計算機アーキテクチャ 152－5
（２００３．３．１０）

分散共有メモリにおける適応型ディレクトリ方式の定量的評価

萩原利英[†] 田中清史[†,††]

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．

Quantitative Evaluation of Adaptive Directory in Distributed Shared Memory

Toshihide Hagiwara[†] and Kiyofumi Tanaka[†,††]

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.

1. はじめに

CC-NUMA 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の位置情報などがあり，このうち共有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，(1) ディレクトリの格納に要するメモリ量の増大，(2) 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，(3) 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，などが問題となる．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

† 北陸先端科学技術大学院大学 情報科学研究科
School of Information Science, Japan Advanced Institute of Science and Technology
†† 科学技術振興事業団，さきがけ研究 21,「機能と構成」領域 “Information and Systems”, PRESTO, Japan Science and Technology Corporation(JST)

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)]，などがある．

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 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コピーを持つプロセッサをリストで結合する方式であり，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．

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[5)]，coarse vector scheme[6)]，疑似フルマップディレクトリ[7),8)]，hierarchical coarse directory[9)] 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☆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．

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．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．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．

疑似フルマップおよび HCD は，情報の不完全性に

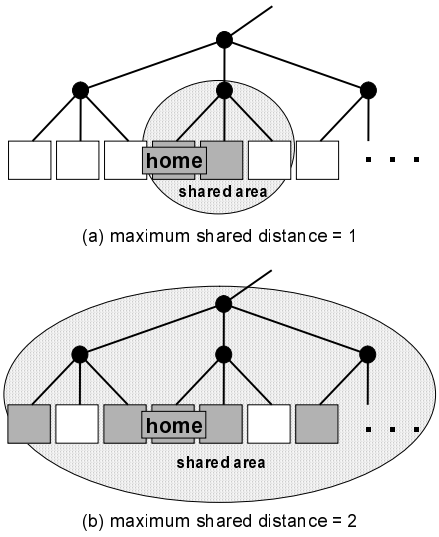

図 1 Hierarchical coarse directory.

☆ 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．

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[7]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，一貫性維持処理時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存在する場合は他の方式と比較してパケット数が増大する．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 home と部分木の root 間の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☆．また HCD と同様，パケット数がコピー数に依存せず，コピーの配置に大きく影響される． 本節では limited ディレクトリを HCD 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．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．

N の値が小さい AHCD 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

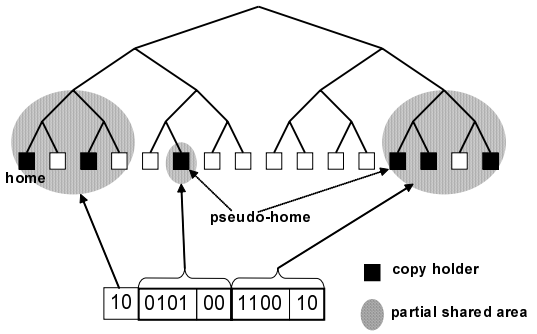

図 2 Adaptive hierarchical coarse directory.

☆ 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．

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 を他の方式と比較し，評価を行う．

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 256 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，“AHCD(1)” と “AHCD(2)” 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，FMD はプロセッサ数に比例したビット数となり，CHD はプロセッサ番号を示すポインタのサイズとなる．一方，PFD，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 256 台のシステムで 3 ビットである．PFD と AHCD(1) はほぼ同じサイズであり，AHCD(2) 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（無効化処理と更新処理）の所要時間について SPLASH-2[11] から FFT，OCEAN の 2 つのベンチマークを使用して評価する．既存のディレクトリ方式からフルマップディレクトリ，HCD，及び疑似フルマップを選択し，ポインタ数 1 個と 2 個の AHCD と比較する．

並列アプリケーションの実行は，augment 方式の並列シミュレータである ABSS[10] 上で行う． メモリブロックサイズを 32B，プロセッサ共有台数を 256 に設定し，プロセッサキャッシュは無限大サイズでライトバック方式でシミュレートし，メモリアクセストレースを生成させる．次にメモリトレースを使用してディレクトリシミュレーションを行う．結合網は 4 進木とし，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[9]．

FFT，OCEAN 実行時の無効化処理のサイクル数を**図 4**，**図 5** に，更新処理のサイクル数を**図 6**，**図 7** に示す．また 1 回の一貫性維持処理時における平均コピー数を**表 1** に示す．図中の横軸は個々の無効化や更新処理を表し，縦軸はそのときの総サイクル数である．可視化のため，無効化処理は FFT，OCEAN に関してそれぞれ連続する 400 回，4800 回の平均値を 1 ドットで表示し，同様に更新処理に関して 1400 回，80000 回で 1 ドットとしている．また，凡例において FMD，HCD，AHCD(1)，AHCD(2) は 4.1 節と同様であり，“SM”，“LARP”，“LPRA” は疑似フルマッ

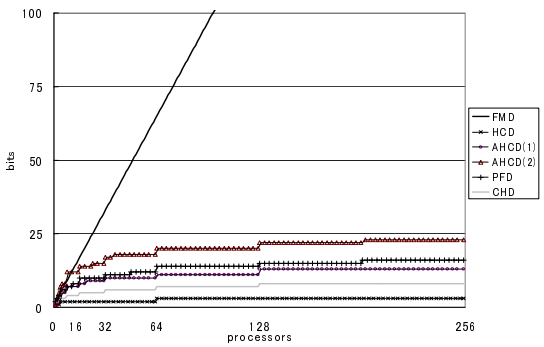

図 3 ディレクトリサイズ

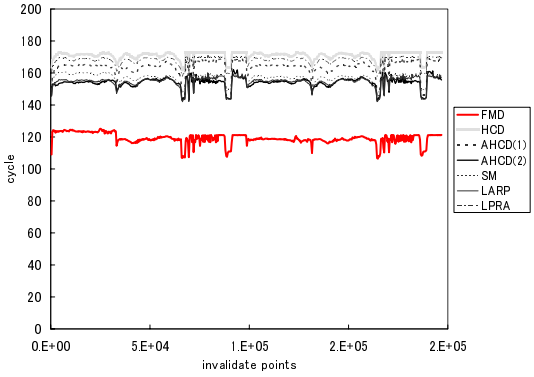

図 4 無効化処理におけるサイクル数 – FFT –

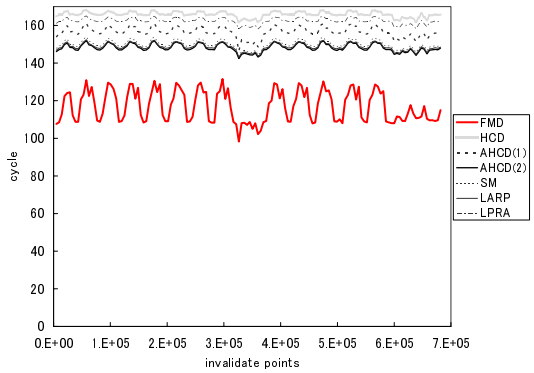

図 5 無効化処理におけるサイクル数 – OCEAN –

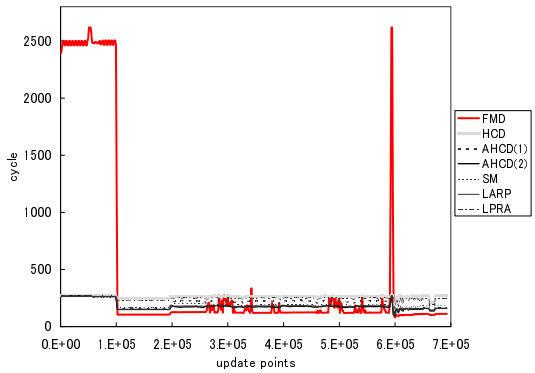

図 6 更新処理におけるサイクル数 – FFT –

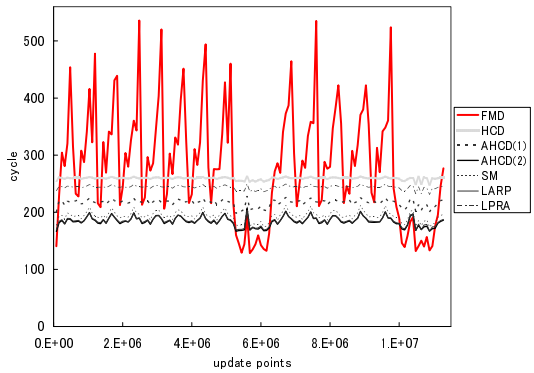

図 7 更新処理におけるサイクル数 – OCEAN –

プにおけるの 3 つの方式である．両プログラムとも無効化処理では AHCD(2) がフルマップについで最も小さな値を示している．また，更新処理では AHCD(2) が平均的に最も小さいサイクル数となっている．各実行結果における一回の一貫性維持処理あたりの平均サイクル数を**表 2** に示す．

4.3 結合網内トラフィック

4.2 節と同じプログラムおよびトレースを使用して一貫性維持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．一対のスイッチ間

表 1 一貫性処理時のプロセッサ共有数

アプリケーション	平均共有数	
	invalidate	update
FFT	3.16	38.62
OCEAN	2.62	17.92

表 2 一貫性処理時の平均サイクル数

ディレクトリ	平均サイクル数			
	FFT		OCEAN	
	invalidate	update	invalidate	update
FMD	119.1	484.2	117.0	283.1
HCD	171.1	261.3	165.9	259.4
AHCD(1)	163.7	211.7	156.5	216.7
AHCD(2)	154.2	181.5	148.2	184.2
SM	157.1	196.1	150.0	193.3
LARP	154.6	184.3	148.8	185.1
LPRA	167.9	247.0	162.4	243.9

表 3 一貫性処理時の平均パケット数

ディレクトリ	平均パケット数			
	FFT		OCEAN	
	invalidate	update	invalidate	update
FMD	45.5	564.9	38.3	262.2
HCD	654.7	624.9	597.1	611.2
AHCD(1)	380.0	342.2	222.8	300.9
AHCD(2)	76.9	141.6	41.0	124.5
SM	138.3	190.2	88.71	161.5
LARP	199.2	249.6	174.7	226.1
LPRA	255.6	285.9	200.7	250.5

を通過するパケットを 1 とし，一回の一貫性維持処理あたりの総パケット数を求める．無効化処理の結果を**図 8**，**図 9** に，更新処理の結果を**図 10**，**図 11** に示す．全ての結果において，AHCD(2) は最も小さいか，あるいは FMD の次に小さい値となっている．各実行結果における一回の一貫性維持処理あたりの平均総パケット数を**表 3** に示す．

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．

メモリトレースを用いて一貫性維持処理のシミュレーションを行った結果，2 つのポインタを有する AHCD は他の不完全情報型ディレクトリ方式よりも所要サイクル数，総パケット数に関して最も小さい値となることを確認した．本方式は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

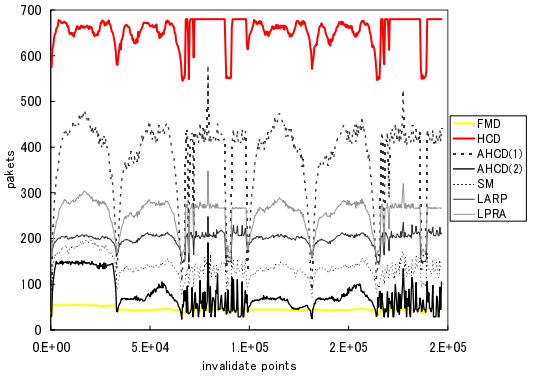

図 8 無効化処理におけるパケット数 – FFT –

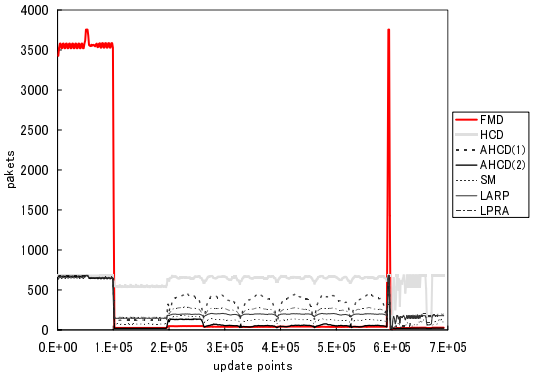

図 10 更新処理におけるパケット数 – FFT –

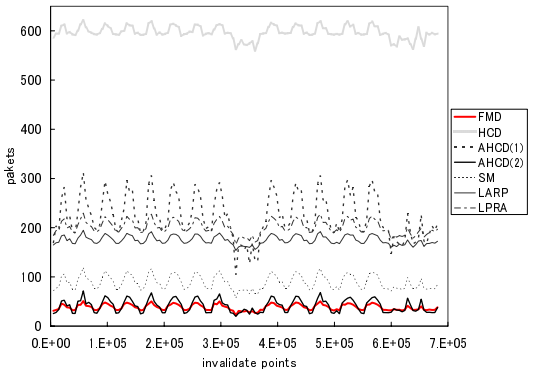

図 9 無効化処理におけるパケット数 – OCEAN –

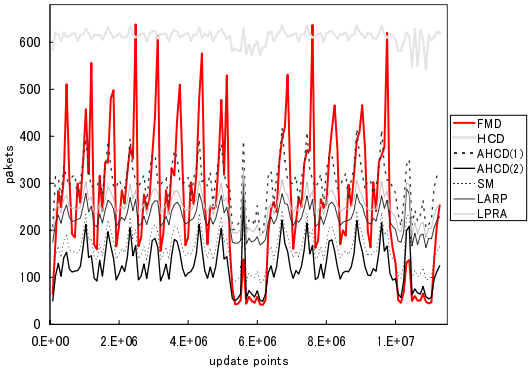

図 11 更新処理におけるパケット数 – OCEAN –

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.

参 考 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) A.Agarwal, R.Simoni, J.Hennessy and M. Horowitz: An Evaluation of Directory Schemes for Cache Coherence. ISCA, 1988.
6) 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.
7) 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.
8) 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.
9) K.Tanaka, T.Matsumoto and K.Hiraki, “Lightweight Hardware Distributed Shared Memory Supported by Generalized Combining”, Proc. of HPCA, pp.90–99, 1999.
10) D.Sunada, D.glascolasco and M.Flynn, “ABSS v2.0: SPARC Simulator”, Technical Report, CSL-TR-98-755, Stanford Univ., 1998.
11) 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.