ICOT Technical Memorandum: TM-0930

TM-0930

Multi-level Cache/Bus Architectureの シミュレーションによる性能評価

村谷博文(東芝)

July, 1990

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

Multi-level Cache/Bus Architectureのシミュレーションによる性能評価

東芝総合研究所情報システム研究所

村谷博文

Multi-level Cache/Bus Architectureはバス共有マルチ・プロセッサのバス・ボトルネックを解消することを目的にしている。本稿では、シミュレーションによるこのアーキテクチャの性能評価を行う。キャッシュ・ミスを原因によりスタート・アップ効果とそれ以外の部分に分け、それぞれのシステム性能への影響を評価する。1-level cacheと2-level cacheの比較で、キャッシュ・ミスの頻度が大きくなるとシステム性能は2-level cacheの方が良い。これはキャッシュ・ミスの頻度が大きいときには2-level cacheがバス・コンテンションを軽減しているのに対し、キャッシュ・ミスの頻度が小さいときには2-level cacheの方がスタート・アップ効果のコストが大きいからである。

Performance Simulation of a Multi-level Cache/Bus Architecture

Hirofumi Muratani

Toshiba R & D center, Information and System Lab.
1 Komukaitoshiba, Saiwai-ku, Kawasaki, 210, Japan

In this paper we investigate a Multi-level Cache/Bus Architecture. Such an architecture should solve the bus-bottleneck problem in bus-shared, shared-memory multiprocessors. Performance of this architecture is estimated by simulation. In the evaluation of system performance we seperately consider the contribution of start-up misses from other misses. Comparing a 2-level cache with a 1-level cache, the former shows better system performance if the frequency of cache misses is high, while giving worse performance for low miss rates. This is because for high miss rates having a 2-leve cache improves bus contension however for low miss rates the start-up costs for a 2-level cache are higher than that of a 1-level cache.

1.はじめに

Wilson Jr.[1]は、バス共有の密結合マルチ・プロセッサのバス・ボトルネックの解消のために、マルチ・レベル・キャッシュ/バス・アーキテクチャ(Multi-level Cache/Bus Architecture)を提案した。このアーキテクチャは図1に示すように、プロセッサと主記憶の間にキャッシュとバスの階層を設け、バスの負荷を分散させることによりプロセッサ数が多くなってもバス・ボトルネックを防ぐことができると考えられている。並列推論マシンPIM/kはこのアーキテクチャを採用している[2]。

Vernon et al.[3]は、解析的手法によりこのアーキテクチャの性能評価を行っている。それによると、システムのトポロジーをうまく選べばプロセッサが千台を越えても性能は飽和しない。ただし、この評価では、いくつかの仮定が用いられており、すべてのワーク(プログラム)で同じ結果が成り立つとは限らない。

マルチ・レベル・キャッシュ/バス・アーキテクチャの性能がワークの特徴にいかに依存するかは余り知られていない。本研究の目的は、マルチレベル・キャッシュ/バス・アーキテクチャの性能が、ハードウェア・パラメータやワークの特徴にどう依存するかを調べることである。このような知見はキャッシュ・メモリの最適な設計およびキャッシュを生かすソフトウェアの設計に有用であろうと考える。

ワークを特徴付けるパラメータとしてはいろいろな量が考えられる。本稿では、キャッシュ・ミスを原因により分類し、各クラスのキャッシュ・ミスの発生頻度をその一つと考えている。システム性能がその頻度にどのように依存性するかを調べるためシミュレーションによる性能評価を行う。なお、本研究は第五世代コンピュータ・プロジェクトの一環である。

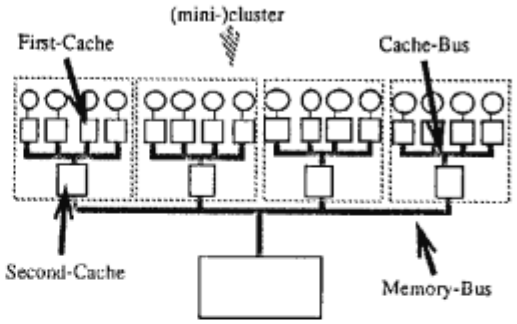

図1：シミュレーション・モデル

以下、2章では今回のシミュレーションで採用したモデルとアドレス・トレースについて説明する。3章ではワークを特徴づけるためのキャッシュ・ミスの分類について述べる。4章ではキャッシュ・ミスの発生頻度を近似的に表していると考えられるパラメータを定義する。5章ではキャッシュ・ミスのキャッシュ・ミスの発生頻度を変化させて得られた2種類のシミュレーション結果を与える。6章では5章の結果に含まれるスタート・アップ効果の影響の推定を行う。7章では、5章、6章の結果に基づいて結論を述べる。

First-Cache(2-level)	
replacement algorithm	direct mapping
block size	4words
Second-Cache(2-level)	
replacement algorithm	set associative
block size	4words
associativity	8
Turn-around(2-level)	
First-Cache(hit)	1clock
Second-Cache(hit)	13clocks
Main-memory	50clocks
Cache(1-level)	
replacement algorithm	direct mapping
block size	4words
Turn-around(1-level)	
Cache(hit)	1clock
Main-memory	38clocks
Turn-around to Bus request	3clocks
Bus latency	1clock

表1：システム・パラメータの設定

2.シミュレーション方法

2.1.シミュレーション・モデル

シミュレーション・モデルを図1に示す。PIM/kのアーキテクチャを参考にしている[2]。コンシステンシ・プロトコルの特徴は、2階層ともcopy-back方式、4状態キャッシュ・ブロック、Multi-level inclusion propertyを満たしていることである。ハードウェア・パラメータの設定は表1のとおりである。キャッシュはすべてスプリット・キャッシュである。データ・キャッシュと命令キャッシュのサイズは同じにした。階層化の効果を調べるために比較対象として1-level cacheと2-level cacheでシミュレーションを行う。

2.2.アドレス・トレース

トイ・プログラムによりアドレス・トレースを生成した。メモリ・アクセスの構成は命令フェッチ:データリード:データライト=7:2:1である。トレース長は約10K~80Kメモリ・アクセスである。今回のシミュレーションではすべてのプロセッサに同じプログラムを実行させた。ただし命令コードは共有していない。発生するキャッシュ・ミスの種類や発生頻度はデータ・アクセスの領域を調整することによって実現する。

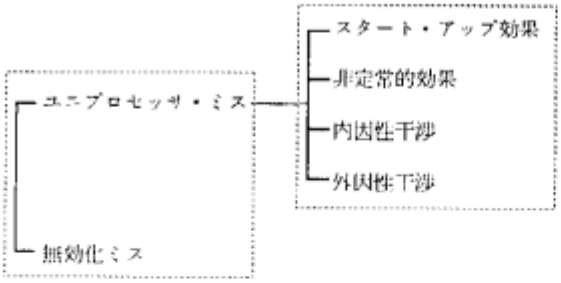

表2：キャッシュ・ミスの分類

3.キャッシュ・ミスの分類

キャッシュ・ミスはその原因により分類できる。このような分類はキャッシュの性能を理解する助けとなるのでしばしば用いられる。Agarwal et al.[4]は、シングル・プロセッサのキャッシュ・ミスの分類をもとに解析的モデルを提案している。また、Eggers and Katz[5]は、マルチ・プロセッサのキャッシュ・ミスの分類をもとにシミュレーションによる性能評価結果の解釈を与えている。

本稿の評価はAgarwal et al.とEgger and Katzの分類を統合した分類に基づいている。統合された分類を表2に示す。各キャッシュ・ミスは次のように定義される:

・ユニプロセッサ・ミスはシングル・プロセッサでも発生するキャッシュ・ミスである。

・無効化ミスはプライベート・キャッシュを持つマルチプロッサにおいて、他のプロセッサにより無効化されたキャッシュ・ブロックへアクセスすることにより生ずるキャッシュ・ミスである。

ユニプロセッサ・ミスはさらに次のように分類される:

・スタート・アップ効果は、プログラムがあるキャッシュ・ブロックを初めて参照することによって生ずるキャッシュ・ミスである。

・非定常的効果は、プログラムのワーキング・セットが非定常なために生ずるキャッシュ・ミスである。

・内因性干渉は衝突により追い出されたデータをアクセスした際に生ずるミスである。

・外因性干渉はマルチ・プログラミングでプロセス間でのキャッシュ・ブロックの追い出しにより生ずるキャッシュ・ミスである。

シングル・プロセッサのキャッシュの性能がかなりよく調べられているのに対して、マルチ・プロセッサでのキャッシュの性能は十分理解されているとは言い難い。その理由のひとつが無効化ミスの存在である。マルチ・レベル・キャッシュ/バス・アーキテクチャの性能評価においても無効化ミスの影響の評価は重要である。

4.潜在的衝突率とデータ共有率

今回はキャッシュ・ミスのクラスのうち内因性干渉と無効化ミスの発生頻度を変えてシステム性能がどう変化するかを調べる。内因性干渉の発生する頻度や無効化ミスの発生する頻度を表すために次のような量を定義した。

(1)潜在的衝突率

potentially colliding block[4]とはメモリ・アクセスのシーケンスをキャッシュ・ブロックにマップした際に衝突が起こるブロックのことである。メモリ・アクセスがpotentially colliding blockへのアクセスである確率を潜在的衝突率と定義する。以下ではこの潜在的衝突率を近似的に内因性干渉の発生する頻度と見なすことにする。

(2)データ共有率

メモリ・アクセスが他のプロセッサとの共有領域へのアクセスである確率をデータ共有率と定義し、近似的に無効化ミスの発生する頻度と見なすことにする。

5.シミュレーション結果

データ・アクセス領域を変えることにより潜在的衝突率やデータ共有率を変化させ、システム性能がどう変化するかをしらべた。プロセッサ台数は16に固定している。

5.1.内因性干渉の多いプログラムの場合

データ共有率=0で、潜在的衝突率を変化させた。図2がその結果である。潜在的衝突率が大きいと2-level cacheの方が性能が良いが、潜在的衝突率が小さくなると1-level cacheの方が性能が良くなっている。

キャッシュ・サイズが大きいほど性能が劣るのはスタート・アップ効果によると考えられる。

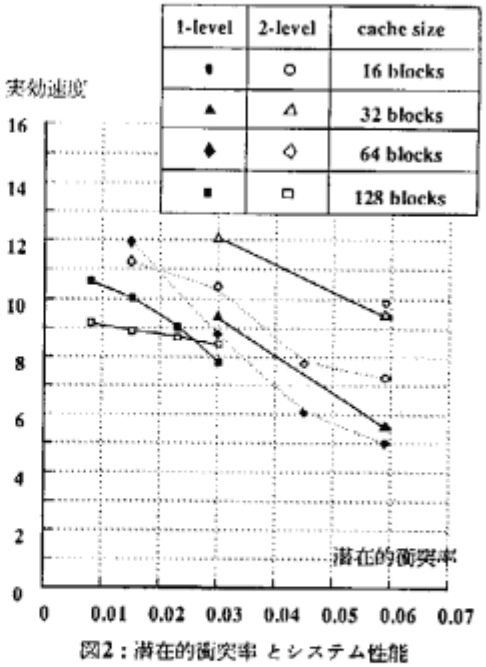

図2：潜在的衝突率とシステム性能

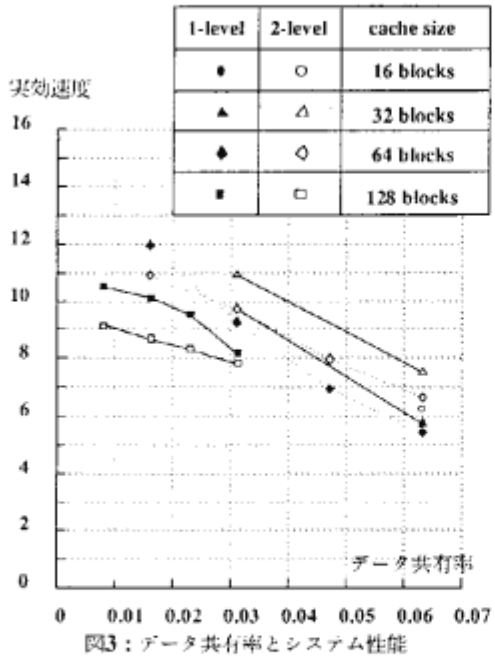

図3：データ共有率とシステム性能

5.2.無効化ミスの多いプログラムの場合

潜在的衝突率=0で、データ共有率を変化させた。図3がその結果である。この場合も、データ共有率が大きいと2-level cacheの方が性能が良いが、データ共有率が小さくなると1-level cacheの方が性能が良くなっている。ただし、1-level cacheと2-level cacheの性能の差は先の場合ほど大きくはない。2-level cacheでの無効化のコストが大きいためと考えられる。

6.スタート・アップ効果

先の二つの結果は、スタート・アップ効果を繰込んだものであった。潜在的衝突率やデータ共有率が小さい場合にはスタート・アップ効果が結果に大きく影響するため、評価結果を解釈するのが難しい。そこでスタート・アップ効果とそれ以外のキャッシュ・ミスの寄与を分離するために以下のような方法を用いた。

6.1.スタート・アップ効果の影響の推定

評価に用いたプログラムは図4のような構造をしている。Aはループ部で、Bはループ部に続く部分でループを含まない。Aの部分はデータ・アクセスがあるが、Bの部分はデータ・アクセスがない。ループがL回繰り返されるときのプログラム実行時間を T(L)、Aの部分の実行時間を $\sum_{i=1}^{L} T_A^{(i)}$、Bの部分の実行時間を T_B とする。$T_A^{(i)}$ はループの第i回目のイタレーションにかかる時間である。

$$T(L) = \sum_{i=1}^{L} T_A^{(i)} + T_B$$

イタレーションの回数 i が十分大きくなるとスタート・アップ効果が小さくなり、$T_A^{(i)} \rightarrow \tau$となる。プログラムがループ部分の実行を終了しBの部分に入ると、実行される命令コードは初めてフェッチされるキャッシュ・ブロックにマップされるのですべてスタート・アップ効果に寄与する。プログラム実行時間のうちスタート・アップ効果による部分Ts(L)を次のように表す:

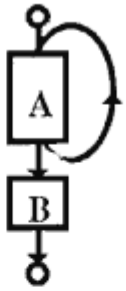

図4：トイ・プログラムの構造

$$Ts(L) = \sum_{i=1}^{L} (T_A^{(i)} - \tau) + T_B$$

従って、プログラム実行時間は、

$$T(L) = Ts(L) + L \times \tau$$

となる。Lが十分大きくなるとTs(L)はある値に収束すると予想される。そこで十分大きな整数L_1、L_2に対して、近似として $Ts(L_1) \simeq Ts(L_2) \equiv Ts$を仮定する。$T(L_1)$、$T(L_2)$を求めることができれば、τおよびTsが求まる:

$$\tau = \{ T(L_1) - T(L_2) \} / (L_1 - L_2),$$
$$Ts = \{ L_1 T(L_2) - L_2 T(L_1) \} / (L_1 - L_2).$$

以下では、Tsをプログラム実行時間へのスタート・アップ効果の寄与と考える。

6.2.内因性干渉の多いプログラムの場合

図5は、潜在的衝突率を変化させたときにA部分の処理時間がどう変化するかを表すグラフである。横軸は潜在的衝突率、縦軸はループ部を1K回実行するのにかかる時間である。潜在的衝突率の値にかかわらず、2-level cacheの方が1-level cacheよりループ部の処理時間は短い。特に潜在的衝突率が大きくなるとその差は顕著である。これはバスの負荷を分散させた効果であると考えられる。

さらに処理時間はキャッシュ・サイズにはほとんど依存していない。これはキャッシュ・サイズが違っても潜在的衝突率が同じならば同じミス率になるためである。

図6は、潜在的衝突率を変化させたときにスタート・アップ効果がどう変化するかを表すグラフである。横軸は潜在的衝突率、縦軸はスタート・アップ効果にかかる時間である。スタート・アップ効果は潜在的衝突率にあまり依存していない。どのキャッシュ・サイズでも2-level cacheのスタート・アップ効果は1-level cacheより大きい。これは今回のパラメータの設定では、スタート・アップ効果によるキャッシュ・ミス一回分のコストが2-level

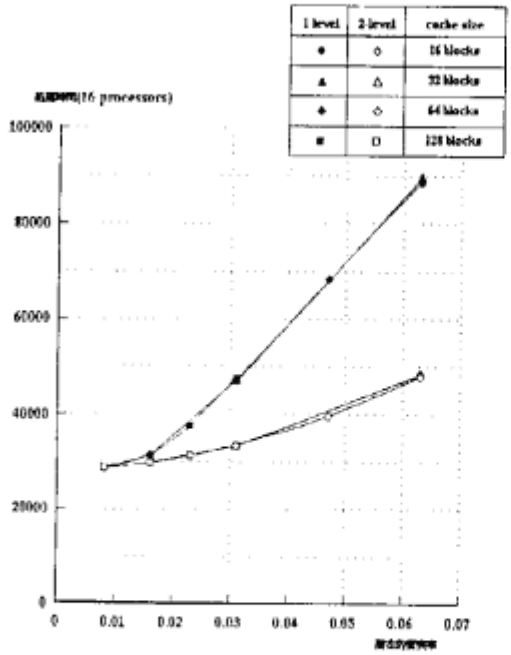

図5：潜在的衝突率と処理時間

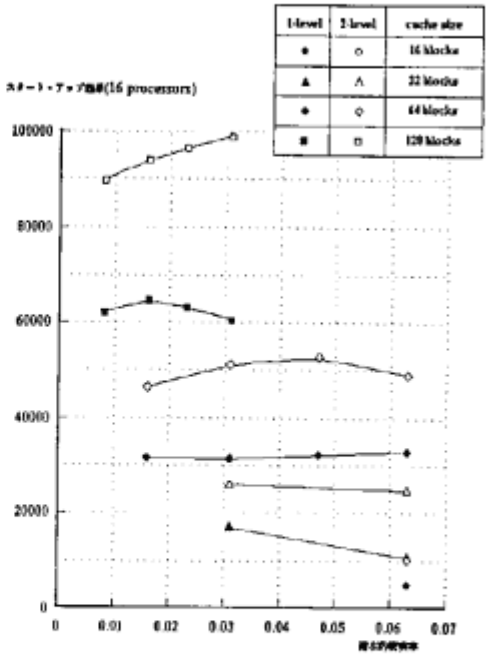

図6：潜在的衝突率とスタート・アップ効果

cacheでの方が1-level cacheより大きいためである。

プログラムはほとんどすべての(First-Cacheの)キャッシュ・ブロックをアクセスするのでキャッシュ・ブロック数の大きな場合ほどスタート・アップ時は大きくなる。

6.3.無効化ミスの多いプログラムの場合

図7は、データ共有率を変化させたときにA部分の処理時間がどう変化するかを表すグラフである。横軸はデータ共有率、縦軸はループ部を1K回実行するのにかかる時間である。この場合も、データ共有率の値にかかわらず、2-level cacheの方が1-level cacheよりループ部の処理時間は短い。特にデータ共有率が大きくなるとその差は顕著である。これはバスの負荷を分散させた効果であると考えられる。

1-level cacheと2-level cacheの差は潜在的衝突率を変化させたときほど大きくはない。これは、キャッシュ・

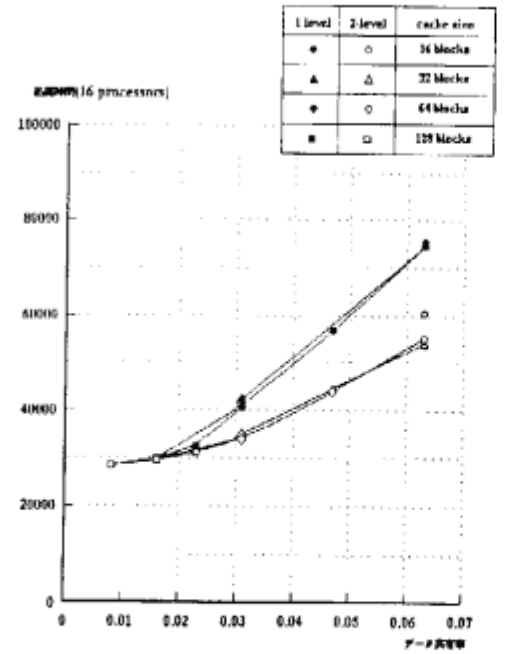

図7：データ共有率と処理時間

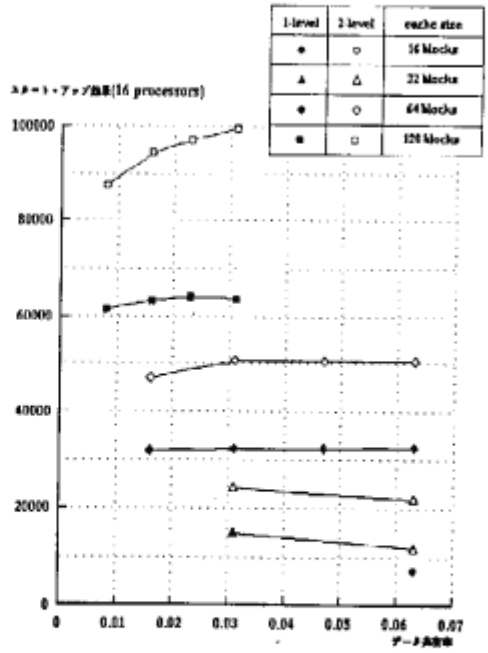

図8：データ共有率とスタート・アップ効果

ミスのコストの違いによると考えられる。1-level cacheの場合には、内因性干渉によるキャッシュ・ミスは必ず主記憶にアクセスするのに対し、無効化ミスは必ず主記憶にアクセスするとは限らず、他のキャッシュをスヌープするだけの場合もある。従って同じミス率でも無効化ミスが多い場合は内因性干渉によるキャッシュ・ミスが多い場合よりもシステム性能はよい。ところが、2-level cacheの場合には、内因性干渉によるキャッシュ・ミスは必ず主記憶にアクセスする必要は無くSecond-Cacheにcopy-backするだけでよい場合があるのに対し、無効化ミスは同じクラスタ内のスヌープで済むとは限らず、他のクラスタのFirst-Cacheをスヌープする場合もある。従って同じミス率でも無効化ミスが多い場合は内因性干渉によるキャッシュ・ミスが多い場合よりもシステム性能はよいとは限らない。実際、今回のシミュレーションでは内因性干渉によるキャッシュ・ミスが多い場合の方が無効化ミスが多い場合よりもシステム性能が良い。

図8は、データ共有率を変化させたときにスタート・アップ効果がどう変化するかを表すグラフである。横軸はデータ共有率、縦軸はスタート・アップ効果にかかる時間である。潜在的衝突率を変化させたときと同じように、スタート・アップ効果はデータ共有率にあまり依存していない。どのキャッシュ・サイズでも2-level cacheのスタート・アップ効果は1-level cacheより大きい。

スタート・アップ効果によるキャッシュ・ミス一回分のコストは潜在的衝突率を変化させた場合とほぼ同じである。

7.まとめ

プライベート・キャッシュを持つマルチ・プロセッサにおけるキャッシュ・ミスを分類した。この分類に基づいて1-level cacheと2-level cacheの性能比較を行なった。今回設定したパラメータの下では内因性干渉、無効化ミスの頻度が小さいうちは1-level cacheの方が2-level cacheよりも性能が良いのに対して、内因性干渉、無効化ミスの頻度が高くなると2-level cacheの方が1-level cacheより性能が良くなる。これは階層化によるバス・トラフィックの軽減の効果と考えられる。一般のアドレス・トレースは両方のミスを含んでいるが、この傾向は保たれると予想される。

スタート・アップ効果とそれ以外のキャッシュ・ミスの効果を分離したシミュレーションをおこなった。スタート・アップ効果を除いたシステム性能は2-level cacheの方が1-level cacheより良い。スタート・アップ効果は2-level cacheの方が1-level cacheより大きい。スタート・アップ効果は内因性干渉や無効化ミスの発生頻度にはほとんど依存せずキャッシュ・サイズによって決まる。内因性干渉、無効化ミスの頻度の小さなときに2-level cacheの性能が劣るのはスタート・アップ効果によると考えられる。階層化によるバスの負荷分散の効果を生かすためには、スタート・アップ効果や非定常的効果の寄与を十分小さくできるようなシステム設計が必要となるであろう。

今回はトイ・プログラムのアドレス・トレースを用いたが、今後は、現在開発中のKL1処理系から得られるアドレス・トレースを用いた評価を検討している。

謝辞

日頃、御指導いただくICOT第1研究室の方々に感謝いたします。

参考文献

[1]A.W.Wilson : Hierarchical cache/bus architecture for shared memory multiprocessors, Proc. of the 14th ISCA, Pittsburg, 1987, pp.244-252.

[2]浅野滋博：並列推論マシンPIM/k,情報処理学会第39会全国大会論文集(III), pp.1772-1773.

[3]M.K.Vernon, R.Jog, and G.S.Sohi : Performance analysis of hierarchical cache-consistent multiprocessors, Performance Evaluation 9 (1989), pp. 287-302.

[4]A.Agarwal et al : An analytic cache model, ACM Trans. on Computer Systems,Vol.7,No.2,1989,pp. 184-215.

[5]R.J.Eggar and R.H.Katz : The effect of sharing on the cache and bus performance of parallel programs, ASPLOS-III, 1989, pp.257-270.