システムＬＳＩ設計技術　107－11
(2002. 11. 28)

適応型ディレクトリによるキャッシュ一貫性管理方式

萩原　利英[†]　田中　清史[†,††]

† 北陸先端科学技術大学院大学 情報科学研究科　〒923-1292 石川県能美郡辰口町旭台 1-1
†† 科学技術振興事業団，さきがけ研究 21,「機能と構成」領域
E-mail: †{t-hagiwa,kiyofumi}@jaist.ac.jp

あらまし　分散共有メモリにおいてキャッシュ一貫性維持管理に使用されるディレクトリは，その方式の選択が必要ハードウェア量，一貫性維持処理の効率，およびネットワークトラフィックに影響を与える．本稿ではこの 3 項目において従来の方式の問題点を解決するディレクトリ方式として adaptive hierarchical coarse directory を提案し，他の方式と比較することにより有効性を示す．
キーワード　分散共有メモリ，キャッシュ一貫性維持，ディレクトリ方式

Cache Coherence Management by Adaptive Directory

Toshihide HAGIWARA[†] and Kiyofumi TANAKA[†,††]

† School of Information Science, Japan Advanced Institute of Science and Technology　1-1, Asahidai, Tatsunokuchi, Ishikawa, 923-1292, Japan
†† "Information and Systems", PRESTO, Japan Science and Technology Corporation(JST)
E-mail: †{t-hagiwa,kiyofumi}@jaist.ac.jp

Abstract　In distributed shared memory systems, the kind of directory scheme for managing the cache coherence affects the amount of hardware, efficiency of coherence processing, and network traffic. In this paper, we propose an directory scheme, "adaptive hierarchical coarse directory", which solves the problems existing schemes have on the three factors, and evaluate the effectiveness in comparison with the others.
Key words　distributed shared memory, cache coherence, directory scheme

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) システムでは，共有情報の管理はキャッシュライン，ページ，あるいはオブジェクトといったブロック単位で行われる．共有情報に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，あるいは同じ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の位置情報などがあり，このうちブロックを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ため，方式の選択が必要ハードウェア量，一貫性維持処理の効率，およびネットワークトラフィックに大きな影響を与える．例えばディレクトリを格納するのに必要なメモリ量がシステム内のプロセッサ数に比例して増加する場合，大規模システムでは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．従って，このようなディレクトリ方式を大規模システムに適用すること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分けすることによるグループ単位の位置指定などの方法が提案されているが，共有数が多い場合に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．また，システムの階層構造を利用したマルチキャストを行うことにより効率の良い通信を実現する方式があるが，コピーの不規則な配置によりトラフィックが増大する傾向がある．

本稿では，共有プロセッサ同士の階層距離により決定されるプロセッサグループを一貫性管理の単位とし，共有プロセッサ数およびアプリケーション実行時のデータの局所性に適応するディレクトリ方式とその動的な構築アルゴリズムを提案する．さらにシミュレーションにより他の方式と比較し，その有効性を示す．

2. ディレクトリ方式

ディレクトリ方式は，共有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．前者ではシステム内のプロセッサ数あるいはコピー数が増大した場合，

- ディレクトリの格納に要するメモリ量の増大
- ディレクトリのサイズが過大となり，ディレクトリメモリの一回のアクセス幅（コントローラとメモリ間のデータ幅）を越えることに起因するアクセスオーバヘッドの増大
- ディレクトリがリスト構造など逐次アクセス構造で構成される場合，ディレクトリの逐次アクセスのオーバヘッドの増大

が問題となる．一方後者は，コピーを持つプロセッサを正確に記憶する数を制限するか，あるいはコピーを持つプロセッサを粗く指定する，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ことによりディレクトリサイズを小さく保つ．このような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，共有数制限による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．

2.1 完全情報ディレクトリ

共有に関して完全な情報を持つディレクトリ方式として，フルマップディレクトリ[1]，LimitLESS ディレクトリ[2]，チェインドディレクトリ[3],[4]，階層ビットマップディレクトリ[5]がある．

フルマップディレクトリは Stanford 大の DASH[6] において採用された方式であり，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 0，1 の値で表す．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 MIT Alewife[7] において採用された方式であり，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．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のソフトウェア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ディレクトリ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．

チェインドディレクトリは Scalable Coherent Interface（SCI[8] に適用されている方式であり，ディレクトリはコピーが存在するプロセッサ間で分散され，それらがリストで結合される．したがってディレクトリサイズはコピー数と一つのポインタのサイズの積となる．この方式は共有しているプロセッサをリストを辿って指定するために逐次アクセスのオーバヘッドがある．LimitLESS ディレクトリやチェインド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抑えることが重要となる．

COMA（Cache-Only Memory Architecture）[5] 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各階層ノードに部分ビットマップとして分散される．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にメッセージを送る際に，各階層ノード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．

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[9]，superset scheme[9]，Cenju-4 のビットパターン構造[10]，coarse vector scheme[11]，疑似フルマップディレクトリ[12],[13]，hierarchical coarse directory[14] などがあり，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．

limited ディレクトリは 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．制限数を越えた場合は，コピーの置換(注2)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2つのポインタの合成ポインタとして表現される．ポインタの各フィールド（2ビット）は 0，1，X のいずれかを表し，X は “両方” として扱われる．すなわち，X の場所以外のビットフィールドに関してプロセッサ番号とディレクトリ値が一致するときに，そのプロセッサはコピーを持つとみなされる．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発生する．

Cenju-4 のディレクトリにおけるビットパターン構造では，プロセッサ番号をフィールド分けし，各フィールドをデコードして得られるビットマップを生成し，これをプロセッサ間で論理和することによってディレクトリを表現する．コピーの位置はこの合成ビットマップから抽出される全てのパターンのプロセッサ番号で指定される．この方式は，プロセッサ番号を1ビット毎のフィールド分けでデコードした場合には superset scheme と同じものとなる．複数ビットのフィールドを用いることにより superset scheme よりも位置指定の正確さは増すが，

(注2)：コピーの置換を行う方式は，完全情報を持つディレクトリとみなされる．

ディレクトリのサイズが増大することになる．

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ビットが割り当てられる．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用意し，各ビットマップを3種類の規則のいずれかに従って対応する階層に適用する．k 進木の場合にサイズが $k \times \log_k n$（n はプロセッサ数）となる．階層ビットマップと異なり，ディレクトリは管理単位毎に割り当てられた home プロセッサが保持する．データ配置に局所性が無い場合にはトラフィックが増大する可能性がある．

階層構造を最大限利用してディレクトリサイズを抑えることを目的とした hierarchical coarse directory（HCD）は，階層最大共有距離のみでディレクトリ情報を表す．ここで階層最大共有距離とは，ブロックの home とコピーを持つ最も遠いプロセッサとの間の階層距離，すなわち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．このような階層情報により，k 進木構造上で n 個のプロセッサが存在する場合，ディレクトリサイズは $\log_2 \log_k n$ となる．例えば4進木結合網上で3ビットのディレクトリを用意することで，65536 プロセッサのシステムに対応可能である．

HCD の構成を図1に表す．灰色の葉はメモリブロックのコピーを持つプロセッサを表し，網掛けの部分はコピーを持つ全てのプロセッサを含む最小部分木（共有空間）を表す．図中の(a) では共有空間が高さ1の木であるため階層最大共有距離は1であり，(b) では同様に2となる．

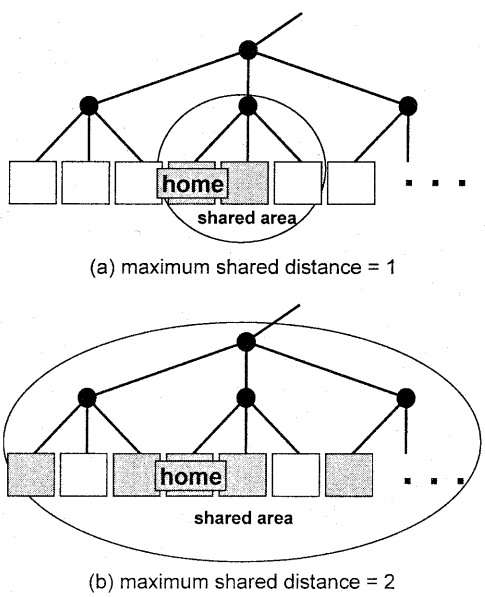

図1 Hierarchical coarse directory.

Cenju-4 のビットパターン構造，疑似フルマップおよび HCD は，情報の不完全性により発生する冗長な通信メッセージを，結合網内のスイッチノードでマルチキャストとコンバイニング [12] を適用することにより削減することを前提としている．

3. Adaptive Hierarchical Coarse Directory (AHCD)

前節の HCD において，一貫性維持処理の際の生成パケット数はコピー数にかかわらず home と最も遠い共有プロセッサとの階層距離によって決定されるため，コピーがまばらで不規則な位置に配置された場合は他の方式と比較してパケット数が増大する可能性が高い．HCD は通信網の木構造の階層性を利用する点で疑似フルマップディレクトリに類似している．一貫性維持の処理時間に関して，マルチキャストおよびコンバイニングを利用することにより両ディレクトリは同等の効率をもたらすが，サイズに関しては情報がより不完全である HCD，生成パケット数に関しては各階層に対応するビットマップを持つ疑似フルマップディレクトリが良い性質を持つ．しかし疑似フルマップディレクトリにおいて，正確な情報（あるいは，ある程度正確な情報）は root ノードに至るパスかそれ以外かのどちらかにのみ適用可能であり，システム内の2つ以上の系統に正確な情報を持たせることはできない(注3)．また HCD と同様，パケット数がコピー数に依存しない．

HCD および疑似フルマップディレクトリともにトラフィックがコピーの配置に大きく影響される．HCD では共有数2，疑似フルマップディレクトリでは n 進木の場合に共有数 n でシステムのほぼ全体にわたってパケットが転送される可能性がある．コピーの位置は実行プログラムのデータ配置およびアルゴリズムやオペレーティングシステムのスケジューリングによって決定される．これらに関して局所性を利用する最適化を行うことが有効であるが，これが困難である場合においてトラフィックがコピー数のみに依存し，配置に依らないディレクトリ方式が望ましい．

文献 [9] によれば，無効化処理において無効化の対象となるキャッシュの数は3以下であることがほとんどである．2. 節で述べたコピーの制限数を設定した limited ディレクトリはその性質に依存している．ただし更新化プロトコルを使用する場合などコピー数が増大する場合には，limited ディレクトリでは制限数を越えてブロードキャストが多発しネットワーク上のパケット数が著しく増加するため効率が悪い．

本節では limited ディレクトリを HCD の方式を利用して拡張し，コピーが不規則に配置されている場合にも軽いトラフィックを実現するディレクトリ方式を提案する．

結合網として木構造が埋め込み可能なものを仮定する．limited ディレクトリの制限数に相当する値を N とする．ディレクトリは N 個のポインタおよび，各ポインタ対応する階層最

(注3)：LPRA 法，LARP 法がこれにあたり，疑似フルマップディレクトリにはこの他に，各階層でターゲット方向同士の論理和をとったビットマップを使用する SM 法がある．

大共有距離を持つ．homeを指すポインタは暗に指定されるためディレクトリには含まないが，homeに対応する階層最大共有距離は含む．各ポインタは“疑似home”として振る舞い，対応する階層最大共有距離により“部分共有空間（partial shared area）”を形成する．

プロセッサPがhomeに対してコピー生成の要求を行った場合，homeはPがいずれかの部分共有空間に含まれるようにディレクトリの再構築を行う．再構築は以下のアルゴリズムに従う．

```
if （Pがいずれかの部分共有空間に含まれる場合）{
    ・再構築なし
}
else if （未使用ポインタが存在する場合）{
    ・Pを疑似ホームとしてポインタに設定
    ・対応する階層最大共有距離を0に設定
}
else {
    ・home，疑似home，Pの間で全対全の階層距離
      計算
    ・最も小さい距離を持つ組み（3つ以上可）を1
      つ選択し，それらを併合
    ・併合された中で1つを疑似homeに設定し，階
      層最大共有距離を更新
    ・併合にPが含まれない場合はPを新たに疑似
      homeに設定し，対応する階層最大共有距離を
      0に設定
}
```

この方式はコピー数がN以下のときはポインタによりコピーを持つプロセッサを正確に指定可能であり，Nより大きい場合には複数の部分共有空間により共有空間を形成する．このディレクトリ方式を**Adaptive Hierarchical Coarse Directory（AHCD）**と呼ぶ．図2にAHCDの例を示す．図は$N=2$の場合であり，ポインタは4ビット，最大共有距離は2ビットであり，homeと2つの疑似homeにより，3つの部分共有空間を形成している．

AHCDのサイズはポインタの数に依存する．疑似フルマップディレクトリのサイズは高さmのn進木のときに$n \times m$であるが，これはnが2あるいは4のときに2つのポインタのサイズと同じである．このことから，ポインタの数を2にすることにより疑似フルマップディレクトリと同程度のサイズに抑えることが可能である．ただしこの場合，3つの最大共有距離を保持するビット数が加わる(注4)．例えば65536プロセッサ，4進木結合網システムにおいて，疑似フルマップが$4 \times \log_4 65536 = 32$ビットであるのに対し，$N=2$のHCDは$2 \times \log_2 65536 + 3 \times \log_2 \log_4 65536 = 41$ビットである．

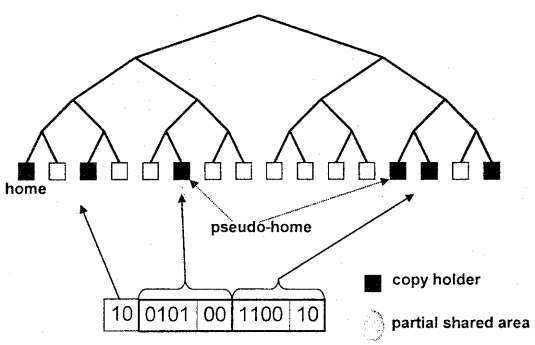

図2　Adaptive hierarchical coarse directory.

Nが小さい値のHCDは，通信パケットにディレクトリ値を含めることによりマルチキャストおよびコンバイニングとの組合わせが現実的に可能である．これにより，HCDと同程度の処理時間で，かつパケット数のより少ない一貫性維持処理が可能となる．

HCDはコピーの配置に関してhomeを基準とした局所性がある場合には効率が良いが，コピーがまばらで不規則な位置に配置された場合はパケット数増大を引き起こす．これに対し，AHCDは複数の疑似homeを設定可能とすることにより分散したデータ局所性に対応し，パケット数増大を緩和する方式である．無効化プロトコルの使用によりコピー数が少ない状況下ではlimitedディレクトリとして振る舞い，更新化プロトコルの使用によりコピー数が増大した場合は複数の局所的な部分共有空間を形成することが可能である．

4. スケーラビリティの評価

本節においてディレクトリに必要なメモリ量，一貫性維持処理の際の所要時間，および通信網のトラフィックに関してAHCDを他のディレクトリ方式と比較し，大規模システム上でのスケーラビリティを評価する．

4.1 ディレクトリサイズ

ディレクトリのサイズに関して，フルマップディレクトリ，チェインドディレクトリ，疑似フルマップ，HCDおよびAHCDを比較する．プロセッサ台数が2から64で，4進木結合網におけるメモリブロックあたりのディレクトリのサイズを図3に示す．

図において，横軸がプロセッサ台数，縦軸がディレクトリのビット数であり，“FMD”がフルマップディレクトリ，“CHD”がチェインドディレクトリ，“PFD”が疑似フルマップ，“HCD”がHCD，“AHCD1”と“AHCD2”がそれぞれポインタ数1，2のAHCDである．なお，チェインドディレクトリはコピー数に依存したサイズとなるため，図にはコピーが存在しない場合のサイズを示す．実際にはオリジナルのブロックを含めてn個のコピーが存在する場合はn倍のサイズになる．図より，フルマップディレクトリはプロセッサ数に比例したビット数となり，チェインドディレクトリはプロセッサ番号を示すポインタのサイズとなる．一方，疑似フルマップ，HCDおよびAHCDは対数のオーダーで増大するが，HCDが最も小さく2ビットで64台のシステムに対応する．疑似フルマップはとAHCD1はほぼ

(注4)：階層最大共有距離を格納するための最小ビット数は木構造の次数およびNの大きさによって決定される．

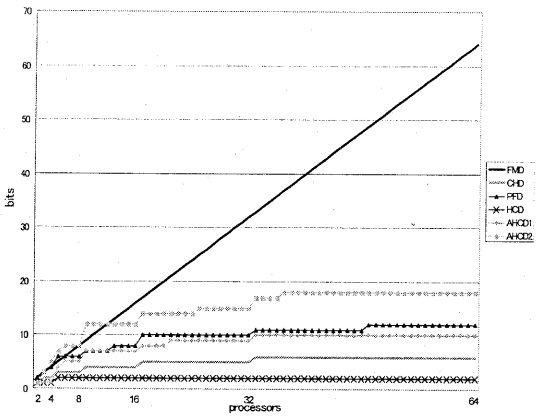

図 3　ディレクトリサイズ

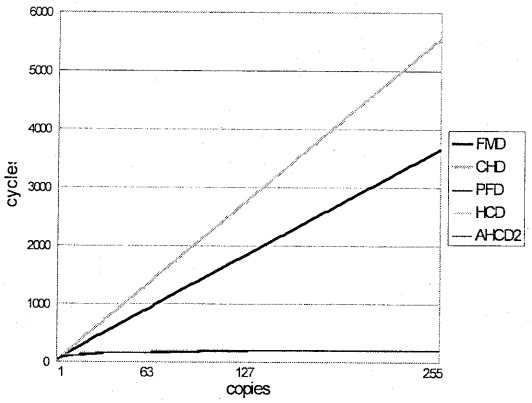

図 4　無効化処理の総サイクル数

同じサイズであり，AHCD2 がポインタ 2 個に付随する階層最大共有距離のサイズだけ大きい．しかし，3. 節で述べたように，41 ビットで 65536 プロセッサシステムに対応可能であることから現実的なサイズである．

4.2　一貫性維持処理のための所要時間

一貫性維持処理（無効化処理）のトータル時間について，フルマップディレクトリ，チェインドディレクトリ，疑似フルマップ，HCD およびポインタ数が 2 の AHCD を比較する．疑似フルマップ，HCD，AHCD はマルチキャストおよびコンバイニング方式を併用する．

プロセッシングノードは，要素プロセッサ，主記憶，メモリコントローラ，ネットワークインタフェースで構成される．ノード間の結合網は木構造であり，中間ノードはマルチキャスト，コンバイニング機構を実現するクロスバスイッチである．これらの各要素におけるパケット発行，通過，受信に要するサイクル数は実機と同じ値を使用する [15]．

コピー数を変化させて，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．
- $Pno = 0$ 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．

フルマップディレクトリにおいて 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．チェインドディレクトリでは home プロセッサは右隣りのプロセッサに無効化メッセージを送り，それが右隣りに Pno_{max} まで伝搬される．すなわち，最左ノードから最右ノードに向かって隣同士を結ぶようにリストが構成されているとする．Pno_{max} のプロセッサは最後に home に Ack を送る．疑似フルマップ，HCD，AHCD 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を含む方向から送る．この場合，転送順に従って遅延時間が付加される．なお，疑似フルマップでは上記のコピー配置のときに最もサイクル数の小さくなる LARP 法を使用した．

結合網が 4 進木のときの総サイクル数を図 4 に示す．横軸がコピー数，縦軸が総サイクル数である．CHD と FMD はコピー数に比例した総サイクル数となっているが，FMD の場合は無効化メッセージの発行と Ack の受信がオーバーラップするのに対し，CHD の場合は完全に逐次化されているために総サイクル数が大きくなっている．疑似フルマップ，HCD，AHCD はほぼ同じサイクル数となった（図中で重なっている）．

4.3　結合網内トラフィック

コピーの無効化処理の際に発生する結合網内の全てのスイッチ間のパケット数を比較する．SPLASH-2 [17] の LU 分解，FFT，Radix ソートの 3 つのプログラムに対して，augment 方式のシミュレータである ABSS [16] を使用してメモリアクセストレースを生成し，これに対するパケット数を求める．結合網は 4 進木を使用し，AHCD のポインタ数を 2 とし，実行するプロセッサ台数を 64 とした．

フルマップディレクトリ，HCD，AHCD について，LU，FFT，Radix の結果をそれぞれ図 5，6，7 に示す．ここで HCD と AHCD にはマルチキャストおよびコンバイニングを適用した．図中の横軸は個々の無効化処理を表し，縦軸はそのときのスイッチ間総パケット数である．可視化のため，LU に関して連続する 700 回の平均値を 1 ドットで表示し，同様に FFT，Radix に関して 700 回，7000 回で 1 ドットとしている．全てのプログラムに関して，AHCD のパケット数は HCD に対して大幅に削減され，完全情報すなわち無駄なメッセージを発生させない FFT とほぼ同等の値となっている．

5.　ま　と　め

本稿で提案した AHCD は，HCD におけるコピーの不規則な配置に起因するトラフィック増大を緩和する方式である．無効化プロトコルの使用によりコピー数が少ない状況下では limited ディレクトリとして振る舞い，更新化プロトコルの使用により

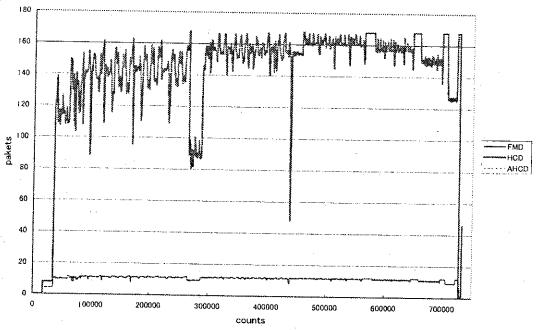

図 5　総パケット数 – LU –

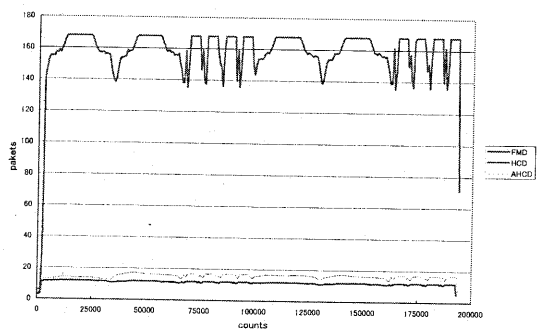

図 6　総パケット数 – FFT –

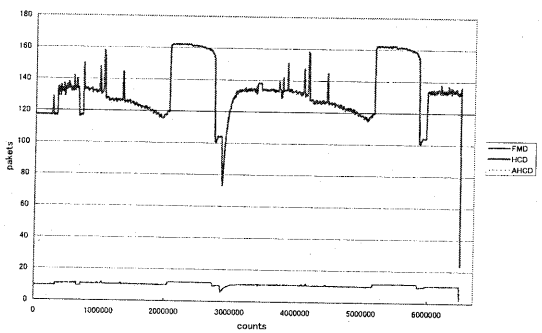

図 7　総パケット数 – Radix –

コピー数が増大した場合は複数の局所的な部分共有空間を形成することが可能である．特に汎用環境においてプロセスがシステム内に不規則に分割されてスケジューリングされた場合に，不完全情報に起因するトラフィック増大による他のプロセスへの阻害を防止することが可能である．

文　献

[1] L.M.Censier and P.Feautrier, "A New Solution to Coherence Problems in Multicache Systems", IEEE Trans. on Computers, Vol.C-27(12), 1978.

[2] D.Chaiken, J.Kubiatowicz and A. Agarwal, "LimitLESS Directories: A Scalable Cache Coherence Scheme", Proc. of ASPLOS, pp.224–234, 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, 1990.

[4] M.Thapar and B.Delagi, "Distributed-Directory Scheme: Stanford Distributed Directory Protocol", Computer, Vol.23, No.6, pp.78–80, 1990.

[5] E.Hagersten, A.Landin and S. Haridi, "DDM–A Cache-Only Memory Architecture", Computer, Vol.25, No.9, pp.44–54, 1992.

[6] "D.Lenoski, J.Laudon, K.Gharachorloo, A.Gupta and J.Hennessy, "The Directory-Based Cache Coherence Protocol for DASH Multiprocessor", Proc. of ISCA, pp.148–159, 1990.

[7] A.Agarwal, R.Bianchini, D.Chaiken and K.L.Johnson, "The MIT Alewife Machine: Architecture and Performance", Proc. of ISCA, pp.2–13, 1995.

[8] IEEE, "IEEE Standard for Scalable Coherent Interface (SCI) 1596–1992", 1993.

[9] A.Agarwal, R.Simoni, J.Hennessy and M. Horowitz: An Evaluation of Directory Schemes for Cache Coherence. ISCA, 1988.

[10] 細見 岳生, 加納 健, 中村 真章, 広瀬 哲也, 中田 登志之, "並列計算機 Cenju-4 の分散共有メモリ機構", 情報処理学会論文誌, Vol.41, No.5, pp.1400–1409, 2000.

[11] 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, 1990.

[12] 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, Vol.92, No.173, pp.47–55, 1992.

[13] T.Matsumoto, K.Nishimura, T.Kudoh, K.Hiraki, H.Amano and H.Tanaka, "Distributed Shared Memory Architecture for JUMP-1 a General-Purpose MPP Prototype", Proc. of I-SPAN, pp.131–137, 1996.

[14] K.Tanaka, T.Matsumoto and K.Hiraki, "Lightweight Hardware Distributed Shared Memory Supported by Generalized Combining", Proc. of HPCA, pp.90–99, 1999.

[15] K.Tanaka, T.Matsumoto and K.Hiraki, "On Scalability Issue of Directory Schemes of Hardware Distributed Shared Memory", The Nineth Workshop on Scalable Shared Memory Multiprocessors, 2000.

[16] D.Sunada, D.glascolasco and M.Flynn, "ABSS v2.0: SPARC Simulator", Technical Report, CSL-TR-98-755, Stanford Univ., 1998.

[17] S.C.Woo, M.Ohara, E.Torrie, J.P.Singh and A.Gupta, "The SPLASH-2 Programs: Characterization and Methodological Considerations", Proc. of ISCA, pp.24–36, 1995.