トレースベースSMPシミュレータMetatoolの開発*

3H−4

鈴木和宏† 岩田靖‡ 安里彰‡ 木村康則‡
(株)富士通研究所† 新情報富士通研‡

1 はじめに

従来の命令レベル並列実行によって性能を向上させるスーパスカラ方式のプロセッサではその限界が指摘されている。こうした中でSMPシステムは更なる性能向上が期待できるものとして注目されており、将来の半導体技術の進展によって1チップに複数のプロセッサを載せたシングルチップマルチプロセッサによるSMPの実現が期待されている。

SMPシステムを設計する際には、プロセッサ間の依存関係を考慮したシミュレーションによって性能予測を行うことが不可欠である。さらに、このシミュレーションはSMPアーキテクチャの設計だけでなく、SMPシステム向けのコンパイラの評価にも重要である。

本論文では我々が開発したトレースベースSMPシミュレータMetatool (Multiprocessor Environment Trace-based Analysis Tool)について述べる。

2 Metatoolの概要

Metatoolは我々が現在までに開発した単体プロセッサのトレースベースシミュレータParatool [1]を、SMPのシミュレーションを行うために拡張したものである。MetatoolはSPARCプロセッサ用のトレースツールShade [2]が生成するトレースデータを元に動作する。

Shadeはアプリケーションを1命令ずつ実行し、UNIXのユーザモードで実行された全ての命令のプログラムカウンタ、命令語、ロード/ストア/分岐命令の実効アドレス等のトレースデータをバッファに貯めていく。バッファがいっぱいになるとShadeはバッファの内容をMetatoolに渡す。Metatoolはこれら逐次的に実行されたトレースデータを元に、SMPにおいてアプリケーションを実行した場合の実行サイクル数を計算する。

図1にMetatoolの構成を示す。Metatoolはシミュレートするアーキテクチャのプロセッサ数だけプロセッサエレメント(PE)を生成する。

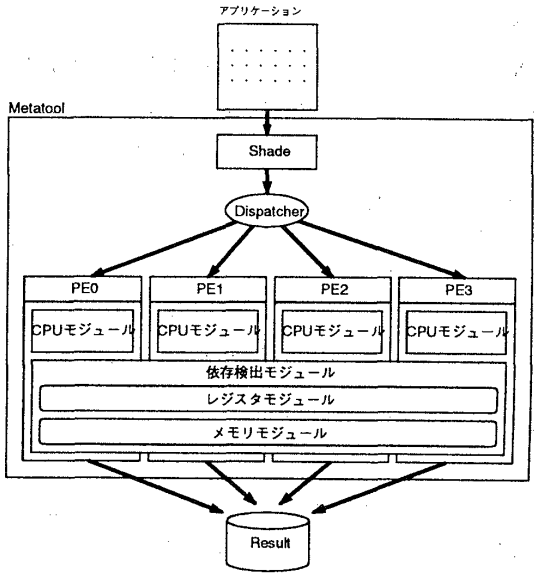

図1: Metatoolの構成

トレースデータはディスパッチャにおいてタスクと呼ばれる処理単位に分割されて各PEに分配される。ディスパッチャでの分割方式を変えることによってタスクの粒度を変化させることができる。現在は基本ブロックごとにタスクを分割し、各PEに対してラウンドロビンにタスクの分配を行っている。

各PEはプロセッサ内のアーキテクチャをシミュレートするCPUモジュールとプロセッサ間に生じる依存関係を検出するモジュールから構成されている。各CPUモジュールはレジスタ等のリソースを独立に持ち、Paratoolと同様にスーパースカラ方式のプロセッサをシミュレートすることができる。依存検出モジュールは各プロセッサのレジスタ間に生じる依存関係と、プロセッサ間で同一メモリアドレスにアクセスすることによって発生する依存関係を検出し、依存が解消するように命令パイプラインを制御する。

Metatoolでは他のプロセッサ上のレジスタに書き込まれたデータを参照するために必要なレイテンシ(refer)を指定することができる。referレイテンシを0とすることで、演算結果が得られたサイクルでレジスタやメモリへの書き戻しをすると同時に他のプロセッサがその値を参照できるようになる。

3 プロセッサ間データ依存

プロセッサ間に生じるデータ依存とその時の動作を、ALU命令で生じるレジスタ依存とロード/スト

* Metatool: a trace based SMP simulator.
† Kazuhiro SUZUKI
‡ Yasushi IWATA, Akira ASATO, Yasunori KIMURA
† FUJITSU LABORATORIES LTD.
‡ RWCP Multi-Processor Computing Fujitsu Laboratory

ア等のメモリアクセス命令で生じるメモリアクセス依存に分けて説明する。

3.1 レジスタ依存

真依存

他のプロセッサで生成された値を参照する場合が真依存である。この時値が生成されるまで実行することができないためパイプラインをストールさせる。

出力依存、逆依存

各プロセッサは独立したレジスタを持っているため、同一の番号のレジスタであってもプロセッサ間では異なるレジスタとなる。したがってプロセッサ間における出力依存や逆依存は考慮する必要はない。

3.2 メモリアクセス依存

真依存

他のプロセッサでメモリにストアした値をロードする場合がメモリアクセスによる真依存である。この時値がストアされるまでロードすることができないためパイプラインをストールさせる。

出力依存、逆依存

他のプロセッサがストアしたアドレスと同じアドレスに対してストアする場合が出力依存となる。また他のプロセッサがロードしたアドレスに対してストアする場合が逆依存となる。依存するタスク内の後続命令に影響を及ぼすことなくこれらの依存関係を解消するために、タスク内の全ての命令の実行が終了するまでパイプラインをストールさせる。

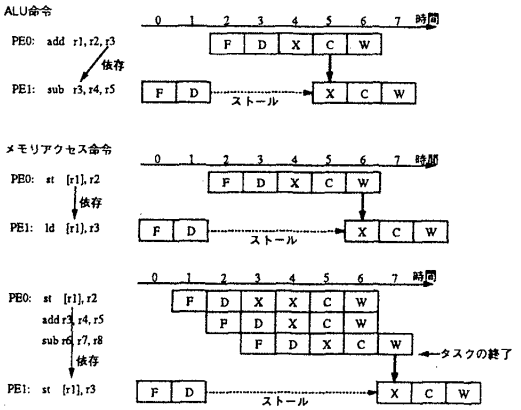

図 2: プロセッサ間の依存による実行遅延

これらの依存が発生した時のパイプライン動作例を図 2に示す。ここで各プロセッサのパイプラインは 5 段階構成 (F: フェッチ、D: デコード、X: 実行、C: キャッシュアクセス、W: ライトバック) で、refer レイテンシは 0 とする。

4 測定結果

Metatool では Paratool で得られる項目に加え、次の様な項目を測定することができる。

- 各 PE の実行サイクル数
- 各 PE が処理したタスク数
- プロセッサ間データ依存による平均遅延
- プロセッサ間データ依存の内訳

図 3に SPECint92 ベンチマークのシミュレーションを行った結果を示す。各ベンチマークに対して、命令発行幅 1 で PE 数 1、命令発行幅 4 で PE 数 1、命令発行幅 1 で PE 数 4 のそれぞれの場合について IPC(Instruction Per Cycle) を測定した結果である。各 PE は UltraSPARC を想定したパラメータを与えた。グラフから eqn 以外のベンチマークでは 4PE で実行した時に IPC の値が増加しており、SMP の効果が現われていることがわかる。ここで eqn はデータのアクセス頻度が高く、タスクに分割したことによってキャッシュのヒット率が低下したために IPC が他のベンチマークほど増えないのではないかと考えられる。

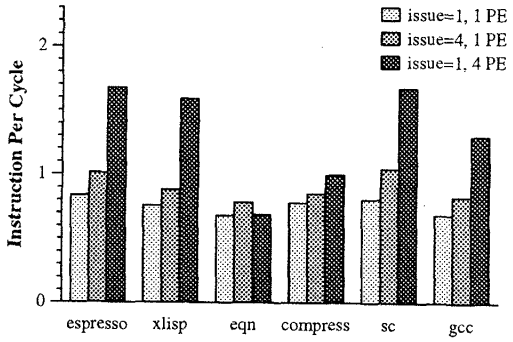

図 3: 測定結果

5 まとめ

トレースベース SMP シミュレータ Metatool の開発を行った。Metatool によってプロセッサ間の依存関係を考慮したシミュレーションが可能となる。

今後はキャッシュを含めたメモリアーキテクチャやバスの競合を評価するためにシミュレータの改良を行う予定である。

参考文献

[1]志村, 西本, 江口, 木村. スーパスカラプロセッサの性能評価 – paratool –. 情報処理学会アーキテクチャ研究会, 10. 1993.

[2]Robert F. Cmelik and David Keppel. Shade: A fast instruction-set simulator for execution profiling. Technical Report SMLI 93-12, UWCSE 93-06-06, SUN Labs, Inc., 1993.