社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－EMB－10 (15)
2008／11／28

キャッシュメモリの性能オーバーヘッドを低減させる タスクスイッチ手法の検討

芹澤 光範　　杉原 真

豊橋技術科学大学大学院工学研究科情報工学専攻, 〒 441-8580 愛知県豊橋市天伯町雲雀ヶ丘 1-1
E-mail: mserizawa@emb.ics.tut.ac.jp, sugihara@ics.tut.ac.jp

要 旨

現在のコンピュータシステムにおいて，キャッシュメモリの導入やマルチタスク処理は必要不可欠である．マルチタスク環境において，タスクスイッチ発生時にキャッシュメモリから主記憶へのデータの退避による性能オーバーヘッドが発生する．本稿では，この性能オーバーヘッドを削減するために，原因となるダーティラインに注目する．プログラム実行時のダーティライン数の変動を解析した結果，ダーティライン数はプログラム特有の変動をすることがわかった．

Examination of a Task Switching Method which Reduces the Performance Overhead of Cache Memory

Mitsunori Serizawa　　Makoto Sugihara

Department of Information and Computer Sciences, Toyohashi University of Technology,
1-1 Hibarigaoka, Tenpakucho, Toyohashi, Aichi 441-8580 Japan
E-mail: mserizawa@emb.ics.tut.ac.jp, sugihara@ics.tut.ac.jp

Abstract

Using cache memory and multitask processing are indispensable to state-of-the-art computer systems. A task switch on a multitasking system makes data evaculated from cache memory to main memory. A dirty line is a major factor to cause a performance overhead. In this paper we examine the number of dirty lines which dynamically change during a program execution. The result of our analysis clearly shows that the number of dirty lines does a change peculiar to every program.

1 はじめに

現在のコンピュータシステムにおいて，高速な処理を行うためのキャッシュメモリの導入，及び，割り込みによる即応性を実現するマルチタスク処理が必要不可欠なものとなっている．キャッシュメモリは，使用頻度の高いデータを蓄積しておくことにより，低速なメインメモリへのアクセスを減らすことができ，処理を高速化することができる．また，マルチタスク処理は，プロセッサの処理時間を非常に短い単位に分割し，優先度に応じてタスクをプロセッサに割り当てることによって，複数の処理を同時に行っているようにみせている．

マルチタスク環境において，タスクスイッチが発生した場合，キャッシュメモリの性能オーバーヘッドが発生してしまう．キャッシュラインとはキャッ

シュメモリと主記憶とのデータのやり取りを行う単位である．キャッシュメモリと主記憶との一貫性が保たれている状態をクリーン，保たれていない状態をダーティという．特に一貫性が保たれていないキャッシュラインをダーティラインと呼ぶ．タスクスイッチ時にダーティラインを主記憶に書き戻す必要があり，その書き戻し作業が性能オーバーヘッドとなりプロセッサの性能低下を引き起こす原因となる．

本研究はタスクスイッチ時に発生する性能オーバーヘッドを低減し，プロセッサの性能向上を目標としている．その性能オーバーヘッドを低減することでプロセッサの性能が向上すると考え，性能オーバーヘッドの原因となるキャッシュライン中のダーティライン数に着目する．その準備段階として，プロセッサシミュレータである SimpleScalar[1] にダーティライン数を観測する機能を実装した．ベンチマークプログラムとして，SPEC CPU2000[2] を用いてキャッシュ動作をシミュレートしダーティライン数の解析を行った．実験の結果を考慮してタスクスイッチ時に発生する性能オーバーヘッドを低減するタスクスイッチ手法の検討する．本稿の構成は次の通りである．2 節で諸準備として，キャッシュメモリ，キャッシュメモリの書き込み方式，マルチタスク処理，ダーティラインについて説明する．3 節では関連研究とタスクスイッチ時の性能オーバーヘッドについて説明する．4 節では実験と考察を行い，5 節で性能オーバーヘッドを低減するタスクスイッチ手法を検討する．6 節で本稿をまとめるとともに，本研究の今後の課題について述べる．

2 諸準備

2.1 キャッシュメモリ

キャッシュメモリとは，図 1 にあるようにプロセッサと主記憶装置の間にある記憶装置であり，アクセスする頻度の高いデータや命令をキャッシュメモリに保存しておく．主記憶よりもプロセッサに近い位置にあるキャッシュメモリからデータや命令を読み込むことで，メモリアクセスの時間を短縮できるので，プロセッサ性能の向上に繋がるというものである．そこで，限られた資源で，どのようにデータ及び命令をキャッシュメモリにマッピングするか，どのようにキャッシュメモリを検索するか，また，キャッシュメモリのデータを置き換える必要が生じた場合に，どのような方法で置き換えるか，といった点を考慮し，最も効率のよい方式を選択する必要がある．

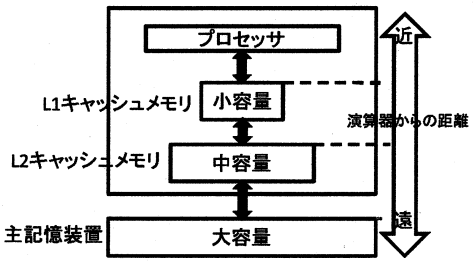

図 1: キャッシュメモリの階層構造

2.1.1 キャッシュメモリの書き込み方式

キャッシュメモリの書き込み方式には次の 2 つが挙げられる．

ライトスルー方式

ライトスルー方式とは，図 2 に示すように，ストア命令によりキャッシュメモリに書き込む際に主記憶にも同時に書き込む方式 [3] である．この方式では，主記憶に常に最新の内容が書き込まれるため，キャッシュメモリと主記憶の内容の一貫性が保たれる．また，書き込みの際の動作が常に同じでその制御がしやすい，置換対象となったラインを主記憶に書き込む必要がない，主記憶にエラー訂正能力があればキャッシュは検出能力だけあれば良い，などの利点がある．しかし，主記憶に書き込みにいくたび，プロセッサが書き込み待ちになり高速化が望めないことが欠点として挙げられる．

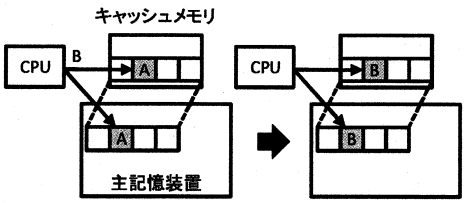

図 2: ライトスルー方式

ライトバック方式

ライトバック方式とは図 3 に示すように，ストア命令の際，キャッシュメモリにのみ書き込む方式 [3] である．この方式では，毎回主記憶に書き込みに行かないので，書き込み待ちの起こる機会が少ないと

いう利点がある．しかし，キャッシュメモリと主記憶間の内容の一貫性が保たれないこと，読み出しミスがあり，置換が必要な場合にまず，キャッシュメモリより追い出すブロックを主記憶に書き込む必要があるなどの欠点がある．

ライトバック方式において，キャッシュメモリから主記憶への書き戻し操作には実行時間がかかってしまう．したがってキャッシュラインを置き換える際に，そのラインを書き戻す必要があるかどうか判定したい．そのためにキャッシュメモリにダーティビットを追加する．キャッシュラインに書き込みが生じた場合，新しい値はキャッシュメモリ内の対応ラインにのみ書き込まれ，主記憶との一貫性が保たれなくなる．この状態をダーティといい，ダーティビットがセットされダーティラインとなる．ダーティラインに他のデータを書き込む際に，一貫性を保つために主記憶への追い出しが生じ，性能オーバーヘッドが発生する．

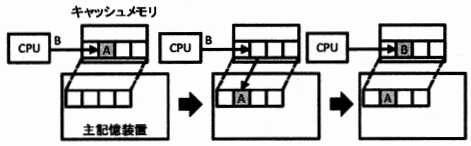

図 3: ライトバック方式

2.2 マルチタスク処理

マルチタスク処理は，図 4 のようにプロセッサの処理時間を非常に短い単位に分割し，タスクを優先度に応じて割り当てることによって，複数のタスクを同時に実行されているようにみせている．タスクの切り替えのオーバーヘッドや，キャッシュのミスヒット率の上昇などのコストがかかるが，入出力待ちなどであるタスクの実行が止まっても他のタスクが実行されるため，全体としてスループットの上昇が期待できる．実行中のタスクを一時的に中断する動作をプリエンプション [4] といい，マルチタスク処理の実現方法のうち，プロセッサを OS が管理しない方式 [4] をノンプリエンプティブマルチタスクという．それに対し，プロセッサを OS が管理する方式 [4] をプリエンプティブマルチタスクという．ノンプリエンプティブマルチタスクではタスクの挙動によって OS の安定性が大きく左右されてしまうため，現在はプリエンプティブマルチタスクが主流である．

図 4: マルチタスク処理

3 タスクスイッチ時の性能オーバーヘッド

これまでのタスクスイッチ手法では，レジスタの退避・復元といったタスク切り替え時間の際に発生する性能オーバーヘッドを削減する方法として，SPARC アーキテクチャで採用されているレジスタウインドという機能を用いた高速なタスクスイッチ手法が提案されている [5]．また，割り込み専用のレジスタセットを複数用いる手法 [6]，リアルタイム OS の処理をハードウェアで実装する手法 [7]，複数のレジスタセットと後続タスクを予測する手法 [8] などがある．これらの手法では，主にコンテキストスイッチによるレジスタの退避・復元の際に発生する性能オーバーヘッドを削減する手法であり，キャッシュメモリについては考慮されていない．そこで我々はタスクスイッチ時にキャッシュメモリから主記憶への書き戻しによって発生するキャッシュメモリの性能オーバーヘッドを削減する手法を検討する．

プログラム実行中にタスクスイッチが発生した場合，キャッシュメモリの内容を主記憶へ書き戻し，次のプログラムのデータを主記憶からキャッシュメモリに格納するという動作が行われる．通常，主記憶への書き込みは 100 サイクル以上の時間がかかってしまい，プロセッサの処理速度が著しく低下してしまう可能性がある．この，余分にかかってしまう時間のことを性能オーバーヘッドという．タスクスイッチが発生した時，図 5 のようにキャッシュメモリがすべてクリーンだった場合，主記憶に追い出すデータが存在しないので性能オーバーヘッドは発生しない．一方で，図 6 のようにキャッシュメモリ中の多くのラインがダーティな場合にタスクスイッチが発生すると，主記憶に追い出すデータが多数存在するので性能オーバーヘッドが発生する．ダーティ

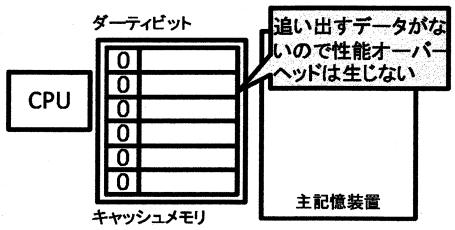

図 5: キャッシュメモリがクリーンな場合

ラインが多ければ，主記憶へ追い出すキャッシュラインが多くなり，ダーティライン数に比例して追い出しによる性能オーバーヘッドは大きくなる．このように，タスクスイッチ時にダーティラインを追い出すことで性能オーバーヘッドが発生してしまうので，タスクスイッチ時には，ダーティライン数を考慮に入れた方が良いと考えられる．

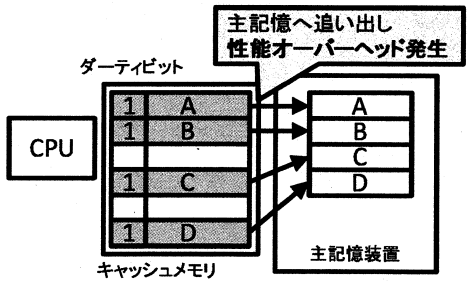

図 6: キャッシュメモリがダーティな場合

4 実験

マルチタスク環境において，タスクスイッチ発生時にキャッシュメモリから主記憶へのデータの退避による性能オーバーヘッドが発生する．我々はこれまでに，マルチタスク環境で性能オーバーヘッドの原因となるダーティライン数について実験的に調査している [9]．本節では，文献 [9] の実験を詳細化するとともに，ダーティライン数が増減する原因を解明する．

4.1 実験環境

本実験では，実行駆動方式のアーキテクチャシミュレータである SimpleScalar 3.0d/PISA[1] を利用した．SimpleScalar スイートの中から，マルチレベルキャッシュシミュレータである sim-cache に L1 データキャッシュのダーティライン数，読み出しミス回数，書き込みミス回数をカウントする機能を実装し，測定を行った．また，ベンチマークプログラムとして，SPEC CPU2000 のうち parser, vortex, gcc, mcf を対象とした．各ベンチマークプログラムの詳細を表 1 に，各キャッシュパラメータを表 2 に示す．

表 1: ベンチマークプログラム

	対象分野	内容
181.mcf	組み合わせ問題	最小コストフロー法による画像生成
197.parser	言語	自然言語処理
255.vortex	DB	オブジェクト指向データベース
176.gcc	言語	C コンパイラ

表 2: 各キャッシュパラメータ

	セット数	ラインサイズ	連想度 n	キャッシュサイズ
L1 I$	256	32	1	8 kB
L1 D$	2048	32	1	64 kB
L2 U$	1024	64	4	256 kB

4.2 実験結果

各ベンチマークプログラムでの実行された命令数に対するダーティライン数，読み出しミス回数，書き込みミス回数を図 7〜図 10 に示す．

ダーティライン数について

図 7 の parser では，プログラム開始直後，ダーティライン数がキャッシュメモリが持つの最大ライン数まで短時間で上昇している．その後，細かく増減を繰り返しながら全体的にダーティライン数は減少していく傾向が見られる．図 8 の vortex では，プログラム開始直後，ダーティライン数は増えるが parser のように短時間には増えず，段階的に増えている．また，vortex ではダーティライン数の減少がほとんど見られない．図 9 の gcc では，parser と同様にプログラム開始直後にダーティライン数がキャッシュメモリが持つの最大ライン数まで短時間で上昇し，その後，大きな増減を繰り返している．図 10 の mcf でも parser と同様にプログラム開始直後，ダーティライン数がキャッシュメモリが持つの最大ライン数まで短時間で上昇する．4×10^7 命

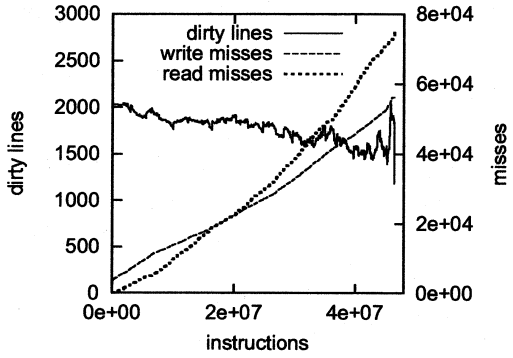

図 7: 197.parser

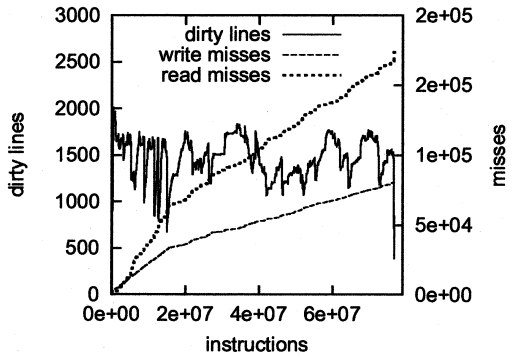

図 9: 176.gcc

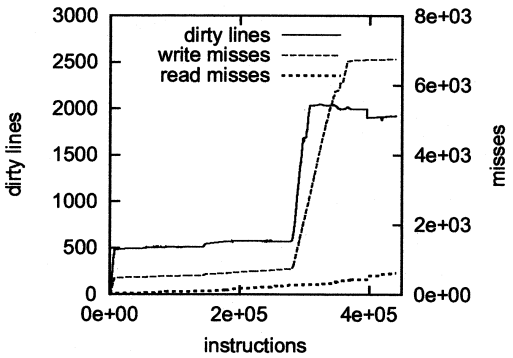

図 8: 255.vortex

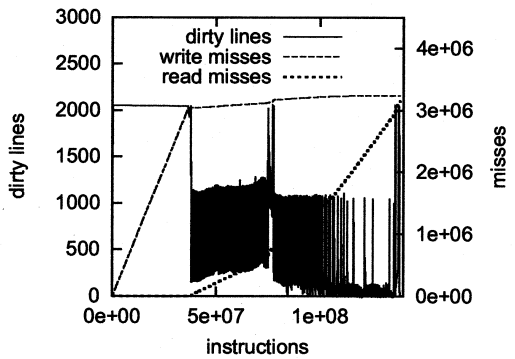

図 10: 181.mcf

令まではダーティライン数の増減はキャッシュライン数の最大値付近で落ち着いているが，4×10^7 命令以降，急激にダーティラインが減少し，さらに急激な増減を繰り返している．上記に述べたように，ダーティライン数はプログラムごとに特有の増減をすることがわかる．

読み出し/書き込みミス回数について

まず書き込みミスについて注目する．図 7 の parser では，プログラム開始直後に書き込みミス回数が 5000 回程度発生している．それに伴い，ダーティライン数が上昇している．他のベンチマークプログラムでも同様に，書き込みミス回数の増加に伴いダーティライン数が増加している．また，図 8 の vortex では，書き込みミス回数の増加と同じようにダーティライン数も増えている．図 10 の mcf では，4×10^7 命令まで書き込みミス回数が増加し続けており，ダーティライン数の増減はキャッシュライン数の最大値付近で落ち着いている．上記に述べたように，ダーティライン数が増える要因となるのは書き込みミスであることがわかる．書き込みミスが生じるとキャッシュメモリにデータの書き込みが生じるため，主記憶との一貫性が保たれなくなりダーティライン数は増加する．

次に読み出しミスについて注目する．最も解りやすいのは図 10 の mcf である．4×10^7 命令までは，読み出しミスはほとんど発生しておらず，ダーティライン数の増減はキャッシュライン数の最大値付近で落ち着いている．4×10^7 命令以降，読み出しミスが多く発生するとともに，ダーティライン数は急激に減少を始める．これは，キャッシュメモリ上に所望のデータが存在せず，主記憶からデータを読み込

むことで，主記憶との一貫性が保たれクリーンになるからである．また，図8のvortexでは，4×10^5あたりで急に読み出しミス回数が100回程度発生し，それに伴いダーティライン数が100ライン程急激に減少している．さらに図7のparserでは，読み出しミス回数が書き込みミス回数を上回り，ダーティライン数が全体的に減少している．上記に述べたように，ダーティライン数が減る要因となるのは読み出しミスであることがわかる．

以上のことをまとめると以下のようになる．

読み出しミス

キャッシュ上に所望のデータが存在しないので主記憶から読み込む．ダーティラインに書き込めば主記憶との一貫性が保たれるのでクリーンになる．よってダーティライン数は減少する．

書き込みミス

書き込みが生じるため主記憶との一貫性が保たれずダーティラインは増加する．もともとダーティなラインに書き込む場合ダーティライン数は変わらない．また，書き込みミス時以外にも書き込みヒット時にも書き込みが生じるためダーティラインは増加する．これらの情報を元にダーティライン数に着目したキャッシュメモリの性能オーバーヘッドを削減するタスクスイッチ手法を検討する．

5 ダーティラインを考慮したタスクスイッチ手法の検討

3節で述べたように，マルチタスク環境において，キャッシュ上にある先行タスクのデータを後続タスクが追い出すために，キャッシュメモリと主記憶との間に通信が生じ，性能オーバーヘッドを生じる．4節では，性能オーバーヘッドの原因となるダーティラインに着目し，ダーティライン数の変動の解析を行った．本節では，この結果を基に性能オーバーヘッドを削減するタスクスイッチ手法を検討する．3節で述べたように，追い出すダーティライン数が多ければ，性能オーバーヘッドも大きくなる．また，4節で示した結果のように，ダーティライン数はプログラム実行中に大きく変動している．そのため，ダーティライン数が少ない時にタスクスイッチを行えば，主記憶に書き戻すラインが少なく，性能オーバーヘッドを小さくすることができる．また，ダーティライン数が多い時には割り込みを禁止し，ダーティライン数が少ない時にのみタスクスイッチを行う手法も考えられる．また，プロセッサ上でダーティライン数を知るためには，ダーティライン数をカウントするカウンタを用いれば良い．今回のシミュレーションでは最大2048ラインあるので11ビットカウンタを使用すればよいことになる．以上のように，タスクスイッチを行う際にダーティラインを考慮することで性能オーバーヘッドを削減できると考えらる．

6 まとめ

本稿ではキャッシュメモリの性能オーバーヘッドの原因となるダーティライン数が読み出しミス時に減少し，書き込みミス時に増加することを明らかにした．また，ダーティライン数が変動する要因である読み出し/書き込みミスについても解析を行い，ダーティライン数を考慮したタスクスイッチ手法の検討した．今後，実際の性能オーバーヘッドの影響はどの程度あるのかを検証すると共に，タスクスケジューリング手法の検討を行っていく予定である．

参考文献

[1] T.Austin, E. Larson, and D. Ernst, "SimpleScalar: an infrastructure for computer system modeling," *IEEE Computer*, Vol. 35, No 2, pp. 59-67, February 2002.

[2] J. L. Henning, "SPEC CPU2000: Measuring CPU performance in the new millennium," *IEEE Computer*, Vol. 33, No. 7, pp, 28-35, July 2000.

[3] David. A. Patterson and John. L. Hennessy, コンピュータの構成と設計 第3版. 日経BP社, 2006.

[4] 前川 守, オペレーティングシステム. 岩波書店, 1996.

[5] 山本 敬, 日高 康雄, 小池 汎平, 田中 英彦, "レジスタウィンドウにおける高速タスクスイッチング手法の実現," 全国大会講演論文集 第45回平成4年後期 (6), pp. 129-130, 1992年9月.

[6] 田中 清史, 松本 尚, 平木 敬, "*Casablanca*: 実時間処理RISCコアの設計と実装," 情報処理学会研究報告, 99-ARC-135-8, Vol. 99, No. 100, pp. 51-56, 1999年11月.

[7] 仲野 巧, ウタマ アンディ, 板橋 光義, 塩見 彰睦, 今井 正治, "リアルタイムOSのVLSI化とその評価," 電子情報通信学会論文誌, Vol. J78-D-1, No. 8, pp. 679-686, 1995年8月.

[8] 永田 真稔, 小林 良太郎, 島田 俊夫, "タスクの予測によりコンテキストスイッチを投機実行する手法に関する検討," 情報処理学会研究報告, 2007-OS-105, Vol. 2007, No. 36, pp. 1-6, 2007年4月.

[9] 芹澤 光範, 杉原 真, "タスクスイッチによって生じるキャッシュメモリの性能オーバーヘッドの解析," 電気関係学会東海支部連合大会, O-080, 2008年9月.