5G−8

共有メモリ型マルチプロセッサにおける多段キャッシュ動作のシミュレーション評価

前田慎司　高橋正人　宮田裕行　菅 隆志
三菱電機株式会社 情報システム研究所

1. はじめに

現在、商用の並列計算機で最も普及しているのは共有バス結合による共有メモリ型マルチプロセッサである。このタイプの並列計算機では、共有バスにアクセスが集中するため、キャッシュが重要な存在となり、アーキテクチャの検討には、その構成、制御法を解析することが必須となる[1]。

本稿では、最適なキャッシュ構成、制御法を用途に応じて定めることを目的として新たに開発した、並列計算機用OSを含めた多段キャッシュ動作の解析が可能なキャッシュシミュレータにより、アプリケーションプログラムとして、マルチプロセッサの汎用的なベンチマークとして定着しているSPEC SDM sdetを実行した際に、キャッシュのコヒーレンシプロトコル[2]、構成を変化させた時のキャッシュ動作を解析した結果を報告する。

2. 並列計算機モデル

本稿で扱う並列計算機モデルを図1に示す。各プロセッサは2段のキャッシュを備え、共有バス結合されている。

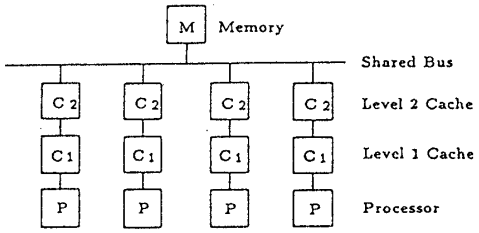

図 1: 並列計算機モデル

3. キャッシュシミュレータ構成

キャッシュシミュレータは2つの部分から構成される。

1) 並列計算機シミュレータ
並列計算機のハードウェア部のシミュレータをワークステーション上に構築し、その上で並列計算機用OSを動作させ、さらにその上でアプリケーションプログラムを動作させる。この時のOSを含めた全てのメモリアクセスのトレースを採取することができる。

2) キャッシュシミュレーション部
並列計算機シミュレータにより採取されるトレースを入力として、キャッシュ動作のシミュレーションを行う。多段構成からなるキャッシュに対応し、コヒーレンシプロトコルとしてBerkeley,Illinois, Write-once,Synapse,Dragon,Fireflyの6種類から選択が可能であり、キャッシュの振舞いの他にバスの負荷の解析が可能である。

4. 評価手法

4.1 トレース採取部

対象ハードウェアは市販のRISCチップ4台からなるマルチプロセッサとし、並列計算機用OSとしてUNIX(OSF/1)を動作させ、アプリケーションとしてSPEC SDM sdetを実行させ、その一部のトレースを採取した。

SPEC SDM sdetはUNIX上でのマルチユーザによるプログラム開発環境を模擬したマルチプロセッサ用ベンチマークプログラムであり、汎用マルチプロセッサの評価に適している。

4.2 シミュレーション部

通常、1次キャッシュはプロセッサに内蔵されているため、今回は構成の変更を行わず、2次キャッシュを以下のように変化させて評価を行った。

1) コヒーレンシプロトコルを変更した時のバスの総負荷の変化
2) 容量を変化させた時のキャッシュミスの変化
3) way数を変化させた時のキャッシュミスの変化

5. シミュレーション結果

5.1 コヒーレンシプロトコルとバス負荷

1次,2次キャッシュを表1に示す構成とし、2次キャッシュのコヒーレンシプロトコルをBerkeley, Write-once(invalidate方式)、Dragon(broadcast方式)の3種類について評価を行った(図2)。

Performance Evaluation of Multi-Level Caches for Shared-Memory Multiprocessors.
Shinji Maeda, Masato Takahashi, Hiroyuki Miyata, Takashi Kan
Computer and Information System Laboratory, Mitsubishi Electric Corporation

バスの負荷の点で、broadcast 方式よりも invalidate 方式が優れる。“SPEC SDM sdet” のような UNIX 上でのマルチタスクを想定した場合、broadcast 方式は、ヒット率が向上する利点はあるが、バスの負荷が膨大となるため不利である。

表 1: 1 次,2 次キャッシュ構成

	1 次キャッシュ	2 次キャッシュ
容量	8KB+8KB	1MB
way 数	4way	1way
linesize	32B	64B
protocol	write-through	(3 種類)

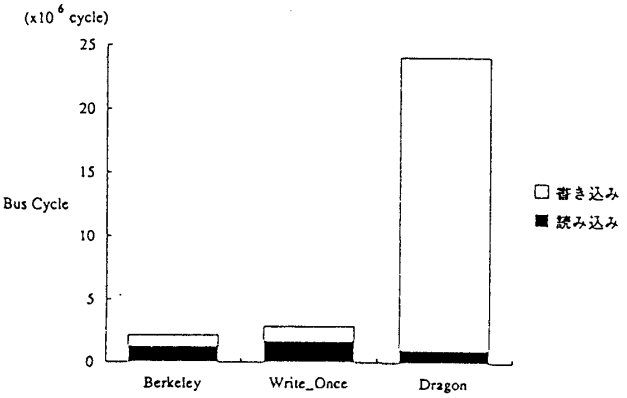

図 2: プロトコルとバス負荷

5.2 キャッシュ容量とキャッシュミス

2 次キャッシュのプロトコルを Berkeley として、容量を 64KB から 16MB まで 5 段階に変化させた時のデータフェッチミス回数をキャッシュミスの原因 [3] により分類したものを図 3 に示す。

- キャッシュ容量の拡大につれて、リプレイスミスが減少し、無効化ミスが増加する。
- キャッシュ容量が大きい時は、ミスの大半を無効化ミス、未使用データ読み込みミスが占め、リプレイスミスは少ないため、キャッシュ容量拡大の効果は小さい。

この場合、容量 4MB の時にリプレイスミスがほぼ 0 になっており、最大の効果が期待できるが、1MB 程度でも充分なキャッシュの効果が得られていると言える。

5.3 way 数とミス

2 次キャッシュの容量を 256KB,1MB として、way 数を 1,2,4 と変化させた時のヒット率を表 2 に示す。

- way 数の拡大により、ダイレクトマップ方式の欠点である特定のエントリの競合のために起こるミスを減少させることができる。
- キャッシュ容量が小さい時 (256KB) は、容量一定にして way 数を拡大するよりも、1way のままで容量を拡大する方が効果的である。
- キャッシュ容量が大きい時 (1MB) は、リプレイスミスが少ないため、way 数拡大、容量拡大による性能向上は小さい。

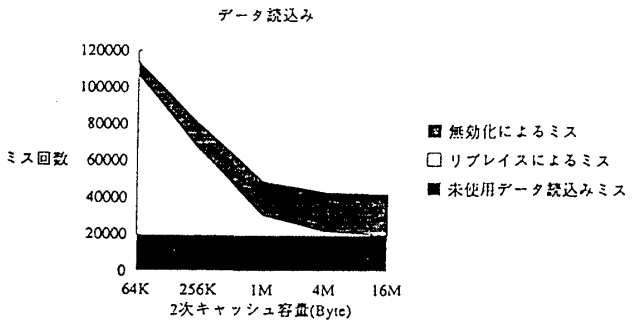

図 3: 容量とミス

表 2: way 数とヒット率 (%)

容量	1way	2way	4way
256KB	59.2	63.8	63.1
1MB	74.9	77.2	77.4
4MB	78.5		

6. おわりに

今後、このキャッシュシミュレータは、ディレクトリ方式によるコヒーレンシ制御機構等を追加することにより、NORMA,NUMA タイプの並列計算機のシミュレータへと拡張していく予定である。

参考文献

[1] S. Eggers and R. Kats, *The Effect of Sharing on the Cache and Bus Performance of Parallel Programs*, Proc. ASPLOS III Conference, 1989, pp.257-270

[2] J. Archibald and J. L. Baer, *Cache Coherence Protocols: Evaluation Using a Multiprocessor Simulation Model*, ACM Trans. on Comp. Sys., Nov., 1986, pp.273-298

[3] Computer Architecture: A Quantitative Approach, David A. Patterson, John L. Hennessy, Morgan Kaufmann Pub. Inc.,1990