3P−4

マルチプロセッサスケジューリングのトレースデータを用いた評価

上脇正　鈴木昭二　恒冨邦彦　山口伸一朗
(日立製作所　日立研究所)

1．はじめに

密結合型マルチプロセッサ（ＴＣＭＰ）の多くは、各プロセッサがキャッシュメモリを持っており、このキャッシュメモリが、システムの性能上非常に重要な役割を果たしている。しかし、スレッドのスケジューリングの方法によっては、キャッシュメモリの性能が大きく低下してしまう。[2]

我々は、スレッドがプロセッサを移動するときに、キャッシュメモリに蓄えられていたデータが無駄になり、キャッシュミスが連続して発生することに着目し、キャッシュメモリにデータが多く存在すると思われるプロセッサにスレッドを割り当てるＳＫＹ−１スケジューリング方式を提案してきた。[1]

しかしながら、スケジューリングが1ms〜1sの間隔で発生するのに対して、キャッシュメモリは10nsのオーダーで動作しており、両者の相互作用をシミュレーションにより、評価するには限界がある。そこで、今回、マルチプロセッサの各プロセッサがアクセスしたアドレスをトレースするトレーサを作成して、スケジューリングがキャッシュメモリの性能に与える影響を評価した。

2．ＳＫＹ−１スケジューリング

ＳＫＹ−１スケジューリングは、必要なデータがキャッシュメモリ最も多くあるプロセッサにスレッドを割り付けることを目標にしている。スレッドのデータは、前回実行されていたプロセッサのキャッシュメモリに最も多く残っている考えられるので、スレッドのプロセッサ間移動を必要最小限にとどめる。ただし、完全にスレッドをプロセッサに固定してしまうと、プロセッサ間の負荷が不均等になり、効率が低下してしまう。そこで、休止してから長く時間が経っているスレッドのキャッシュメモリ内データは、ほとんど別のスレッドのデータに置換されてしまっている考えて、そのようなスレッドは、別のプロセッサへの移動を認めた。また、前回実行されたプロセッサがビジーであるが、別にアイドルのプロセッサが存在する場合にもプロセッサ間移動を認めた。アルゴリズムは以下の通りである。

[スレッドのウェイクアップ処理]

(1) 実行可能となったスレッドの前回実行されていたプロセッサが、アイドルであれば、そのプロセッサに直接、そのスレッドの処理を依頼する。

(2) アイドルでなければ、待ち行列にそのスレッドを入れる。

[ディスパッチ処理]

(1) 前回、自プロセッサが実行したスレッド、または、休止時間がしきい値を越えているスレッドが、待ち行列中に存在すれば、その中で優先度が最高のものを実行する。

(2) 存在しなければ、待ち行列中で最高優先度のスレッドを実行する。

(3) 待ち行列にスレッドが存在しなければアイドルとなる。

3．評価方法

３３ＭＨｚの３２ビットμプロセッサを４台接続したＴＣＭＰ上で、命令トレーサを用いてスケジューリングの効果を評価した。使用したμプロセッサには１命令毎にトラップを発生するトレース例外の機能が内蔵されている。評価に用いたトレーサは、この機能を利用している。図１にトレーサの動作を示す。

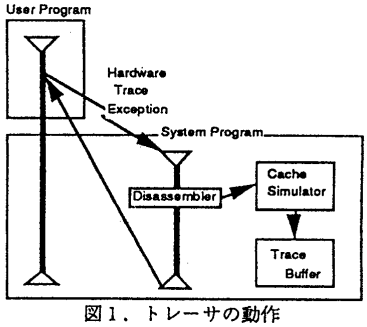

図１．トレーサの動作

トレース例外が発生すると、プログラムカウンタが指している命令を逆アセンブルする。その命令の内容と現在のレジスタの値から、その命令がアクセスしたアドレスを計算し、その結果をキャッシュシミュレータに入力する。キャッシュシミュレータは、そのアドレスへのアクセスがキャッシュにヒットするかどうかの判定を行う。キャッシュシミュレータは、以下のパラメータを設定可能である。１度に４種類の構成のキャッシュのシミュレーションが可能となっている（括弧内の値はデフォルト値である）。

(1) キャッシュサイズ（５１２ｋＢ）
(2) ラインサイズ（１６Ｂ）
(3) セット数（１）
(4) 一致化プロトコル（無効化）

ただし、後述の評価結果は、キャッシュサイズ以外、デフォルトの値を用いた。キャッシュシミュレータは、１００命令毎に各プロセッサのキャッシュミス率を計算して、トレース用バッファに結果を書き出す。

トレース時には、実行速度が通常時の1/1500になる。スレッドのスケジューリングは、クロック割り込みによっても、起動されるので、通常時と同じ条件でトレースするために、クロック割り込みの周期を実行速度に合わせて遅くした。

4．評価結果

4．1　キャッシュミス率の時間変化

マルチプロセッサの内の１台のプロセッサを選び、そのキャッシュミス率の時間変化を比較した。実行したプログラムは、８個のプロセスが、それぞれ４ｋＢの配列を計算をするプログ

Evaluation of Multiprocessor Scheduling by Trace Data
Tadashi KAMIWAKI, Shoji SUZUKI, Kunihiko TUNEDOMI, Shinichiro YAMAGUCHI
Hitachi Research Laboratory, Hitachi, Ltd.

ラムである。図２が従来のスケジューリング方式を用いた場合であり、図３がＳＫＹ−１スケジューリング方式を用いた場合である。いずれの図も、８０万命令トレースした中の４０万命令から６０万命令の部分を表示したものである。

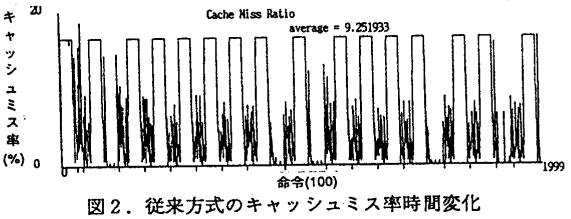

図２．従来方式のキャッシュミス率時間変化

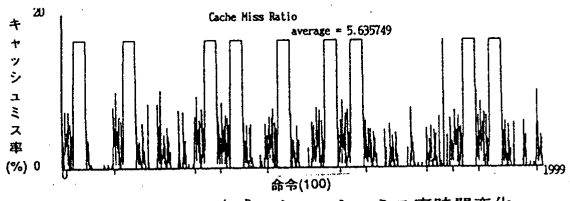

図３．ＳＫＹ−１方式のキャッシュミス率時間変化

Ｘ軸の下に描かれている縦線は、そこでスレッドのディスパッチが発生したことを示している。従来のスケジューリングでは、スレッド切り替え毎に配列アクセスに起因したキャッシュミスが連続して発生するが、ＳＫＹ−１スケジューリングでは、配列の内容がキャッシュに載ってくるにしたがって、徐々に配列アクセスに起因したキャッシュミスが発生しにくくなってくる。平均のキャッシュミス率も、従来のスケジューリングが9.25%に対して、ＳＫＹ−１スケジューリングの方は5.64%であり、39%低くなっている。

４．２　キャッシュミスのキャッシュ容量依存性

図４に、図２と同じ配列計算を行なったときのキャッシュミス率のキャッシュ容量依存性のグラフを示す。トレースした命令数は８０万命令である。キャッシュ容量が、８ｋＢのときには、配列がキャッシュメモリに入りきらないので、両スケジューリング方式でキャッシュミス率に殆ど差がない。キャッシュ容量が３２ｋＢ以上と大きくなると、スレッド切り替え後も配列やプログラムが、キャッシュメモリに残るようになるので、ＳＫＹ−１方式のミス改善も大きくなる。また、キャッシュ容量が大きくなると、キャッシュミス率自体も低くなってくるので、ミス率改善の影響も大きい。

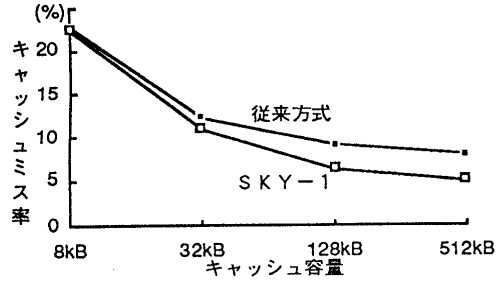

図４．キャッシュ容量依存性（配列計算）

キャッシュ容量５１２ｋＢのときのキャッシュミス率の改善は、３７％である。

図５は、Linpackベンチマーク（１００ｘ１００、倍精度）を４スレッドに並列化して、実行したときのキャッシュ容量依存性である。トレースした命令数は８０万命令である。このプログラムも配列計算なので、図４と傾向が似ているが、図４のプログラムと比較して、シリアルに実行される部分が多いために、スケジューリングによる効果が小さくなっている。キャッシュ容量５１２ｋＢ時のキャッシュミス率改善は１２％である。

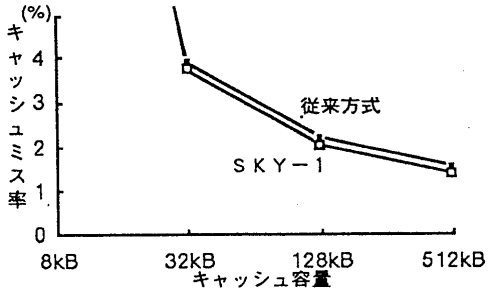

図５．キャッシュ容量依存性（Linpackベンチマーク）

図６は、SPECが提供しているSDM(Software Development Multitasking)ベンチマークを実行したときのキャッシュミス率のキャッシュ容量依存性である。トレースした命令数は、８００万命令である。両方式によるキャッシュミス率の違いはほとんどない。この原因は２つあると考えられる。１つは、図４、図５のプログラムと比較して、スイッチの間隔が長いことである。このプログラムは、１つのジョブをさらに並列化することをしていないため、実行の粒度が大きい。もう１つの理由は、システムプログラムを実行している部分が多いことである。システムのプログラム及びデータは、プロセス間で共有されている部分が大部分であるため、ＳＫＹ−１スケジューリングによる効果が出にくい。

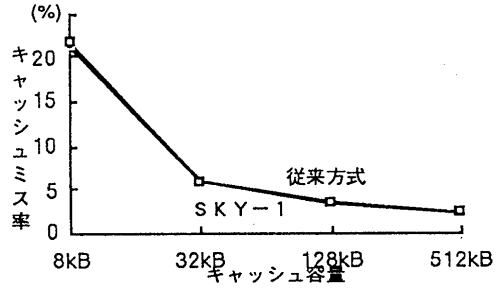

図６．キャッシュ容量依存性（SPEC-sdmベンチマーク）

５．おわりに

キャッシュメモリの状態を考慮して、スレッドをプロセッサにスケジューリングすることにより、大幅なキャッシュミスの削減が得られることを、マルチプロセッサの命令トレーサを用いることにより確認した。

キャッシュ容量の増大、キャッシュメモリと主メモリの速度差拡大により、この効果は増大するため、このようなスケジューリングは今後、必須になると考えられる。

［参考文献］

１）上脇他：並列処理用ＯＳ　ＳＫＹ−１のスケジューリング方式、情報処理学会第３９回全国大会論文集

２）M. Devarakonda, et al:Issues in Implementation of Cache-Affinity Scheduling, Proc. of the Winter Usenix, 1992