マルチスレッド処理におけるキャッシュ構成方式の検討*

4P－3

浦田 卓治、中村 宏、朴 泰祐、 中澤 喜三郎†
筑波大学 電子・情報工学系‡

1. はじめに

近年、プロセッサの処理速度と主記憶アクセスレーテンシのギャップがプロセッサの実効性能に与える影響は、極めて大きくなっている。このためにキャッシュメモリが用いられ、プログラムの制御フローが予測可能な場合には、プリフェッチなどの手段でメモリアクセスレーテンシを隠蔽することが可能である。しかし、一般にプログラム制御フローが予測不能な場合には、キャッシュミス時のメモリアクセスレーテンシは隠蔽できない。そこで、マルチスレッド処理 [1] によりレーテンシの隠蔽を行なう手法が考えられるが、逆にキャッシュミスが増大して、十分な性能が発揮できない恐れもある。

本稿では単一プロセッサ上での、単位時間当りの処理件数を問題とするようなマルチスレッド処理を対象とし、victim cache[2] を含む種々のキャッシュ構成方式の有効性を、シミュレーションにより検討する。

2. マルチスレッド処理方式

マルチスレッドアーキテクチャとは、キャッシュミスまたはリモートメモリアクセスなどによる、比較的長時間にわたるプロセッサストールを隠蔽するために、異なる処理をオーバーラップさせながら実行する機構をアーキテクチャレベルでサポートするものである。

本稿では、遅延隠蔽の対象をメモリアクセスによる遅延に限定し、キャッシュミスでスレッドを切替える方式を考える。この場合、隠蔽可能なキャッシュミスペナルティの大きさは、並行処理されるスレッド数と各スレッドの平均連続実行時間とで制限される。また、スレッドの平均連続実行時間がキャッシュミスレートに依存するため、スレッド数を一定とすると、隠蔽可能なペナルティを大きくするには、ミスレートを低く抑えることが重要となる。

しかし、このような小さな粒度で、複数のスレッドが同一のキャッシュに対してアクセスを行なうと、キャッシュミスが著しく増加してしまう恐れがある。ここでは、マルチスレッド処理における各種キャッシュメモリ構成方式について、検討を行なうことにする。

3. 評価

3.1 評価モデル

本稿では時間的概念は考えず、まずキャッシュミスレートのみを評価する。プロセッサモデルとしては、高速にスレッドの切替えが行なえるよう、スカラプロセッサを拡張したものを仮定する。このプロセッサモデル上で、以下に示すキャッシュモデルに対して評価を行なった。

<n−way(shared)>　すべてのスレッドで共有される、n−way set associative キャッシュ。キャッシュラインの追い出しは LRU で行なう。

<n−way(separate)>　各スレッド毎に独立した、n−way set associative キャッシュを与えるモデル。

<1−way+m(shared)>
<1−way(shared)> に、m ラインの victim cache を付加したモデル。victim cache とは、主キャッシュ (ここでは<1−way(shared)>) から追い出されたメモリブロックを、一時的に保持しておくための full associative キャッシュである。victim cache からのキャッシュラインの追い出しは LRU で行なう。

<1−way+m(separate)>　<1−way(separate)>の各スレッドに割り当てられたキャッシュに、それぞれ m ラインの victim cache を付加したモデル。

その他、全モデルに対し以下の仮定をおいた。

- 命令キャッシュ: warm start
- データキャッシュ: ラインサイズ 32bytes、write back、write allocate 方式を採用
- 主記憶: FIFO 性を保証

3.2 評価プログラム

以下の4種類のプログラムを、独立な4つのスレッドとして与えることで、評価を行なった。

(1) 行列積を求めるプログラム。
(2) クイックソートを行なうプログラム。
(3) ハイパクロスバ・ネットワーク・シミュレータ [3]
(4) 文字列検索を行なうプログラム。

各プログラムとも、ワーキングセットが 100KB 程度になるように、問題サイズを設定してある。

3.3 評価環境

シミュレーションは、トレースドリブン方式によって行なった。各評価プログラムのアドレストレースの収集にあたっては、PA-RISC1.1 アーキテクチャを対象とするプロセッサシミュレータ [4] を使用した。

シミュレータは、合計 100M 個の命令を実行した時点でのミスレートを計算する。この時、各評価プログラムをそれぞれ繰り返して実行することにより、シミュレーションが終了するまで、常に4つのスレッドが、処理を継続している状態を保つようにする。繰り返しの際、対象となるプログラムは、データキャッシュに関して cold start となるようにした。

3.4 評価結果

図1に、<1−way(shared)>を用いて、シングルスレッド処理とマルチスレッド処理を行なった場合の、ミスレートの比較を示す。ここで [Multithread] は、2節で述べ

---

*Preliminary Evaluation of Cache Configurations for Multi-threaded Architecture

†Takuji Urata, Hiroshi Nakamura, Taisuke Boku, Kisaburo Nakazawa
‡Institute of Information Sciences and Electronics, University of Tsukuba

たようなマルチスレッド処理を行なったものを表す。また[Singlethread]は、[Multithread]と同量の計算を、スレッド切替えを行なわずに実行した場合のミスレートを表している。

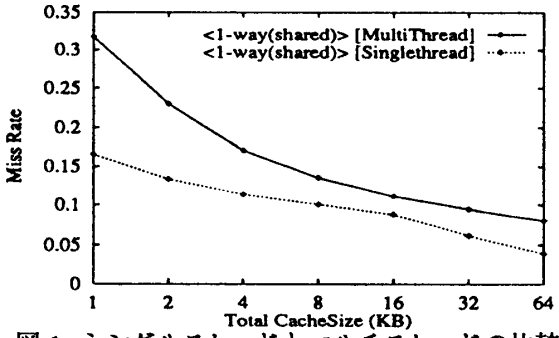

図 1: シングルスレッドとマルチスレッドの比較

[Singlethread]に比べ、[Multithread]の方がキャッシュミスが増加していることが分かる。ただし、キャッシュサイズが大きくなるにつれて、その差は小さくなっている。また今回の例では、マルチスレッド化したことは、キャッシュから見ればワーキングセットが約4倍になったことに等しい。そのため、キャッシュサイズをシングルスレッド時の4倍にすることで、マルチスレッド処理でも、シングルスレッドとほぼ同程度のミスレートが実現できていることが分かる。

このように、キャッシュ容量をスレッド数に合わせて大きくしてやることで、キャッシュミスの増加を抑えることはできる。しかし、スレッド数が多くなると、キャッシュ容量のみによる対応には限界があろう。そのため、キャッシュ構成方式の変更を考える必要が生じてくる。

図2に、キャッシュ構成を変化させた時の評価結果を示す。なお、これ以降のグラフは全て、マルチスレッド処理を行なった場合のものである。また横軸は、victim cacheを含め、システム中の全てのキャッシュ容量の合計を表している。

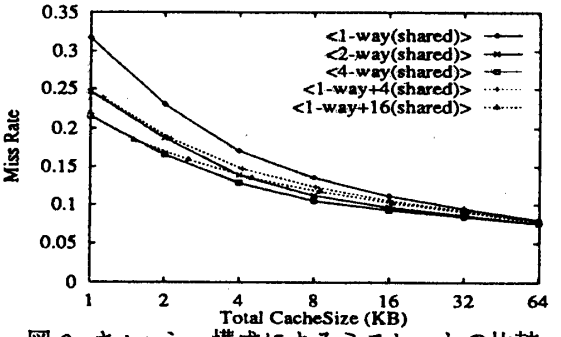

図 2: キャッシュ構成によるミスレートの比較

<n−way(shared)>において、n=2,4の時は、n=1の場合に比べ、かなり低いミスレートが実現できている。特に、キャッシュサイズがごく小さい場合を除き、n=1のままキャッシュサイズを2倍にするよりも、nを2倍にする方がミスレートが低いことが分かる。

victim cacheを付加したモデルは、キャッシュサイズが小さい場合には、set associativeモデルと同程度のミスレートが実現できているが、キャッシュサイズが大きくなるにつれ、その効果は薄れている。victim cacheが有効に働くのは、主キャッシュ上でラインコンフリクトを起こす複数のメモリブロックに対して、短時間に集中してアクセスが起こるような場合である。キャッシュサイズが大きくなり、そのようなラインコンフリクト自体が減少すると、victim cacheは十分な効果を発揮しない。ただし注意すべきは、victim cacheはその利点の一つとして、主キャッシュのアクセスタイムにほとんど影響を与えることなく、見かけ上のassociativityを増加できることである。したがって、時間の概念を考慮しない今回の評価だけで、その優劣を述べるべきではない。

上記は全スレッドでキャッシュを共有する場合であるが、各スレッドのワーキングセットが完全に異なるので、スレッド毎に専用のキャッシュを与えた方が、スレッド間での衝突がなく、ミスレートが低くなる可能性がある。そこで図3に、スレッド間でキャッシュを共有(shared)する場合と、独立(separate)の場合の結果を示す。

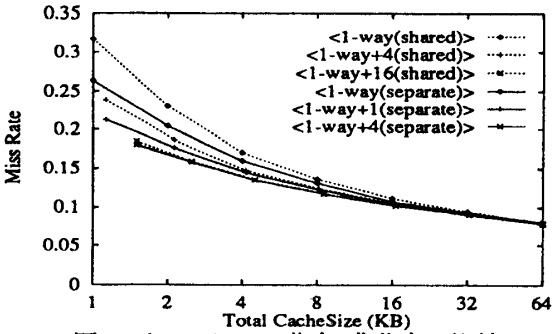

図 3: キャッシュの共有/非共有の比較

キャッシュミスは、スレッド間の競合のために生ずるものと、スレッド内の競合によるものに分けて考えることができるが、separateにはスレッド間の競合は存在しない。sharedの方は、キャッシュ容量が小さい時はスレッド間の競合の影響を大きく受け、ミスレートが増大する。しかしキャッシュ容量が大きい時には、この影響が緩和され、sharedとseparateの差はなくなってくる。また逆に、スレッド内の競合はsharedの方が少ないことが影響して、僅かではあるが逆転も起こっている。

4. おわりに

本稿では、トレースドリブン方式のシミュレーションを用いて、マルチスレッド処理における各種キャッシュ構成方式の検討を行なった。今後の課題としては、時間軸を考慮し、プロセッサ利用率を指標とした評価を行なうことなどがあげられる。

謝辞

本研究に関し貴重な御意見を頂いた筑波大学アーキテクチャ研究室の諸氏に深く感謝致します。なお、本研究は一部文部省科学研究費(奨励研究(A) 07780222)によるものである。

参考文献

[1] R.A.Iannucci, "Multithreaded Computer Architecture: A Summary of the State of the Art", Kluwer Academic Publishers, 1994.

[2] N.P.Jouppi, "Improving Direct-Mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers", 17th ISCA, pp.290-298, June 1990.

[3] 朴 泰祐 他, "ハイパクロスバ・ネットワークの性能評価", 信学技法 CPSY93-25, 1993年

[4] 廣野 哲 他, "マルチバンクメモリ上における擬似ベクトルプロセッサPVP-SWの性能評価", 情処研報 ARC-111-3, 1995年