計算機アーキテクチャ 139－2
（2000．8．3）

ハードウェア分散共有メモリにおけるスケーラブルなディレクトリ方式の定量的評価

田中清史[†] 松本尚[†,††] 平木敬[†]

プロトタイプ並列計算機上での実際のハードウェアDSMシステムの実装から、メッセージのスイッチ通過時間などの様々な所要時間が得られた。本稿では、得られた値を使用して大規模システムにおける一貫性維持（無効化）処理を扱い、階層最大共有距離ディレクトリとマルチキャストおよびコンバイニングの組み合わせによる方式をフルマップディレクトリシステムと比較する。また、ディレクトリに必要なメモリ量、およびディレクトリ構造がもたらす通信網のトラフィックを考察する。

Quantitative Evaluation of Scalable Directory Schemes in Hardware Distributed Shared Memory

KIYOFUMI TANAKA ,[†] TAKASHI MATSUMOTO [†,††] and KEI HIRAKI[†]

From the implementation of the hardware DSM system on the prototype machine, various values were obtained, such as the time required for a message to pass through a switch. In this paper, coherence processing (invalidation) on a large-scale system is considered in terms of the obtained values, and the hierarchical coarse directory with multicasting and combining is compared with the full-map directory. Moreover, we consider the size of memory required for the directories and network traffic which the structure of the directories causes.

1. はじめに

CC-NUMA(Cache Coherent Non-Uniform Memory Access)を実現する分散共有メモリ(Distributed Shared Memory: DSM)システムにおいて、共有されるメモリ空間をブロックに分割して共有情報が管理される。共有情報とは、ブロックが共有されているかどうか、更新されているかどうか、あるいはブロックのコピーを保持するプロセッサが何処にあるかを示すものである。このうち、共有しているプロセッサの位置情報を表すものをディレクトリと呼ぶ。

ハードウェアDSMにおけるディレクトリ方式は、共有プロセッサの指定方法を決定するほかに、必要ハードウェア量、一貫性維持処理の効率、ネットワークトラフィックに大きく影響する。例えば、ディレクトリを格納するのに必要なメモリ量はシステムの規模にしたがって大きくなる。システムの規模に比例するサイズを必要とするディレクトリ方式によって大規模並列システムを構築する場合、ディレクトリのためのメモリ量がデータ用のメモリ量を越えるため、このようなディレクトリ方式の適用は事実上困難である。ディレクトリサイズを小さくするために、共有数の制限、リスト構造によるディレクトリの表現、あるいはプロセッサ集合をグループ化することによる粗い指定方法などの方法が挙げられるが、一貫性維持処理の際の通信レイテンシやトラフィックの増大といった効率面での代償を払うことになる。

† 東京大学大学院理学系研究科情報科学専攻
Department of Information Science, Faculty of Science, University of Tokyo
†† 科学技術振興事業団さきがけ研究21「情報と知」領域
PRESTO, Japan Science and Technology Corporation

我々はハードウェアDSMシステムにおいて、サイズの小さいディレクトリ方式およびそれと協調動作する一貫性維持のための通信方式を提案および実装してきた。本稿ではプロトタイプにおける実装から得られた各部所要サイクルを使用して、大規模なハードウェアDSMシステムにおいて我々が提案してきたディレクトリおよび通信方式が、ハードウェア量、一貫性維持の効率、ネットワークトラフィックの観点から既存の方式よりも効率が良いことを示す。

2. ハードウェアDSMにおけるディレクトリ方式

ディレクトリ方式は、メモリブロックのコピーを持つプロセッサの位置に関して完全な情報を持つタイプと、不完全に持つタイプの2つのタイプに分類される。前者は、1) システム内のプロセッサ数が増大するにしたがってディレクトリの格納に要するメモリ量が増大、2) ディレクトリのサイズがディレクトリメモリの一回のアクセス幅を越えることに起因するアクセスオーバヘッド、あるいは3) ディレクトリがリスト構造で構成されることによるディレクトリアクセスのオーバヘッドといった問題がある。一方、ディレクトリサイズを小さく保つために不完全な情報で構成するディレクトリには2種類の方式がある。一つはコピーを持つプロセッサの数を制限する方式で、もう一つはコピーを持つプロセッサを粗く指定する方式、すなわちコピーを持つプロセッサを含むプロセッサグループを指定する方式である。これらの方式は完全情報のディレクトリよりも使用メモリ量を小さくすることが可能であるが、キャッシュの置換あるいは一貫性維持処理の際の冗長なブロードキャストあるいはマルチキャストの発生といった問題がある。

表 1　完全情報を持つディレクトリのサイズ

Directory scheme	Size	Notes
フルマップ	n	n processors in a system
LimitLESS	*limit* × *a pointer size*	overflow invokes software exec.
chained	*a pointer size*	traces a linked list
階層ビットマップ	$\sum_{k=1}^{m}(n+1)^k$	n-ary tree of height m

2.1　完全情報ディレクトリ

フルマップディレクトリ[1]、LimitLESS ディレクトリ[2]、chained ディレクトリ[3),4)]、階層ビットマップディレクトリ[5)] は共有に関して完全な情報を持つディレクトリ方式である (図 1)。フルマップディレクトリはシステム内の各プロセッサに 1 ビットを割り当て、そのプロセッサが当該メモリブロックのコピーを持つかどうかを表す (図 1(a))。この方式はシステム内のプロセッサ数に比例したビット数を使用するため、大規模システムには適さない。また、プロセッサ数が一回のメモリアクセスの幅を越えた場合にはメモリに対する多段アクセスが必要となり効率が悪い。

LimitLESS ディレクトリはメモリブロックを共有するプロセッサの数に制限を設けることによりディレクトリサイズを小さくする方式である (図 1(b))。ディレクトリには制限数と同数のポインタがあり、これによりコピーを持つプロセッサを指定する。制限数を越えてコピー生成の要求が生じた場合は、プロトコルプロセッサあるいは要素プロセッサがフルマップ方式をエミュレートする。この方式は多数のプロセッサが存在する場合にフルマップ方式に比べ使用メモリ量が小さいが、ソフトウェア実行のオーバヘッドが存在する。chained ディレクトリのサイズは一つのポインタと同サイズである (図 1(c))。この方式は共有しているプロセッサをリストをたどって指定するために、順次アクセスのオーバヘッドがある。LimitLESS ディレクトリや chained ディレクトリは共有数が大きい場合にオーバヘッドを生ずるため、無効化プロトコルとの組み合わせで使用することにより共有数を小さくおさえることが重要となる。

COMA(Cache-Only Memory Architecture) における階層ビットマップディレクトリでは、フルマップに相当する共有情報が木構造結合網の階層に部分ビットマップとして分散される (図 1(d))。メモリブロックあたりのディレクトリサイズは、高さ m の n 進木の場合に $\sum_{k=1}^{m}(n+1)^k$ となり、これはフルマップディレクトリのサイズより大きい。この方式は共有プロセッサを指定するために各階層でディレクトリアクセスが必要となるためレイテンシの増大を引き起こす。このアクセスオーバヘッドを小さくおさえるために高速なメモリを使用する必要がある。

表 1 に完全情報を持つディレクトリのサイズを示す。

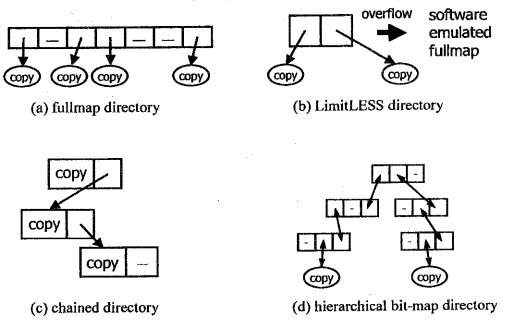

図 1　完全情報を持つディレクトリ

2.2　不完全情報ディレクトリ

不完全な共有情報を持つディレクトリとして、limited ディレクトリ[6)]、superset scheme[6)]、coarse vector scheme[7)]、疑似フルマップディレクトリ[8),9)] があり、これらは情報の不完全性によりプロセッサ数に比例しないサイズを持つ (図 2)。

limited directory は LimitLESS ディレクトリ同様、ハードウェア制御下でのコピー数に制限を設ける (図 2(a))。制限数を越えた場合は、キャッシュブロックの置換あるいは全てのプロセッサへのブロードキャストによって一貫性を維持するためオーバヘッドが大きく、無効化プロトコルとの併用により共有数をおさえることが必須である。

superset scheme では、ディレクトリが複数のポインタの合成ポインタとして表現される (図 2(b))。ポインタの各フィールド (2 ビット) は 0、1、X のいずれかであり、X は "両方" として扱われる。この方式はコピーを持つプロセッサのスーパーセットを構成し、長さ $2 \times k$ のポインタを使用することで最大 2^k 個のプロセッサを指定可能である。一貫性維持処理の際には、合成ポインタが指し得る全てのプロセッサにメッセージが送信されるため冗長な通信が存在する。

coarse vector scheme において、プロセッサはグループに分割され各グループに 1 ビット割り当てられる (図 2(c))。すなわち、グループがフルマップ方式により指定される。グループ内で一つ以上のプロセッサがコピーを持つ場合には、グループ内の全てのプロセッサが一貫性維持の対象となるため、冗長な通信が存在する。

疑似フルマップディレクトリは、結合網に埋め込まれた木構造の各階層に対応するビットマップを使用することによりサイズが $\log n$ (n はプロセッサ数) である。ディレクトリは各ブロックに割り当てられた home プロ

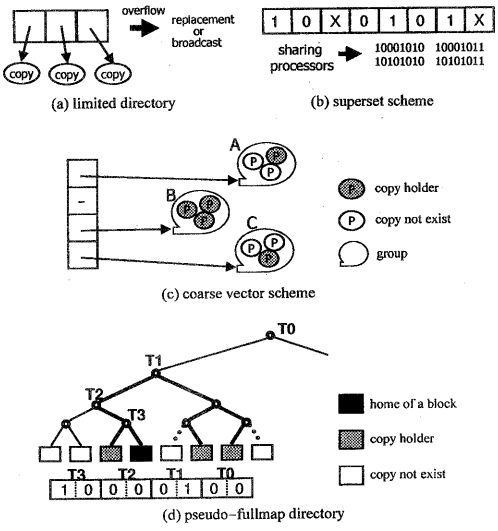

図 2　不完全情報を持つディレクトリ

表 2　不完全情報を持つディレクトリのサイズ

Directory scheme	Size	Notes
limited	$limit \times a\ pointer\ size$	overflow invokes hardware broadcast or replacement
superset scheme	$2 \times \log n$	n processors in a system
coarse vector scheme	n/k	k is the size of a group
疑似フルマップ	$n \times m$	n-ary tree of height m

セッサが保持する。図 2(d) は疑似フルマップディレクトリ方式の一つである LPRA (Local Precise Remote Approximate) 法である。情報の不完全性により、一貫性維持処理の際に図中の点線のパスにおいて冗長な通信が生ずる。

表 2 に不完全情報のディレクトリのサイズを示す。

3. 軽量ハードウェア DSM

我々は軽量ハードウェアによって効率のよい CC-NUMA を構築する方式を提案してきた[10),11)]。提案する方式では、既存のいかなるディレクトリ方式よりもサイズの小さい☆階層最大共有距離ディレクトリ (Hierarchical Coarse Directory) を用い、ディレクトリの情報の不完全性による通信の増加を補うために、階層マルチキャストおよびコンバイニング機構[8)] を組み合わせて一貫性維持処理を行う。

3.1 階層最大共有距離ディレクトリ

階層最大共有距離ディレクトリは、"最大共有距離 (maximum shared distance)" によって表される。最大共有距離はコピーを持つ全プロセッサを含む最小部分木の高さである (**図 3**)。各メモリブロックにつき home プロセッサを割り当て、home プロセッサがディレクトリを保持する。k 進木構造が埋め込まれた結合網を持つシステム内に N 台のプロセッサが存在する場合、ディレクトリのサイズは $\log_2 \log_k N$ である。

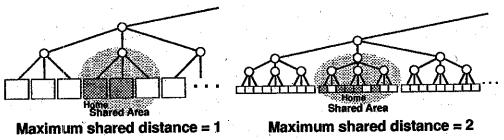

図 3　階層最大共有距離ディレクトリ

3.2 階層マルチキャスト／コンバイニング

階層最大共有距離ディレクトリでは、最大共有距離が示す部分木内の全てのプロセッサがコピーを持つとみなされるため、一貫性維持処理の際に冗長な通信が生じる。home プロセッサがこれら全ての通信を逐次的に処理するとオーバヘッドを生じるため、結合網内のスイッチノードが **図 4** のように、適宜メッセージのマルチキャストおよびコンバイニングを行うことによりトラフィックをおさえる。これにより home プロセッサは一貫性維持処理の際、1 個のメッセージの発行および受信のみを行う。

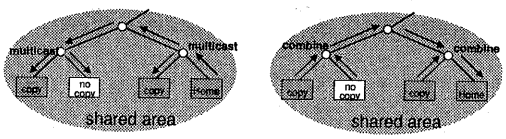

図 4　階層マルチキャスト／コンバイニング

☆ 一貫性維持をブロードキャストによって行う場合はこの限りでない。

4. ディレクトリサイズ

ディレクトリのサイズに関して、階層最大共有距離ディレクトリとフルマップディレクトリを比較する。**図 5** において、"FMD" がフルマップディレクトリ、"HCD2"、"HCD4" がそれぞれ 2 進木結合網、4 進木結合網における階層最大共有距離ディレクトリである。フルマップディレクトリがプロセッサ数に比例したサイズであるのに対し、階層最大共有距離ディレクトリは 2～3 ビットで 64 台のシステムをカバーする。より大規模なシステムにおいて、4 ビットのディレクトリで 64,000 台以上に対応可能であり、フルマップディレクトリと比較し大幅に小さいメモリ量でディレクトリが格納される。

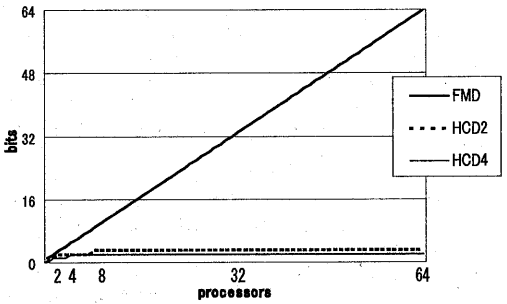

図 5　ディレクトリサイズ

5. 一貫性維持処理のための所要サイクル数

一貫性維持処理のトータル時間について、フルマップディレクトリ方式と階層最大共有距離ディレクトリ＋マルチキャスト・コンバイニング方式を比較する。

5.1 パラメータおよび所要時間

フルマップディレクトリ方式を "FMD"、階層最大共有距離ディレクトリ＋マルチキャスト・コンバイニング方式を "HCD" とし、以下のパラメータを使用する。

- *copies* はメモリブロックのコピー数であり、1 から 255 の値をとる。
- *COM* はコンバイニングの有無を表す。FMD の場合に 0、HCD の場合に 1 である。
- *TREE* は 2 進木結合網の場合に 2、4 進木結合網の場合に 4 である。
- *WIDTH* は結合網のデータ幅を表す。1、2、4(バイト) の値をとる。
- *PLENGTH* は通信パケットのバイト数である。一貫性維持の通信を対象としていることから、固定値 8 を使用する。

表 3 にシステムの各要素における所要サイクルを示す。これらの値は実機上での実装から得られたものである。表中の各要素は以下のように定義される。

- *H_MCTL_REQ* は home ノードにおいてプロセッサが無効化要求を発行してから、ノード内のメモリコントローラがネットワークインタフェースに通知するまでのサイクルである。
- *H_NETIF_OUT* は home ノードにおいてネットワークインタフェースが外部結合網に無効化メッセージを発行するのに要するサイクルであ

表 3　各要素での所要時間

Required time	Cycles
H_MCTL_REQ	7
H_NETIF_OUT	3
$SWITCH$	$4 + COM \times (PLENGTH/WIDTH + 4)$
D_NETIF_IN	$3 + PLENGTH/WIDTH$
D_MEMCTL_IN	7
D_NETIF_OUT	3
H_NETIF_IN	$3 + PLENGTH/WIDTH$
H_MEMCTL_ACK	9

る。FMD の場合はコピーを持つプロセッサ全てに対して順次的に発行するため、i 番目のターゲットプロセッサへのメッセージに対して $(PLENGTH/WIDTH) \times (i-1)$ サイクルが付加される。

- $SWITCH$ はメッセージが結合網内のスイッチノードを通過するためのサイクルである。FMD の場合は 4 である。HCD の場合はスイッチ内のコンバイニングユニットを通過するサイクル $(PLENGTH/WIDTH + 4)$ が付加される。
- D_NETIF_IN はターゲットノード内のネットワークインタフェースが外部からのメッセージを受け取ってから、ノード内のローカルバスにリクエストを発行するまでのサイクルである。
- D_MEMCTL_IN はターゲットノード内のメモリコントローラがバスリクエストをデコードし、ネットワークインタフェースに返答するのに要するサイクルである。
- D_NETIF_OUT はターゲットノード内のネットワークインタフェースが外部に acknowledgement メッセージ (以下 "Ack") を送るのに要するサイクルである。
- H_NETIF_IN は home ノードにおいてネットワークインタフェースが外部から Ack を受取り、それをローカルバスに転送するサイクルである。
- H_MEMCTL_ACK は home ノードにおいて、メモリコントローラが Ack をデコードしてネットワークインタフェースに返答するサイクルである。

5.2 算出方法

共有数 ($copies$) を変化させて、無効化処理が完了するのに要するサイクル数を算出する。データ配置の最適化の観点から、コピーを持つプロセッサを以下のように配置する。

- 各プロセッサにプロセッサ番号 (Pno) を左から順に割り当てる (図 **6**)。
- $Pno = 0$ を home プロセッサとする。
- $copies = n$ のときに、$Pno = 1, ..., n$ のプロセッサがコピーを持つ。

FMD において home プロセッサがコピーを無効化する際、トータル時間を削減するために共有しているプロセッサのうち Pno の大きい順に無効化メッセージを送る。

以上の仮定から、FMD において home プロセッサが一つのコピーを無効化するのに要するサイクル数 (RTL_{FMD}) は図 **7** (a) のように表される*。

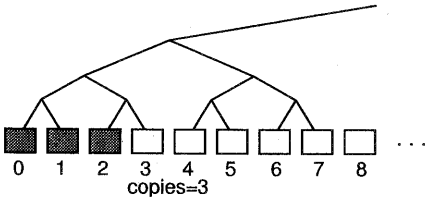

図 **6**　プロセッサ番号

ここで、$home$ は home プロセッサの番号、すなわち 0 であり、Pno はターゲットプロセッサの番号である。$height(Pno, home)$ は home プロセッサとターゲットプロセッサを含む最小部分木の高さから 1 を引いた値である。例えば、2 進木結合網において $height(10, 0)$ は 3 となる。Pno_{max} はコピーを持つプロセッサの中で最も大きいプロセッサ番号である。図 7 (a) をコピーを持つ各プロセッサに対して計算し、その最大値をトータルサイクル数の候補とする。更に home における逐次処理を考慮するために、home ノードにおいて Ack の処理 $(H\_NETIF\_IN + H\_MEMCTL\_ACK)$ がオーバーラップした場合はトータルサイクル数に付加し、最終的な結果をトータルサイクル数とする。図 **8** にフルマップディレクトリの場合のトータルサイクル数を求める例を挙げる。図において、Copy1 が Copy2 よりも遠隔にあるとする。

HCD においてスイッチノードでマルチキャストを行う場合、番号の大きいプロセッサの方向から送る (図 **9**)。この場合、転送順に従って遅延時間 $(PLENGTH/WIDTH)$ が付加される。以上から home プロセッサが一つのコピーを無効化するのに要するサイクル数 RTL_{HCD} は図 7 (b) によって計算される。ここで、$order_i$ $(0 \leq order_i < TREE)$ は通過スイッチにおけるマルチキャストの転送順を表す。コピーを持つ各プロセッサに対して計算し、その最大値をトータルサイクル数とする。

5.3 結　果

結合網が 2 進木、4 進木の場合の FMD のトータルサイクル数をそれぞれ図 **10**、図 **11** に示す。横軸がコピー数 ($copies$) で、縦軸が無効化に要するトータルサイクル数 ($cycles$) である。"FMD2(1)"、"FMD2(2)"、"FMD2(4)" は 2 進木結合網のデータ幅がそれぞれ 1、2、4 バイトであることを示す。以下、同様の記述法を使用する。これらの結果より、トータルサイクル数はコピー数に比例していることがわかる。これは、home ノードにおいて Ack を逐次的に処理していることに起因する。また、2 進木の場合と 4 進木の場合を比較すると、結合網のトポロジーはトータルサイクル数に影響しないことがいえる。

図 **12**、図 **13** に HCD の結果を示す。図より、トー

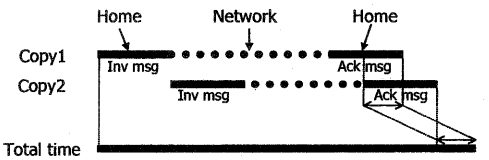

図 **8**　トータル時間 — フルマップディレクトリ

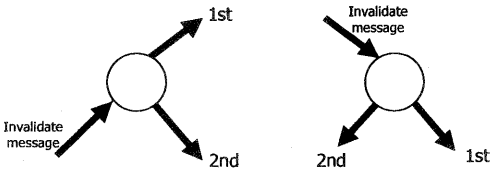

図 **9**　スイッチにおけるマルチキャスト

* ここでは、フルマップディレクトリのサイズが大きくなった場合でも、ディレクトリメモリに対して一回のアクセスで情報の読み込みが可能であると仮定する。

$$
\begin{aligned}
(a)\quad RTL_{FMD} &= H\_MCTL\_REQ + H\_NETIF\_OUT \\
&\quad + (PLENGTH/WIDTH) \times (Pno_{max} - Pno) \\
&\quad + (2 \times height(Pno, home) + 1) \times SWITCH \\
&\quad + D\_NETIF\_IN + D\_MEMCTL\_IN + D\_NETIF\_OUT \\
&\quad + (2 \times height(Pno, home) + 1) \times SWITCH \\
&\quad + H\_NETIF\_IN + H\_MEMCTL\_ACK
\end{aligned}
$$

$$
\begin{aligned}
(b)\quad RTL_{HCD} &= H\_MCTL\_REQ + H\_NETIF\_OUT \\
&\quad + \sum_{i=0}^{i<2\times height(Pno,home)+1} (SWITCH + order_i \times (PLENGTH/WIDTH)) \\
&\quad + D\_NETIF\_IN + D\_MEMCTL\_IN + D\_NETIF\_OUT \\
&\quad + (2 \times height(Pno, home) + 1) \times SWITCH \\
&\quad + H\_NETIF\_IN + H\_MEMCTL\_ACK
\end{aligned}
$$

図 7 (a) フルマップディレクトリ
(b) 階層最大共有距離ディレクトリ ＋ マルチキャスト・コンバイニング

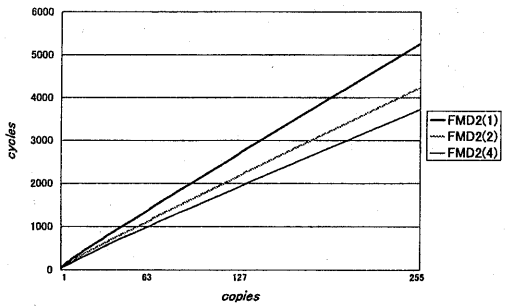

図 10 フルマップディレクトリ － 2 進木結合網

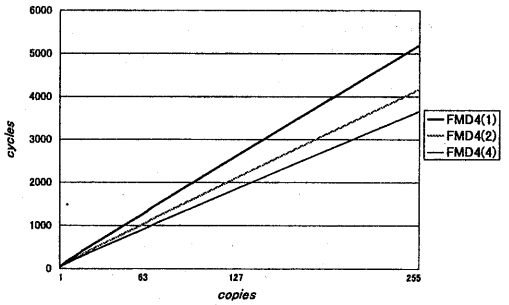

図 11 フルマップディレクトリ － 4 進木結合網

タルサイクル数はコピーを持つプロセッサ群を含む最小部分木の高さに依存することがわかる。また、2 進木と 4 進木を比較すると、大きな差が確認される。

次に FMD と HCD を比較する。**図 14** は 4 バイト幅の結合網の場合の結果である。図より全体として HCD のトータルサイクル数は FMD と比較し大幅に小さい。**図 15** にコピー数が 15 以下の場合を示す。これは図 14 を拡大したものである。2 進木の場合は、コピー数が 9 を越えたときに FMD のサイクル数が大きくなる。4 進木の場合はコピー数が 5 を越えたときに FMD のサイクル数が大きくなる。コピー数が小さいとき HCD のサイクル数のほうが大きい場合があるが、最大で 1.6 倍程度であり、ディレクトリ情報を不完全にした代償は大きくない。

以上の結果から、ディレクトリの情報が不完全な場合でも、マルチキャストおよびコンバイニングによって home プロセッサにおける逐次処理を削除することにより、効率良く一貫性維持処理が行われることがいえる。

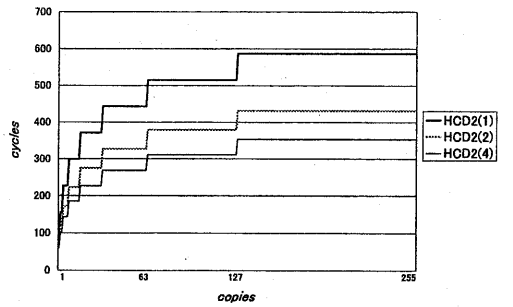

図 12 階層最大共有距離ディレクトリ ＋ マルチキャスト・コンバイニング － 2 進木結合網

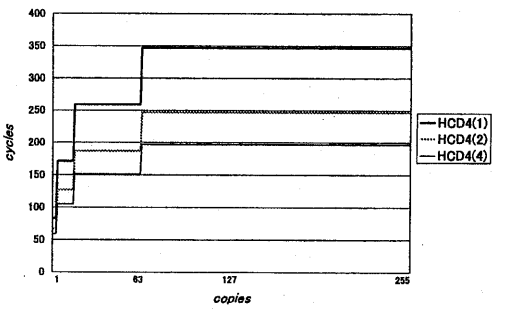

図 13 階層最大共有距離ディレクトリ ＋ マルチキャスト・コンバイニング － 4 進木結合網

6. ネットワークトラフィック

ネットワークトラフィックとして前節の一貫性維持処理の状況下での、結合網内全てのスイッチ間のパケット数を扱う。2 進木、4 進木結合網における総スイッチ間パケット数をそれぞれ図 **16**、図 **17** に示す。図において、“HCD2–”、“HCD4–” は階層最大共有距離ディレクトリ方式において、結合網でマルチキャストおよびコンバイニングを行わないものである。図より、FMD はコピー数に対して比例オーダーよりも多くのスイッチ間パケットを生成している。一方、HCD は対数オーダーで生成している。

7. おわりに

本稿ではプロトタイプ計算機の実装から得られた各要素における所要時間を使用し、我々が提案してきた階

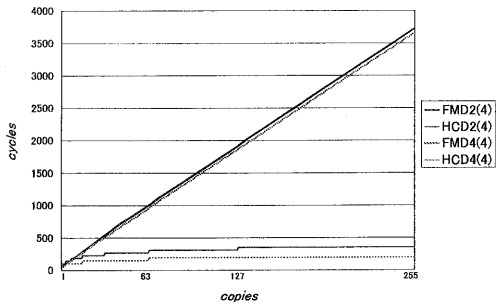

図 14 フルマップ vs. 階層最大共有距離ディレクトリ + マルチキャスト・コンバイニング

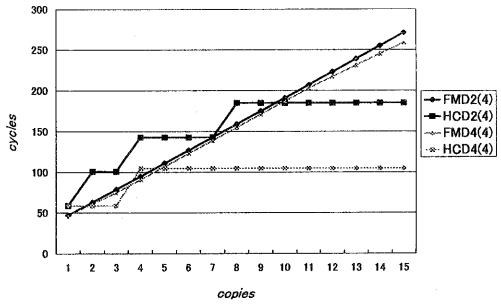

図 15 フルマップ vs. 階層最大共有距離ディレクトリ + マルチキャスト・コンバイニング (拡大図)

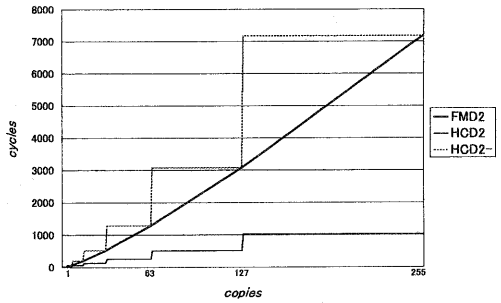

図 16 総パケット数 − 2進木結合網

層最大共有距離ディレクトリとメッセージのマルチキャストおよびコンバイニングを組み合わせた方式と、フルマップディレクトリによる方式を比較した。大規模なシステムにおいて、我々の方式がディレクトリのためのメモリ量、一貫性維持処理のトータル時間、およびネットワークトラフィックの3項目全てにおいてフルマップディレクトリ方式よりも優位な結果を示した。特にシステム規模あるいはコピー数が大きいときにその差は顕著となった。このことから、特にコピー数が大きくなる傾向にある更新型 (update) プロトコルにおいて我々の方式は有効であるといえる。

謝辞

本研究は通商産業省 RWC プロジェクトの一環として行われた。

参 考 文 献

1) L.M.Censier and P.Feautrier: A New Solution to Coherence Problems in Multicache Systems. IEEE Trans. Computers, pp.1112–1118, (Dec 1978).

2) D.Chaiken,J.Kubiatowicz and A.Agarwal: LimitLESS Directories: A Scalable Cache Coherence Scheme. Proc. of Int'l Conf. ASPLOS, pp.224–234, (Apr 1991).

3) D.James,A.T.Laundrie,S.Gjessing and G.S. Sohi: Distibuted–Directory Scheme: Scalable Coherent Interface. Computer, Vol.23, No.6, pp.74–77, (Jun 1990).

4) M.Thapar and B.Delagi: Distributed–Directory Scheme: Stanford Distributed Directory Protocol. Computer, Vol.23, No.6, pp.78–80, (Jun 1990).

5) E.Hagersten,A.Landin and S.Haridi: DDM – A Cache-Only Memory Architecture. Computer, Vol.25, No.9, pp.44–54, (Sep 1992).

6) A.Agarwal,R.Simoni,J.Hennessy,M.Horowitz: An Evaluation of Directory Schemes for Cache Coherence. Proc. of ISCA, pp.280–289, (Jun 1988).

7) A.Gupta,W.Weber and T.Mowry: Reducing Memory and Traffic Requirements for Scalable Directory-Based Cache Coherence Schemes. Proc. of ICPP, pp.I-312–321, (Aug 1990).

8) 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 CPSY, Vo.92, No.173, pp.47–55 (Aug 1992).

9) T.Matsumoto, et al.: Distributed Shared Memory Architecture for JUMP-1 a General-Purpose MPP Prototype. Proc. of I-SPAN, pp.131–137, (Jun 1996).

10) K.Tanaka,T.Matsumoto and K.Hiraki: Lightweight Hardware Distributed Shared Memory Supported by Generalized Combining. Proc. of the 5th Int'l Symp. HPCA, pp.90–99, (Jan 1999).

11) 田中, 松本, 平木: 軽いハードウェアによる分散共有メモリ機構. 情報処理学会論文誌, Vol.40, No.5, pp.2025–2036, (May 1999).

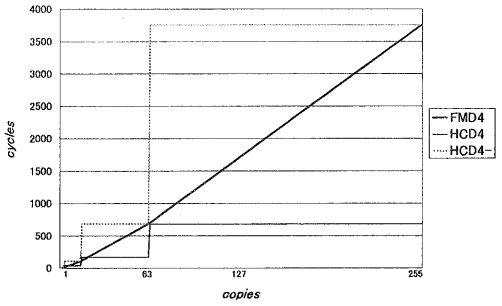

図 17 総パケット数 − 4進木結合網