オペレーティング・システム 51−15
（１９９１．７．１８）

アドレストレースの統計情報を用いた バスアーキテクチャの評価

松岡 浩司　堀川 隆
日本電気(株)　C&C システム研究所

バスアクセスに関する統計情報を用いたバスシミュレーションについて述べる。この統計情報はアドレストレースを用いたキャッシュシミュレーションにより得られるバスアクセスの時間分布であり、バッファフルによる遅れを含めた性能を評価できる。さらに、時間を要するキャッシュのシミュレーションと、バスのシミュレーションを分離したことによって、繰り返し行なわれるバスのシミュレーションを効率良く行なうことができる。このバスシミュレーションによって、アクセスを起動するアドレスの転送とリードデータの転送を分割したスプリットアクセスの評価を行なった。さらに、バンクの状態を通知することによって効率良くバスを使用するアクセス発行制御方式を提案する。この方式によると、バスボトルネックによる性能の飽和傾向をある程度緩和することができる。

Bus Architecture Evaluation using Address Trace Statistics

Hiroshi Matsuoka　Takashi Horikawa
C&C Systems Reserch Laboratories　NEC Corporation
1-1,Miyazaki 4-Chome, Miyamae-Ku Kawasaki, Kanagawa 216, Japan

Bus simulations using bus access statistics are disscussed. These statistics are obtained from address trace driven cache simulations. Using time distributions of bus accesses, delays caused by buffer fulls can be analized. Evaluation efficency is increased by separating cache simulations which need long execution time from bus simulations which are to be done repeatedly. The bus simulations can briefly evaluate the effectiveness of split accesses-read data transfers are done separately from address transfers which activate memory acccess cycles. Furthermore, we proposed an effective bus access control method using memory bank status. It improves performance by decreasing degration caused by bus bottlenecks.

1 はじめに

近年、商用機において、システムのマルチプロセッサ化が積極的に進められている。最も簡単に複数のプロセッサを実装する方法の1つとして、バス結合マルチプロセッサシステムがある。ところが、バス結合マルチプロセッサシステムでは、複数のプロセッサがバスを介して主記憶を共有するため、バスがボトルネックとなることが指摘されている。この問題を解決するために、最近のバス結合マルチプロセッサシステムでは大容量のコピーバックキャッシュを実装するようになっている。これは、ローカリティのあるアクセスを吸収し、バスを介して主記憶に対し行なわれるアクセスの頻度を低く抑えることを目的としている。

マルチプロセッサシステムにおける性能向上率(台数効果)は、バスを介して主記憶に対して行なわれるアクセスの頻度と、バスのデータ転送能力のバランスによって決まる。台数効果を発揮させるためには、バスを介して主記憶に対して行なわれるアクセスの頻度を低く抑えるとともに、プロセッサの性能に見合ったデータ転送能力を持ったバスを実装することが重要である。

バスのデータ転送能力を向上させるためには、例えば、データ転送周期を短くしたり、I/O用に独立したバスを用意すれば良い。当然のことながら、これには、大きなハードウェアコストを必要とする。このため、バスを介して主記憶に対して行なわれるアクセスの頻度と、バスのデータ転送能力をうまくバランスさせることがシステムの設計上1つの重要な課題となっている。設計初期の段階では、バランスをとるための選択枝が多岐にわたるため、それらの性能を定量的に、また、簡便に評価するためのツールが必要とされている。

我々は、要求仕様に最も適したマルチプロセッサシステムの構成を決めるための性能評価技法の研究[4]と、その性能評価技法を用いたマルチプロセッサシステムの性能評価[5]を行なってきた。実計算機のプログラム実行履歴であるトレースデータを用いて、アプリケーションの特性やOSの動作を含めた実効性能による評価を行なうことを1つの特徴としている。

本稿では、まず、トレースデータを用いたキャッシュのシミュレーションによりバスアクセスに関する統計情報を求め、この統計情報を用いて繰り返し行なわれるバスシミュレーションを効率良く行なう方法について述べる。次に、高速バスにおけるスプリットアクセスの有効性を示し、さらに、バンクの状態を通知することによってバスを効率良く使用するアクセス発行制御方式を提案する。

2 シミュレーション方式

大容量のキャッシュを有するシステムの性能を評価する場合には、大規模なシミュレーション[1]を行なう必要があり、長い時間を必要とする。この大容量キャッシュのシミュレーションとバスのシミュレーションを分離することにより、繰り返し行なうバスシミュレーションを効率良く行なう。

2.1 大容量キャッシュのシミュレーション

大容量のキャッシュを有するシステムの性能評価は2つの問題点を有している。1つは、評価のために長い時間を必要とする点で、いま1つは、システムの"定常的な"性能を評価することが難しい点である。

(1) 評価時間

キャッシュを有するシステムの性能を評価するためには、キャッシュミスヒットによるアクセスの遅延の影響を見積り、性能低下を求める必要がある。評価手段としてよく用いられているのは、プロセッサがプログラムを実行するために発行するメモリアクセス系列が記録されたアドレストレースを使うトレースドリブンシミュレーションである。トレースドリブンシミュレーションでは、まず、アドレストレースに記録されたメモリアクセスのアドレスを用いてキャッシュのタグシミュレーションを行ない、メモリアクセスのヒット/ミスを決定する。ところが、キャッシュの容量が大きくなると、短いアドレストレースでは、使用しないタグが存在したままシミュレーションが終了してしまうため、評価が"過渡的な"ものとなってしまう。例えば、キャッシュが空の状態からシミュレーションを開始(コールドスタート)すると、短いトレースでは、キャッシュタグの追い出しにともなうデータの書き戻しは生じない。実際には、以前に実行されたプログラムのデー

タがキャッシュ上に保持されているので、キャッシュタグの追い出しにともなう書き戻しが生じる可能性がある。このためバスを介して主記憶に対して行なわれるアクセスの回数が変化し、プログラム実行の遅れの見積りが異なってくる。図1は、コールドスタートの影響を概観するために、メモリアクセスの回数とキャッシュのタグの使用率との関係を調べた結果である。容量128KBのキャッシュの場合、キャッシュが80%以上使われた状態になるために300Kステップを要することが判る。キャッシュの初期状態を考慮してシステムの性能を評価するためには、1Mステップ程度のメモリアクセスが必要となる。このようなデータを用いたシミュレーションは、用いるマシンの性能に大きく依存するが数十分程度の時間を要する大規模なシミュレーションである。

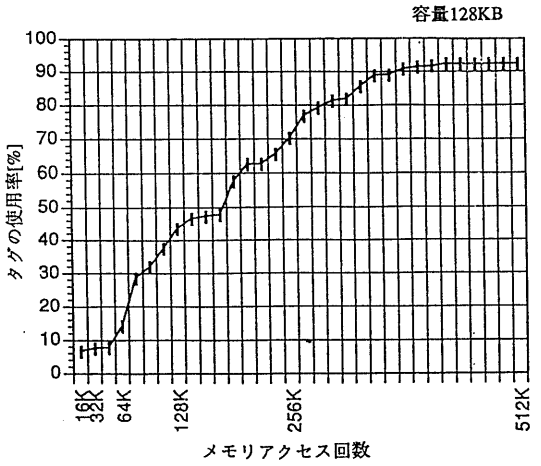

図 1: アクセス数とタグの使用率の関係

(2) 定常的な性能の評価

システムの"定常的な"性能を評価するためには、キャッシュの初期状態を考慮するだけでは不十分である。これは、キャッシュの初期状態を考慮してシステムの性能を評価するために必要とする数のメモリアクセスを発行するまでに、プロセスが切替るからである。プロセスが切り替わると、ワーキングセットが変化するため、キャッシュのヒット率が一時的に低下する。また、極めて大容量のキャッシュでは、前回のプロセスの実行の際にロードされたデータが生き残っていたりするため、ヒット率の変動はさらに複雑なものとなる。これが、システムの"定常的"な性能を評価することが難しいという第2の問題点である。

この問題点を解決するためには、OSによるプロセスの実行制御の影響を含めた評価が必要である。OSによるプログラム実行制御を考慮した性能評価を行なうために、我々は、実計算機が実行していたアプリケーションの特性やプロセス切替などのプログラム実行制御に関する情報が記録されたトレースデータをキャッシュの評価に用いている。トレースデータは、実際に稼働している計算機システムからロジックアナライザを改造したハードウェアトレーサを用いて低オーバヘッドで採取された[3]ものである。このトレースデータを用いることによって、より実際に即した評価が行なえると考えている。

今回行なった大容量キャッシュのシミュレーションでは、キャッシュは、プロセス番号とプロセッサ番号を識別するようになっている。プロセス番号はトレースデータに記録されたプロセス切替えに関する情報から得たものである。また、プロセッサ番号は、プロセス切替が生じた時点で、乱数によって決めたものである。これにより疑似的にマルチプロセッサ環境のシミュレーションを行った。

シミュレーションにおいてはユーザの空間は完全に独立しているものとして扱った。つまり、アドレスが一致してもプロセス番号が一致しない場合は異なるデータであるとした。また、プロセス番号を含めてアドレスが一致したが、プロセッサ番号が異なる場合には、他系のキャッシュ上にデータのコピーが存在することを意味している。この場合には、必要があれば、キャッシュ間のデータ転送や無効化のためのバスアクセスが行なわれる。

独立したユーザプログラムを複数のプロセッサ上で同時に実行することには問題はない。しかし、カーネルを複数のプロセッサ上で同時に実行しようとすると、共用資源管理上の問題が生じる。カーネルを並列実行するためには、共用資源の排他制御を考慮する必要があるが、今回のシミュレーションでは、これを考慮していない。従って、今回のシミュレーションでは、OSによるプログラム実行制御が効率良く行なわれた場合の、ハードウェアの性能を見積もることになる。

2.2 シミュレーションの分割

今回目的としたのは、バスアーキテクチャの評価である。バスアクセスを効率良く処理するバスを含めたメモリシステムの構成の検討を目的としている。

このような目的のためには、構成の決まったキャッシュのシミュレーションを毎回行なうのはいかにも非効率的である。このような理由から、キャッシュのシミュレーションとバスのシミュレーションを分離した(図2)。バスシミュレーションでは、バスを含むメモリシステム上で、バスアクセスが処理されていく過程のシミュレーションを行なう。これによって、主として、マルチプロセッサ化にともなうプログラム実行の遅れを求め、システムの性能の評価を行なう。なお、バスアクセスはキャッシュのシミュレーションにより得られた統計情報から確率的に発生させた。

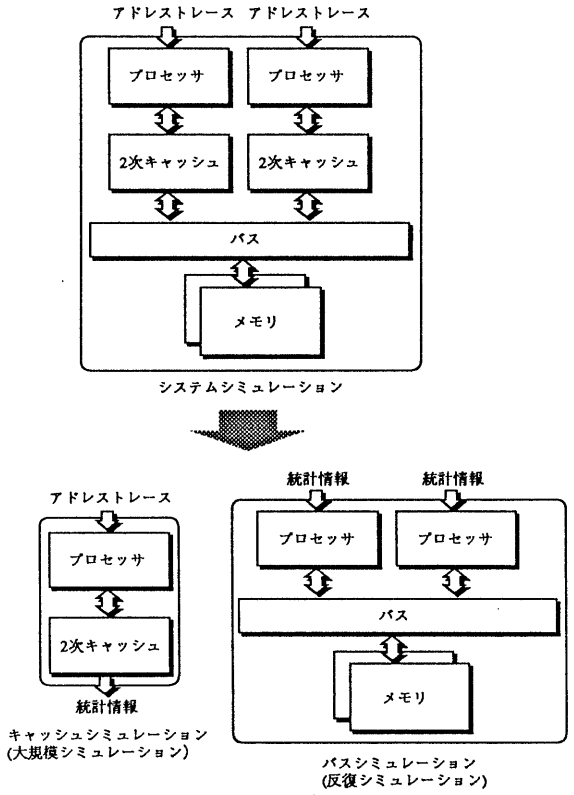

図 2: シミュレーションの分割

データの共有は、本来、動的なものであるため、キャッシュのシミュレーションとバスのシミュレーションを分離できるかは、考察の余地を残している。例えば、コピーバックキャッシュでは、共有データへの書き込みが生じると、無効化のためのバスアクセスが行なわれる。しかし、今回のシミュレーションでは、プロセスは完全に独立していて、カーネルのデータ領域は常に共有としたので、無効化による性能低下は十分に考慮されていると考えている。

統計情報は、システムの"定常的な"性能を評価するために十分な数のアクセスを用いたキャッシュのシミュレーションから得られたものである。このため、この統計情報を用いて確率的に発生させたバスアクセスは少数でもシステムの挙動をそれなりに反映させたものとなる。つまり、この統計情報を用いることによって、少ないバスアクセスで疑似的にではあるが"定常的な"性能の評価を行なうことができる。

この特徴を積極的に利用すると、次のような、柔軟な評価を行なうことができる。すなわち、まず、少ないバスアクセスを用いた短時間のシミュレーションによって評価対象の特性を把握し、ある程度選択枝を絞り込む。それから、多くのバスアクセスを用いて"定常的な"性能を評価するためのシミュレーションを行なったり、必要ならば、バスシミュレーションと同時にキャッシュのシミュレーションを行ない、時間をかけて精度の高い評価を行なう。システム設計の初期の段階では選択枝が多く、短時間で簡便に選択枝を絞り込めることは大いに魅力がある。

2.3 統計情報

統計情報としては、バスアクセスの特徴をパラメータ化したアクセス発行間隔の平均値、あるいは、実際のアクセス発行間隔の時間分布などが考えられる。今回のシミュレーションでは、バッファフルにともなう待ちに起因する性能の低下を正しく評価するために、キャッシュシミュレーションからバスシミュレーションに対しバスアクセスの時間分布を受け渡している。

(1) バスアクセスの時間分布

図3の実線は、バスを介して主記憶に対して行なわれる命令ブロックリードアクセスの発行間隔分布の例である。この例では、アクセス発行間隔

の平均値は60[T]であり、著しく発行間隔が短い方に偏った時間分布となっている。これは、例えば、分岐などによって、ワーキングセットが変わる場合に、キャッシュミスが集中することを意味している。

(2) 時間分布を用いる効果

システムを効率的に動作させるためにバッファがよく使用される。例えば、図5に示すシステムではリクエストバッファ(RB)が使用されている。

バスのデータ転送周期を短くしたことによって、アクセスタイムに比較して短い周期でバンクが起動されることになる。このため、メモリシステムは複数のバンクから構成されている。バンク競合が生じると、メモリアクセスがバンクに受理されないためバスを閉塞してしまう。リクエストバッファ(RB)を用意し、バンク競合が生じても、バンクにメモリアクセスが引き取られるようにする。これによって、バンク競合を生じさせるメモリアクセスに後続する、空いている他のバンクに対するメモリアクセスを発行できるようになる。アクセスの処理効率が高まり、バスの閉塞による性能の低下を回避することができる。

しかし、バッファの段数は有限なので、アクセスの発行間隔が短いと、バッファフルが生じる。この場合、バッファフルによる待ちだけではなく、バスの閉塞による待ちが生じ、性能が低下する。

図3の実線で示したような時間分布の偏りがあると、バッファを有するシステムのシミュレーションで平均値を用いた場合、ほとんどバッファフルによる待ちは生じないことになってしまう。実際には、バスアクセスが時間的に集中する場合があり、有意の頻度でバッファフルが生じ、待ちが生じ性能が低下する。

図3では、実線でキャッシュシミュレーションによって得られた実際のバスアクセス発行間隔の時間分布を、破線でキャッシュシミュレーションの結果から得られたバスアクセス発行間隔の平均値近傍にバスアクセスの発行が正規分布することを仮定したバスアクセス発行間隔の時間分布を示している。図4は、これら分布の異なる2つの統計情報を用いた場合の性能推定値の差を示している。メモリの負荷が高い領域で差が観測され、バッファを有するシステムの評価には実際の時間分布を用いる必要があることが判る。

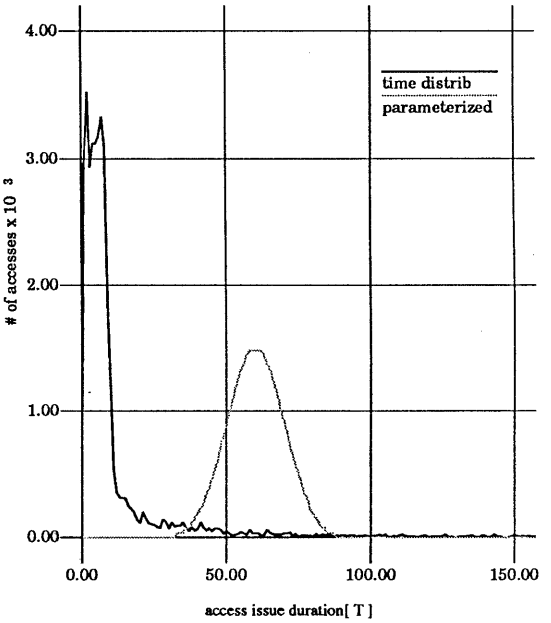

図 3: バスアクセス間隔の時間分布

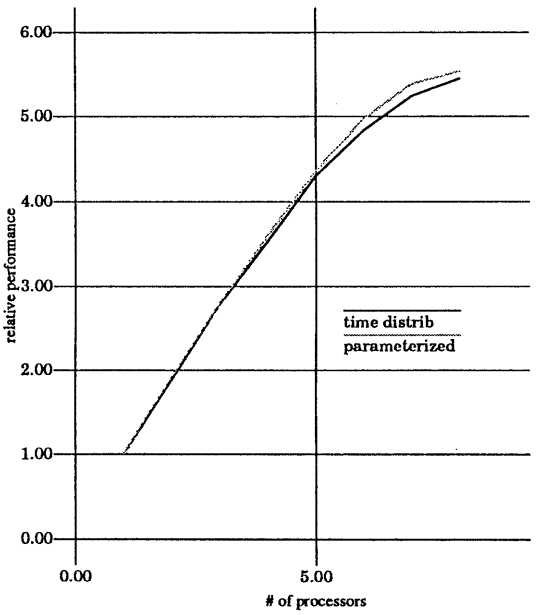

図 4: 時間分布による性能推定値の差異

3 バスアーキテクチャの評価

2で述べたバスシミュレーションによって下に示す方式の検討を行なった。

- スプリットデータ転送
- アクセス抑制制御方式

3.1 バスシミュレーションモデル

図5に、バスシミュレーションに用いたシステムのモデルを示す。

プロセッサは発行したアクセスの完了を待ち、乱数で決めたアクセス間隔時間だけ時間が経過するとアクセスを発行するようになっている。2次キャッシュにミスすると、バスを介した主記憶へのアクセスが行なわれる。バスには複数台のプロセッサが接続されているため、主記憶のアクセス時間だけではなく、バス占有待ちが生じた場合には、アクセスに要する時間が更に長くなり、プロセッサの単体性能が低下していく。

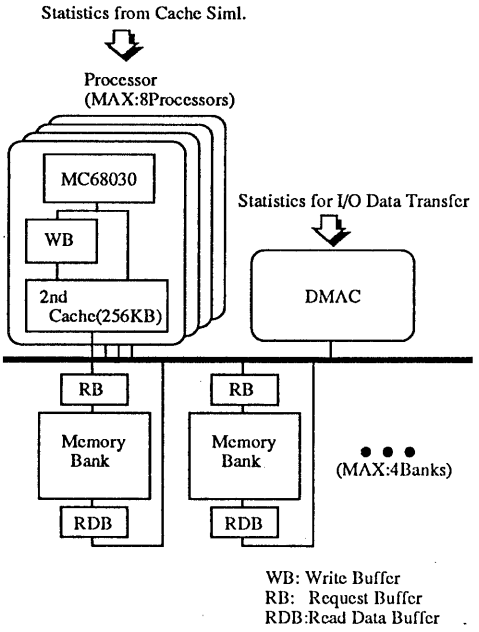

図 5: バスシミュレーションモデル

評価対象システムのプロセッサは68030(25MHz)である。68030をシミュレートするために、68030を採用した実稼働計算機から採取したトレースデータを用意した。

2次キャッシュは、コピーバックキャッシュで、スヌーピングプロトコルはイリノイ方式を採用している。2048セットを有し、4ウェイセットアソシアティブ、16Bラインで、容量は128KBである。

バスは、基本的にはFuturebus+のような非同期バスで、メモリアクセスを起動するアドレスの転送とリードデータの転送を分割したスプリットデータ転送方式を採用している。

メモリは、サイクルタイム200nSのバンク4つから構成されたインターリーブドメモリである。バンク競合が生じた場合、バスが閉塞されるのを防ぐために、バンクはリクエストバッファを有している。

3.2 スプリットデータ転送

従来のマイクロプロセッサベースのシステムではアクセスを起動するアドレスの転送とデータの転送は1組として扱われている。

バスのデータ転送能力を高めるために、データ転送周期を短くすると、アクセスタイムが相対的に長くなる。なぜなら、主記憶は一般的にDRAMによって構成されており、データ転送の高速化に対し、アクセスタイムがさほど早くならないためである。

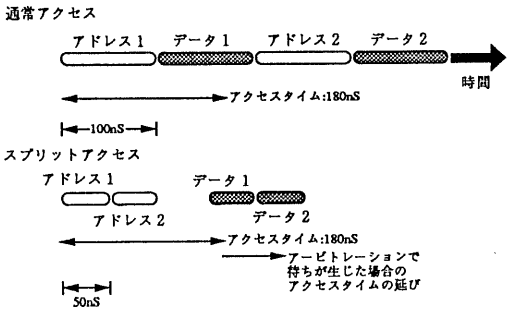

図 6: スプリットアクセス

スプリットアクセスではリードアクセスを起動するアドレスの転送とリードデータの転送を分割して行なう。データ転送周期を50nsとしアクセスタイムを180nSとすると、タイミングは図6で示すようになる。リードアクセスを起動するためにアドレスを発行した後、バスを解放すると、リードデータが得られるまでの時間に、他系のプロセッサからのアクセスにバスを割り当てることができ、データの転送効率が向上する。反面リードデータの転送にもアービトレーションが必要と

なり、待ちが生じると、アクセスタイムが若干長くなってしまう。スプリットアクセスの採用に当たっては、データ転送効率の向上による性能の向上とアクセスタイムの伸びによる性能の低下のトレードオフを考慮する必要がある。

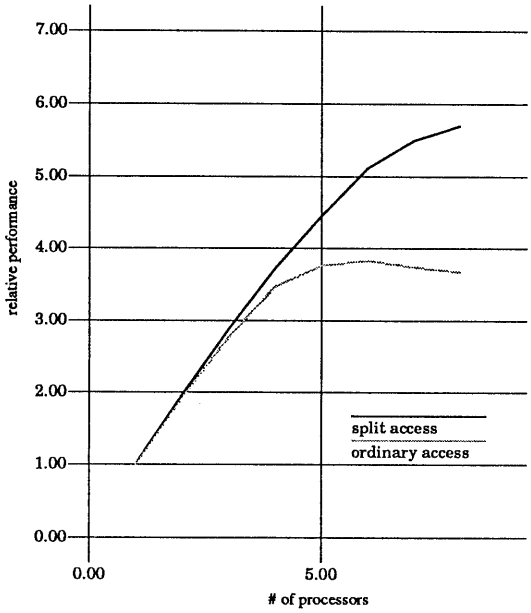

図 7: スプリットアクセスの効果

図7はシミュレーションによって求めた評価対象システムにおけるスプリットアクセスの効果を示す図である。バスの負荷が高い領域での性能向上が顕著である。

3.3 アクセス発行制御

シミュレーションに用いたモデルでは、各バンクはリクエストバッファを有している。このため、バンク競合が生じても、バスが閉塞されることが無い。ただし、バッファは有限なので、メモリの負荷が高くなってくると、今度は、バッファフルによるバスの閉塞が生じる。

リードアクセスの場合、プロセッサがリードデータを得るまで、プログラム実行に待ちが生じる。このため、リードデータ転送には最も高い優先順位が設定され、できるだけ早くプロセッサがリードデータを得ることができるようになっている。リードデータの転送に最も高い優先順位が与えられているにもにもかかわらず、バッファフルによって、バスが閉塞されると、場合によってはデッドロックが生じてしまう。図8はこの状態を示した図である。リクエストバッファフルによりバス閉塞を生じさせたバンクから、リードデータの転送が要求されている。リードデータが転送されるまで、バンクが解放されないため、リクエストバッファフルの状態は解除されない。

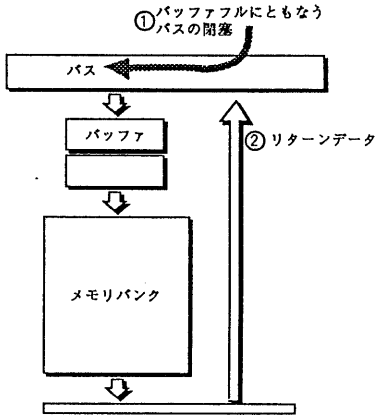

図 8: バスアクセスのデッドロック

このデッドロックを回避するためには、バッファフルによってバスを閉塞している状態でリードデータ転送要求が生じた場合、まず、バスを閉塞しているアクセスをキャンセルし、リードデータの転送を行ない、その後に起動に失敗したアクセスを再起動する必要がある。しかし、この方法ではバスを無駄に使用することになり、データの転送効率が低下してしまう。

この効率低下を回避するため、アービタが各メモリバンクのバッファの状態を監視し、現時点で発行するとバスを閉塞してしまうアクセスの発行を抑制し、かわりに、バスを閉塞しない、つまり、バッファフルを生じないバンクへのアクセスを発行するような制御方式が考えられる。これが、今回提案するバスアクセス発行制御方式である。

今回提案したバスシミュレーションを用いれば、このような方式の効果を簡便に評価できる。図9にシミュレーションによる再起動を行なう場合とバンクフルを生じるアクセスの発行を抑制する場合の性能の比較を示す。

バスの性能が不足気味の時は発行抑制制御が有効であることが判る。なお、メモリバンクの性

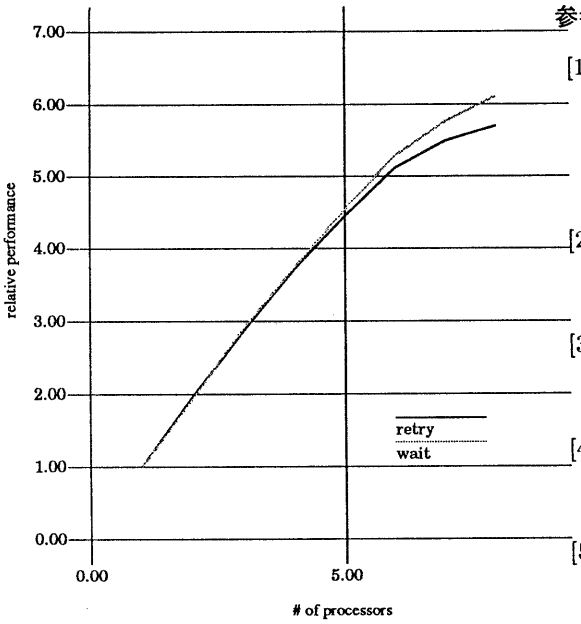

図 9: バスアクセス発行の抑制

能が不足して、全バンクのバッファがフル状態になってしまうと性能は向上しない。

4 まとめ

大容量キャッシュを有するマルチプロセッサシステムにおける、バスアーキテクチャの評価を効率良く行なうシミュレーション方式を提案した。この方式のバスシミュレーションによって、スプリットアクセスの有効性を示し、バスを効率良く使用するためのアクセス制御方式を提案した。

今回用いたバスシミュレーションでは、特に、バスアクセスの時間的局所性に着目しているが、アクセスの発行時間間隔は2つのアクセスの時間関係を規定するものであり、3つ以上のアクセスの時間関係に関する情報が欠落している。今後の課題として、厳密な時間的局所性を表現する統計情報の設定が挙げられる。

システム設計の初期段階では、選択枝が多く、簡便に選択枝を絞り込めるツールが必要である。シミュレーションを分割することによって、評価に要する時間を短縮することができ、必要に応じて、精度の高いシミュレーションを行なえる柔軟な評価環境を構築できたと考えている。

参考文献

[1] Archibald,J.,Badr,J., "Cache Coherence Protocols: Evaluation Using a Multiprocessor Simulation Model", ACM Trans. on Computer Systems, Vol.4,No.4,pp.273-298, Nov(1986).

[2] McKerrow,P., "Performance Measurement of Computer Systems", ADDISON-WESLEY PUBLISHING COMPANY,(1988).

[3] 堀川ほか, "ハードウェア・トレーサを用いた計算機アーキテクチャ評価システム", 情報処理学会,OS 研究会,Feb(1987).

[4] 松岡ほか, "マルチプロセッサシステムの評価技法と評価システム", 情報処理学会,OS 研究会,Feb(1989).

[5] 松岡ほか, "動的トレース駆動シミュレーションによるコピーバックキャッシュの評価", 情報処理学会, 第 41 回全国大会,(1990).