2007－ARC－174 (6)
2007／8／1

ウェイアロケーション型共有キャッシュ機構の性能評価

小寺 功[†] 江川隆輔[††]
滝沢寛之[†] 小林広明[††]

我々は，キャッシュパーティショニングと部分的に電力供給を止める消費電力削減手法を組み合わせることで，性能を維持しつつ低消費電力で動作するマルチコアプロセッサ用ウェイアロケーション型共有キャッシュ機構を提案している．本提案機構ではキャッシュの参照局所性の評価量を定義し，キャッシュパーティショニングと消費電力削減の指標として用いる．この評価量を用いることにより，提案するキャッシュ機構は柔軟に性能指向と省電力指向に設定することができる特徴を持つ．

本論文では，キャッシュ参照の特徴が異なるアプリケーションを用いて本提案機構の有効性を評価する．その評価の結果，提案機構は高い参照局所性を持つアプリケーションでは適切なキャッシュパーティショニングを実現可能であることが示された．また，性能指向の設定にすることで，平均約0.3%の速度向上しつつ，約 28% の消費エネルギを削減できることを明らかにした．

The Evaluation of A Way-Allocatable Shared Cache Mechanism

Isao Kotera,[†] Ryusuke Egawa,[††] Hiroyuki Takizawa[†]
and Hiroaki Kobayashi[††]

We have proposed a way-allocatable shared cache mechanism for chip multiprocessors, which can save power consumption with remaining the performance by employing cache partitioning and power gating. In the proposed mechanism, a metric of cache access locality is defined and used for the cache partitioning and the power gating. Based on the metric, the proposed mechanism can flexibly change the configuration to be either performance-oriented or power-oriented.

This paper evaluates the validity of the proposed mechanism, using some benchmarks with different cache access behaviors. The evaluation results show that the proposed mechanism can appropriately partition the shared cache for applications with high localities. In addition, our proposal at the performance-oriented mode can reduce energy consumption by 28% while improving the performance by 0.3%.

1. 緒 言

近年，半導体集積技術の進歩により，大量のトランジスタを 1 チップに集積できるようになった．これにより，マイクロプロセッサは豊富なハードウェア資源を搭載することが可能になり，それを利用して実行性能を急速に向上させてきた．マルチコアプロセッサ (Chip Multiprocessor, CMP) は，この豊富に用意されたハードウェア資源を有効利用すると共に．スレッドレベル並列性を利用した高速化を期待できるアーキテクチャである．

しかし，高集積化によって速度向上が得られる一方で，それに起因する消費電力と熱排出の増加が深刻な問題となっている．特に，消費電力の中でもリーク電流による静的電力の占める割合が増加している．このため，電力消費の削減には動的電力以上に，静的電力の抑制が求められている．チップ上で大きな面積を占めるオンチップキャッシュは，静的電力の抑制できる可能性が大きい要素である．

キャッシュのリーク電流を削減する手法は，多く提案されている．Selective cache ways[1] は，アプリケーションのプロファイル情報を基に，必要の無いウェイの電力供給を静的に止めることで電力消費を削減する手法である．しかし，アプリケーションのプロファイル情報が必要で，あらゆるアプリケーションに応じた動的な制御ができない．Cache Decay[2] はアクセスが一定期間無いキャッシュラインの電力供給を停止することで消費電力を削減する．しかし，電力を停止することで，データの保持ができず，性能低下を引き起こす．Drowsy Cache[3] はラインへの電力供給を停止するのではなく，データが保持できる最低限の電圧を供

† 東北大学情報科学研究科
Graduate School of Information Sciences, Tohoku University
†† 東北大学情報シナジーセンター
Information Synergy Center, Tohoku University

給し続けることで，Cache Decay の欠点を克服している．しかし，電力制御が複雑になるため，そのオーバヘッドが大きくなることが報告されている．

本論文では CMP の共有キャッシュに着目する．チップ上の複数のコアでキャッシュを共有することで，各コアが大きな容量を確保でき，各コア間で共有するデータをより高速に処理できる．しかし，データを共有しない場合にはコア間でアクセスの競合が起こり，各コアの実行性能が不均一に低下する．これは各コアが実行するアプリケーションのデータ参照の振る舞いの違いに起因するものである．その結果，CMP 全体のスループットも低下してしまう．

この問題を解決するために Dynamic Cache Partitioning[4] が提案されている．これは各コアがアクセスできるキャッシュの範囲を決めることで，コア間の不均一な性能低下を防ぐ手法である．しかし，いずれの提案も，現在重要な問題となっている消費電力削減の観点からの検討はなされていない．

本論文は，これまで提案された Cache Partitioning と電力供給を部分的に停止する消費電力削減手法を組み合わせることにより，性能を維持しつつ低消費電力を実現するウェイアロケーション型共有キャッシュを提案する．本提案手法は，消費電力削減手法のひとつである Way-adaptable Cache[5] に Cache Partitioning 機能によって構成される．Way-adaptable Cache では，アプリケーションの参照局所性の高さを実行時に測定することで，そのアプリケーションが必要としているキャッシュ容量を適切に予測できる．この機能を Cache Partitioning に拡張することで，キャッシュ競合による性能低下を防ぐことができる上に，データの検索をするエリアを制限できるため，動的電力の削減が期待できる．さらに電力供給をウェイ単位で動的に停止することで動的電力に加え，静的電力の削減が期待できる．その結果，性能を維持した上で，消費電力の削減が可能となる．

本論文では，キャッシュアクセスの振る舞いの異なるベンチマーク用いて本提案手法の電力評価を行い，その有効性と限界明らかにする．

2. ウェイアロケーション型共有キャッシュ

2.1 前提条件

はじめに，提案するウェイアロケーション型共有キャッシュ機構について述べる前に前提となる条件を以下に述べる．

- 本論文で想定しているキャッシュメモリは，一般的なセットアソシアティブキャッシュであり，さらに各ウェイ単位で電力供給を停止，再開が可能なハードウェア機構を有している．電源供給を止めることによりリーク電流を削減でき，動的電力と静的電力をいずれも削減することができる．また，各ウェイを構成するサブアレイは，各コアが排他的にアクセスすることが可能である．つまり，各ウェイ単位でアクセスを許可するコアを指定できる機構を有しているとする．
- キャッシュを共有するスレッド間でいずれのアドレススペースも共有しないものとする．つまり，各コアで異なるプロセスを実行するものとする．同じデータを共有するような場合にはそれらのスレッドデータ参照の特徴が似ており，キャッシュの共有が各コアの実行性能に不均一な影響を与えることが少ないためである．
- 提案機構は LRU 置換を用いる．これは実際のプロセッサにおいても，pseudo-LRU 置換のように LRU を近似的に実現する置換方法が用いられているからである．また，提案機構は排他的アクセスを行うため，アクセスするコアがアクセス可能であり，かつ電力供給されている領域内での LRU ブロックを置換ブロックとする．

2.2 提案機構の概要

2.1 節で述べた前提条件の下で，本論文ではウェイアロケーション型共有キャッシュ機構を提案する．図 1 にその概念図を示す．本機構は，大容量の共有 2 次キャッシュのウェイを各コアに専用のキャッシュメモリとして動的に割り当てるものである．つまり，仮想的なパーティションをキャッシュに設け，各コアはパーティションを越えて許可されていない領域にアクセスすることはできない．さらに，割り当てられた領域の全容量を必要としていない場合は，必要の無いウェイ数を不活性にすることができる．

2 コア CMP のときの制御フローを図 2 に示す．本提案機構は，2 種類の制御機能を用いる．1 つ目のウェイアロケーション機能は，各コアが実行性能を維持するために必要なキャッシュ容量を推定し，全コアが性能を発揮できるようにキャッシュ容量を公平に割り当てる機能である．2 つ目の電力制御機能は，各コアで実行されているスレッドが割り当てられた全容量を必要としていない場合に，消費電力削減のために一部のウェイを不活性化する電力制御機能である．この 2 つの制御をある一定の期間毎に行い，通常の実行時はアクセスをサンプリングして統計情報を収集する．q その情報を基に，局所性の程度を評価し，各ウェイに対して割り当てられるコアと電力供給の ON，OFF を決定する．ただし，全コアが不活性ウェイを保持している場合，全ウェイが活性化していない限り，ウェイアロケーション制御は性能に影響を及ぼさない．そのため，コア i に割り当てらている不活性ウェイ数を $Inact_i$ とした場合，全てのコアが不活性ウェイを有するとき ($Inact_i > 0$)，ウェイアロケーションは行わない．

2.3 局所性評価量

まず，ウェイアロケーションと電力制御の双方で用いられる参照の時間的局所性の定量的評価について議

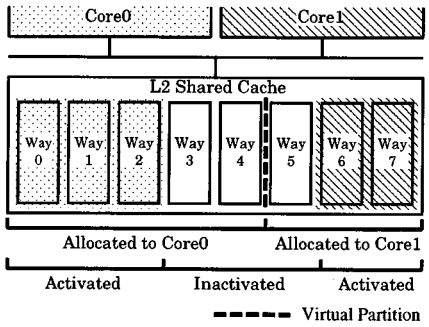

図 1 提案機構の概念図

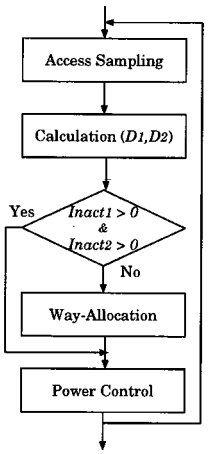

図 2 提案機構の制御フロー (2 コア CMP)

論する．図 3 は同一ブロックへの参照の間隔をヒストグラム化したものである．横軸は参照の間隔を縦軸はアクセス数を示している．左図は時間的局所性が高い場合，右図は低い場合である．参照の時間的局所性から一般に MRU ブロックへのアクセス数が多く，LRU ブロックへのアクセス数は少ない．局所性が高い場合，LRU ブロックにヒットする確率は小さく，ウェイ数を減らしても性能の低下は小さいと推測できる．逆に分散していて局所性が小さい場合，LRU ブロックへのヒットする割合から考えて，キャッシュを広範囲に利用しているので，より多くのキャッシュを必要としていると推測され，ウェイ数を増加させることで性能向上が期待できる．

ここで，局所性の程度は MRU ブロックと LRU ブロックへのアクセスの比で近似できることに着目し，アプリケーションのキャッシュ上での時間的局所性の振る舞いを定量化するために，局所性評価量

$$D = \frac{LRUcount}{MRUcount} \tag{1}$$

を用いる[5]．$LRUcount$ は LRU ブロックへのアクセス数，$MRUcount$ は MRU ブロックへのアクセス数である．

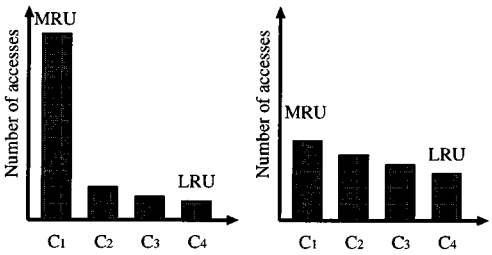

図 3 アクセスの局所性分析

2.4 ウェイアロケーション機能

次に，ウェイアロケーション型共有キャッシュ機構の 1 段階目であるウェイアロケーション機能について述べる．一般的なキャッシュメモリはウェイと呼ばれる単位に分割されている．ウェイアロケーションはこのウェイ単位で各コアにキャッシュを割り当てる．

この制御手法として前節で述べた局所性評価量 D を用いて，キャッシュ参照の局所性を考慮に入れたアロケーションを提案する．この手法はキャッシュ参照の局所性の高さがアプリケーションの必要としているキャッシュサイズに比例していることに着目している．まず，コア $i(i = 0, 1)$ からのアクセスについて D_i を求める．D_i について，

$$\text{if } D_k < D_l \quad \begin{cases} Alloc_k & +=1 \\ Alloc_l & -=1 \end{cases} \tag{2}$$

を行う．ここで，$Alloc_i$ は，コア i にアロケーションされたウェイ数であり，全ウェイ数 $Alloc_{all}$ をとしたとき

$$Alloc_{all} = \sum Alloc_i \tag{3}$$

を満たす．また，$D_0 = D_1$ の場合，再アロケーションは行わない．

この手法により，参照局所性の程度に応じたウェイアロケーションを実現することができる．この割り当てられた容量について，次節で述べる電力制御を適用することで必要に応じて不活性 (電力供給停止) 状態のウェイを作りキャッシュ全体の消費電力の削減を図る．

2.5 電力制御機能

ウェイアロケーション機能においては，D_i の相対的な比較によって適切なウェイアロケーションを行った．本節の電力制御機能では，Way-adaptable Cache の制御法[5] を利用し，D_i を絶対値との比較によって，D_i に応じたウェイ数を活性化し，性能低下を許容範囲内に留める．

効果的で適切なウェイ数の制御を行うためには，アプリケーションが今必要としているキャッシュ容量を利用状況から予測する局所的判断と，過渡的な要求の変化に過剰に反応しないように時系列から全体的な傾

向を把握する大域的判断の二つが必要である．提案機構の制御には，局所的判断のために局所的評価量と，大域的判断のために N ビットステートマシンをそれぞれ用いる．

2.5.1 局所的判断

局所的なキャッシュの必要量を求めるために式 (1) で定義された局所性評価量 D を再び用いる．閾値 $(t_1, t_2)(t_1 < t2)$ を導入し，D がある閾値 t_1 以下の場合は，MRU ブロック近辺に著しくアクセスが集中していると判断し，1 ウェイを不活性化するべきと判断する．また，D が別の閾値 t_2 を越えている場合は，参照局所性を示すヒストグラムの分布が比較的広く分散していて局所性が低いといえる．この場合，ウェイ数を増加させることで大きな性能向上が得られると考えられることから，不活性化されているウェイを 1 つ活性化する．

t_1 と t_2 の値は大きくなるほど不活性化判断する範囲が大きくなり，相対的に活性化判断する範囲が小さくなる．そのため，より少ない活性ウェイ数で稼働する．つまり，電力消費を削減することができ，省電力指向のパラメータ設定で．あると考えられる．一方で t_1, t_2 の値が小さいと活性ウェイがより多くなることになる．つまり，性能を維持することを優先する，性能指向の設定であると考えられる．

2.5.2 大域的判断

2.5.1 節で，D を閾値と比較することで，増加，減少，維持の 3 つの判定を行うことを示した．このような判定方法は，常に安定したアクセス分布を示すアプリケーションに対しては有効に機能する．しかし，アクセスの分布が急激に変化するアプリケーションの場合，2.5.1 節で示した局所的評価だけで制御すると活性化している容量が増減を短期間に繰り返すことになる．しかも，これは本来の目的とするタイミングとは異なるので，ミス率の増加やキャッシュの増減制御のオーバヘッドの著しい増加の原因となる．このような場合，より長い期間で安定した状況を見つけ出し，キャッシュを安定して制御することでウェイ数の増減によって引き起こされるオーバヘッドを抑える必要がある．キャッシュアクセスの大域的な振る舞いを評価し，制御を安定させるために N ビットステートマシンを用いる．

例として図 4 に非対称型の 2 ビットステートマシンの状態遷移図を示す．まず，2 ビットカウンタで表すことのできる 4 個の状態を考える．00 を増加状態，11 を減少状態とし，それ以外の場合を維持状態とする．入力は局所的判断で得られた結果であり，入力に応じた遷移後の状態に対応する処理（増加，減少，維持）を出力をとする．局所的判断によりウェイ数を増加するべきと判定された場合，状態 00 に遷移し，逆に減少するべきと判定した場合 +1 した状態に遷移する．このようにフィルタにかけることで，高い周期で

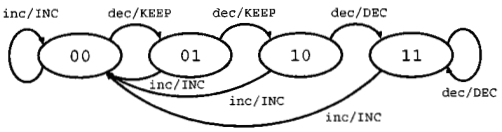

図 4 非対称 2 ビットステートマシン

増加・減少を繰り返すことを防ぐことができ，安定した制御を行うことができる．また，増加判定に傾いた非対称型にすることで，ウェイを減らすような制御は慎重に，ウェイの増加要求に対しては早急に対応することが可能であり，不用意な性能低下を防ぐことができる．

3. 評価実験と考察

3.1 実験環境

提案手法の評価のために CMP のシミュレーションツールである M5[6] と，キャッシュの電力評価ツールである CACTI4.2[7] を用いる．評価するプロセッサモデルは L2 キャッシュを共有する 2 コアの CMP であり，Alpha 命令セットアーキテクチャを用いている．実行命令数は 5 億命令とする．提案キャッシュ機構の制御間隔は 100,000 アクセス毎とする．主なシミュレーションパラメータを表 1 に示す．

ベンチマークアプリケーションとしては SPEC2006 から表 2 に示す 6 個のアプリケーションを用いて，各コアで異なるアプリケーションをそれぞれ 1 個づつ実行させる．さまざまな性質を持つベンチマークを公平に選択するために，キャッシュリソースの utility グラフを使用する[8]．utility グラフとは，活性ウェイ数が各アプリケーションの実行性能に与える影響を示したグラフである．図 5 に評価に用いるアプリケーションの utility グラフを示す．横軸が活性化ウェイ数，縦軸が IPC である．この utility グラフの形状により，SPEC2006 のアプリケーションは low-utility と，high-utility, saturating utility を持つアプリケーションの 3 種類に分類できる．low-utility は活性ウェイ数が性能に影響をほとんど及ぼさないものである．high-utility は逆に容量とウェイ数の増減が大きく性能に影響するものである．saturating utility を持つものは最初の数ウェイだけが影響を受け，残りはほとんど影響を受けないものである．本論文では，それぞれから 2 つのアプリケーションを選択している．表 2 に評価を行ったベンチマークとその utility を示す．

提案キャッシュの実行性能の評価として，*Weighted Speedup* を用いる．CMP で複数のアプリケーションが同時実行している時，アプリケーション i の実行したコアの IPC を IPC_i，アプリケーション i を単独で実行したときの IPC を $SingleIPC_i$ とすると，*Weighted Speedup* は，

表 1　主なシミュレーションパラメータ

Parameters	Value
fetch width	8 instrs
decode width	8 instrs
issue width	8 instrs
commit width	8 instrs
Inst. queue	64 instrs
LSQ size	32 entries
L1 Icache	32kB, 2-way, 32B-line
	1 cycle latency
L1 Dcache	32kB, 2-way, 32B-line
	1 cycle latency
L2 cache	1024kB, 32-way, 64B-line
	14 cycle latency, 1 R/WPort
main memory	100 cycle latency
Frequency	1GHz
Technology	70nm
Vdd	0.9V

表 2　ベンチマークアプリケーション

Name	Function	Utility
`gcc`	C Compiler	High
`bzip2`	Compression	High
`dealII`	Finite Element Analysis	Low
`sjeng`	Artificial Intelligence: chess	Low
`milc`	Quantum Chromodynamics	Saturating
`libzuantum`	Quantum Computing	Saturating

$$Weighted\ Speedup = \sum(\frac{IPC_i}{SingleIPC_i}) \quad (4)$$

で定義される．

3.2　ウェイアロケーション機能の評価

まず，ウェイアロケーション機能のみについて，その性能を評価する．図 6 に全てのベンチマークの組み合わせについての結果を示す．縦軸は *Weighted Speedup* を示しており，各ベンチマークの組み合わせについて従来のキャッシュ(Normal) とウェイアロケーション機能を有するキャッシュ(Walloc) を比較している．いくつかのベンチマークの組み合わせで Walloc が Normal を上回っていることが分かる．平均すると，性能が約 2%向上している．つまり，従来のキャッシュよりも競合が少なく，性能を維持することができ，適切なウェイアロケーションができるといえる．また，従来の各コアへウェイを割り当てる際の基準として参照の局所性を用いることの妥当性も明らかになった．

`bzip2` が含まれる組み合わせに注目すると，性能を大きく低下させる場合と向上させる場合があることがわかる．これは図 5(b) で示すように，`bzip2` が活性化ウェイ数が増加しても性能の向上が収束することないため，適切なアロケーションの決定が難しいためと考えられる．

3.3　提案機構の電力評価

次に電力制御機能を加えた提案キャッシュ機構の消費電力を評価する．評価指標としては提案キャッシュ機構の消費エネルギーと *Weighted Speedup* を用いる．電力制御機能の閾値 (t_1, t_2) は，性能指向から省電力指向までの設定の機能を確認するために $(0.1, 0.5)$, $(0.01, 0.05)$, $(0.001, 0.005)$ の 3 種類を用いる．ステートマシンは 3-bit の非対称型を用いる．不活性化の際にデータのコヒーレンシを保つために必要になるデータのライトバックにかかるオーバヘッドは考慮しているが，ウェイの活性化・不活性化にかかるオーバヘッドは考慮していない．

図 7 と図 8 に評価結果を示す．図 7 は全アプリケーションの組み合わせにおける消費エネルギーの平均を従来のキャッシュでの消費エネルギで正規化している．同様に，図 8 は *Weighted Speedup* の平均を正規化している．

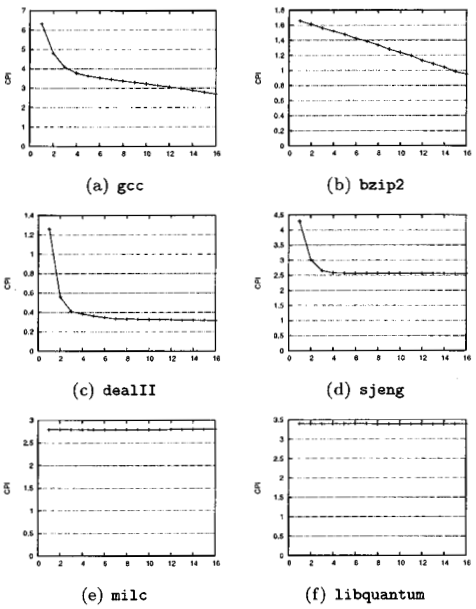

図 5　Utility グラフ

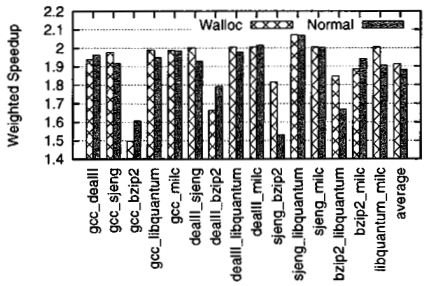

図 6　ウェイアロケーション評価

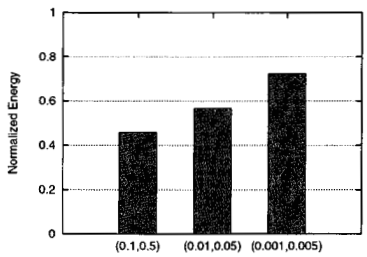

図 7 評価結果 (消費エネルギ)

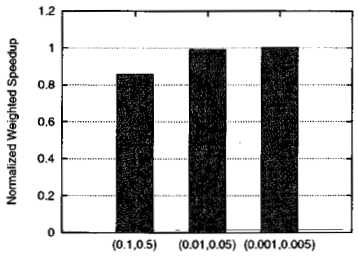

図 8 評価結果 (*Weighted Speedup*)

閾値が $(t_1, t_2) = (0.001, 0.005)$ のとき，0.3%の性能向上に対して，消費エネルギを平均 28%削減することができた．また，閾値が $(t_1, t_2) = (0.1, 0.5)$ のとき，消費エネルギを平均 54%削減することができたが，性能は約 15%低下した．これらのことから，閾値が小さい場合は性能指向の設定であり，大きい場合は省電力指向の設定であることが示された．したがって，閾値 (t_1, t_2) を操作することで，性能と消費電力の間でトレードオフをとることができる．

4. 結　　言

本論文では CMP の共有キャッシュメモリの電力消費を低減し，最適なキャッシュサイズを提供できるウェイアロケーション型共有キャッシュ機構を提案した．参照の局所性を考慮したウェイアロケーションを導入することにより，従来の共有キャッシュよりも CMP 全体の IPC を向上させることが可能であることを示した．また，電力制御機能を導入し，閾値を性能指向に設定することにより共有キャッシュを用いた場合で約 0.3%の性能向上で 28%の消費エネルギの削減を達成した．また，閾値を省電力指向に設定することにより約 54%の消費エネルギの削減を約 15%の性能低下で達成した．

今後の課題としては，high-utility を持つアプリケーションに対しても適切なウェイアロケーションが可能な手法に修正を加えていく共に，ハードウェア構成に関して詳細な検討を加え，追加ハードウェアによるオーバヘッドを評価する．また，他のキャッシュパーティショニング手法や電力削減手法とも比較評価していく．さらに，今後主流になると思われる 4 つ以上のコアを持つ CMP アーキテクチャに対する，本提案手法の有効性の評価を予定している．

謝辞　本論文を書くにあたり，議論に協力頂いた東北大学小林・滝沢研究室の船矢祐介，佐藤雅之両名に感謝致します．本研究の一部は、文科省科研費「ハードウェア・ソフトウェア協調型高効率マルチスレッドスケジューリングに関する研究」(課題番号 18300011) の補助を受けている。

参 考 文 献

1) Albonesi, D.: Selective cache ways: on-demand cache resource allocation, *Microarchitecture, 1999. MICRO-32. Proceedings. 32nd Annual International Symposium on*, pp.248–259 (1999).
2) Kaxiras, S., Hu, Z. and Martonosi, M.: Cache decay: exploiting generational behavior to reduce cache leakage power, *Computer Architecture, 2001. Proceedings. 28th Annual International Symposium on*, pp.240–251 (2001).
3) Flautner, K., Kim, N.S., Martin, S., Blaauw, D. and Mudge, T.: Drowsy caches: simple techniques for reducing leakage power, *Computer Architecture, 2002. Proceedings. 29th Annual International Symposium on*, pp.148–157 (2002).
4) Suh, G.E., Rudolph, L. and Devadas, S.: Dynamic Partitioning of Shared Cache Memory, *Journal of Supercomputing*, Vol.28, No.1, pp. 7–26 (2004).
5) Kobayashi, H., Kotera, I. and Takizawa, H.: Locality analysis to control dynamically way-adaptable caches, *SIGARCH Comput. Archit. News*, Vol.33, No.3, pp.25–32 (2005).
6) Binkert, N., Dreslinski, R., Hsu, L., Lim, K., Saidi, A. and Reinhardt, S.: The M5 Simulator: Modeling Networked Systems, *Micro, IEEE*, Vol.26, No.4, pp.52–60 (2006).
7) Wilton, S. and Jouppi, N.: CACTI: an enhanced cache access and cycle time model, *Solid-State Circuits, IEEE Journal of*, Vol.31, No.5, pp.677–688 (1996).
8) Qureshi, M.K. and Patt, Y.N.: Utility-Based Cache Partitioning: A Low-Overhead, High-Performance, Runtime Mechanism to Partition Shared Caches, *MICRO 39: Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, Washington, DC, USA, IEEE Computer Society, pp.423–432 (2006).