計算機アーキテクチャ 116－1
（1996. 1. 26）

分散メモリ型マルチプロセッサ用キャッシュ一致保証方式の提案と評価

森岡道雄　　黒沢憲一
日立製作所 日立研究所

階層バス型の分散メモリ型マルチプロセッサでは、キャッシュ一致保証方式としてディレクトリ方式を採用するものが多い。これは、分散された各主メモリごとに該主メモリのコピーがどのキャッシュに存在するかを記録する方式である。しかし、本方式では、主メモリ容量の増大に伴ってディレクトリの物量が大きくなる問題がある。本論文では、バススヌープ機能をベースに分散メモリ型マルチプロセッサのキャッシュ一致保証を効率良く実装する方式としてエクスポートディレクトリ方式を提案する。本方式は、マルチプロセッサを複数のクラスタ（複数のプロセッサ及び主メモリからなるグループ）に分割し、各クラスタ毎にエクスポートディレクトリを設け、システム全体でキャッシュ一致保証すべきかどうかを判定する方式である。タスクスイッチの影響も含めてOLTPプログラムを評価できるシミュレータを構築し、提案した方式の性能評価を行った。

Design and Evaluation of the Cache Coherent Mechanism for the Distributed Shared Memory Multi-Processor

M. Morioka　　K. Kurosawa
Hitachi Research Laboratory, Hitachi, Ltd.

Directory-based cache coherent mechanism is famous approach for the distributed shared memory multi-processor which has the layered bus architecture. However, when the capacity of the main memory become large, the total size of the directory memory also grows. And it causes a expensive storage hardware for a large scale machine.

We propose an snoop-based *export-directory* architecture in order to realize cost-effective cache-coherent mechanism for the distributed shared-memory multi-processor. In this architecture, the system is composed of some clusters which have some processors, a part of the main memory and a export-directory. The export-directory is accessed in order to decide inter-cluster or intra-cluster cache coherency for each memory accesses. Proposed architecture are evaluated by trace-driven simulator which can evaluate the OLTP including task activities.

1．はじめに

近年、数十台から数百台のプロセッサを効率良く動作させるスケーラブルな共有メモリマルチプロセッサの研究が盛んである。これらのシステムでは、Non-uniform Memory Access Architecture (NUMA) と呼ばれる分散メモリ型システム構成を採用するものが多い。これは、共有メモリを分散させるシステム構成により、共有メモリアクセスのボトルネックを避けるとともに、プロセッサの近くに主メモリを実装可能とし、メモリレイテンシを改善することを狙ったものである。

分散メモリ型マルチプロセッサにおける重要な課題として、各プロセッサに分散されたキャッシュメモリの一致保証制御がある。分散メモリ型マルチプロセッサ用のキャッシュ一致保証方式として各種のディレクトリ方式が提案されている[2)5)]。これは、分散された各主メモリごとに該主メモリのコピーがどのキャッシュに存在するかをディレクトリとして保持する方式である。これにより、全てのプロセッサに対するブロードキャストやバス監視の機構は不

要となり、キャッシュ一致保証を大幅に高速化できる。ディレクトリ方式の有効性は確認されているが、主メモリ容量の増大に比例してディレクトリの物量が大きくなる問題を含んでいる[4]。特に、昨今主メモリ容量が数ＧＢまで拡大してきており、ディレクトリ容量の問題は無視できなくなってきている。

本論文では、バススヌープ機能をベースに分散メモリ型マルチプロセッサのキャッシュ一致保証を効率良く実装する方式としてエクスポートディレクトリ方式を提案する。本方式は、マルチプロセッサを複数のクラスタ（複数のプロセッサ及び主メモリからなるグループ）に分割し、各クラスタ毎にエクスポートディレクトリを設け、キャッシュ一致保証をシステム全体で実施すべきか、クラスタ内のみで実実施すべきかを判定する方式である。提案した方式の性能評価に関しては、ビジネス向けアプリケーションである OLTP (On-Line Transaction Processing)の評価に適したトレースドリブンシミュレータを構築する。これは、OLTPの命令トレースを入力として、タスクスイッチの影響も含めて評価できるシステムである。

２．マルチプロセッサシステムの概要と課題

２．１ システム概要

対象とした分散メモリ型マルチプロセッサの構成を図１に示す。４個のプロセッサと１ＧＢ程度の主メモリを１クラスタとし、最大４つのクラスタをクラスタ間バス及び入出力システムバスで接続する。各プロセッサはシステム内の全ての主メモリにアクセスできる。物理アドレス空間は、各クラスタ内の分散メモリが連続した領域として見える構成とした。プロセッサは、120MHzのRISCプロセッサとし、キャッシュは命令・データ分離で、容量はそれぞれ256KBとした。ラインサイズは32バイトでストアイン方式を採る。クラスタ内バス及びクラスタ間バスは８バイト幅のアドレス・データマルチプレックスであり、60MHzで動作する。スプリットバス方式により、実効性能で384Mバイト/秒が可能とした。

システム全体にわたってキャッシュ一致を保証するハード機構を設ける。クラスタ内・クラスタ間ともに、スヌープ方式によってキャッシュ一致保証を行う。ライト無効化方式を基本とし、キャッシュメモリ内の各ラインがModified、Exclusive、Shared、Invalidの４状態をとりうるキャッシュ一致保証プロトコル[1]を採用する。命令が発行された順序で実行されるメモリモデルであるストロングオーダーを保証するために、クラスタ間キャッシュ一致保証が実行できるプロセッサは唯一とする。即ち、複数のクラスタから同時にクラスタ間キャッシュ一致保証を実行しようとした場合、最初にクラスタ間バスの使用権を獲得したプロセッサのみがシステム全体にわたるキャッシュ一致保証を実行できる。獲得できなかったプロセッサは、実行中のキャッシュ一致保証を中断し先行プロセッサが終了後再度リトライする方式を前提とした。

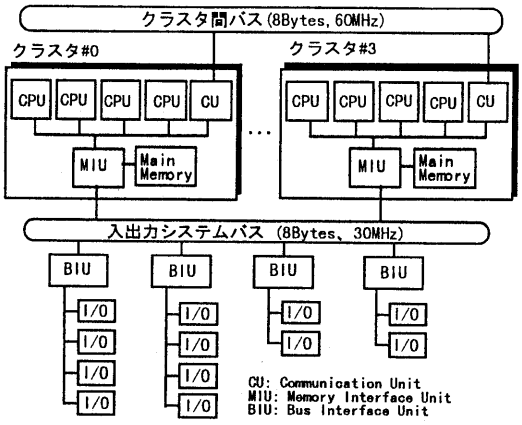

図１ 分散メモリ型マルチプロセッサの構成

図２に階層型マルチプロセッサバスの基本動作を示す。クラスタ内のあるプロセッサが他クラスタ内のデータを読み出すトランザクションを発行すると、自クラスタ内でクラスタ内キャッシュ一致保証制御(intra-cache coherent control)が起動される（図中②）。自クラスタ内の通信ユニット(CU)は、これを取り込み、クラスタ間キャッシュ一致保証制御(inter-cache coherent control)を起動する（図中③）。全ての他クラスタの通信ユニットはこれを取り込み、クラスタ内キャッシュ一致保証制御を起動する（図中④）。この時対象クラスタでは主メモリの読みだしも並列に起動される（図中⑤）。キャッシュ一致保証の結果は、全ての他クラスタの結果が集計されクラスタ間バスを経由して報告される。キャッシュ一致保証の結果、最新データが他プロセッサのキャッシュになければ、対象クラスタの主メモリから呼び出されたデータが返送される。一方他プロセッサのキャッシュにあれば、プロセッサ間で直接データが転送される（キャッシュ間コピー）。

２．２ キャッシュ一致保証における課題

OLTPプログラムは、大規模データベースに対する読みだし・更新要求を受け付け処理するといった比較的単純なタスクが、多数個並列に処理され

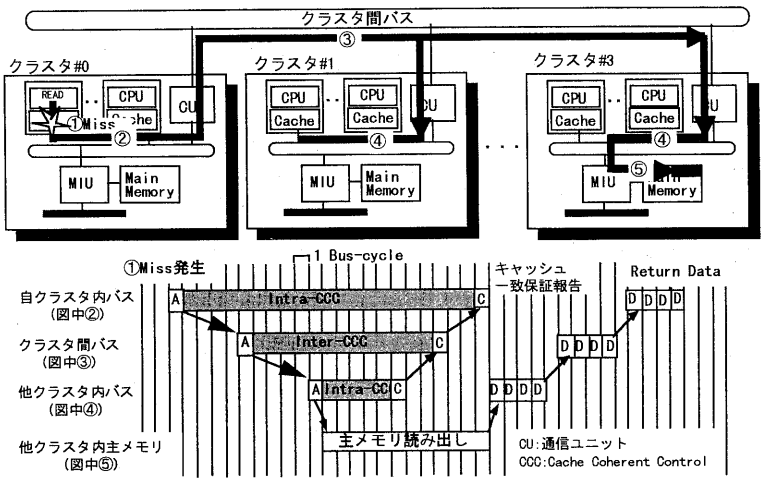

図2　階層型マルチプロセッサバスの基本動作

るプログラムである。従って、クラスタを意識したタスクスケジューリングを行えば、分散メモリマルチプロセッサの利点を最大限に引き出せる可能性がある。しかし、キャッシュ一致保証に関連して、以下の問題が考えられる。1つは、全プロセッサを対象としてキャッシュ一致保証を行う方式では、キャッシュ一致保証の遅延のためにクラスタのローカリティを十分に生かせず、メモリレイテンシが低下してしまう。また2つ目は、ストロングオーダーメモリモデルを実現するためには、全プロセッサを対象としたキャッシュ一致保証を並列に実行できない。このため、全プロセッサ対象のキャッシュ一致保証が一旦発行されると、他のトランザクションは待たされてしまい、バスの利用率が低下する。

これらの問題を解決する手段として、各種のディレクトリ方式が提案されている[2)5)]。ディレクトリ方式とは、クラスタ内の各主メモリごとに該メモリのコピーがどのキャッシュに存在するかを示すディレクトリを持つ方式である。これによって、クラスタ内だけでキャッシュ一致保証を完了できるか、またできない場合は、どのクラスタに対してキャッシュ一致保証を実行すべきかを判定でき、クラスタのローカリティを引き出すことが可能となる。

しかし、ディレクトリ方式では、主メモリ容量の増大に比例してディレクトリの物量が大きくなるといった問題がある。例えば、16台のプロセッサが1GBの主メモリを共有する場合、キャッシュラインサイズを32バイトとすると、ディレクトリサイズは約数十Mバイトとなる。ディレクトリサイズを低減する方式が各種提案されているが[3)4)]、最も少ないものでも数Mバイトは必要となる。この問題は今後主メモリ容量やプロセッサ台数の増大にともなって、更に無視できないものになってくると予想される。

3．エクスポートディレクトリ方式

3．1　基本概念

従来のディレクトリ方式は、命令や共有データ、非共有データの区別なく、主メモリ上の全てのデータをディレクトリとして管理する必要があり、その容量が主メモリ容量と関連するといった問題を引き起こしている。これを解決するために、エクスポートディレクトリ方式を提案する。図3にエクスポートディレクトリの構成を示している。エクスポートディレクトリは、各クラスタ毎に設けられ、担当クラスタが有する主メモリのデータの内、他クラスタのキャッシュメモリに登録されたデータの物理アドレス及びその状態を記録するディレクトリである。状態ビットは、Shared、Dirty、Invalidの3つの状態を示す。Sharedは、該当データが他クラスタに登録されているが、変更はされていないことを示す。Dirtyは、該当データが他クラスタに登録され且つ内容が変更されていることを示す。エクスポートディレクトリがオーバーフローした場合は、もっとも古いエントリを追い出すとともに、追い出されたエントリに対応したデータをシステム内の全てのキャッシュメモリから無効化する。

クラスタ内でトランザクションが発行された場合、エクスポートディレクトリを検索することにより、クラスタ間キャッシュ一致保証が必要かどうかを即座に判定できる。即ち、対象データが、自クラスタの主メモリに割り当てられており、且つ他クラスタに輸出されていないか、あるいは、輸出されていてもShared状態であれば、クラスタ内キャッシュ一致保証のみで処理を終えることができる。従来ディレクトリ方式との最も大きな違いは、クラスタ間で共有されるデータのみディレクトリで管理すればよく、主メモリ容量やプロセッサ数に依存しないディレクトリ方式が可能に

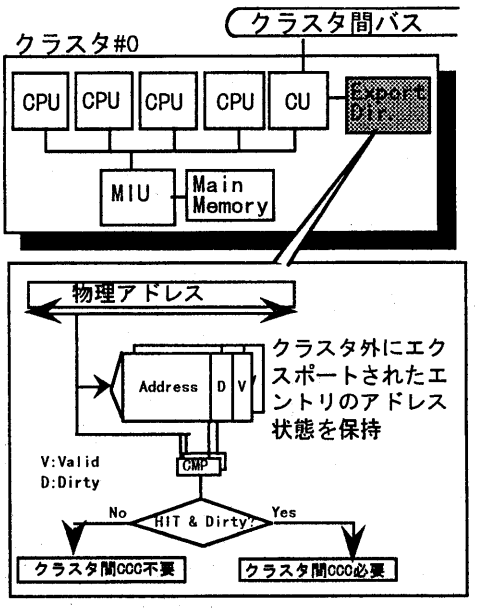

図３ エクスポートディレクトリ方式

なることである。

本方式の効果は、タスクスケジューリング方式と密接に関連する。即ち、タスクスケジューリングによって、タスクを、関連するローカルデータが存在するクラスタに割り当てることができれば、ローカルデータアクセスの大部分は、クラスタ内キャッシュ一致保証のみで処理可能であろう。エクスポートディレクトリに登録されるのは、タスク間で共有されるデータや、入出力処理に必要なＯＳ管理テーブルになる。一方、クラスタを意識しないスケジューリングのために、タスクがクラスタ間を行ったりきたりする場合は、クラスタ間キャッシュ一致保証の頻度が増大し、エクスポートディレクトリの効果はあまり期待できないと予想される。

３．２ プロトコル仕様

以下に、エクスポートディレクトリ方式における代表的なトランザクション処理の例を示す。

３．２．１ エクスポートディレクトリ登録

あるクラスタに属する主メモリに対し、クラスタ外から読みだしがあると、対象データのアドレスが、該クラスタのエクスポートディレクトリに登録される。このとき、クラスタ外からのアクセスの種類によって、登録されたエントリの状態が決定される。すなわち、参照のみであればShared状態として登録され、ストア命令のキャッシュミスに伴う更新のための読み出しであればDirty状態で登録される。クラスタ外にエクスポートされたデータが無効化された場合、あるいは該データがキャッシュメモリから追い出されてホームのクラスタに書き戻される時に、エクスポートディレクトリ内の該当エントリが無効化される。

３．２．２ リードミス処理

自クラスタ内主メモリに割り当てられたデータへのリードミス処理の例を以下に示す。あるプロセッサがロード命令実行中にキャッシュミスが発生すると、該プロセッサが属するクラスタ内バスにリード要求が発行される。クラスタ内の他プロセッサ及び、通信ユニット(CU)はこれをスヌープし、クラスタ内キャッシュ一致保証機構を起動する。プロセッサは自キャッシュ内を検索しその結果を報告する。主メモリ制御ユニット(MIU)は、受け取ったリード要求がクラスタ内主メモリに対するものであることを識別し、主メモリの読み出しを行う。また、通信ユニット(CU)は、エクスポートディレクトリを検索し、クラスタ間キャッシュ一致保証が必要かどうか判定する。

エクスポートディレクトリにミスヒットかあるいは、ヒットでも状態がSharedの場合は、クラスタ間一致保証は発行せず、クラスタ内バスにエクスポートディレクトリの検索結果を報告する。クラスタ内の全てのプロセッサ及び通信ユニットからのキャッシュ一致報告が揃った時点でこれを集計し、主メモリからデータを読み出すか、クラスタ内でキャッシュ間コピーを実行するか決定する。

一方、通信ユニットにおいて、エクスポートディレクトリがヒットし、且つ状態がDirtyであれば、自クラスタ内のキャッシュ一致保証処理を引き伸ばすように指示するとともに、クラスタ間バスにリード要求を発行し、クラスタ間キャッシュ一致保証機構を起動する。自クラスタ内ではキャッシュ一致保証が引き伸ばされている間、他のトランザクションは発行できない。他クラスタの通信ユニットは、クラスタ間バスのリード要求をスヌープし、各クラスタ内バスにブロードキャストしてクラスタ内キャッシュ一致保証を起動する。各クラスタ内のプロセッサは、これをスヌープし、Dirty状態のデータを持っているプロセッサが、キャッシュ間コピーでデータを通信ユニットに転送する。各クラスタからのキャッシュ一致保証が揃った時点で、アクセス元クラスタはこれを集計しDirty状態のデータを該当クラスタから受け取る。その後アクセス元クラスタで中断されてい

たクラスタ内キャッシュ一致保証が再開され、通信ユニットから、アクセス元プロセッサにキャッシュ一致保証の結果及びデータが転送される。この時、自クラスタの主メモリから読み出されていたデータは廃棄される。また、エクスポートディレクトリの該当エントリの状態はDirtyからSharedに変更される。

4．性能評価環境

4．1 評価システム

OLTPプログラムでは、タスクスイッチを含めた評価が不可欠である。そこでOLTPに適したトレースドリブンシミュレータを構築した。本システムは、命令トレーサとマルチプロセッサシミュレータから構成される。命令トレーサにより、単一プロセッサ実機上でOLTPプログラムを実行し1トランザクション相当の命令列を複数採取する。この命令列は、タスクスイッチ処理を含み、ロードストア命令ではそのメモリアドレスも一緒に採取する。マルチプロセッサシミュレータでは、採取した複数の命令トレースをそれぞれプログラムカウンタを持ったサーバタスクとみなしサーバプールを定義する。そして、複数のサーバタスクを並列に実行し、マルチプロセッサバスやメモリシステムの動作を1サイクル単位で模擬する。タスクスイッチポイントに来ると指定されたタスクスケジュールに従い、サーバプールから次に実行すべきサーバタスクが選択される。

4．2 マルチプロセッサモデル

プロセッサのキャッシュは、命令/データそれぞれ容量256KBとし、ダイレクトマップ方式、コピーバック型で、ラインサイズは32バイトとした。競合をさけるために、キャッシュ検索アドレスはタスク番号によりハッシュされる。キャッシュはウオームスタートとする。待機するサーバタスク数は1プロセッサ当たり16タスクとした。タスクスケジューリングは、メモリアフィニティと、ラウンドロビンで評価した。メモリアフィニティとは、タスクのデータが割り当てられているクラスタに優先的に割り付ける方式である。

メモリアクセスのレイテンシは、競合が無い場合キャッシュヒットで1マシンサイクル、キャッシュミスで39マシンサイクルとした。キャッシュ一致保証のタイミングは図2に示したモデルを仮定する。また、各タスクは共有データを持たないものと仮定した。実際にはOS管理テーブルやI/O処理等タスク間共有データは存在するが、最適化により共有データを絞り込むことが可能と判断したためである。

表1 命令出現頻度の比較

Instruction		TPC-B	SPEC89	
			Integer Gr.	Floating Gr.
Load		23.2%	26.8%	35.6%
Store		12.1%	10.5%	10.0%
Branch	Conditional	17.3%	17.9%	4.9%
	Un-conditional	5.8%	4.1%	1.4%
Others		41.6%	40.7%	48.2%

評価には、OLTPの代表的なベンチマークであるTPC-Bを用いた。表1は、TPC-BとSPEC89の命令出現頻度を比較したものである。TPC-BはSPEC89の整数系プログラムとほぼ同等の命令出現頻度となっているが、ＯＳ動作などかなり異なる特徴を持つと予想される。

5．　評価結果

5．1 キャッシュミス率

表2は、2クラスタ構成においてエクスポートディレクトリが無い場合とある場合のキャッシュミス率を比較したものである。各クラスタ内のプロセッサ数は2で、エクスポートディレクトリは、容量256エントリで2セットアソシアティブ構成とした。タスクスケジューリングポリシーとしては、メモリアフィニティとラウンドロビンで比較している。キャッシュミスの要因を分析した結果、大部分がタスク間競合によるミスであることが明らかになった。また、データキャッシュにおいて、エクスポートディレクトリ方式でラウンドロビンスケジューリングを採用した場合、ミス率が極端に増大している。これは、タスクのプロセッサ割り当てが不適切で、タスクに割り当てられたメモリ空間が、他クラスタに存在した状態で実行されることに起因する。この状態では、エクスポートディレクトリがオーバーフローするケースが頻発し、せっかくキャッシュに登録したタスクのデータが無効化されるためミス率が極端に増大する。

表2 キャッシュメモリのミス率

キャッシュ一致保証方式	スケジューリングポリシー	キャッシュミス率 (容量:命令256KB/データ256KB)	
		命令(%)	データ(%)
単一クラスタ構成		3.5	6.4
2クラスタ構成 エクスポート無し	アフィニティ	3.7	6.8
	ラウンドロビン	3.6	6.9
2クラスタ構成 エクスポート有り	アフィニティ	3.6	6.7
	ラウンドロビン	3.5	9.3

(備考) キャッシュミス率はアクセス当たりのミス率

5．2 エクスポートディレクトリ方式の効果

図4は、2クラスタ構成においてエクスポートディレクトリが無い場合とある場合のCPI(Cycle Per Instruction)を比較したものである。各クラ

スタ内のプロセッサ数は2で、エクスポートディレクトリは、容量256エントリで2セットアソシアティブ構成とした。各CPIは、要因ごとに分類して示している。要因のなかで、ＣＣリトライは、クラスタ間キャッシュ一致保証の権利獲得に失敗し、リトライしたオーバヘッドを示す。その他には、データパージ処理の待ち時間や、キャッシュビジーに伴う待ち時間などが含まれる。

メモリアフィニティスケジュールでは、タスクの割り当てが最適化され、エクスポートディレクトリにより、ほとんどクラスタ内で処理が終了する。このため、単一クラスタ構成に比べても５％程度CPIがが低下するだけに留まり、台数効果は単一クラスタの約1.9倍となる。一方、エクスポートディレクトリが無い場合、クラスタ間キャッシュ一致保証の遅延のために、ロードミス・ストアミス処理ともに増大する。また、クラスタ間キャッシュ一致保証の権利獲得失敗によるリトライも増加し、エクスポートディレクトリ方式に比べ約25%CPIが低下し、単一クラスタに対する台数効果は約1.5倍となる。

一方、ラウンドロビンスケジュールでは、タスクの割り当てが最適化されず、エクスポートディレクトリの効果が期待できない。逆に、エクスポートディレクトリが無い場合に比べて性能が低下している。これは、キャッシュミス率の章で述べたが、エクスポートディレクトリのオーバーフローに起因して、データキャッシュのミス率が増大し、性能低下を招いている。表３には、エクスポートディレクトリのエントリ数256と64に関して、エクスポートディレクトリのヒット率及びデータキャッシュのヒット率を比較したものである。メモリアフィニティでは、クラスタのローカリティが高く、エクスポートディレクトリがほとんどヒットせず、クラスタ内で処理が終了していることが解る。一方、ラウンドロビンでは、エクスポートディレクトリのヒット率は、30%前後となっている。エントリ数を64と少なくすると、エクスポートディレクトリのオーバフローの頻度が上がり、データキャッシュのミス率が増大していることが解る。

表３ エクスポートディレクトリのヒット率

スケジューリングポリシー	Export Dir. 容量(エントリ)	Export Dir. ヒット率(%)	Data Cache ミス率(%)
アフィニティ	64	0.2	6.7
	256	0.3	6.7
ラウンドロビン	64	36.4	13.0
	256	30.5	9.3

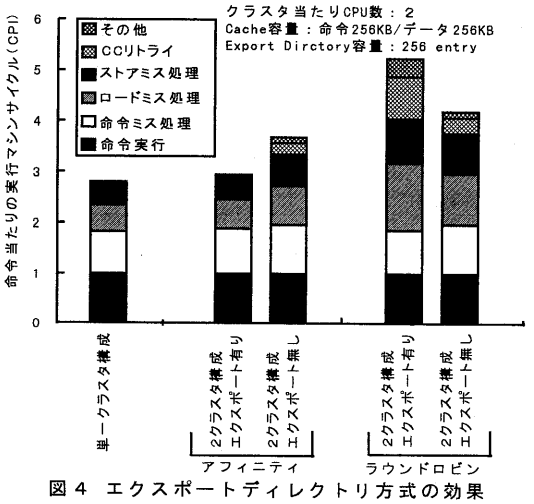

図４ エクスポートディレクトリ方式の効果

６．おわりに

階層バス型の分散メモリ型マルチプロセッサを対象に、バススヌープ機能をベースにキャッシュ一致保証を効率良く実装する方式としてエクスポートディレクトリ方式を提案した。エクスポートディレクトリは、各クラスタ毎に設けられ、担当クラスタが有する主メモリのデータの内、他クラスタのキャッシュメモリに登録されたデータの物理アドレス及びその状態を記録するディレクトリである。これにより、クラスタ間で共有されるデータのみディレクトリで管理すればよく、主メモリ容量やプロセッサ数に依存することなく、小容量のディレクトリで、キャッシュ一致保証を効率よく実装できることが明らかになった。

タスクスイッチの影響も含めて評価が可能なトレースドリブンシミュレータを構築し、OLTPのベンチマークであるTPC-Bプログラムを用いて性能評価を行った。その結果、クラスタを意識したスケジューリングを行えば、エクスポートディレクトリにより約２５％性能を改善できることを明らかにした。

参考文献

1) Sweazey, P. and Smith, A. J.: *A Class of Compatible Cache Consistency Protocols and their Support by the IEEE Futurebus*, Proc. of 13th ISCA, pp. 414-423 (1986).

2) Lenoski, D. et al.: *The DASH Prototype: Implementation and Performance*, Proc. of 19th ISCA, pp. 92-103(1992).

3) Archibald, J. et al.: *An Economical solution to the Cache Coherence Problem*, Proc. of 11th ISCA, pp. 355-362 (1984).

4) Lilja, D. J. et al.: *A Superassociative Tagged Cache Coherence Directory*, Proc. of ICCD, pp. 42-45(1994).

5) Chaiken, D et al.: *Limit-LESS Directory: A Scalable Cache Coherence Scheme*, Proc. of ASPLOS, pp. 224-234(1991).