キャッシュ非共有型マルチコアプロセッサにおけるキャッシュの性能改善に関する研究

伊藤, 義崇
九州工業大学情報工学部

千代延, 昭宏
九州工業大学情報工学部

佐藤, 寿倫
九州大学LSI研究センター

出版情報：電子情報通信学会技術研究報告, CPSY2006-51. 106 (436), pp.63-68, 2006-12. 電子情報通信学会CPSY研究会
バージョン：
権利関係：

キャッシュ非共有型マルチコアプロセッサにおける キャッシュの性能改善に関する研究

伊藤　義崇[†]　千代延　昭宏[†]　佐藤　寿倫[‡]

†九州工業大学情報工学部 〒820-8502 福岡県飯塚市川津 680 番 4
‡九州大学システム LSI 研究センター 〒814-0001 福岡市早良区百道浜 3-8-33-3F
E-mail: †yitou@mickey.ai.kyutech.ac.jp, †chiyo@mickey.ai.kyutech.ac.jp, ‡tsato@slrc.kyushu-u.ac.jp

あらまし　オフチップアクセス速度が問題を解決するために，チップマルチプロセッサ（CMP）上には大容量のキャッシュが搭載されている．オンチップキャッシュ容量を有効に利用するためには，共有型キャッシュが望ましい. しかし微細化が進み配線遅延の問題が深刻化すると, 大容量な共有型キャッシュでのレイテンシが問題となる．そこで今回，オンチップキャッシュとして非共有型キャッシュを用いることを提案する．非共有型キャッシュは共有型と比べて配線遅延の影響を受けにくいが，大きなミス率が問題である．本稿では非共有型キャッシュにデータ圧縮技術を用いることでこの問題の解決を図る．データ圧縮技術を用いることで，オフチップとの通信量も削減することができる．

キーワード　CMP，非共有型キャッシュ，データ圧縮

Utilizing Data Compression to Improve Cache Performance in Multicore Processors with Dedicated Caches

Yoshitaka Ito[†]　Akihiro Chiyonobu[†] and　Toshinori Sato[‡]

†Kyushu Institute of Technology,　680-4 Kawazu, Iizuka, 820-8502 Japan
‡Kyushu University, 3-8-33-3F Momochihama, Sawara-ku, Fukuoka, 814-0001 Japan
E-mail: †yitou@mickey.ai.kyutech.ac.jp, †chiyo@mickey.ai.kyutech.ac.jp, ‡tsato@slrc.kyushu-u.ac.jp

Abstract　Chip multiprocessors (CMPs) have large on-chip caches in order to mitigate large off-chip access latency.　For efficiently utilizing on-chip cache capacity, on-chip caches should be shared.　However, increasing wire delay due to advanced process technologies, increasing on-chip access latency of the shared caches becomes serious.　In this paper, we propose to use data compression for on-chip dedicated caches.　Using data compression reduces on-chip cache miss rate and off-chip memory traffic.

Keyword　CMP,　Dedicated cache,　Data Compression

1. はじめに

マイクロプロセッサ性能の向上はトランジスタの微細化による動作周波数の向上に支えられてきた．しかし近年では，動作周波数の向上によるチップ温度の上昇や消費電力の増加が問題となっている. そのため，マイクロプロセッサの性能向上に動作周波数の向上を利用することは難しい．この打開策として，一つのチップ上に複数のプロセッサコアを搭載するチップマルチプロセッサ（CMP）アーキテクチャが注目を集めている．低速な動作周波数のコアを採用することで消費電力や熱の発生を抑制しながら，チップ内並列処理によって高い性能を実現する．しかし，搭載コア数を増やしても必ずしもプロセッサ性能が向上するわけではない．この理由として，メモリウォール問題がある．プロセッサとメモリには現在大きな速度差があり，その向上率にも大きな差がある．コアの処理性能が高くてもコアの要求するデータを十分に供給できなければ, プロセッサ全体の性能は向上しない. プロセッサ･メモリ間のデータ転送能力がボトルネックとなるわけである．さらに，処理量の増加によりメモリアクセス量が増えると, 十分なメモリバンド幅の確保が難しくなる．バンド幅はパッケージにより物理的に制限される I/O ピン数で決まるため, バンド幅を増やすことは難しい．性能の向上に必要な十分なメモリバンド幅の確保は難しくなることが考えられる．

これらのオフチップメモリアクセスの問題を解決

するために,CMP ではオンチップメモリとして大容量キャッシュを載せている．オンチップキャッシュでのミス率を減らし，オフチップへのアクセスを減らすことが目的である．しかし，大容量キャッシュでは配線遅延が問題となる[3][7]．配線遅延とは，配線の抵抗や容量，隣接する配線間の容量などによって生じるもので，微細化に伴い増大する．このために大容量キャッシュになることで，コアとキャッシュ間の物理的距離が増加すると，ヒットレイテンシが大きくなる．この問題はキャッシュウォールと呼ばれる[10]．CMP では代表的なオンチップキャッシュ構成として，各コア間でキャッシュを共有する共有型キャッシュと，各コアが専用のキャッシュを持つ非共有型キャッシュの方式がある．共有型キャッシュでは多くがバンク分けした構造を持つ[7]が，レイテンシは物理的に遠いバンクに対してのアクセスで決定する．これに対し，非共有型キャッシュではコアの近くに位置するキャッシュへアクセスするため，配線遅延が小さい．しかし，非共有型キャッシュではキャッシュ全体を有効利用できず，ミス率が増加するためにオフチップアクセスが共有型に比べて増える．

本稿では，非共有型キャッシュにデータ圧縮手法を適用することで，キャッシュミス率の低減とオフチップバンド幅の増大を狙う方式を提案する．本稿の構成は以下のとおりである．2 節で関連研究を述べる．3 節で提案する圧縮を利用したキャッシュの方式について述べる．4 節で評価手法について述べ，5 節で評価の結果を示す．6 節でまとめとする．

2. 関連研究

CMP でのオンチップキャッシュの性能改善を図っている研究は多く存在する．

2 次キャッシュの大きなアクセスレイテンシを解決する方法として,D-NUCA[4][6]や NuRapid[12][13]ではデータマイグレーションを利用している．[3]では，加えてプリフェッチや伝送線路も検討している．CMP のオンチップの共有型キャッシュに対して，圧縮とプリフェッチを組み合わせることでキャッシュ有効容量の増大やバンド幅を軽減し，性能向上を図っているものもある[1]．非共有型キャッシュの利用効率を上げる方法として，コヒーレンスプロトコルを利用して，非共有のキャッシュ間で協調して他コアのキャッシュを利用する方法がある[7]．

キャッシュ内データ圧縮では，ファイル圧縮とは異なり圧縮率を重視しない．プロセッサの動作中に行うため高速性が求められる．そのため，LZ-Base のような辞書式圧縮方式ではなく Significance-Based Compression（SBC）アルゴリズム[2]が多く採用される．SBC では，符号の情報しか持たない上位ビットを 1 ビットで表現することでデータサイズを圧縮する．SBC で有名なキャッシュ圧縮方法として Frequent Pattern Compression(FPC)がある．これは，キャッシュブロック内で頻繁に使われるデータパターンを，それを示す 3 ビットの接頭辞と圧縮データの形で圧縮するものである[2]．当然ながら，SBC は辞書式圧縮方式に比べると圧縮効果は低い．データサイズを半分以下にできるブロックの数はそれほど多くないため，一部の圧縮できないデータを保持する領域を付け加える方法が提案されている[8]．

本稿では，圧縮を適用することで非共有型キャッシュでのキャッシュ領域有効活用し，キャッシュ有効容量の増大とオフチップバンド幅の増大を狙う．

3. 提案方式

本節では，本稿で提案する非共有型キャッシュ圧縮方式について述べる．最初に，CMP 上での圧縮キャッシュの構成を述べ，次に圧縮方法について述べる．最後に提案方式の動作を説明する．

3.1. 非共有型キャッシュ上でのデータ圧縮

共有型 2 次キャッシュと非共有型 2 次キャッシュのモデルを図 1 に示す．この図では 1 チップ上にスーパースカラ型プロセッサコアを 2 個搭載している．オンチップメモリ階層において，1 次キャッシュは命令・データ分離型で各コア占有の非共有型となっている．対して，2 次キャッシュは，命令・データ統合型でキャッシュを 2 コアで共有する共有型と，1 コアで占有する非共有型のモデルを使う．オフチップのメモリは全て共有型である．

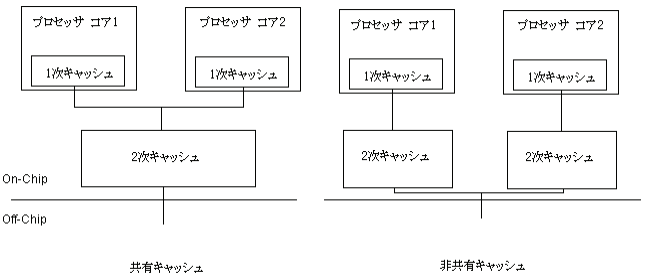

図 1:オンチップキャッシュ構成

共有型の利点は，キャッシュの持つ物理的容量の全てを活用できるため，容量性のミスを削減できることである．欠点は，最近の大容量化と微細化による配線遅延によって，アクセスレイテンシが大きくなることである[7]．また，共有することで競合性ミスが発生する可能性がある．

非共有型の利点は，共有型より物理的に小さいために配線遅延によるレイテンシの増加が小さいことである．欠点は，キャッシュ容量の全てが有効に利用されるわけではないため，容量性のミス率が共有型キャッシュよりも高いことである．また，オフチップへアクセスするバスを共有しているので，バス競合が生じる問題がある．

非共有型キャッシュの大きなミス率とバス競合の問題を解決するために，データ圧縮を適用することを検討する．オンチップキャッシュ上でデータを圧縮すればキャッシュ内の有効容量を増やすことができるので，2 次キャッシュでのミス率を低減できる．オフチップメモリとのデータの転送を圧縮形式で行うことでオフチップバストラフィックを軽減し，オフチップバンド幅を増大する．キャッシュの構成を図 2 に示す．2 次キャッシュ上ではデータを圧縮することで，オンチップキャッシュの有効容量を増加する．一般に 1 次キャッシュは高速な動作を期待される．ブロックが圧縮されていると展開による遅延がアクセスレイテンシに追加されるため，データは展開形式で格納される必要がある．2 次キャッシュから読み出された圧縮キャッシュブロックは展開されて 1 次キャッシュに転送される．

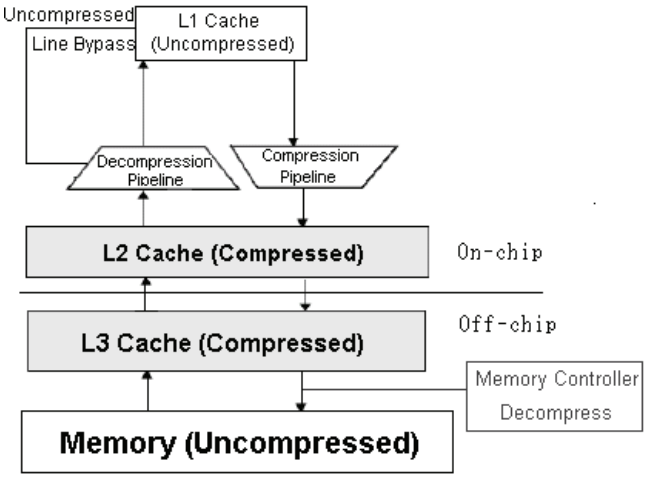

図 2:データ圧縮を利用するメモリ階層

オフチップキャッシュ(図では 3 次キャッシュ)とのデータ転送では，バストラフィックを軽減するために圧縮可能なブロックは圧縮形式のままで転送する．

3.2. Significance-Based データ圧縮手法

プロセッサ･メモリ間の速度差を埋めるためには，大容量のキャッシュが必要である．そのため，データ圧縮によってキャッシュ内データの有効容量の増大を行うことは有用である．データを圧縮すると，プロセッサで使用する際に展開することが必要になる．圧縮展開が遅いと，その遅延がキャッシュアクセスレイテンシに加わり性能の向上の妨げになる．そのため，通常使用されるファイルの圧縮などとは違い，高速な圧縮展開を行うアルゴリズムである必要がある．

我々は圧縮方式として，SBC アルゴリズムを採用している．キャッシュブロック内のデータを 1 語（32 ビット）単位のデータとして分割し，その 32 ビットデータが 16 ビットの符号拡張データとして表現できれば，符号情報と 16 ビットデータに圧縮する．また，1 語データが符号情報しか持たない場合は，符号情報のみ格納することになる．各語には 3 ビットの符号・圧縮情報を付加する．圧縮されているか否かを表す 1 ビット（圧縮ビット），16 ビットに圧縮されているのかそれとも符号情報のみなのかを表す 1 ビット（完全圧縮ビット），そして符号を表す 1 ビット（符号ビット）である．この拡張された符号・圧縮情報も通常のデータ領域に保持される．符号・圧縮情報が拡張されたキャッシュブロックで圧縮を実施した様子を図 3 に示す．

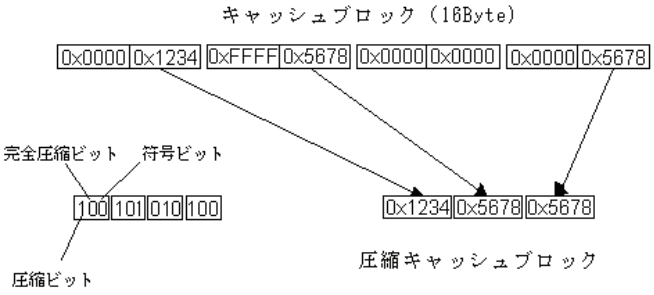

図 3:圧縮キャッシュのブロック

図 3 では,最初のデータが 16 ビットの正符号拡張，次のデータが 16 ビットの負符号拡張となる．3 番目のデータは符号情報しか持たないので，完全圧縮ビットを立てる.最後のデータは 16 ビットの正符号拡張である．こうすることで，12 ビットの符号・圧縮情報を追加しても元のブロックサイズの半分以下に圧縮できている．

圧縮データを詰めるために 2 次キャッシュではタグ領域を 2 倍もつ必要がある．符号・圧縮情報を考慮しても元のサイズの半分以下に圧縮できるブロックは圧縮形式で保持される．その同じインデックスのブロックに，別の圧縮形式のデータを書き込む時には，現在のデータは追い出されず，同じブロック内に保持される．二つの圧縮形式データを区別するために，もう一つのタグが必要になる．

展開方法は以下の通りである．ブロックの先頭に格納されている圧縮情報から，容易に圧縮されているかどうかを知ることができる．圧縮時には，各語の符号情報に基づいて符号拡張を行う．

上述した基本的な SBC アルゴリズムでは，圧縮できるデータはそれほど多くはない．そこで，圧縮できないデータの保持を許容する方法を利用する[8]．そのために，許容領域と呼ばれる領域を付け加える．例えば図 4 のようにブロックサイズの 1/8 に相当する保持領域を追加すると，図 3 を用いて説明した SBC アルゴリズムでは圧縮形式で保持できないブロックも，図 4 のように圧縮形式で保持可能になる．この追加された保持領域を許容領域と呼ぶ．

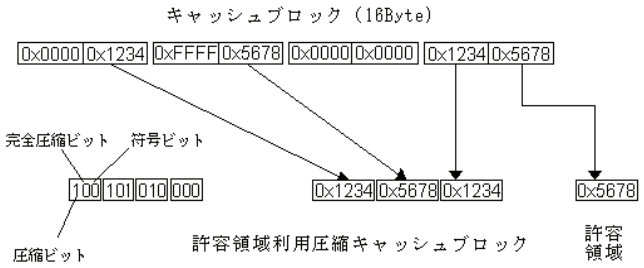

図 4:許容領域を利用する SBC アルゴリズム

このブロックでは最後のデータが圧縮できないため，ブロック全体としては元のサイズの半分以下に圧縮出来ない．このような圧縮できないデータを少数持つブロックは多く存在するため，わずかな許容領域を追加するだけで，SBC アルゴリズムでは圧縮不可能なブロックを圧縮できるようになる．

3.3. データの圧縮と展開

キャッシュアクセス時のデータの圧縮･展開動作について説明する．プロセッサからのデータリード要求で 1 次キャッシュにおいてミスが起こり 2 次キャッシュでヒットした場合，該当ブロックが圧縮されていれば展開回路を通してデータが転送される．圧縮されていなければ展開回路はスルーされて転送されるため，展開レイテンシはかからない．1 次キャッシュでデータの追い出しが必要になったときは，圧縮回路を使ってデータの圧縮を試みる．ブロックサイズの半分以下に圧縮できたデータは 2 次キャッシュへ圧縮形式で格納される．圧縮できなかったデータは展開形式で渡される．

2 次キャッシュがオフチップキャッシュのデータを読み込むときには,圧縮形式でデータが受け渡される．よって，オフチップバスは圧縮形式でデータの受け渡しが可能となり，バスのトラフィックを減らすことが出来る．

プロセッサからのライト要求によって圧縮不可能なデータに変化することがある．2 次キャッシュにヒットしたブロックが圧縮形式であるにも関わらず，ライトデータが圧縮できない場合には，2 次キャッシュの同じブロックに圧縮されて配置されているもう一方のデータを，下位レベルのメモリへ追い出す． 2 次キャッシュにヒットしたブロックが展開形式であり，ライトデータが圧縮できる場合には，展開形式のデータを破棄し，圧縮形式で格納する．余ったデータ領域は未使用の状態になる．

2 次キャッシュからデータが追い出されるときには，圧縮データならばそのままオフチップキャッシュへ圧縮形式で渡すことで，バストラフィックを減らす[1]．

メインメモリ内部でも圧縮形式で扱うためには OS との協調動作が必要になるため，今回は検討していない．

4. 評価手法

ミシガン大学で開発されたマルチプロセッサシミュレータ M5[9]を用いる．M5 はイベントドリブン方式のサイクルアキュレートなシミュレータである．本評価では，システムコールをエミュレーションした走行モードで実験を行う．命令セットは Alpha ISA である．シミュレーションには SPEC2000 CPU ベンチマークセット[11]を用いる．入力データには reference を用いる．今回の評価では 2 コアチップを仮定し，同時に 2 つのプログラムを独立に実行する．どちらか一方のプログラムがプログラムの先頭から 5 億命令の実行を完了した時点でシミュレーションを終了する．実行したベンチマークの組を表 1 に示す．

表 1：ベンチマークセット

CINT ベンチマーク
bzip - parser
mcf - gcc
vpr - twolf
CFP ベンチマーク
equake - galgel
applu - lucas
galgel - swim

プロセッサとキャッシュの構成を表 2 に示す．ここで，メインメモリレイテンシはメモリの読み出しにかかるレイテンシである．共有型と非共有型キャッシュのアクセス速度の差は CACTI4.0[5]で求めた．共有型キャッシュのアクセスレイテンシには，バスのアービトレーションとして 1 サイクルを考慮している．今回の評価では 2 コアチップを仮定しているので，共有型キャッシュ容量は非共有型の二倍に過ぎない．そのため配線遅延の問題が顕在化していない．コアが増えて共有型キャッシュと非共有型キャッシュのサイズに大きな差が現れると，配線遅延の深刻さが明確になると考えられる．

表 2：プロセッサ構成

命令発行幅・デコード幅	8 命令
命令キューサイズ	64 エントリ
ROB サイズ	192 エントリ
LSQ サイズ	32 エントリ
iALU	4
iMul/Div	1
fALU	4
fMul/Div/Sqrt	1
L1$(データ･命令)	32KB，64B ブロック，2way, LRU, 2cycles
L2$(非共有)	512KB，64B ブロック，8way, LRU, 10cycles
L2$(共有)	1MB，64B ブロック，8way, LRU, 12cycles
オフチップバス幅	32B
オフチップバス周波数	500MHz
オフチップ L3$	2MB，64B ブロック，32way, LRU, 25cycles
メモリレイテンシ	200cycles

5. 実験結果

最初に各ベンチマークの圧縮率のデータ，次に 2 次キャッシュミス率のデータとバスアクセスレイテンシのデータを示す．最後に実行したベンチマークのサイクル数を示す．

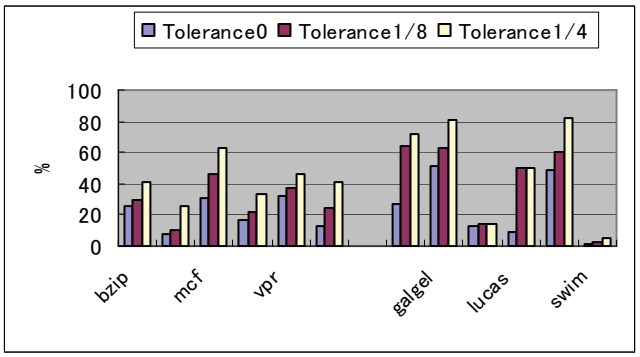

図 5:キャッシュブロック圧縮率

図 5 にデータ圧縮率を示す．縦軸は圧縮できた 2 次キャッシュのブロックの割合，横軸はベンチマークを示す．グラフは **Tolerance0** が SBC アルゴリズムの圧縮手法のみ適用した場合，**Tolerance1/8** が許容領域として元のキャッシュ領域の 1/8 のサイズの領域を加えた場合，**Tolerance1/4** は 1/4 の領域を加えた場合である．グラフから **Tolerance0** では圧縮率が高くないことがわかる．許容領域を加えることで **equake** や **lucas** で大きく圧縮率を改善できていることがわかる．その他のベンチマークでも圧縮率向上に貢献している．したがって許容領域は必要である．以降示すデータはキャッシュ領域の 1/8 のサイズを許容領域として加えている場合のデータである．

図 6 に 2 次キャッシュのミス率を示す．縦軸はキャッシュミス率を示す．横軸はベンチマークプログラムを示している．**Dedicated** が圧縮を行わない非共有型キャッシュの場合で，**Compressed** が圧縮を行う非共有型キャッシュの場合である．**Shared** は共有型キャッシュの場合である．容量は非共有型キャッシュの総量に等しいが，アクセスレイテンシが増加している．**Double size cache** は非共有型キャッシュのブロックサイズを二倍にした場合である．

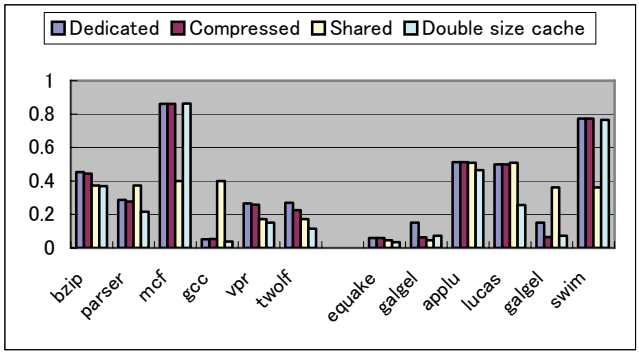

図 6:2 次キャッシュミス率

Dedicated と **Compressed** を比較すると，一部のベンチマークでは圧縮することでキャッシュミス率を低下できることが確認できる．特に **galgel** では，キャッシュ領域を二倍にした **Double size cache** と同程度までミス率を削減できている．圧縮率が低いものでは効果が小さいが，圧縮率が高くキャッシュ容量を増やすことでミス率の削減が測れるものには効果があることがわかる．

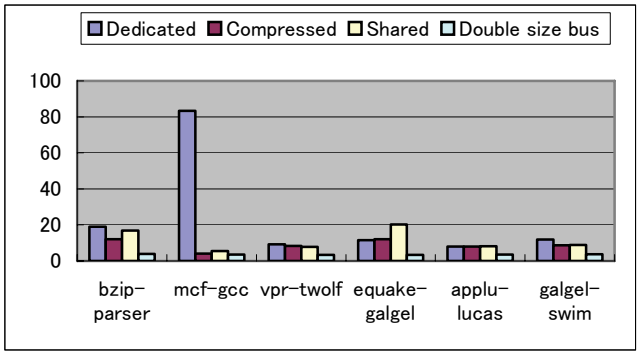

図 7:バスアクセスレイテンシ

バスアクセスレイテンシを図 7 に示す．縦軸はオフチップバスアクセスにかかったレイテンシの平均，横軸はベンチマークの組み合わせを示している．グラフ中の **Dedicated**, **Compressed**, **Shared** は図 6 の場合と同様である．**Double bus size** はオフチップバス幅を 2 倍

にした場合を示す. **Dedicated** と **Compressed** とを比較すると，一部のベンチマークでは，圧縮することによってバスレイテンシの削減ができていることがわかる. このことから，バストラフィックも削減できていると予想できる. 特に **mcf-gcc** では顕著に削減出来ている. この理由は以下のように考察できる. **mcf** の 2 次キャッシュミス率が高いのでオフチップアクセス回数が多く，バスの競合が多く発生していると考えられる. 圧縮形式での転送によって，バストラフィックが減ったために競合も減らすことができたと考えられる. **mcf-gcc** は **Shared** では **Dedicated** 程レイテンシが高くない. これは，2 次キャッシュを共有することでオフチップバスを利用する際の，競合によるバス待ちの発生がないためだと考えられる.

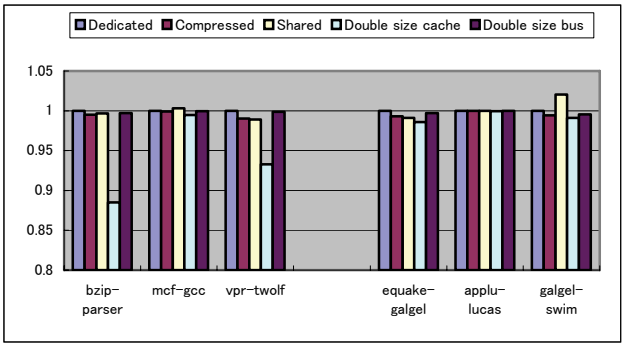

図 8:実行サイクル数

最後に図 8 に実行サイクル数を示す. もともとの構成の場合(**Dedicated**）のサイクル数で正規化している. キャッシュミス率の低下やバストラフィックの減少を果たしているにも関わらず. **Compressed** ではわずか 1%のサイクル数減少しか見られない. 性能向上には更なる改善が必要である.

6. まとめ

本稿では, CMP 上のオンチップキャッシュの問題に着目し，非共有型キャッシュに圧縮手法を適用した. オフチップアクセスを減らし，キャッシュ性能の改善によるプロセッサ性能の改善を目指した. 圧縮率が高いプログラムでの 2 次キャッシュミス率の削減と，オフチップバストラフィックの軽減ができている.

謝辞

本研究の一部は，科学研究費補助金（No.16300019, No.176549）の援助によるものである.

文　献

[1] A. R. Alameldeen, “Using compression to improve chip multiprocessor performance”, PhD thesis, University of Wisconsin-Madison, 2006.

[2] A. R. Alameldeen and D. A. Wood, “Frequent pattern compression: a significance-based compression scheme for L2 caches”, Technical Report 1500, Computer Sciences Department, University of Wisconsin-Madison, 2004.

[3] B. M. Beckmann and D. A. Wood, “Managing wire delay in large chip-multiprocessor caches”, 37th Internatiomal Symposium on Microarchitecture, 2004.

[4] C. Kim, D. Burger, and S. W. Keckler, "An adaptive, non-uniform cache structure for wire-dominated on-chip caches", 10th International Conference on Architectural Support for Programming Languages and Operating Systems, 2002.

[5] D Tarjan, S Thoziyoor, and N P. Jouppi, "CACTI 4.0", Technical Report HPL-2006-86, Hewlett-Packard Laboratories, 2006.

[6] J. Huh, C. Kim, H. Shafi, L. Zhang, D. Burger, and S. W. Keckler, "A NUCA substrate for flexible CMP cache sharing", 9th Internatiomal Conference on Supercomputing, 2005

[7] J. Chang and G. S.Sohi, “Cooperative caching for chip multiprocessors”, 33th International Symposium on Computer Architecture, 2006

[8] N. S. Kim, T. Austin, and T. Mudge, “Low-energy data cache using sign compression and cache line bisection”, 2nd Workshop on Memory Performance Issues, 2002

[9] N. L.Binkert, E. G. Hallnor, and S. K. Reinhardt, “Network-oriented full-system simulation using M5”, 6th Workshop on Computer Architecture Evaluation using Commercial Workloads, 2003.

[10] P. Michaud, "Exploiting the cache capacity of a single-chip multi-core processor with execution migration", 10th International Symposium on High Performance Computer Architecture, 2004.

[11] SPEC, http://www.spec.org/

[12] Z. Chishti, M. D. Powell, and T. N. Vijaykumar, "Distance associativity for high-performance energy-efficient non-uniform cache architectures", 36th International Symposium on Microarchitecture, 2003.

[13] Z. Chishti, M. D. Powell, and T. N. Vijaykumar, "Optimizing Replication, Communication, and Capacity Allocation in CMPs", 32nd International Symposium on Computer Architecture, 2005.