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

2006－ARC－169 (15)
2006／8／1

TLBを用いたCPUキャッシュ利用分布の推定法

小川周吾[†,☆] 平木敬[††]

プロセッサのキャッシュはメモリアクセス性能を向上させるために用いられる．しかしキャッシュは，実行するプログラムによって使用状況が変化する．キャッシュの使用状況は，キャッシュヒット率に影響する．特にCMPやSMT等を用いたプロセッサでキャッシュを共有した場合，各スレッドのキャッシュ使用状況が全体の性能に影響する．そのため，キャッシュミスによる性能低下を回避するためには，各プログラムのキャッシュ使用状況の把握が必要である．本稿ではプロセッサのTLBを用いたプログラムのキャッシュ使用領域の推定方式を提案する．我々は提案方式をシミュレータ上でSPEC CINT2000を用いて評価した．その結果，TLBの内容からキャッシュの使用状況が推定可能であることが判明した．

An Estimation for CPU cache usage condition with TLB Information

Shugo Ogawa[†,☆] and Kei Hiraki[††]

Processor cache is used to improve memory access performance. However, each executed program have differnt cache usage condition. Cache usage condition have an affect on cache hit rate. Performance of shared cache on CMP or SMT processor is particularly affected by cache usage condition of each thread. Thus it is important for system to get cache usage information for avoiding cache miss increase and performance decrease. In this paper, we propose an estimation method for CPU cache usage condition with TLB information. We evaluate our new method with SPEC CINT2000 on a simulator. As a result, we show the fact that we can estimate cache usage condition with TLB information.

1. はじめに

現在の多くのプロセッサがキャッシュを搭載する．それは，低速なメモリへのアクセス頻度を低減してメモリアクセス性能を向上させるためである．メモリアクセスはプログラム実行中に頻繁に発生するため，キャッシュの搭載はシステム全体の性能向上につながる．

しかし，キャッシュヒット率はキャッシュの容量不足，使用領域の偏りによって低下する．キャッシュ使用量，使用領域は実行中のプログラムのアクセスパターンにより変化する．キャッシュ使用量の増加は，キャッシュの容量ミスによるヒット率の低下をもたらす．キャッシュ使用領域の偏りは，多くのプロセッサが用いるセットアソシアティブ方式，ダイレクトマップ方式のキャッシュで，アクセス頻度の高い領域での競合増加に伴うヒット率の低下をもたらす．これらのキャッシュ方式では，データの格納場所がアドレスで決まるため，使用領域の偏りがヒット率低下につながる．

キャッシュヒット率低下の問題は，特にCMP，SMT[1)2)]プロセッサで現れやすい．これらのプロセッサは複数のスレッドを同時実行する．一方，キャッシュは資源節約のためにプロセッサ内で共有することが多い．この場合，キャッシュの使用量，使用領域はキャッシュを共有する全スレッドからの影響を受ける．そのため，キャッシュ使用量の増加，使用領域の偏りが発生しやすく，キャッシュヒット率が低下しやすい．

よってこれらのプロセッサ上でのキャッシュミス低減には，各プログラムのキャッシュ使用状況を把握することが重要である．キャッシュミスは前述の，各プログラムが持つ容量不足，使用領域の偏り等のキャッシュミス発生要因によって引き起こされる．そのため，各プログラムが使用するキャッシュ容量，使用領域を調べることでキャッシュミスの発生状況を調べられる．CMPやSMTを用いたキャッシュ共有型のプロセッサでは，同時実行中の各スレッドのキャッシュミス発生

† 元・東京大学
Formerly with the University of Tokyo
†† 東京大学大学院情報理工学系研究科
The University of Tokyo, Graduate School of Information Science and Technology
☆ 現在，NEC システムプラットフォーム研究所
Presently with NEC System Platform Research Laboratories

要因を調べることで，ミスの発生状況を把握できる．

2. 関連研究

本章では関連研究として，キャッシュミス発生要因の特定，CMP，SMT プロセッサでのキャッシュミス削減，ページ配置とキャッシュの関係の順に説明する．

2.1 キャッシュミス原因の特定手法

プログラムのキャッシュミス発生要因を調べる方法には，静的にミスの発生要因を特定する方法，実行時にキャッシュの使用状況を動的に調べる方法がある．

静的な方法は，動作中のプログラムのキャッシュミスの低減に必要な情報収集手段としては使用できない．その理由は，静的な方法ではプログラムの実行前にキャッシュミスの発生要因を特定するためである．静的な方法には，プログラムソースを分析する方法，キャッシュミスをプロファイリングする方法がある．

一方，動的な方法は実行中のプログラムのキャッシュミスの低減に必要な情報収集手段として適している．その理由は，実行中のキャッシュ使用情報を収集するためである．しかし，動的な方法にはプログラムのキャッシュ使用情報を収集する機構が必要である．

動的に実行中のプログラムのキャッシュ使用情報を収集するには，ハードウェアまたは OS のカーネルレベルでの情報収集が必要である．汎用性の面では，特殊なハードウェアが不要な OS のカーネルレベルでの情報収集が有利である．しかしハードウェアを使用せず，かつ OS のカーネルレベルでプログラム実行中にキャッシュミス要因を調べる方法は研究されていない．

2.2 CMP, SMT プロセッサのキャッシュミス低減

CMP，SMT プロセッサ上の共有キャッシュでは実行中の複数のスレッド間で競合が発生する．この競合によるミス回数の低減について研究がなされている．

我々は文献[8] で，SMT プロセッサ上の共有キャッシュのミスを削減する方式を提案している．この方式では，プロセッサの性能カウンタを用いて各スレッドのキャッシュミス回数を収集する．この情報から，ミスが少なくなるように同時実行するスレッドを選択する．

キャッシュに限らず，資源一般の競合回避も研究されている．Snavely らは文献[3] で，SMT プロセッサ上で同時実行するプロセスの組を，ハードウェア資源の観点で決定することの有用性を指摘している．Zaki らは文献[4] で，SMT プロセッサ上の資源競合を無視したスケジューリングでの性能低下を指摘している．

これらの研究では，キャッシュミス回数のみからキャッシュ競合を検出する．他の情報は使用せず，同様の方式からキャッシュミス発生要因を調べることは難しい．

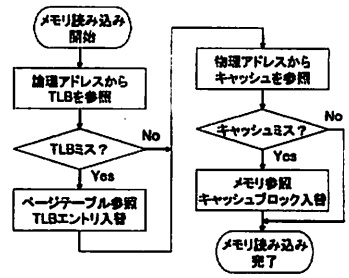

図 1 TLB，キャッシュを介したメモリアクセス手順 (Read)

2.3 ページとキャッシュの関係

ページとキャッシュの関連性についても研究が行われている．Kessler らは文献[5] においてページのメモリ上での配置改善による，キャッシュミス削減について述べている．この研究ではページがキャッシュ上の偏った領域に格納されないようにページ配置を変更する．これにより必要なキャッシュ連想度を削減している．

3. 提案方式

本稿では，実行中のプログラムのキャッシュ使用状況の推定方式を提案する．提案方式の説明に先立って，まずキャッシュと TLB，ページの関係を述べる．次に，提案方式を支える概念である必要連想度とその算出アルゴリズムを述べる．最後に，必要連想度を用いたキャッシュ使用領域，ミス要因の特定方法を述べる．

3.1 キャッシュと TLB の関係

プロセッサがメモリにアクセスする際に，キャッシュと TLB は連携して動作する．それにより，TLB にページアドレスが記録されている場合，対応するページのデータがキャッシュ上に存在する可能性が高くなる．

仮想記憶機能を持つプロセッサでは，各プログラムに別々の論理アドレス空間が与えられる．論理アドレスと物理メモリとの対応は，メモリを一定容量毎に区切った，ページを単位として管理される．ページテーブルと TLB はこの対応関係を記録する．ページテーブルはメモリ上に存在し，プログラムが使用する各ページの先頭の論理アドレスと，対応する物理アドレスを記録する．TLB はプロセッサ上に存在し，ページテーブル中の頻繁に使用する情報を記録する．

一方，キャッシュはメモリ上のアクセス頻度の高いデータを格納する．これは，低速なメモリへのアクセスの低減によるメモリアクセスの高速化を目的とする．

TLB とキャッシュを持つプロセッサでは，メモリアクセスはアドレス変換，データアクセスの順で処理される (図 1 参照)．処理中に TLB，キャッシュ共に必要な情報が見つからない場合にはページテーブルやメ

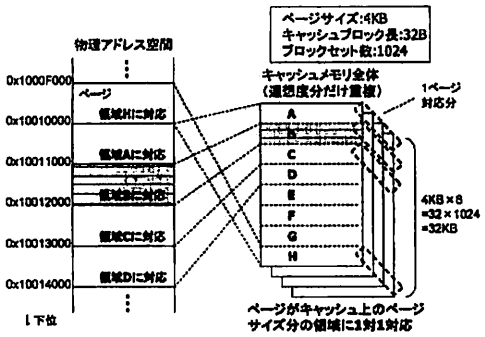

図 2 キャッシュとページの関係

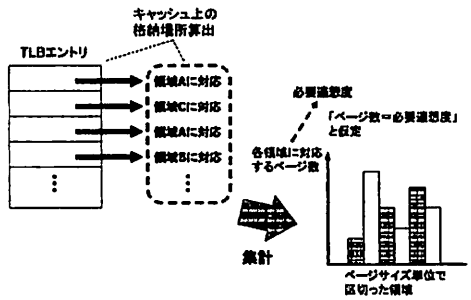

図 3 TLB から必要連想度を算出する手順 (概要)

for $i = 1$ **to** n **do begin**
  $area = f(T_i)$
  $P_{area} = P_{area} + 1$
end

図 4 キャッシュの各領域の必要連想度算出アルゴリズム

モリから必要な情報を参照して新たに格納する.

よってメモリアクセス時にはキャッシュ上にアクセスされたデータ, TLB には対応するページアドレスが記録された状態になる. そのため, TLB エントリにアドレスが記録されているページは, ページ内のデータがキャッシュ上に存在する可能性が高い.

3.2 キャッシュとページの関係

多くのプロセッサではキャッシュの構成上, ページ内のデータのキャッシュ上での格納場所が一定である. キャッシュとページの関係を図 2 で示す. 各ページは, アドレスで決まるキャッシュ上の領域に格納される.

多くのプロセッサでは, キャッシュの構成としてダイレクトマップまたはセットアソシアティブが用いられる. これらの構成では, データは格納されているアドレスによって, キャッシュ上の格納位置が限定される.

また, 近年キャッシュとメモリの間にキャッシュから溢れたデータを格納する victim キャッシュ[6] を追加して, ミスペナルティを削減する方式が用いられる.

この場合, データが victim キャッシュ上にある可能性を持つ. しかし, victim キャッシュにデータが移動する前後でデータの上位キャッシュ上での格納先ブロックは変化しない. よって victim キャッシュの有無に関わらず, キャッシュ上の格納先領域に変化はない.

3.3 キャッシュの必要連想度, 及び算出方式

キャッシュ, TLB, ページの関係により, TLB に記録されたページアドレスからページのキャッシュ上の格納先領域を特定できる. また, その領域に実際にページ内のデータが格納されている可能性が高い.

ここで, TLB に記録された全ページのデータを同時にキャッシュに格納するのに必要な連想度を, 必要連想度と定義する. また各キャッシュ領域の必要連想度を, 各領域に対応する, TLB に記録された全ページをキャッシュに格納するのに必要な連想度と定義する.

TLB から必要連想度を算出する手順は図 3 に示した通りである. 手順では TLB に記録されたページアドレスから各ページのキャッシュ上の格納先領域を求める. この格納先情報から各領域の必要連想度を求める.

図 4 は必要連想度の算出アルゴリズムである. まず, キャッシュ全体をページサイズ単位で分割した領域を想定する. 領域総数はキャッシュ総容量をページサイズと連想度で割った値である. この値を N とする. 分割した領域を先頭から, 0 から $N-1$ とする. 次に, TLB エントリ i のページアドレスを T_i, TLB エントリ数を n とする. ページアドレスから, 格納先のキャッシュ領域番号を求める関数は, $f(addr) = addr \bmod N$ と定義される. 最後に, 領域 i についての TLB 中の対応ページ数を P_i とする. すると必要連想度の定義により, P_i は各領域の必要連想度に対応する.

3.4 キャッシュ使用領域・ミス要因の特定

算出された必要連想度から, キャッシュ使用領域の推定, キャッシュミス発生要因の特定を行う. 本節では必要連想度から推定可能な情報について説明する.

3.4.1 キャッシュミスが発生しやすい領域

必要連想度から, キャッシュミスが発生しやすい領域を推定できる. 必要連想度がキャッシュの連想度を上回る場合, ワーキングセット中に同じ領域に格納されるデータがキャッシュ連想度より多く含まれることを表す. よって, 必要連想度がキャッシュの連想度以下の領域と比較して, ミスを引き起こす可能性が高い.

3.4.2 頻繁にアクセスされる領域

必要連想度の情報から, 頻繁にアクセスされるキャッシュ領域を推定できる. TLB には頻繁に利用するペー

CPU 命令セット	Alpha 互換
Instruction/Data (cache)	共用
キャッシュライン長	512-bit
キャッシュブロックセット数	128-16384
キャッシュブロック配置	1-16 way
キャッシュ置換アルゴリズム	LRU
Instruction/Data (TLB)	共用/分離
TLB エントリ数	最大 128
TLB エントリ配置	Fully associative
TLB 置換アルゴリズム	LRU
ページサイズ	1K-128Kbytes

図 5 シミュレーションパラメータ

ジのアドレスが記録されている．よって，TLB に記録されたページは，頻繁にアクセスするページである．よって，TLB エントリから算出される必要連想度が高いほど同一キャッシュ領域を頻繁にアクセスする．

3.4.3 発生するキャッシュミスの種類

キャッシュ上の各領域の必要連想度から，発生するキャッシュミスの種類を特定できる．必要連想度が高い領域がキャッシュの一部の場合は，ミスがこれらの領域に集中して競合ミスが発生する．一方で，必要連想度が高い領域が広範囲に渡る場合は，ミスの発生領域がキャッシュ全体に分散して，容量ミスが発生する．

4. 提案方式の評価

提案方式の正当性，有効性をシミュレータで評価する．本章では提案方式の評価環境，評価結果を述べる．

4.1 評価環境

提案方式による，キャッシュ使用状況の推定精度を SPEC CINT2000 のシミュレータ上での実行結果で評価する．シミュレータは SimCore/Alpha Functional Simulator[7]Version 2.0r1 に TLB とキャッシュのシミュレーションを独自に追加して使用する．ハードウェアのシミュレーション条件は図 5 に示した通りである．

一方でソフトウェアのシミュレーションパラメータは以下の通りである．評価では SPEC CINT2000 から，正常に動作しなかった 186.crafty を除いた 11 種類のテストを用いる．入力データセットは ref を用いる．

各テストは，実行開始から最大 200 億命令で実行を中断する．また，提案方式によるキャッシュ使用状況の推定を 10 億命令毎に行って効果を評価する．

4.2 評価結果・考察

本稿の提案方式である TLB を用いたキャッシュ使用状況の推定法について，効果，正当性を評価する．

まず，TLB エントリにアドレスが記録されている

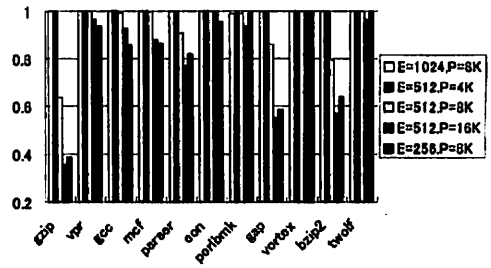

図 6 TLB に含まれるページ内容がキャッシュされている率

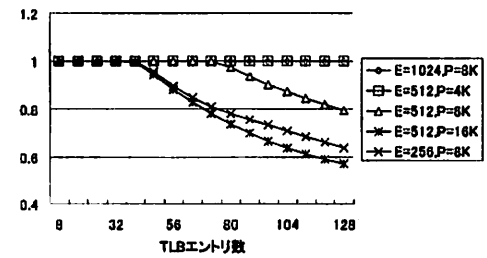

図 7 bzip2 において TLB 中のページがキャッシュされる率

ページに含まれるデータが，キャッシュ上にある確率を評価する．次に，キャッシュの各領域の必要連想度とアクセス頻度との関係を評価する．最後に，必要連想度と実際のキャッシュミス回数との関係を評価する．

各テストの評価は，10 億命令毎の評価結果をテスト全体で平均したものを結果とする．また，複数回のプログラム実行を含むテストについては，全実行の結果を平均したものを結果とする．

各評価では特に断りがない限り，キャッシュはブロックセット数を 1024，連想度を 16，TLB は Instruction/Data 共用型，参照するエントリ数を 128，ページサイズを 8192 バイトとする．

4.2.1 TLB 内の該当ページとキャッシュの関係

TLB エントリに対応するページ内のデータがキャッシュに格納されている確率を評価した結果を図 6 に示す．グラフでは，必要連想度を求める際に参照する TLB エントリ数，ページサイズ，キャッシュブロックセット数別に結果を示している．グラフ中の E はキャッシュブロックセット数，P はページサイズを表す．

ページサイズを大きくした場合，キャッシュのブロックセット数を減らした場合に TLB エントリに対応するページ内容がキャッシュされる率が低下する．

また，256.bzip2 を例に参照する TLB エントリ数とページ内容がキャッシュされる率との関係を図 7 に

示す．全条件において，32エントリまでは全てのページについて内容の一部がキャッシュされている．

評価結果から，ページ内容がキャッシュされる率が低くなる条件はTLBエントリ数分のページの合計サイズがキャッシュサイズを上回ることだと分かる．このとき，キャッシュサイズがTLBエントリに対して不足しているために，率が低下すると考えられる．実際に各条件におけるキャッシュサイズを計算すると，ブロックサイズ数が512の場合には16連想度で512KBである．これをページサイズ8KB，16KBで割るとそれぞれ64，32となる．グラフと比較すると，TLBエントリがこの値を超えた部分で単調減少を始める．

この他，キャッシュの連想度を低くした場合にも，全体的に率が低下する以外には，評価結果に連想度が16の場合と同様の傾向が現れることを確認している．

以上の結果から，キャッシュにデータが残されている場合，TLBにもそのデータを含むページのアドレスが記録されていることが分かる．

4.2.2 必要連想度とアクセス頻度の関係

キャッシュ領域の必要連想度と，アクセスされる頻度との関係を評価した結果は図8に示した通りである．グラフでは，キャッシュの各領域の必要連想度以内でヒットした回数を評価している．TLBから求めた必要連想度と，キャッシュヒット回数が最大になるように必要連想度を理想的に設定した場合とを比較している．また，参照したTLBエントリ数は32とする．

評価の結果，キャッシュのブロックセット数の違いによる大きな差は見られない．一方でページサイズが大きくなると，必要連想度によってカバーされるヒット回数は理想値に近づく．また，ほぼ全てのテストでTLBから算出した必要連想度によってカバーされるヒット回数は，理想的な必要連想度によってカバーされるヒット回数と比べて約98%以上の値を示している．181.mcfで他より低い値を示しているのは，ヒット率が相対的に低いことが関係している．mcfは他のテストより高い連想度でのヒットが多い．そのためより多くのTLBを参照して必要連想度を上げたことでカバーされる，ヒット回数の収束が他より遅れる．

更に，176.gccを例に参照するTLBエントリ数と，必要連想度以内でのヒット回数の対理想値比率の関係を図9に示す．参照するTLBを増やすほど，各領域の必要連想度以内で生じるミス回数が理想値に近づく．

以上の結果から，必要連想度が高いキャッシュ領域へのアクセス回数が多くなることが分かる．

4.2.3 必要連想度とキャッシュミスの関係

各領域の必要連想度と，その領域でのキャッシュミス

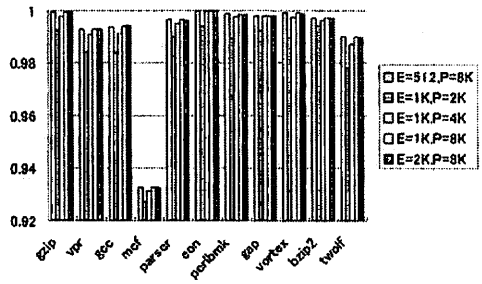

図8 必要連想度以内でのキャッシュヒット回数の対理想値比率

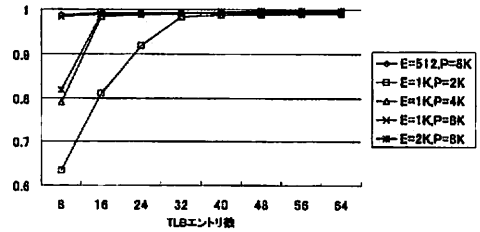

図9 gccにおける必要連想度以内でのヒット回数比率

スの関係を評価する．評価の基準として，「求められた全領域中の最大連想度以上でのキャッシュヒットに占める，最大連想度を持つ領域のキャッシュヒットの割合」を用いる．これは，必要連想度が高いことがキャッシュミスの要因になるのであれば，最大必要連想度を要するキャッシュヒットの多くは連想度を最も必要とする領域に集中することを利用している．以下ではこの割合を，「最大必要連想度以上のヒット分布」と呼ぶ．

最大必要連想度以上のヒット分布は以下のように求める(図10参照)．領域iの連想度jでヒットする回数をa_{ij}，領域数をn，キャッシュの連想度の上限をN，最大必要連想度とそれを持つ領域をそれぞれM，A_{max}とする．このとき，キャッシュ全体の最大必要連想度以上でのヒット数は$S = \sum_{i=1}^{n} \sum_{j=M}^{N} a_{ij}$と求められる．一方，最大連想度を持つ領域の，必要連想度以上でのキャッシュヒット数は$S_{max} = \sum_{i=M}^{N} a_{max\,i}$と求められる．全体のヒット数を$T$，最大必要連想度を持つ領域の全ヒット数を$T_{max}$とすると，最大必要連想度以上のヒット分布は$\frac{S_{max}}{T_{max}} / \frac{S}{T}$と表される．

TLBエントリ数を32とした場合の，最大必要連想度以上のヒット分布は図11に示した通りである．評価結果から，最大必要連想度を持つ領域では，それ以上の連想度でヒットするアクセスが他の領域より多いことが分かる．多くのテストでは結果の値が1以上で

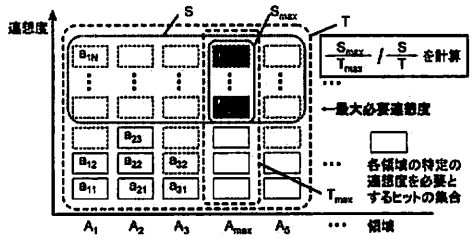

図 10 最大必要連想度以上のヒット分布の算出

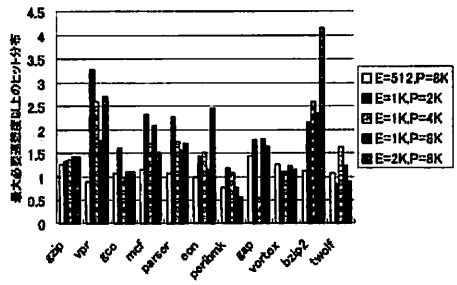

図 11 最大必要連想度以上のヒット分布

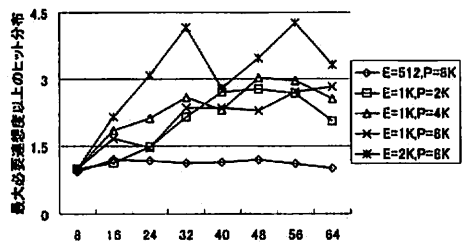

図 12 bzip2 における最大必要連想度以上のヒット分布

あるため，最大必要連想度を持つ領域では最大必要連想度以上のヒットが集中していることが分かる．

更に，176.gcc を例に参照する TLB エントリ数と，最大必要連想度以上のヒット分布の関係を図 12 に示す．参照する TLB を増やすほど，最大必要連想度以上のヒット分布の値が増加している．よって，最大必要連想度を持つ領域は他の領域に比べてヒットするのにより高いキャッシュの連想度を要するアクセスが多い．

以上の結果から，高い連想度を必要とする領域は他の領域に比べて最大必要連想度以上のヒット分布の値が高い，つまり連想度が有限であるキャッシュ上ではミスになる可能性が高いアクセスが多いことが分かる．つまり，高い連想度を必要とする領域はキャッシュミスを起こしやすい．

5. まとめと今後の課題

本稿では，キャッシュ使用状況を動的に調べることでミスを回避することの有用性を述べた．我々は，プログラムのキャッシュ使用状況を動的に調べる方法として，TLB を用いたキャッシュ使用状況の推定方式を提案した．評価の結果，提案方式によりキャッシュの各領域が必要とする連想度を正しく算出されることを確認した．また，各領域が必要とする連想度と，各領域で発生するキャッシュミスとの関連性を確認した．以上の評価により，提案方式がキャッシュ使用状況を動的に把握する方法として有用であることを確認した．

今後は，TLB の置換アルゴリズム，エントリの配置を変更する，他のプログラムを用いた評価をすることで，ハードウェア，ソフトウェアパラメータが提案方式に与える影響についてより明確にする予定である．

参 考 文 献

1) Hiroaki Hirata, Kozo Kimura, Satoshi Nagamine, Yoshiyuki Mochizuki, Akio Nishimura, Yoshimori Nakase, Teiji Nishizawa: An elementary processor architecture with simultaneous instruction issuing from multiple threads, Proc. Annu. Int. Symp. on Computer Architecture, pp.136-145, 1992
2) Dean M. Tullsen, Susan Eggers, Henry M. Levy: Simultaneous Multithreading: Maximizing On-Chip Parallelism, Proc. Annu. Int. Symp. on Computer Architecture, pp.392-403, 1995
3) Allan Snavely, Dean M. Tullsen: Symbiotic Jobscheduling for a Simultaneous Multithreading Processor, 9th International Conference on Architectural Support for Programming Languages and Operating Systems, pp.234-244, 2000
4) Omer Zaki, Matthew McCormick, Jonathan Ledlie: Adaptively Scheduling Processes on a Simultaneous Multithreading Processor, Technical report, University of Wisconsin - Madison, 2000
5) R. E. Kessler, Mark D. Hill: Page Placement Algorithms for Large Real-Indexed Caches, ACM Transactions on Computer Systems, Vol. 10, No. 4, pp.338-359, 1992
6) Norman P. Jouppi: Improving Direct-Mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers, Proc. Annu. Int. Symp. on Computer Architecture, pp.364-373, 1990
7) Kenji Kise, Takahiro Katagiri, Hiroki Honda, Toshitsugu Yuba: The SimCore/Alpha Functional Simulator, Workshop on Computer Architecture Education (WCAE-2004) held in conjunction with the ISCA-31, 2004, http://www.arch.cs.titech.ac.jp/~kise/SimCore/index.htm
8) 小川 周吾, 平木 敬: プロセスの実行時情報を用いたスケジューラによる高速化手法, 情報処理学会論文誌, Vol. 46, No. SIG 12(ACS 11), pp.161-169, 2005