J. Kubiatowicz, and A. Agarwal, "LimitLESS Directories: A Scalable Cache Coherence Scheme," in *Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 224–234, 1991.

[6] Goodman, J. R., "Using Cache Memory to Reduce Processor-Memory Traffic," in *Proceedings of the 10th Annual International Symposium on Computer Architecture*, pp. 124–131, 1983.

[7] Gupta, A., W.-D. Weber, and T. Mowry, "Reducing Memory and Traffic Requirements for Scalable Directory-Based Cache Coherence Schemes," *SCALABLE SHARED MEMORY MULTIPROCESSORS*, 1991.

[8] James, D. V., A. T. Laundrie, S. Gjessing, and G. S. Sohi, "Distributed-Directory Scheme: Scalable Coherent Interface," *IEEE Computer*, pp. 74–77, June 1990.

[9] Lenoski, D., J. Laudon, K. Gharachorloo, A. Gipta, and J. Hennessy, "The Directory-Based Cache Coherence Protocol for the DASH Multiprocessor," in *Proceedings of the 17th Annual International Symposium on Computer Architecture*, pp. 148–159, 1990.

[10] Stenstrom, P., "A Survey of Cache Coherence Schemes for Multiprocessors," *IEEE Computer*, pp. 12–24, June 1990.

[11] Tang, C. K., "Cache Design in the Tightly Coupled Multiprocessor System," in *AFIPS Conference Proceedings*, vol. 45, pp. 749–753, June 1976.

[12] Thapar, M. and B. Delagi, "Distributed-Directory Scheme: Stanford Distributed-Directory Protocol," *IEEE Computer*, pp. 77–80, June 1990.

[13] 工藤智宏, 松本尚, 平木敬, 西村克信, 楊愚魯, 天野英晴, "超並列計算機でのコヒーレンス維持のためのマルチキャスト手法の検討," 情報処理学会計算機アーキテクチャ研究会報告, July 1994.

[14] 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.

[15] 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," 電子情報通信学会研究会, no. CPSY 92-26, pp. 47–55, Nov. 1992.

[16] 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.