投機的コヒーレンス制御機構SCCMの性能解析

鈴木圭介[†]　古川文人[††]　大津金光[†]
横田隆史[†]　馬場敬信[†]

DSMシステムではリモートメモリへのアクセスレイテンシはローカルのものに比べ格段に長いことが問題となっている．我々はこのリモートレイテンシの短縮を目的とした，投機的コヒーレンス制御方式spec-allとその実現方法の一つである投機的コヒーレンス制御機構SCCMを提案してきた．従来のreadアクセスのみを投機の対象としたものと違い，writeとreadの両方のメモリアクセスにより発生するリモートレイテンシの短縮を目的とするものである．

これまでの研究では，投機しないものに対し最大1.59倍，readのみ投機を行うものに対し，最大1.18倍の速度向上が出ている．しかし，現実的なベンチマークであるSPLASH-2のBarnes, Raytraceでは十分な速度向上が得られていない．本稿では，投機の全実行回数に対する対象共有データへの投機実行の割合と，投機の成功，失敗の割合の調査を行い，SCCMの性能解析を行なった．その結果，最大で約90%の成功率を得ることができているが，評価アプリケーションによって，約30%の成功率しか得られていないことがわかった．また，投機の失敗の原因や対象共有データへの投機実行の割合を調査し，そのデータを元に今後の高速化の可能性について検討する．

Performance analysis of the Speculative Coherence Control Mechanism SCCM

Keisuke SUZUKI,[†] Fumihito FURUKAWA,[††]
Kanemitsu OOTSU,[†] Takashi YOKOTA[†] and Takanobu BABA[†]

In DSM systems, the access latency to a remote memory is very long compared to a local memory. We have proposed the Speculative Coherence Control Mechanism SCCM which is one of Speculative Coherence Control system spec-all aiming at shortening of this remote latency, and the realization method of its. This aims at shortening of the remote latency generated by memory access of both write and read.

In old research, the 1.18 times as many improvement in a performance as this has come out to that to which only 1.59 times and read speculate to what does not speculate at the maximum. However, improvement in a performance which is considered is not obtained in Barnes and Raytrace of SPLASH-2 which are a realistic benchmark.

In this paper, the rate of the speculation execution to the object share data to all the number of times of execution of speculation and the rate of a success/failure of speculation were investigated, and performance analysis of SCCM was performed. Consequently, about 90% of rate of a success was able to be obtained at the maximum. However, evaluation application showed that only about 30% of rate of a success was obtained. Moreover, the rate of the speculation execution to the cause and object share data of failure of speculation is investigated, and the possibility of the future improvement in the speed of the data to origin is examined.

1. はじめに

分散共有メモリシステム(以下，DSMシステム)ではリモートメモリへのアクセス(以下，リモートアクセス)は相互結合網を介して行われるため，そのレイテンシ(以下，リモートレイテンシ)はローカルのものに対して格段に長い[1),2)]．

この問題解決のアプローチの一つとして，Cache Coherent DSMシステムにおける，投機的コヒーレンス制御がある．これは過去のメモリアクセスの履歴をもとに予測をし，コヒーレンス制御を投機実行することによってそのオーバーヘッド(以下，コヒーレンスオーバーヘッド)を軽減し，リモートレイテンシを短縮する方法である．

† 宇都宮大学工学部情報工学科
Department of Information Science, Faculty of Engineering, Utsunomiya University
†† 宇都宮大学サテライト・ベンチャー・ビジネス・ラボラトリー
Satellite Venture Business Laboratory, Utsunomiya University

我々は，新たな投機的コヒーレンス制御方式spec-allとその実現方法である投機的コヒーレンス制御機構SCCMを提案している[3),4)]．spec-allは，Producer-Consumer共有状態(以下，PC共有状態)とMigratory共有状態(以下，M共有状態)における，readとwriteの両方のリモートメモリアクセスを投機の対象とする点に新規性がある．

これまでの研究において，spec-allを評価した結果，定型的なメモリアクセスを行うアプリケーションでは，投機を行わない方式および既存の投機方式に比べて，極めて高い性能向上が得られることを確認している．しかし，現実的なアプリケーションでは十分な性能向上が得られていない．

本稿では，この問題の原因を明らかにするために，1) 投機の全実行回数に対する対象共有データへの投機実行の割合，2) 投機成功/失敗の割合の2点を調査することで，SCCMの挙動を解析する．

2. 投機的コヒーレンス制御方式spec-allとSCCM

2.1 投機的コヒーレンス制御方式spec-all

我々の提案する投機的コヒーレンス制御方式spec-allは，PC共有状態と，M共有状態における，readおよびwriteアクセス両方のリモートアクセスを対象としている．以後，投機を全く行わないnon-spec，readのみを投機の対象とするspec-readを比較対象とする．

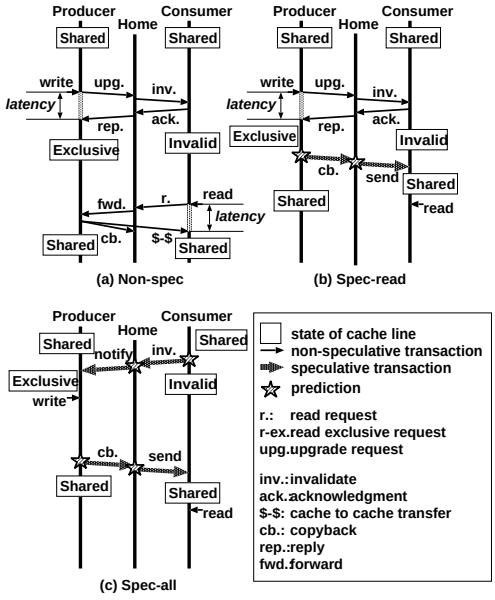

図1 PC共有状態における投機的コヒーレンス処理の例

図1は，コヒーレンス処理の流れを示したものである．PC共有状態とはあるメモリブロックに対し，Producerノードがwriteを行い，その後Consumerノードがreadを行う共有パターンである．

図1(a)，(b)，(c)は，それぞれnon-spec，spec-read，spec-allでのコヒーレンス処理の流れである．図1(a)では，Producerによるwrite アクセスが開始された後，Consumerの所有するラインの無効化が初めて開始される． また，Consumerによるreadアクセスが開始された後，Producerへのread要求が初めて開始され，それぞれレイテンシが発生する．それに対し，図1(c)では，予め投機を行い処理をすることにより，readおよびwrite アクセスによるリモートレイテンシを完全に除去することができる．

投機を行うには，処理の種類，対象アドレス，処理を開始するタイミングの予測が必要である．これらの予測が失敗した場合，本来必要の無い処理が行われ，状態が変更されてしまうため，変更前の状態に戻すための回復処理が必要となる．これにより余計なリモートアクセスが発生し，レイテンシが増大する．

よって，高速化を見込むためには，投機失敗によるオーバヘッドよりも成功によるリモートレイテンシの短縮の方が大きくならなければならない．そのためには，十分に精度の高い予測が必要となる．

2.2 投機的コヒーレンス制御機構

2.2.1 DSMシステムの構成

我々が前提しているDSMシステムの構成を図2に示す．このDSMシステムは，複数のノードがネットワークで結合されたものである．各ノードは，プロセッサコア，キャッシュ，DSMコントローラ，ディレクトリおよびメモリ，ネットワークインターフェース，および投機的コヒーレンス機構(以下，**SCCM**(Speculative Coherence Contorol Mechanism))から構成される．

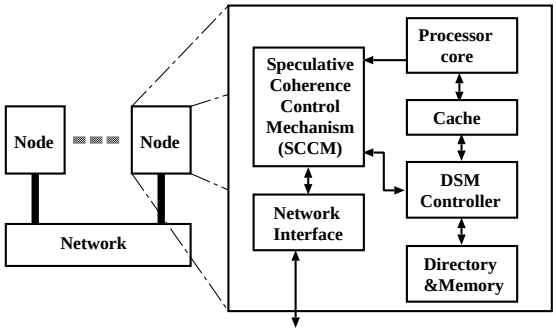

図2 DSMシステムとノード内構成

キャッシュコヒーレンスプロトコルは無効化型のMESIプロトコルである．キャッシュはModified，Exclusive，Shared，Invalid(以下，M，E，Sc，I)の4状態をとる． また，メモリブロックは，Uncached，Shared，Private，Busy-Shared，Busy-Private（以

下，U，Sm，P，BS，BP)の5状態をとる．

2.2.2 SCCMによる投機的コヒーレンス処理

SCCMでは，spec-allを実現するために，以下の5つの投機的コヒーレンス処理を行う．

(1) speculative self-invalidate(以下，spec.self-inv.)
PC共有データへのwriteアクセスおよびMigratory共有データ(M共有データ)へのreadとwrite両方のメモリアクセスによって発生するコヒーレンスオーバーヘッドを軽減するための処理である．

(2) speculative self-downgrade(以下，spec.self-dwg.)
Producer-Consumer共有データ(以下，PC共有データ)へのreadアクセスによって発生するコヒーレンスオーバーヘッドを軽減するための処理である．

(3) speculative send in shared state(以下，spec.send-sh.)
PC共有データへのreadアクセスによって発生するコヒーレンスオーバーヘッドを軽減するための処理である．

(4) speculative send in exclusive state(以下，spec.send-ex.)
M共有データへのreadおよびwriteアクセスによって発生するコヒーレンスオーバーヘッドを軽減するための処理である．

(5) speculative upgrade(以下，spec.upg)
PC共有データおよびM共有データへのwriteアクセスによって発生するコヒーレンスオーバーヘッドを軽減するための処理である．

SCCM内では，**VMSP**[6](Vector Memory Sharing Predictor)と**LTP**[7](Last-Touch Predictor)をそれぞれ独自に拡張することにより，投機的コヒーレンス処理の種類，開始のタイミングを予測する．

3. 評　　価

3.1 評価方法

評価は，non-spec，spec-read，spec-allを実現するシステムをそれぞれNON-SPEC，SPEC-READ，SPEC-ALLとし，それをシミュレートすることにより行った．

本論文では，マイクロベンチマークのpcとmgrおよびSPLASH-2のBarnes，RaytraceをCC-NUMA型並列計算機シミュレータRSIM[8]にSCCMの機能を実装することによりシミュレーションを行った．pcとmgrはPC共有状態とM共有状態を繰り返すプログラムである．BarnesとRaytraceは現実的な並列ベンチマークでありそれぞれPC共有状態，M共有状態をとるデータへのメモリアクセスを行う．

表1　シミュレーションパラメータ

ノード数	32 台
プロセッサコア	4命令同時発行スーパースカラ (2GHz動作)
命令キャッシュ	完全ヒット
データキャッシュ（ノンブロッキング）	
ラインサイズ	64バイト
最大ミス未解決数	4
L1	
サイズ	64Kバイト
アクセスレイテンシ	1サイクル
L2	
サイズ	8Mバイト
タグアクセスレイテンシ	3サイクル
データアクセスレイテンシ	4サイクル
メモリアクセスレイテンシ	
ローカルメモリ	80サイクル
ネットワーク	
トポロジ	2次元メッシュ
パケットヘッダサイズ	8バイト
フロー制御方式	ワームホール
レイテンシ/ホップ	
データなし	88サイクル
(2ホップ以降)	(+36サイクル/ホップ)
データ付き	184サイクル
(2ホップ以降)	(+36サイクル/ホップ)

評価パラメータを表1に示す．

3.2 評価結果

図3にNON-SPEC，SPEC-READ，SPEC-ALLの性能向上比を示す．また，図4にそれぞれのreadアクセス，writeアクセスにおける平均メモリアクセスレイテンシを示す．速度向上比，平均メモリアクセスレイテンシともに，NON-SPECの実行速度を基準とした．また，平均メモリアクセスレイテンシは，リモートとローカル全てのメモリアクセスレイテンシの平均としている．

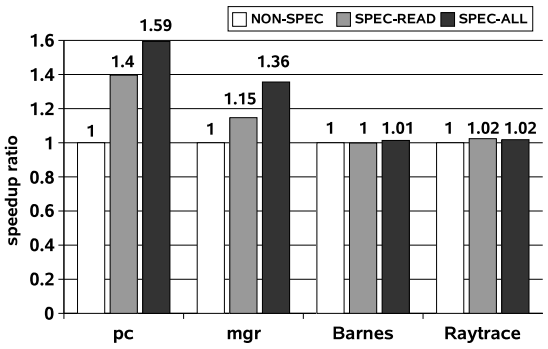

図3　速度向上比

図3から，SPEC-ALLがNON-SPECに対し，pcで1.59倍，mgrで1.36倍，Barnesで1.01倍，Raytraceで1.02倍の速度向上となった．また，SPEC-READに対してpcにおいて1.14倍の速度向上となった．また図4より，平均メモリアクセスレイテンシが，NON-SPECに比べ，SPEC-ALLではpcで77%，Raytraceで6%削減されている．

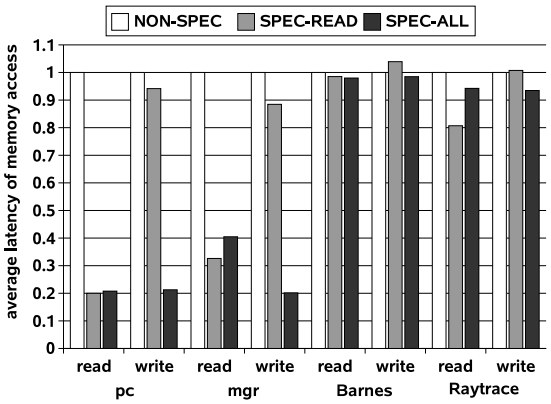

図4　平均メモリアクセスレイテンシ

3.3　SCCMの性能解析

評価結果より，pc，mgrに比べBarnes，Raytraceではわずかな速度向上しか得られていない．その原因を探るため1) 対象共有データへの投機実行の割合がどれくらいか，2) 投機の成功/失敗の割合がいくらかの2点について調査を行った．

3.3.1　対象共有データへの投機実行の割合

本方式spec-allでは，PC共有パターンとM共有パターンという，特定のパターンを対象としている．そのため，対象としているデータ領域に対して投機が行われていなければ，性能の向上は見込めない．そのため，投機先のデータ領域を調べ，対象とする領域への投機の状況を調べた．以下にそれぞれの投機の行われている領域を示す．これ以降，spec-allが対象としているPC共有データ，M共有データを含む領域を対象領域，それ以外の領域を非対象領域と呼ぶ．

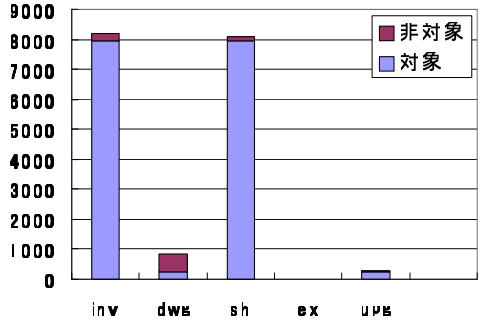

図5　領域ごとの投機の回数(pc)

図5，6，7，8では，SCCMによる投機的コヒーレンス処理ごとに各領域への投機の回数を示している．pc，mgr，Barnesでは対象領域への投機の割合がそれぞれ94%，52%，84%と頻繁に行われていることがわかる．しかし，Raytraceでは対象領域への投機がわずか3%しか行われていない．そこで，Raytraceでの主な投機先を調べたところ，lock変数に対しての割合が約80%を占めていることがわかった．

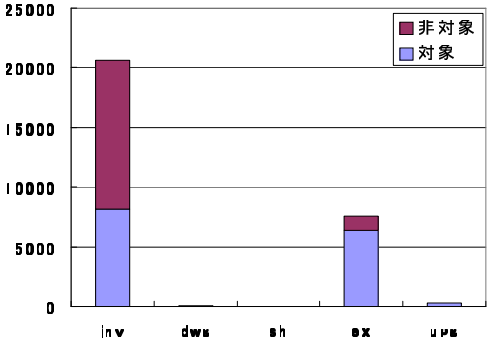

図6　領域ごとの投機の回数(mgr)

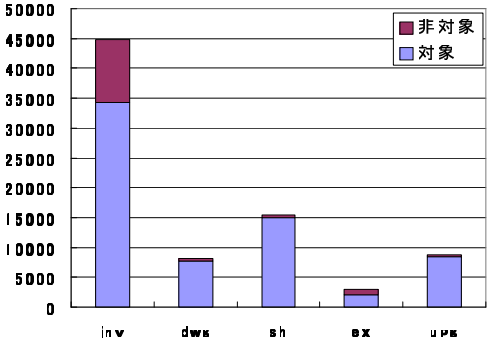

図7　領域ごとの投機の回数(Barnes)

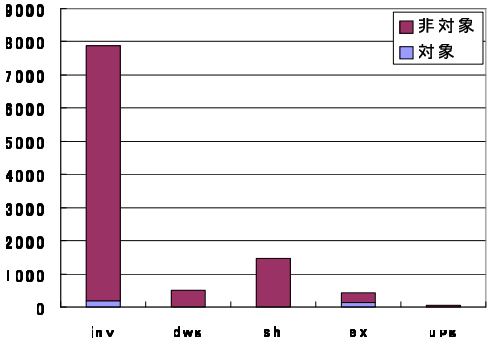

図8　領域ごとの投機の回数(Raytrace)

3.3.2　成功/失敗の割合

次に，投機の成功と失敗の割合について調べた．ここではspec-allと対象領域と非対象領域での違いを見るためにそれぞれの領域ごとに割合を調べた．

まず，投機の失敗の原因について説明する．投機の

失敗の原因を分類すると以下のようになる．

投機失敗の原因の内訳

(1) spec.self-inv.

予測の失敗による原因

・ディレクトリコントローラにおいて失敗からの回復処理のために，非投機要求が到着した．または回復処理は発生しないが不要だった

その他の原因

・L2キャッシュで当該ラインが処理中

(2) spec.self-dwg.

予測の失敗による原因

・ディレクトリコントローラにおいて，失敗からの回復処理のために，非投機要求 (read-ex，upgrade) が到着した．または回復処理は発生しないが不要な spec.self-inv の回数

・L2キャッシュにおいて，shared状態のラインに対して行った

その他の原因

・L2キャッシュにおいて，当該ラインが処理中

(3) spec.self-sh.

予測の失敗による原因

・キャッシュにコピーを送信し，read可能な状態に関わらず，使われなかった (readがなかった)

・仮定しているメモリブロックの状態が違うために，ディレクトリコントローラにおいて破棄

・L2キャッシュにおいて，ラインが既に存在していた

その他の原因

・ディレクトリコントローラ内のバッファに空きがない

・write-back bufferが足りなかった

・L2キャッシュにおいて，当該メモリブロックに対して既に要求を出力していて，その応答待ちだった

・L2キャッシュにおいて，置換の対象となるラインが全てUPGRADEの応答待ちであるために置換できなかった

(4) spec.self-ex.

予測の失敗による原因

・キャッシュにコピーを送信し，かつ，専有状態にも関わらず，使われなかった (read or writeがなかった)

・仮定しているメモリブロックの状態が違うために，ディレクトリコントローラにおいて破棄

その他の原因

・ディレクトリコントローラ内のバッファに空きがない

・L2キャッシュにおいて，置換が必要なのに，write-back buffer が足りなかった

・L2キャッシュにおいて，当該メモリブロックに対して既に要求を出力していて，その応答待ちだった

・L2キャッシュにおいて，置換の対象となるラインが全てUPGRADEの応答待ちであるために置換できなかった

(5) spec.self-upg.

予測の失敗による原因

・キャッシュラインが専有状態であることを送信したにも関わらず，使われなかった (writeがなかった)

・仮定しているメモリブロックの状態が違うために，ディレクトリコントローラにおいて破棄

・送信先のノードにキャッシュラインが存在しなかった

その他の原因

・ディレクトリコントローラ内のバッファに空きがない

・当該ラインに対して既に要求を出力していて，その応答待ちだった

これらの投機失敗の原因は大きく分けて2つに分けることができる．1つは予測の失敗によるもの，1つは応答待ちなど，予測と関係の無いその他の原因によるものである．

図9，図10にそれぞれ対象領域，非対象領域での投機の成功率を示す．図中では，投機の成功をsuccess，予測の失敗による投機の失敗をfailure，その他の原因による投機の失敗をotherとした．

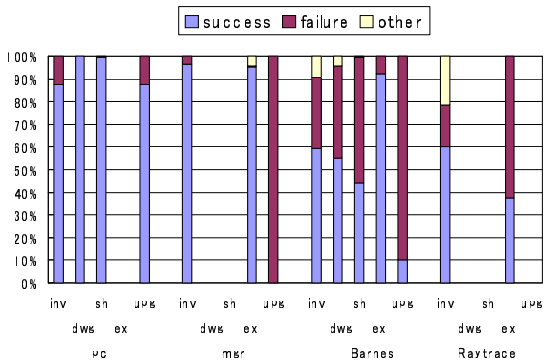

図9 対象領域での成功率

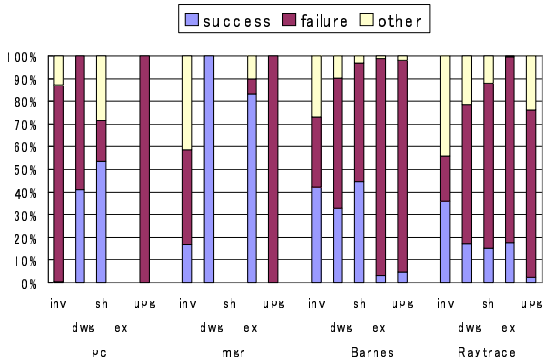

図10 非対象領域での成功率

3.4 考　　察

図9，図10より，非対象領域では対象領域に比べ成功率が低いことが分かる．また，非対象領域では予測の失敗以外の理由による投機の失敗も目立つ．非対象領域にはbarrier変数やlock変数など含まれており，予測の困難なものも含まれており，lock変数が約80%を占めるRaytraceでの成功率が低い理由の一つであると考えられる．

Raytraceにおいて，Barnesに比べ対象領域への投機も少なく，成功率も低いにも関わらず，性能が低下していない．その原因として，Barnesに比べ，投機失敗時の回復処理など，オーバーヘッドの影響ある失敗の原因の割合が低いことが挙げられる．

以上の結果より，より高速化を目指すには次のことが考えられる．

第1に，spec-allが対象とする領域以外への投機に関して，barrier変数やlock変数などの予測の困難なものに対し，投機を行なわないなどの措置を取り，投機の失敗によるオーバーヘッドを極力抑えることが考えられる．

第2に，予測の成功率をあげることである．これは，予測器を改善することにより予測の精度をあげることで可能である．

また，この2つは予測失敗による投機失敗の改善であるが予測が成功していても投機が行われないなどの，それ以外の原因による投機失敗の改善を行なうことにより，さらなる高速化が可能であると考える．

4. ま と め

本論文では，4つのベンチマークプログラムを用い，投機的コヒーレンス制御機構SCCMの性能評価を行った．これまでの研究により，定型的なメモリアクセスを行うアプリケーションでは，最大でNON-SPECに対し1.59倍，SPEC-READに対し1.14倍の速度向上比が得らている．しかし，現実的な並列ベンチマークであるBarnesとRaytraceにおいては，NON-SPECと比べ，それぞれ1.01倍，1.02倍の速度向上比しか得ることができていない．その原因を調べるために，投機の全実行回数に対する対象共有データ領域への割合，成功/失敗の割合，また，失敗の原因を調査することによりSCCMの性能解析を行い，さらなる高速化の可能性を検討した．その結果，投機の失敗する確率の高い領域に対し，投機を行わないことや，予測器を改良し，成功率をあげることによって，高速化の可能性があることがわかった．

今後の課題として，より高精度な予測を可能とするための予測器の改良と，spec-allで対象としている領域以外での予測の不可能な部分での，投機の回避方法を考えることある．

謝辞　本研究は，一部日本学術振興会科学研究費補助金(基盤研究(B)14380135，同(C)14580362，若手研究14780186)の援助による．

参 考 文 献

1) 細見岳生，他: 並列計算機Cenju-4の分散共有メモリ機構，並列処理シンポジウムJSPP'99論文集，pp. 15-22(1999).
2) Landon, J. et al.: The SGI Origin: A cc-NUMA Highly Scalable Server, *Proceedings of the 24th Annual International Symposium on Conputer Architecture* (1997).
3) 古川文人，他: DSMシステムにおける投機的コヒーレンス制御機構の提案と評価，情報処理学会研究報告2001-ARC-142 2001-HPC-85，pp. 169-174(2001).
4) 鈴木圭介，他: 投機的コヒーレンス制御行うDSMシステムの性能評価，情報処理学会第65回全国大会論文集(1)，pp.135-136(2003)
5) Woo, S.C., et al. A.: The SPLASH-2 Programs; Characterizarion and Methodological Considerations, *Proceedings of the 22th Annual International Symposium on Computer Architecture*, pp. 179-190(1998).
6) Lai, A., et al. :Memory Sharing Predictor: The Key to a Speculative Coherent DSM, *Proc. of the 26th Ann. Int. Symposium on Computer Architecture* (1999)
7) Lai, A., et al. :Selective, Accurate and Timely Self-Invalidation Using Last-Touch Prediction, *Proc. of the 27th Ann. Int. Symposium on Computer Architecture*(2000)
8) Pai, V. S., Ranganathan, P. and Adve, S. V.: RSIM; An Execution-Driven Simulator for ILP-Based Shared-Memory Multiprocessors and Uniprocessors, *Proceedings of the Third Workshop on Conputer Architecture Education*(1997).