2008 年 6 月 23 日 情報システム基盤学専攻合同輪講発表資料

Symbiotic Jobscheduling for a Simultaneous Multithreading Processor

著者： Allan Snavely,Dean M.Tullsen
出典： *Architectural Support for Programming Languages and Operating Systems-IX,pp.233-244,2000*
発表者： 本多 · 近藤研究室　0853013　鈴木翔平

1　導入

Simultaneous Multithreading (SMT)[1] とは単一 CPU 上で複数のスレッドを同時に実行することで、スレッドレベル並列性を向上させる技術である。これにより CPU 内の実行ユニット使用率が向上しシステム全体のパフォーマンスが向上する。

SMT 上で同時に実行されるスレッドは CPU 内実行ユニットを共有しているため、資源競合による性能低下が発生する。また、たまたまそのとき同時実行している命令の組合せによって競合が発生したりしなかったりするため、システム性能はダイナミックに変化し予測がしにくいという欠点がある。

本論文では、SMT プロセッサにおいて同時実行できるスレッド数 (SMT レベル) 以上の実行スレッドが存在した場合に、同時実行すると性能が低下しないスレッドの組合せを、CPU の性能カウンタによって得られる実行時情報を利用して動的に選択し実行するスレッドスケジューラを提案し、その有用性を検討する。

2　WeightedSpeedup

性能評価の方法について述べる。SMT のスケジューラ性能を評価するのに最も単純な方法はシステム全体の IPC の向上率といえる。なぜならスレッドレベル並列性の向上は IPC の向上にほかならないからである。しかしシステム全体の IPC 向上率を考えると、不公平なスケジュールが高い評価を得てしまうことがある。

例えば IPC の高いスレッド A と低いスレッド B を同時に実行する場合、A ばかり実行していればシステム全体では一時的に高い IPC が得られるが、B の IPC は極端に下がってしまう。本論文のスケジューラではスレッド間の優先度は同じであるため、このようなスケジュールは不適である。

そこで、各スレッドに IPC による重みを付けた WeightedSpeedup(WS) という評価指標を提案する。これは式 (1) によって計算される。

$$WS(t)\text{`}WeightedSpeedup\ in\ interval\ t\text{`} = \sum_{i=1}^{n} \frac{realized\ IPC\ job_i}{Single-threaded\ IPC\ job_i} \qquad (1)$$

式 (1) において、n は実行スレッドの総数、interval t はスレッドごとに定めるプログラム中で IPC を測定する部分、realized IPC はスレッドを同時実行している時の IPC、single-threaded IPC はスレッドを単一に実行した時の IPC である。なお single-threaded IPC は事前に測定した既知の値である。

3　スケジューラの概要

本論文で提案されているスケジューラは、実行時情報を取得する Sample フェーズと得られた情報に基づいて同時実行するスレッドを選択し実行する Symbiosis フェーズに分かれている。

3.1　Sample フェーズ

ランダムに選んだ 10 通りのスケジューリングについて、それぞれ 10^7 サイクル動かして CPU の性能カウンタから実行時情報を取得する。

3.2　Symbiosis フェーズ

Sample フェーズで得られた情報を基に 10 通りの中からベストなスケジュールを選択し、以降の処理をそのスケジュールによって実行する。

4　実験概要

実験では、実行スレッドの総数 ·SMT レベル · タイムスライス間で入れ替えるスレッドの数が違う 13 種類の SMT プロセッサを仮定して、シミュレーションによる評価を行った。

5　スレッドの相性が性能に及ぼす影響

予備実験として Sample フェーズのみを動かして WS を測定し、スレッドの相性が性能に及ぼす影響を測定した。結果を図 1 に示す。この図より、WS の差が平均 8%、最大 25%あることがわかる。このことから、スレッドの組合せが性能に大きな影響を与えていることがわかった。

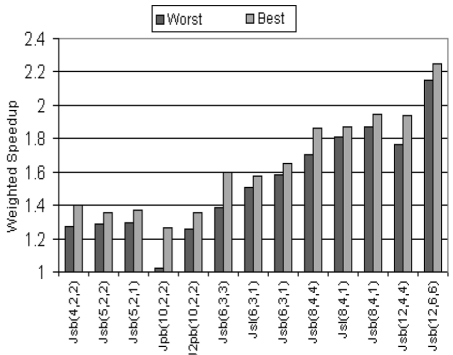

図 1: 各 Jobmix における WS の Best と Worst

6 実行時情報に基づいたスケジューリング

プログラム実行中に WS を測定することはできない。そこで本論文の提案するスケジューラは、CPU の性能カウンタの情報により WS が向上するスレッドの組合せを予測する手法を取る。性能カウンタから得る情報は IPC、Dcache、FQ、FP、Sum2、Allconf、Diversity、Balance、Composite、Score の 10 種類である。

6.1 実験

各 Jobmix において Sample フェーズで情報を取得し、Symbiosis フェーズで WS を測定する。そして性能カウンタの情報がどの程度 WS の向上を予測できているかを検討する。

6.2 Jsb(6,3,3) の結果および考察

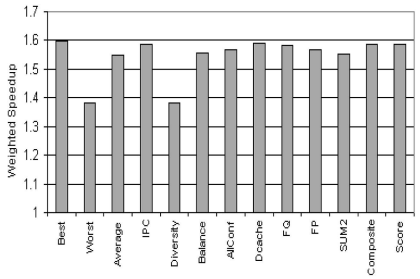

図 2: 性能カウンタの情報と WS

図 2 より、Diversity を除く全ての情報が Worst を避けられていること、特に IPC、Dcache,FQ、Composite、Score では Best の 2%以内、Average より 9%の向上、Worst より 17%の向上を達成している。

6.3 性能カウンタから得られた情報の考察

性能カウンタから得られた各情報についての考察を以下に列挙する。

- IPC···WS が高いスケジュールを選択する確率はあまり良くなかった。これは Sample フェーズにおいて IPC の高いスレッドがシステムを占有してしまう場合があるためである。
- Diversity···WS が高いスケジュールは選択できなかった。Sample フェーズ実行中の命令の割合が同程度でも実行ユニットの競合は発生するため、実行ユニットの利用率向上とは関係が無かった。
- Balance··· 高確率で WS の高いスケジュールを選択した。
- AllConf、FP、FQ、SUM2··· 確率はあまり高くなかった。原因として 2 つ考えられる。1 つは高い資源競合が発生することは実行ユニット使用率が高いということを示していることである。もう 1 つは実験で用いたプロセッサモデルは浮動小数点数ユニットおよび命令キューの競合を回避する仕組みを持っているためである。しかし、FP、FQ、SUM2 が低くなることは WS 向上に少なからず関連があると思われる。
- Dcache··· 一貫性のある選択ができなかった。これは、カーネルがキャッシュミスを少なくするように働くため、同時実行するスレッドの相性とは関係が無かったためである。
- Composite···WS が Average 以上になるスケジュールを選択する確率が最も高かった。複数の情報を用いれば良い結果が得られることを示している。
- Score··· 最も Best を選ぶ確率が高かった。Composite と同様の理由で、複数の情報を利用することが良い結果が得られるにつながっている。

Score によるスケジューリングでは WS が Worst から平均 22%、Average からは平均 7%向上するスケジュールを選択することができた。このことから、複数の情報を利用することで WS の高いスケジュールを選択する確率があがること、また CPU 性能カウンタによる動的なスケジューリングが効果的であることがわかった。

7 まとめ

本論文では SMT における、CPU 性能カウンタの情報による動的なスケジューリングを行うスケジューラを提案した。またシミュレータ上の実験によってそれの有用性を示した。

参考文献

[1] Tullsen,D.M.:Simultaneous Multithreading :Maximizing On-Chip Parallelism.Proc.22nd Annual International Symposium on Computer Architecture,1995