LC-010

SMT プロセッサの実行時性能予測のためのハードウェアリソース競合解析
Analysis of hardware resource conflicts for runtime performance prediction of SMT processors

佐藤雅之* 船矢祐介* 小寺功* 滝沢寛之*† 小林広明*†
Masayuki Sato Yusuke Funaya Isao Kotera Hiroyuki Takizawa Hiroaki Kobayashi

1 緒言

これまで，マイクロプロセッサの性能向上には，演算ユニットの増加，クロック周波数の向上といったアプローチがとられてきた．演算ユニットを増やす事によって，命令レベル並列性 (Instruction Level Parallelism, ILP) に基づき，より多くの依存関係のない命令を並列に処理することが可能である．しかし，プログラム中に含まれる並列実行可能な命令数には限界があり，既に十分な数の演算ユニットを搭載している近年のマイクロプロセッサにおいては，演算ユニットの増加による性能向上を期待できない．また，クロック周波数の増加は回路の動作速度を上げ，命令の処理速度を速くすることができるが，実際には消費電力と発熱量の増加の問題により，これ以上クロック周波数を上げるのは困難である．

そのような状況の下，Simultaneous Multi-Threading(SMT) プロセッサ [1] が登場した．SMT プロセッサは，スレッドレベル並列性 (Thread Level Parallelism, TLP) に基づき，1 サイクル中に複数の異なるスレッドから依存関係のない命令を抽出し，実行する機構を備えている．この機構により，より多くの命令を同時に実行できるようになった．そのため，演算ユニットの使用率が向上し，プロセッサ全体の実効性能が向上した．

しかし，SMT ではプロセッサのリソースを複数のスレッドで共有するため，1 スレッドのみ実行する場合と比較するとスレッドあたりの性能が低下する．また，その性能低下率はスレッドの性質やプロセッサの構成に依存する．図 1 は 2 つのスレッドを SMT プロセッサ上で同時実行した際の性能低下を示している．ただし，この図においてスレッドあたりの性能とは，単独実行の場合のそれぞれのスレッドの平均 IPC(Instruction Per Cycle) により正規化した値である．このグラフから SMT プロセッサ上で 2 スレッドを実行した場合，その組合せによってスレッドあたりの性能低下率に差があることがわかる．よって，1 つ 1 つのスレッドの性能低下を抑え，SMT プロセッサ全体の実効性能を向上させるには，組み合わせた際の性能が高くなるスレッドの組を選択・実行するスレッドスケジューリングが必要である．

関連研究として，船矢ら [2] によるスレッドスケジューリング手法が挙げられる．船矢らの手法は，SMT 優先度という指標に基づいて最も高い性能を発揮するスレッドの組を選択する．しかし，SMT 優先度を取得するためには，そのスレッドを事前に実行し，スレッドの性能を取得（プロファイリング）しなければならない．そのため，あるスレッドの組がどのくらいの性能を持つかは，実際にプロファイルするまでわからない．また，どのリソースにおいてどのように競合が発生しているかについては述べられていない．

本報告では，SMT プロセッサ上で 2 つのスレッドが実行された場合，どの共有リソースでどのように競合が起きているか解析する．その解析の結果から，SMT の性能を最大限に引き出すために，実行時性能予測に基づくスレッドスケジューリングが必要である事を明らかにする．

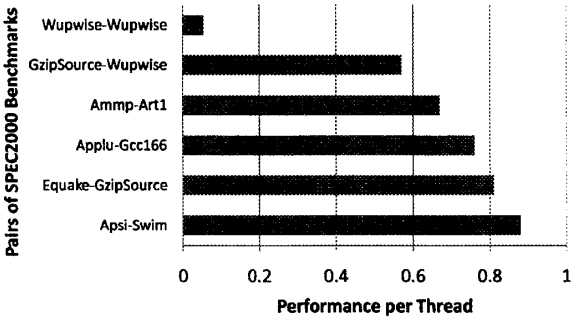

図 1 SMT プロセッサで 2 スレッド同時実行した場合の 1 スレッドあたりの性能低下

2 SMT におけるリソース競合

2.1 SMT プロセッサのアーキテクチャ

図 2 に一般的なスーパスカラと SMT プロセッサの構成を示す．プロセッサはスレッドの実行状態を保存するためにアーキテクチャステートを有する．アーキテクチャステートはレジスタ，プログラムカウンタ，その他のマシンステートレジスタで構成される．一般のスーパスカラでは同時に 1 スレッドのみ実行されるためアーキテクチャステートは 1 つしか持たない．しかし，SMT プロセッサは複数スレッドを同時実行するため複数のアーキテクチャステートを持っている．

SMT プロセッサの利点は，演算ユニットを複数スレッドで共有する事である．このため，演算ユニットの利用効率が上がり，全体の実効性能が向上する．また，SMT を実現するために必要なダイサイズの増加は小さく，ハードウェアコストが少ない．しかし，他のスレッドとリソースを共有しているため，多くの場合に個々のスレッドの性能が低下することが欠点としてあげられる．相性の悪いスレッド同士の組合せの場合には，全体の実効性能も低下する可能性がある．本研究では、共有する主なリソースとして演算ユニットとキャッシュメモリに着目する。

2.2 演算ユニット

演算ユニットはプロセッサで実際に演算を行う部分である．さまざまな種類の演算に対応するため，演算の種類やデータ型によってどのユニットを使用するか異なってくる．

もし，2 つのスレッドが同じ演算ユニットを使用する場合は，片方のスレッドが演算ユニットを使用している間，もう一方のスレッドはその演算ユニットを利用することはできない．そして第 1 のスレッドの演算実行が終了した後に，第 2 のスレッドの命令が実行に移ることができる．このように，第 2 のスレッドで実行待ちが発生した場合，そのスレッドの実行時間が延び

* 東北大学大学院情報科学研究科
† 東北大学情報シナジー機構

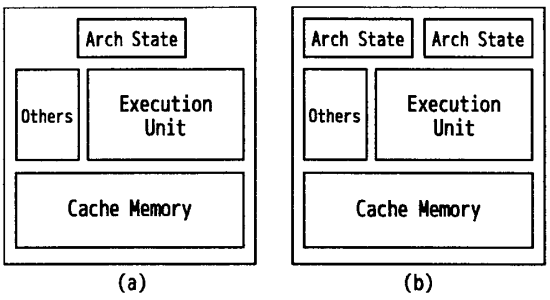

図 2　(a) スーパスカラプロセッサ (b)SMT プロセッサ

るので性能低下につながる．

特に，SMT プロセッサでは複数のスレッドから依存性の低い命令をより多く抽出する事ができるため，並列に実行可能な命令が増加し，演算ユニットの使用率が向上する．演算ユニットの使用率が高くなることによって，競合が発生する可能性も高まる．

2.3　キャッシュメモリ

キャッシュメモリはプロセッサ内にある高速なメモリである．メインメモリに保存されている命令やデータの中で再利用されやすいデータの複製を保持し，プロセッサコアからの要求に対してより速くデータを供給できるようにする．しかし，要求された命令やデータがキャッシュ上にない場合はキャッシュミスとなり，より遅延の大きいメインメモリからデータを供給しなければならない．そのためキャッシュミス率の増加によって実行に必要な命令やデータの供給が遅れるため，性能が著しく低下する．

SMT プロセッサにおいて複数のスレッドを実行する場合，それぞれのスレッドが他のスレッドの存在を考慮せずに独自にキャッシュを利用するため，競合性および容量性のキャッシュミスが増加する．

3　リソース競合の解析と考察

SMT プロセッサでの共有リソースの競合を確認するため，シミュレーションを行った．まず 1 つのスレッドを単独で実行し，各ベンチマークプログラムのリソースの使用状況を明らかにする．次に 2 スレッドを同時に実行した結果と比較し，どのリソースで競合が発生しているかを確認する．

実験では M5 シミュレータ [3] を用いて SMT プロセッサをシミュレートする．また，M5 上で実行するプログラムは SPEC CPU2000[4] から選択し，1 つのプログラムをプロセッサ上で 1 スレッドとして実行する．2 スレッド同時実行の場合は 2 つのプログラムを選んで同時に実行する．プロセッサのパラメータを表 1 に示す．

以下，結果が特徴的な組に着目して議論する．図 3，図 4 及び図 5 に GzipSource と Wupwise を同時に実行した場合の実験結果を示す．図 3 は 1 スレッド実行時の性能と比較した場合の Wupwise の性能低下を表している．ここで性能は，2 スレッドを同時に実行した場合の IPC を，1 つのスレッドを単独で実行した場合の IPC で正規化した値 (Normalized IPC) で表している．このグラフから，断続的に性能が 1 割程度まで低下している．図 4 は Wupwise の L1 命令キャッシュのミス率を表している．1 スレッド時のミス率 (Wupwise) と比較して，SMT プ

表 1　実験に用いたプロセッサのパラメータ

Parameter	Value
L1 I-cache	32KB, 1 cycle latency
L1 D-cache	32kB, 1 cycle latency
L2 cache	1MB, 14 cycle latency
Main memory	100 cycle latency
fetch width	8 insts
fetch policy	icount
branch predictor	hybrid, 3 cycle penalty
issue width	8 insts
int ALUs	6
int mult/div units (issue latency)	2 19 cycle (div)
fp ALUs	4
fp mult/div units (issue latency)	2 12 (div), 24 cycle (sqrt)
load/store units	4
commit width	8 insts

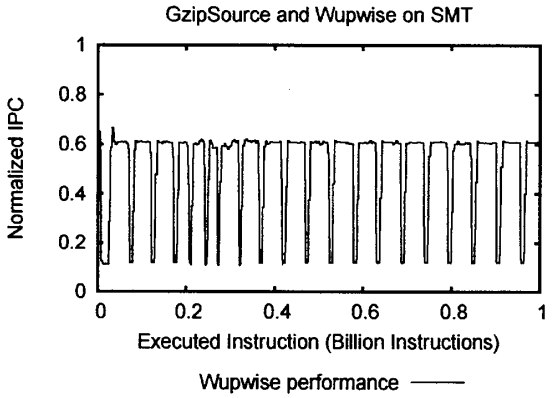

図 3　Wupwise(vs GzipSource) の性能低下

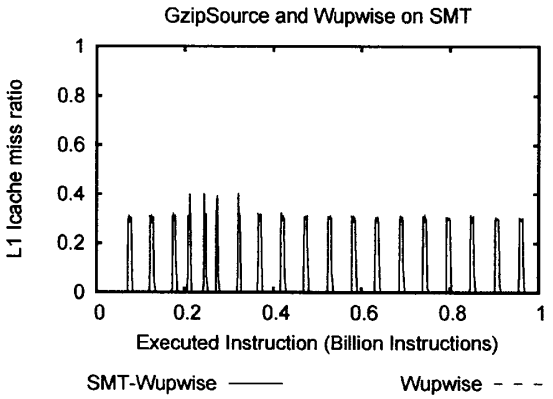

図 4　Wupwise(vs GzipSource) の L1 命令キャッシュミス率

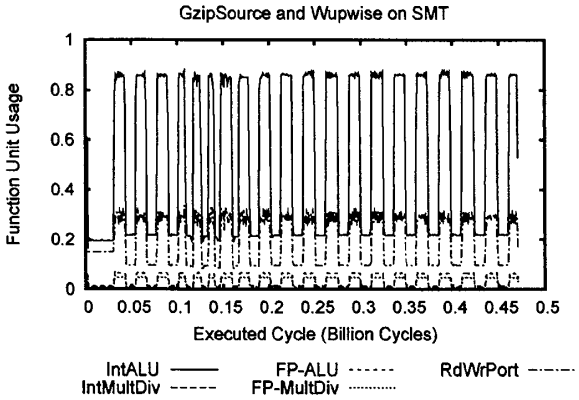

図 5　GzipSource と Wupwise の組での演算ユニット使用率

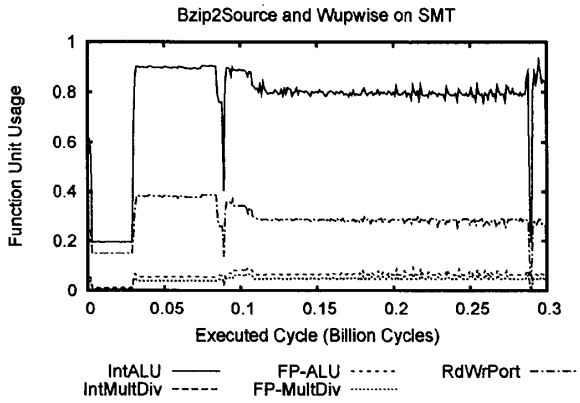

図 6　Bzip2Source と Wupwise の組での演算ユニット使用率

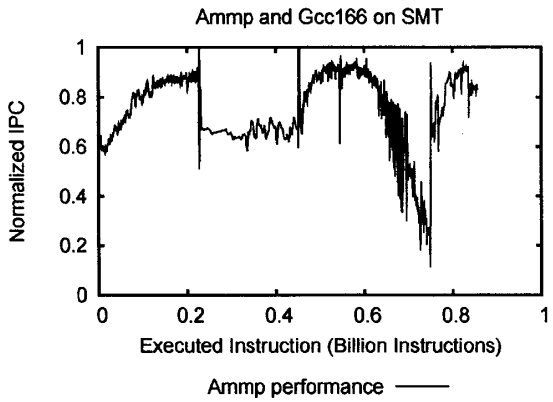

図 7　Ammp(vs Gcc166) の性能低下

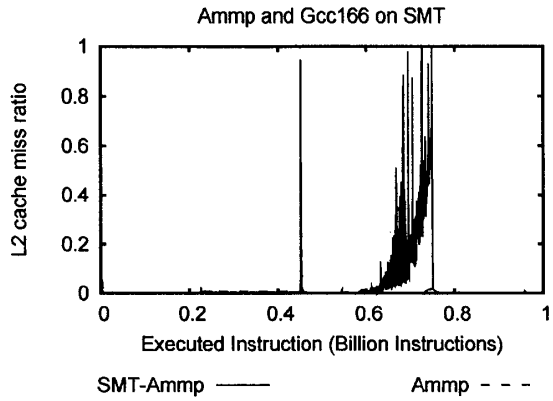

図 8　Ammp(vs Gcc166) の L2 キャッシュミス率

ロセッサによる 2 スレッド同時実行の場合 (SMT-Wupwise) のミス率が，図 3 と同様に断続的に上昇している事がわかる．この時，L1 データキャッシュと L2 キャッシュではミス率に変化が表れなかった．よって，図 3 の性能低下は L1 命令キャッシュにおけるミスの増加が原因になっていると考えられる．同時実行した GzipSource についても同様の傾向が表れている．次に，図 5 はこの組での演算ユニットの使用率を表したグラフである．整数加減演算ユニット (IntALU) に注目すると，使用率の増減が繰り返し発生している．使用率が低い場合の性能低下は，前述の通り L1 命令キャッシュミスの影響が大きいと考えられる．使用率の高い場合では 9 割程度まで増加している．この時，各キャッシュにミス率の変化がなかったので，命令やデータが演算ユニットに淀みなく供給された分，IntALU ユニットの使用率が上昇したと考えられる．しかし，この場合でも性能が 6 割まで低下している．これは，IntALU ユニットに処理が集中することによって競合が発生し，これがボトルネックとなって性能低下を招いたためと考えられる．

演算ユニットの使用率が特に高い他の例として Bzip2Source と Wupwise の組が挙げられる．図 6 はこの組を同時実行した際のプロセッサの演算ユニットの使用率を表している．このグラフから整数加減演算ユニット (IntALU) の使用率が非常に高い．しかし，調査したすべての組のうち，この組のスレッドあたりの性能は比較的低い事が分かった．この組では，L1·L2 キャッシュについてはミス率の変化は見られなかったため、キャッシュによる性能低下は考えられない．このため IntALU ユニットにおける処理の集中がボトルネックとなって，スレッドあたりの性能が低く抑えられたと考えられる．

図 7 及び図 8 は Ammp と Gcc166 の組を実行した際の Ammp について表したグラフである．図 7 は Ammp の性能低下を表したグラフである．6 億命令を経過した付近から大きく性能が低下しはじめている．この時，L1 キャッシュには目立った変化はなかった．図 8 は Ammp の L2 ミス率を表している．このグラフから，6 億命令付近から L2 ミス率が次第に上昇しており，性能低下のグラフとほぼ似たような傾向を示している事がわかる．このため，L2 キャッシュのミス率上昇による性能低下が発生したと考えられる．

図 9，図 10，及び図 11 は Twolf と Mcf の組を同時実行した際の Twolf について表したグラフである．図 9 のグラフより，5 億命令付近から大きく性能低下しているのがわかる．図 11 は Twolf の L2 キャッシュミス率を表しているが，5 億命令付近から高い事がわかる．この事から，5 億命令以降の性能低下は L2 キャッシュミス率の上昇によるものだと考えられる．また，この L2 キャッシュミス増加による性能低下が大きいのに対して，2 億命令から 5 億命令の区間では L1 キャッシュのミス率が上昇しているにもかかわらず，性能低下はそれほど大きくない．このため，L2 キャッシュのミス率の増加は，L1 キャッシュのミス率増加より性能へ与える影響が大きいと考えられる．

以上の実験結果より，スレッドの組合せによって異なる共有リソースで競合が発生することが明らかになった．特に，L1

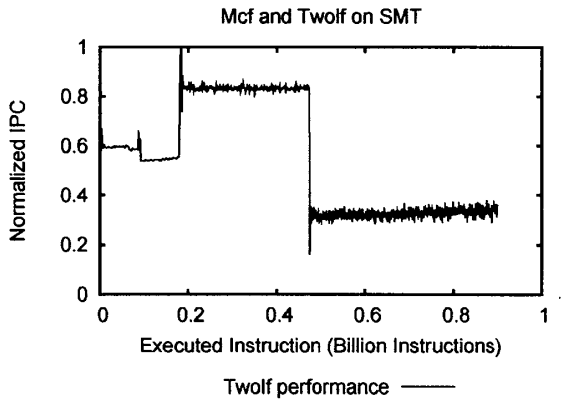

図 9　Twolf(vs Mcf) の性能低下

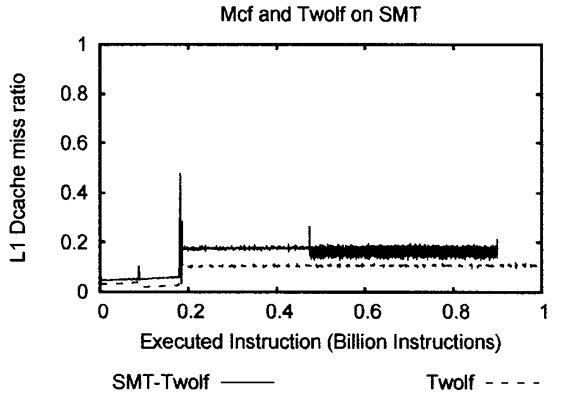

図 10　Twolf(vs Mcf) の L1 データキャッシュミス率

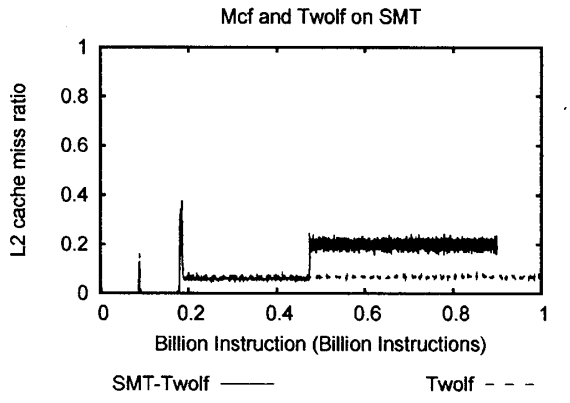

図 11　Twolf(vs Mcf) の L2 キャッシュミス率

命令キャッシュと L2 キャッシュでの競合が特に大きく影響を及ぼすことがわかった．また，L1 データキャッシュと演算ユニットにおける競合による性能低下もあったが，これらの競合による性能低下への影響は比較的小さいことがわかった．さらに，各共有リソースの利用状況は実行時に動的に変化することも観測された．

これらの競合を回避する事ができれば SMT プロセッサの実効性能を向上させる事ができる．実験結果から競合の発生は組合せ・リソース・時間によって異なるため，それらを考慮して実行中にスレッドの組合せを変更する動的なスレッドスケジューリングの必要性が明らかになった．

4　結論

本報告では，SMT プロセッサ上で実行されるスレッドの性能を向上させるため，スレッドの性能低下の要因となっているリソース競合を解析した．

実験の結果，SMT プロセッサの共有リソースである L1 キャッシュ，L2 キャッシュ，演算ユニットにおいて競合が発生している事を確認した．また，これらの競合はスレッドの組合せだけでなく，実行時間の経過に従って同じスレッドでも競合が発生する場合としない場合がある事が明らかになった．これらの競合を回避すれば性能を向上させる事ができるため、競合の影響が大きいスレッドの組を回避するようにスレッドを入れ換える事ができる動的なスレッドスケジューリングを行う必要がある．

今後の課題として，動的なスレッドスケジューリングを行うために，各共有リソースの利用状況を動的に観測・定量化し，それを基に性能予測を行う事が必要である．

謝辞

本研究の一部は，文部科学省科学研究補助金基板研究 (B)（課題番号 18300011）による．

参考文献

[1] Susan J. Eggers, Joel S. Emer, Henry M. Levy, Jack L. Lo, Rebecca L. Stamm, and Dean M. Tullsen. Simultaneous Multithreading: A Platform for Next-Generation Processors. *IEEE Micro*, Vol. 17, No. 5, pp. 12–19, 1997.

[2] 船矢祐介, 小寺功, 滝沢寛之, 小林広明. スレッド特徴量に基づくマルチコアプロセッサスケジューリング. 第 5 回情報科学技術フォーラム 情報科学レターズ, pp. 37–40, 2006.

[3] Nathan L. Binkert, Ronald G. Dreslinski, Lisa R. Hsu, Kevin T. Lim, Ali G. Saidi, and Steven K. Reinhardt. The M5 Simulator: Modeling Networked Systems. *IEEE Micro*, Vol. 26, No. 4, pp. 52–60, 2006.

[4] John L. Henning. SPEC CPU2000: Measuring CPU Performance in the New Millennium. *IEEE Computer*, Vol. 33, No. 7, pp. 28–35, 2000.

[5] Isao Kotera, Hiroyuki Takizawa, and Hiroaki Kobayashi. A Fair-Sharing and Power-Aware L2 Cache System for Chip Multiprocessors. *CoolChips X Proceedings*, p. 139, 2007.

[6] Deborah T. Marr, Frank Binns, David L. Hill, Glenn Hinton, David A. Koufaty, J. Alan Miller, and Michael Upton. Hyper-Threading Technology Architecture and Microarchitecture. *Intel Technology Journal Q1*, Vol. 6, pp. 4–15, 2002.

[7] Tipp Moseley, Dirk Grunwald, Joshua L. Kihm, and Daniel A. Connors. Methods for Modeling Resource Contention on Simultaneous Multithreading Processors. *ICCD*, Vol. 00, pp. 373–380, 2005.

[8] Francisco J. Cazorla, Peter M.W. Knijnenburg, Rizos Sakellariou, Enrique Fernández, Alex Ramirez, and Mateo Valero. Predictable Performance in SMT Processors: Synergy between the OS and SMTs. *IEEE Transactions on Computers*, Vol. 55, No. 7, pp. 785–799, 2006.