Vol.2010-ARC-187 No.3
Vol.2010-EMB-15 No.3
2010/1/28

マルチコアプロセッサのコアごとのアクセス局所性を利用した共有キャッシュの消費電力削減

佐藤　公紀[†1]　　阿部　公輝[†1]

L2 キャッシュに共有キャッシュ方式を用いたマルチコアプロセッサにおいて, ラインごとのコア局所性に着目し, タグ比較の回数を減らすことにより動的な消費電力を削減することを考える. L2 キャッシュの各ラインごとに前回アクセスしたコアの番号を記憶させ, 次回のアクセスに利用する手法を提案する. 本手法の有効性を調べるために, ベンチマークプログラムを用いてコア局所性とタグ比較回数の削減割合を測定した. 実験の結果,L2 キャッシュヒット時のタグ比較にかかる動的消費電力を平均して最大約 25%削減できる可能性があることが分かった.

Dynamic Power Reduction of Shared Cache Utilizing Access Locality of Cores in Chip Multiprosessors

Koki Sato[†1] and Koki Abe[†1]

We try to reduce dynamic power consumption of a shared L2 cache in chip multiprocessor by reducing the number of tag comparisons, exploiting locality of line accesses. We provide a table memorizing for each line the identifier of a core which accessed the line previously. The table is refered for accessing the L2 cache next time. In order to examine the effectiveness of the method, we measured the locality of line accesses by cores as well as the reduction of the number of tag comparisons in executing benchmark programs. Experimental results revealed that the proposed method can reduce the dynamic power consumed for cache hits by up to 25% in average.

†1 電気通信大学 情報工学科
Department of Computer Sience, The University of Electro-Communications

1. はじめに

近年の半導体技術の進歩により大幅に増加した半導体資源を利用し, 単一チップに複数のプロセッシングエレメントを搭載したチップマルチプロセッサ (CMP) が開発されている.CMP の処理能力の向上の他に消費電力の削減についての研究も広く行われている. 特にキャッシュメモリはチップ内の電力消費に占める割合が大きくなっている[1)].

CMP のキャッシュは大容量化の傾向にあり, またボトルネックの解消のためキャッシュが多層化する傾向にある. また共有方式の場合は複数のコアの使用するデータが混在するため, 空間的局所性も薄くなると考えられる. そのため共有キャッシュの特徴を利用した従来とは違う観点からの消費電力削減手法が必要になると考えられる.

本研究では L2 キャッシュに共有キャッシュ方式を用いた CMP において, ラインごとのコア局所性に着目し, タグ比較の回数を減らすことにより動的な消費電力を削減することを考える.L2 キャッシュの各ラインごとに前回アクセスしたコアの番号を記憶させ, 次回のアクセスに利用する手法を提案する. 本稿では本手法の有効性を調べるために, ベンチマークプログラムを用いてコア局所性とタグ比較回数の削減割合を測定した結果を示す.

2. 関連研究

キャッシュの省電力化として, 静的消費電力を減らすことと動的消費電力を減らすことがある. 静的消費電力を削減する方法としては, 長時間アクセスのなかったラインを待機状態にすることで省電力化を図る方法[2)] 等が知られている.

動的消費電力を削減する手法には, 過去のタグ比較結果を再利用して参照データが存在するウェイのみを活性化する方法[3)] 等があるが,L1 命令キャッシュのみを対象としている.CMP の共有 L2 キャッシュに着目した手法には, スケジューリング手法を拡張し,L2 キャッシュミスを削減することにより, 結果として消費電力も削減する方法が提案されている[4)]. 文献[4)] はスケジューリングの改良により L2 キャッシュでのラインの競合を防ぐ手法であるが, ヒット率上昇の結果として消費電力の削減を図るものであり, 本研究とはアプローチが異なる.

3. 提案手法

プログラム中のスレッド並列性を利用し,CMP を用いてプログラムを高速低消費電力で実行するための研究がなされている. しかし現存する大半のプログラムは CMP に最適化されておらず, 現実には複数のプログラムでコアの資源を分割しながら実行していることが多

Vol.2010-ARC-187 No.3
Vol.2010-EMB-15 No.3
2010/1/28

いと考えられる. この場合共有キャッシュ内には複数のプログラムが使用するデータが混在する形となっていると予想される. 各コアが使用するデータのキャッシュ内での局所性 (コア局所性) を利用し, 消費電力を削減することを考える.

多くの場合,L2 共有キャッシュにはセットアソシアティブ方式が用いられる.L2 キャッシュで用いられる場合は一般にまずタグを全て比較し, 一致したラインのデータのみ読みだす. 連想度を上げればヒット率は上がるが比較するタグ数も増え消費電力が増加する. 連想度を下げればヒット率が低下し CPU-メモリ間のボトルネックによる速度低下が避けられない.

そこでコアごとの局所性を用いた低消費電力手法を考える. 各ラインごとに最後にそのラインにアクセスしたコアの番号を Last Access Core(LAC) 情報として保存する. コアからデータの要求が来たとき, 各ラインの LAC を見比べそのコアが前回アクセスしたラインのタグ比較のみを先に行う. ヒットした場合は該当ラインのデータ読み出しを行い, ミスした場合は残りのタグに対して比較を行う.

図 1 に動作例を示す. 図はコア数 4, 連想度 8 のキャッシュを示している. 図の例では core1 から L2 キャッシュにアクセスがあったため, まず LAC 情報を調べ core1 と一致するライン 0,3,7 のみタグ比較を行う. ヒットしたライン 7 からデータを core1 へ転送する.

本手法が電力削減に有効であるためには, コア局所性がどの程度存在するかを実際に調べる必要がある. 次章ではベンチマークプログラムを用いて, コア局所性により比較するタグ数がどの程度削減されるかを調べる.

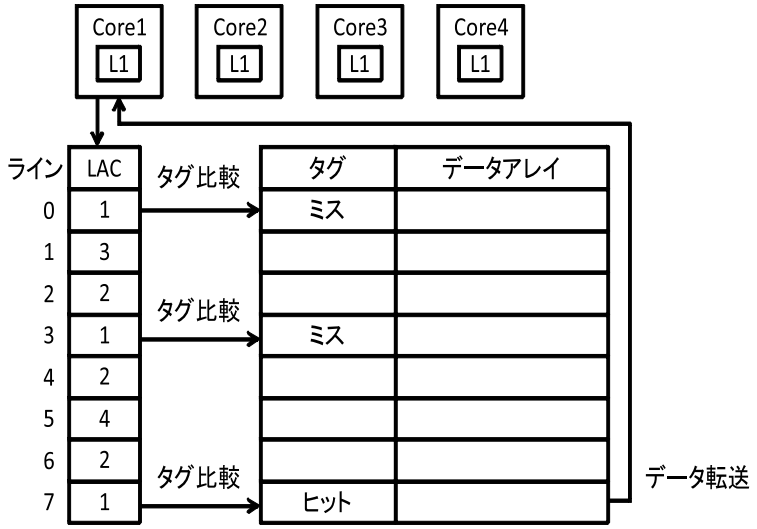

図 1 提案手法の動作例
Fig. 1 Illustration of proposed method.

4. ベンチマークを用いたシミュレーション実験

4.1 実 験 環 境

提案手法の有効性を確認するために, ベンチマークプログラムを用いたシミュレーションを行う. マルチプロセッサシュミレータ M5[5] に提案手法を実装した. **表 1** に実験環境を示す. ベンチマークプログラムは Splash2[6] を使用した.Splash2 は, キャッシュコヒーレントな集中または分散共有アドレス空間を持つ並列計算機のためのベンチマークであり広く用いられている. ベンチマークからは FFT,LUContig,Radix,Barnes,FMM,Ocean,WaterNSquared,WaterSpatial の 8 つのプログラムを使用した. **表 2** にそれらの内容を示す.

表 1 実験環境
Table 1 Experimental conditions.

全体構成	コア数	4
	ラインサイズ	64byte
	動作周波数	1GHz
キャッシュサイズ	L1 命令キャッシュ	32kB〜512kB
	L1 データキャッシュ	32kB〜512kB
	L2 データキャッシュ	256kB〜4096kB
アクセスレイテンシ	L1 命令キャッシュ	1ns
	L1 データキャッシュ	1ns
	L2 データキャッシュ	10ns
連想度	L1 命令キャッシュ	1
	L1 データキャッシュ	4
	L2 データキャッシュ	4〜64

4.2 実験結果と考察

変数 $core_n$ を次のように定義する.

$core_n$ =(コア n のアクセスにおいて L2 キャッシュの $LAC = n$ のラインでヒットした回数)/(すべてのコアのアクセスにおいて L2 キャッシュにヒットした回数).

Vol.2010-ARC-187 No.3
Vol.2010-EMB-15 No.3
2010/1/28

表 2 ベンチマークの内容
Table 2 Benchmark programs used in experiments.

SPLASH2 ベンチマーク	
FFT	高速フーリエ変換
LUContig	LU 分解
Radix	Radix ソート
Barnes	Barnes-Hut 法
FMM	Fast Multipole Method
OceanContig	海洋の移動問題の解法
WaterNSquared	水分子の力とポテンシャルの計算
WaterSpatial	水分子の力とポテンシャルの計算

また,

$$\Sigma_{n=1}^{N} core_n$$

は,L2 キャッシュにおけるデータのコア局所性を表す. **図 2** はコア数 $N = 4$, 連想度 8,L1 キャッシュ容量 32kB,L2 キャッシュ容量 256kB のときの $core_1, \ldots, core_N$ を測定した結果を表す.

プログラムによっては 60%以上, 平均しても 40%のアクセスは前回そのラインにアクセスしたコアと今回アクセスしたコアが同じであることが分かった. そのコアごとの割合も FFT のようにほぼ一定のものもあれば,WaterSpatial のように偏りがあるものもあった. コア局所性の高いプログラムは, コアごとに計算に使用するデータの独立性が高いと考えられる. 一方, コア局所性の低いプログラムは, データを複数のコアで共有する割合が高いか, 計算結果の受け渡しが頻繁に行われるものだと考えられる.

このようにベンチマークプログラム単体の中でもある程度のコア局所性が確認できた. 前章で述べたように複数の関係性の薄いプログラムを同時実行するような状態ならコア局所性はさらに上がると考えられる.

次に変数 $agree_n$ を

$agree_n$ =(コア n のアクセスにおいて L2 キャッシュの $LAC = n$ のラインの平均数)/(連想度)

で定義する. また変数 $agree$ を次のように定義する.

$$agree = \Sigma_{n=1}^{N} agree_n$$

変数 $agree$ は本来タグ比較をするはずだったラインのうち, コア局所性が存在するラインのみタグ比較をする割合を表す. この値が小さいとタグ比較時の消費電力の削減が期待される.

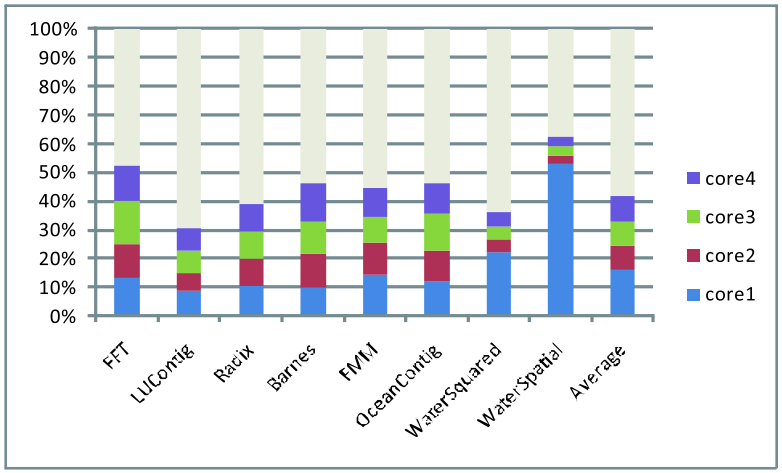

図 2 連想度=8,L1=32kB,L2=256kB でのコア局所性
Fig. 2 Access locality of cores when associativity=8, L1=32kB,L2=256kB.

図 3 はコア数 $N = 4$, 連想度 8,L1 キャッシュ容量 32kB,L2 キャッシュ容量 256kB のときの $agree$ を測定した結果を示す. $agree$ の値は平均して 40%程度であることが分かった. コア局所性と合わせて考えるなら 40%のアクセスに対して 60%のタグ比較数の削減, つまり L2 キャッシュヒット時にタグ比較にかかる動的消費電力を最大約 25%程度削減できる. コア局所性が大きく,$agree$ の小さい FFT の場合最大約 35%の動的消費電力の削減が望める.

次に連想度と L1 キャッシュ容量を固定し,L2 キャッシュ容量を変化させてコア局所性,$agree$ の変化を調べた. 連想度 8,L1 キャッシュ容量を 32kB とし,L2 キャッシュ容量を 256kB から 4096kB まで変化させたときの結果を**図 4** と**図 5** に示す. コア局所性の値は L2 キャッシュ容量が 2048kB のときに最大値 50%となった.$agree$ はほとんど変化はないが,L2 キャッシュ容量が 1024kB のときに最少となる.2048kB までコア局所性が上昇したのは L2 キャッシュ容量が増えたことでライン入れ替えが減り, 今までキャッシュミスになっていたコア局所性のあるラインへのヒットが増えたためと考えられる.$agree$ が L2 キャッシュ容量が 1024kB 以上で増加するのは, キャッシュ内に残る比較不要なデータが増えるためと考えられる.

L1,L2 キャッシュ容量の比率を一定とし, 両者を変化させた場合も調べた. 連想度 8, キャッシュの比率 1:8 とし,L1 キャッシュ容量を 32kB から 512kB まで,L2 キャッシュを 256kB から 4096kB まで変化させた結果を**図 6**, **図 7** に示す. コア局所性が山なりのグラフになった

Vol.2010-ARC-187 No.3
Vol.2010-EMB-15 No.3
2010/1/28

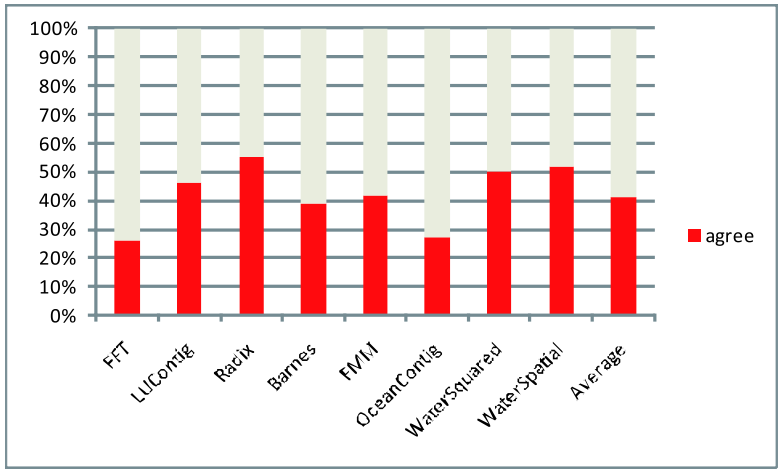

図 3 連想度=8,L1=32kB,L2=256kB での *agree*
Fig. 3 Values of *agree* when associativity=8,L1=32kB,L2=256kB.

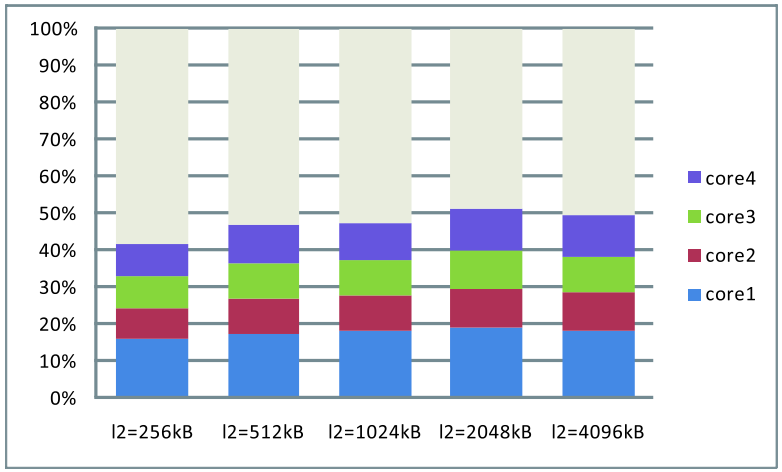

図 4 連想度=8,L1=32kB でのコア局所性の L2 キャッシュ容量による変化
Fig. 4 Access locality of cores depending on L2 cache size with associativity=8, L1=32kB.

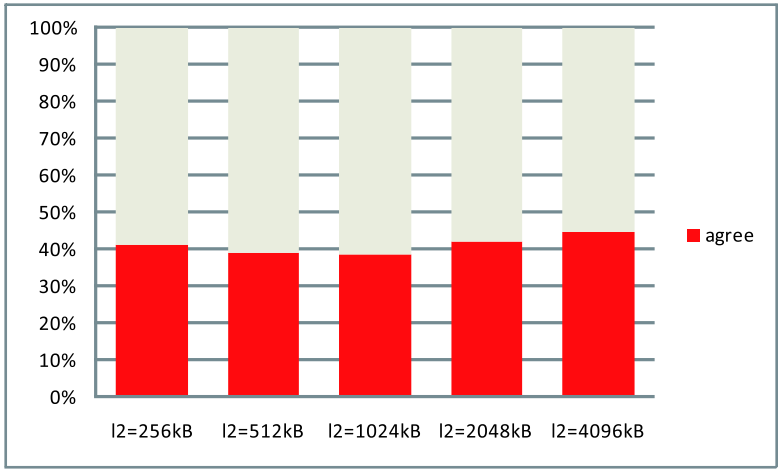

図 5 連想度=8,L1=32kB での *agree* の L2 キャッシュ容量による変化
Fig. 5 Values of *agree* depending on L2 cache size with associativity=8, L1=32kB.

のに対し,*agree* は容量が大きくなるにつれ減少する傾向になった. このように L2 キャッシュのみならず L1 キャッシュの容量によってもコア局所性,*agree* の値が変化する. これは L1 キャッシュの容量が大きければ L2 キャッシュへのアクセス自体が減るためと考えられる.

連想度についての比較も行った.L1,L2 キャッシュ容量をそれぞれ 32kB,256kB に固定し, 連想度を 4 から 64 まで変化させてコア局所性と *agree* を調べた. **図 8**, **図 9** に結果を示す. コア局所性は連想度を上げてもあまり変化がなく,*agree* は連想度の増加に伴い減少することがわかった. キャッシュの容量が固定されていれば連想度を変えてもラインの数は変わらないので, コア局所性は変わらないと考えられる. 一方, 連想度の増加に対して,$LAC = N$ のラインの数はそれほど増加しないため,*agree* は減少すると考えられる.

本手法は L1 キャッシュに対する L2 キャッシュの容量が大きく, 連想度が高い場合により有効であると考えられる.

5. おわりに

CMP の共有キャッシュ内におけるコアごとのデータ局所性を利用したタグ比較数の削減による省電力化手法を提案し, その有効性を調べるためのシミュレーション実験を行った. 実験の結果,L2 キャッシュヒット時のタグ比較にかかる動的消費電力を平均して最大約 25%削

Vol.2010-ARC-187 No.3
Vol.2010-EMB-15 No.3
2010/1/28

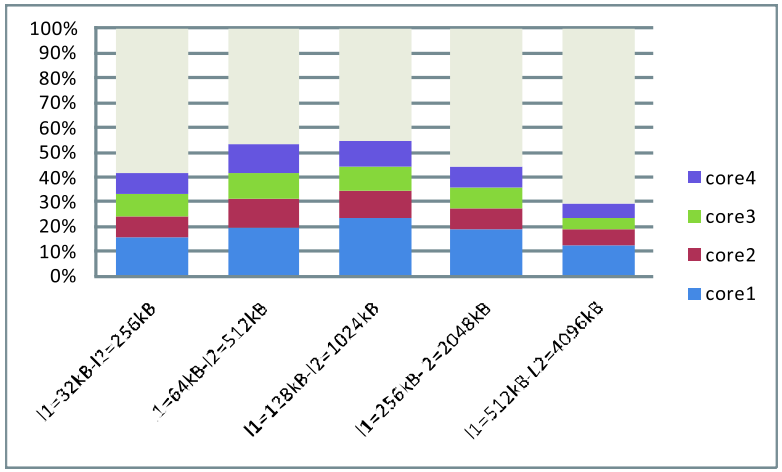

図 6 連想度=8 でのコア局所性の L1,L2 キャッシュ容量による変化
Fig. 6 Access locality of cores depending on L1 and L2 cache sizes with associativity=8.

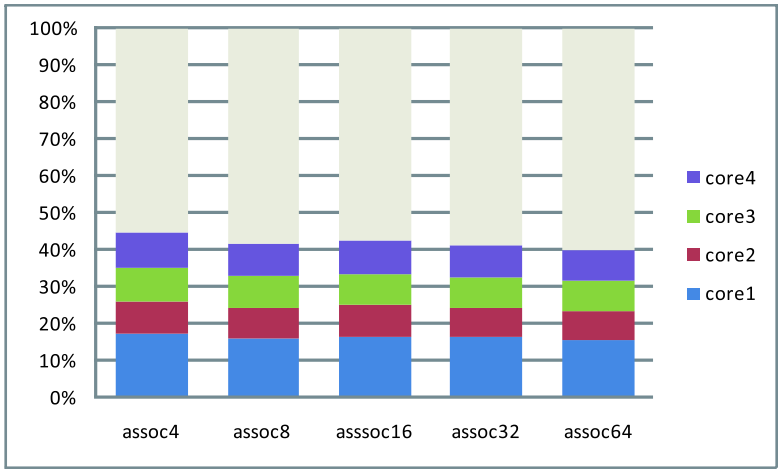

図 8 L1=32kB,L2=256kB でのコア局所性の連想度による変化
Fig. 8 Access locality of cores depending on associativity when L1=32kB,L2=256kB.

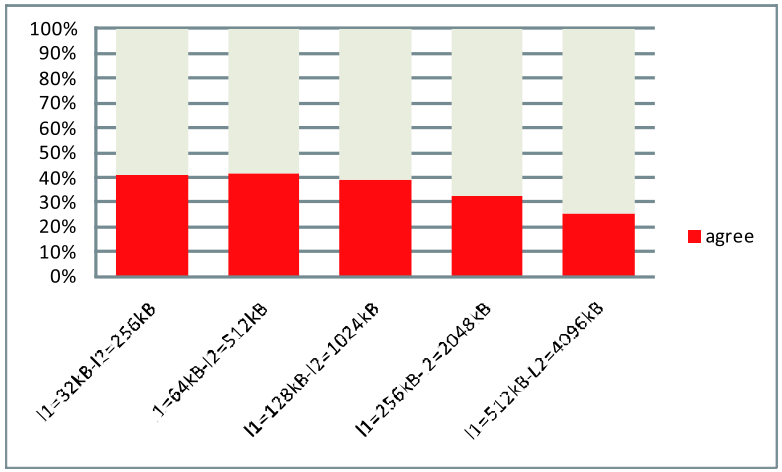

図 7 連想度=8 での *agree* の L1,L2 キャッシュ容量による変化
Fig. 7 Values of *agree* depending on L1 and L2 cache size with associativity=8.

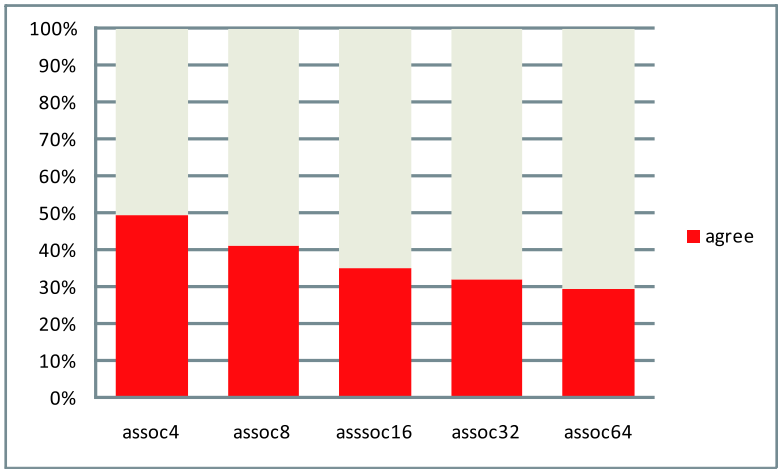

図 9 L1=32kB,L2=256kB での *agree* の連想度による変化
Fig. 9 Values of *agree* depending on associativity when L1=32kB,L2=256kB.

Vol.2010-ARC-187 No.3
Vol.2010-EMB-15 No.3
2010/1/28

減できる可能性があることが分かった. 本手法は L1 キャッシュに対する L2 キャッシュの容量が大きく, 連想度が高い場合により有効であると考えられる. 今後は複数のプログラムを同時実行したときの評価実験や L2 キャッシュヒット率を考慮に入れた実験,LAC 情報を保存または参照するのにかかる動的および静的消費電力, また実行速度等を考慮に入れた評価実験を行う予定である.

参 考 文 献

1) M. B. Kamble, and K. GhoseAnal: "ytical Energy Dissipation Models For Low Power Caches", Proc. of the 1998 Int. Symp. on Low Power Electronics and Design, pp.143-148, 1998.
2) S.Kaxiras, Z.Hu, and M.Martonosi: "Cache Decay: Exploiting Generational Behavior to Reduce Cache Leakage Power", Proc. of the 28th Int. Symp. on Computer Architecture, pp.240-251, 2001.
3) 田中 秀和, 井上 弘士, モシニャガ ワシリー, 村上 和彰: "ヒストリ・ベース・タグ比較キャッシュの設計と評価", 第 66 回情報処理学会全国大会講演論文集（1）, p83-84, 2004.
4) 志村 嘉洋: "マルチコアプロセッサ上でのマルチスレッドタスクスケジューリングとキャッシュ性能", 東北大学大学院情報科学研究科修士論文, 2008.
5) N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi and S. K. Reinhardt: "The M5 Simulator: Modeling Networked Systems", IEEE Micro, 26, 4, pp.52-60, 2006.
6) S. C. Woo, M. Ohara, E. Torrie, J. P. Singh and A. Gupta: "The SPLASH-2 programs: characterization and methodological considerations", Proc. of the 22nd Int. Symp. on Computer Architecture, pp.24- 36, 1995.