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

2006－ARC－169 (14)
2006／8／1

チップマルチプロセッサにおけるキャッシュメモリの特性解析

三原 智伸† 井上 弘士†† 村上 和彰††

† 九州大学大学院 システム情報科学府 〒816-8580 福岡県春日市春日公園 6-1
†† 九州大学大学院 システム情報科学研究院 〒816-8580 福岡県春日市春日公園 6-1
E-mail: †mihara@c.csce.kyushu-u.ac.jp, ††{inoue,murakami}@i.kyushu-u.ac.jp

あらまし 近年，より高い性能の実現を目的として1つのチップ上に複数のプロセッサコアを搭載したCMP(Chip MultiProcessor)アーキテクチャが注目されている．メモリバンド幅の制約，メモリウォール問題のさらなる深刻化を背景として，今後プロセッサシステムの高性能化を実現するには，CMPに適したメモリシステムを構築することは不可欠となる．メモリシステムの中でもオンチップ・キャッシュの構成は性能に与える影響が大きく，その主な設計選択肢としてプロセッサコア間での共有/非共有がある．本稿では，CMPにおけるキャッシュの共有と非共有によるメモリ性能への影響を解析し，L2キャッシミス率の差によりメモリ性能の優劣が異なる事を明らかにした．
キーワード チップマルチプロセッサ，メモリアーキテクチャ，オンチップ・キャッシュ

Analysis of Cache Memory Characteristics on a Chip MultiProcessor

Tomonobu MIHARA†, Koji INOUE††, and Kazuaki MURAKAMI††

† Graduate school of Information Science and Electrical Engineering, Kyushu University
†† Faculty of Information Science and Electrical Engineering, Kyushu University
E-mail: †mihara@c.csce.kyushu-u.ac.jp, ††{inoue,murakami}@i.kyushu-u.ac.jp

Abstract To achieve higher performance, CMP(Chip MultiProcessor) is focused today. Because of narrow bus bandwidth and the memory wall problem, it is necessary to design the memory system which is suitable for CMP. In the system, on-chip cache architecture has a large impact on performance, and to decide sharing/dedicating an on-chip cache among multiple processor cores is a important choice. In this paper, we studied the difference of performance in shared-cacheCMP and dedicate-cacheCMP. We analyzed the factor which impacts memory-access-time qualitatively and quantitatively, and revealed that L2cache miss rate makes the largest gap between them.
Key words CMP, memory architecture, On-chip cache

1. はじめに

1970年代初頭に世界初の1チップ・マイクロプロセッサが開発されて以来，順調な動作周波数の向上はその高性能化に大きく寄与し続けてきた．しかしながら，この手法も限界に達しつつある．動作周波数の向上に伴い消費電力が増加し，動作可能なチップ温度を超えるためである．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CMPアーキテクチャが注目されている．CMPでは，低速な動作周波数により消費電力を抑制しつつ，チップ内並列処理により高い性能を実現する．実際，サーバー処理やメディア処理を目的として，3個以上のプロセッサコアを搭載したCMPも開発されている[5]．

しかしながら，搭載するコアの数を増加させた場合，必ずしもプロセッサシステム全体の性能が向上するわけではない．その主な理由として，メモリウォール問題のさらなる深刻化が挙げられる．一般に，搭載するコアの数を増加させた場合，オフチップアクセス頻度は高くなる．しかしながら，プロセッサチップのI/Oピン数はパッケージにより物理的に制限される．そのため，十分なメモリバンド幅を確保することが難しくなる．また，オフチップメモリの読出し速度はオンチップ・メモリのそれと比較して数十倍から数百倍遅く，頻繁な主記憶アクセスは性能低下の原因となる．したがって，CMPに適したメモリシステムを構築することは性能向上のために不可欠である．

メモリシステムの中でもオンチップキャッシュの構成は性能に与える影響が大きく，その主な設計選択肢としてプロセッサコア間での共有/非共有がある[2][3]．そこで本研究では，CMPにおけるキャッシュの共有/非共有がプロセッサ性能へ与える影響を定性的かつ定量的に評価する．

複数のプロセッサコアがオンチップキャッシュを共有する場合，スレッドが使用可能な最大キャッシュ容量は非共有キャッシュ方式のそれに比べて大きくなる(総キャッシュ容量が等しい場合)．しかしながら，スレッド間でのキャッシュ領域の奪い合い，および，キャッシュ-プロセッサ間の共有バス競合が原因で性能が低下することがある．一方，オンチップキャッシュを共有しない場合，スレッドは各プロセッサコアに占有のキャッシュ領域のみしか使用できないため，共有キャッシュ方式に比

べて最大使用可能容量が制限される．さらに，主記憶への共有バス競合は性能低下要因になりうる．これらの共有/非共有の得失は，実行されるスレッドの性質によって異なる．解析の結果，共有キャッシュ型の利用可能容量増加による利得を受け，他のスレッドからの影響が利得を下回る場合，メモリ性能が高まることが明らかになった．

本稿の構成は以下の通りである．まず，第 2. 節にて，CMP における構成要素の分類をし，性能評価対象モデルならびに性能式を示す．次に，第 3. 節にて，性能式に基づく定性的評価を行う．続いて，第 4. 節では，キャッシュ構成法による性能への影響を定量的に評価する．第 5. 節にて，関連研究と本研究を比較する．最後に，第 6. 節で本稿をまとめる．

2. チップマルチプロセッサ

本節では，まず CMP におけるオンチップ・ネットワーク構成およびオンチップ・メモリ構成の選択肢を挙げる．その後，本稿で対象とする性能評価対象モデルを示し，その性能式を構築する．

2.1 CMP におけるオンチップ・ネットワーク構成およびオンチップ・メモリ構成の選択肢

CMP の主な構成要素として，プロセッサコア，オンチップ・ネットワーク，オンチップ・メモリがある．本節では，これらのうちオンチップ・ネットワークおよびオンチップ・メモリに着目し，構成方式の違いにより分類する．

2.1.1 オンチップ・ネットワーク構成

オンチップ・ネットワークとはプロセッサコアとオンチップ・メモリを相互に接続する通信経路網である．CMP における代表的なオンチップ・ネットワークの構成に，共有バス型とクロスバースイッチ型がある．

バス型ではある時刻においてアクセス権限を取得できるプロセッサコアが 1 つに限られるので，チップ上で通信が集中する場合に競合が起きやすい．そのため多くのプロセッサコアが接続される場合，バスを多重化する事がある．一方，クロスバースイッチは伝送速度が高速であるという側面を持つが，コストが高いため，多くのプロセッサコアを接続するのに向かない側面もある．

2.1.2 オンチップ・メモリ構成

CMP における代表的なオンチップ・メモリ構成として，キャッシュ共有型，キャッシュ非共有型，スクラッチパッド型の 3 つがある．

まず，キャッシュ共有型は，複数のコアが 1 つのオンチップ・キャッシュに接続された形態である [2]．搭載されているオンチップ・キャッシュ容量を M とおくと，各コアが利用できるオンチップ・キャッシュ容量の最大値は M である．そのため，他方のプロセッサコアでスレッドが実行されていない場合など，オンチップ・キャッシュ容量を効果的に利用できる利点がある．しかしながら，キャッシュメモリとプロセッサコアとを接続するオンチップ・ネットワークの競合，キャッシュ領域の競合が発生する可能性は高まる．

次に，キャッシュ非共有型は，各プロセッサコアが占有のオンチップ・キャッシュに接続された形態である [3]．キャッシュ非共有型では，オンチップ・ネットワークの競合やキャッシュ領域の競合は発生しない．しかしながら，各コアが利用できるオンチップ・キャッシュ容量は，同一キャッシュ容量の共有型よりも少ない．例えば，総容量 M のオンチップキャッシュを n 個のプロセッサコアで均等に分割した場合，各プロセッサコアが利用できるオンチップ・キャッシュ容量は M/n となる．また，I/O ピンおよび I/O インターフェースの競合，および，オンチップ・キャッシュとオフチップメモリとを接続するバスの競合により，性能低下を招く可能性もある．

最後に，スクラッチパッド型とは，主記憶に接続されたメモリ階層とは別にプロセッサコア占有の記憶領域を有する形態である [5]．スクラッチパッド型では，オンチップ・キャッシュと違い，記憶領域にデータが存在しない場合でも，ハードウェアはオフチップ・メモリからデータを読込まない．そのため，ソフトウェアの制御により，必要なデータを明示的に読込む必要がある．スクラッチパッド型では，各プロセッサコアが独立した占有記憶領域を持ち，明示的にデータの通信を行うため，オンチップ・メモリの競合やオンチップ・ネットワークの競合をより避けることができる．そのため性能の予測をしやすいという利点がある．

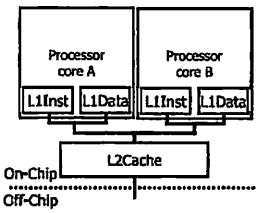

図 1 キャッシュ共有型 CMP モデル

2.2 性能評価対象モデル

本節では，キャッシュ共有/非共有型の 2 つの CMP の性能を評価する．各 CMP のモデル概略図を，それぞれ図 1，図 2 に示す．各モデルの構成は以下の通りである．

- プロセッサコア:
  - 1 チップ上にスーパースカラ型プロセッサコアを 2 個搭載する．以下，それぞれをプロセッサコア A/B と呼ぶ．
- オンチップ・メモリ:
  - オンチップ・メモリ階層は 2 階層からなる．L1 キャッシュ(命令・データ分離型) は，各コア占有 (非共有型) とする．以降，L1 キャッシュ容量は固定とする．L2 キャッシュ(命令・データ統合型) は，キャッシュ共有型および非共有型の 2 種類を想定する．
  - ノンブロッキング・アクセスが可能である．
  - データ書込み方式はライトバック方式である．
  - L2 キャッシュでのデータ追出し時，当該データに対応する L1 キャッシュデータの無効化は行なわない．そのため，同時実行するスレッドの影響により L2 キャッシュでデータの追い出しが発生しても，L1 キャッシュミス率には影響を与えない．
- オンチップ・ネットワーク:
  - 共有バス型を採用しており，プロセッサコア，各オンチップメモリ間を接続する．ただし，アドレスバスとデータバスは分離されている．
  - バスのアクセス権限は，アクセス要求を発生させた順に与える．
- 実行するスレッドの制限:
  - スレッド間でデータの共有はない．
  - 2 つのスレッドが同一時刻に実行を開始する．

2.3 性 能 式

第 2.2 節で示した 2 つの CMP モデル (キャッシュ共有型/非共有型) について，単一スレッドの性能式を構築する．

あるスレッドの実行時間 T を式 (1) で定義する．

$$T = T_{exe} + T_{mem} \quad (1)$$

ここで，

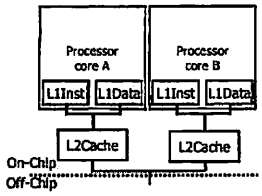

図2 キャッシュ非共有型 CMP モデル

- T_{exe}: メモリ読出しレイテンシが 0 のときの実行時間
- T_{mem}: 実際のメモリ読出しを考慮した際に増加する実行時間

とする．また，式 (1) の T_{mem} を式 (2) で定義する．

$$T_{mem} = AC \times AMAT \quad (2)$$

ここで，

- AC: 総メモリアクセス数
- $AMAT$: 平均メモリアクセス時間

となる．第 2.2 節で示した 2 つの CMP モデルに対し，式 (2) における $AMAT$ をそれぞれ式 (3)，式 (4) で定義する．

$$AMAT_{share} = HT_{L1} + MR_{L1} \times \{C_{L2share} + HT_{L2share} + (P_{L2} + MR_{L2share}) \times (C_{memshare} + MP_{mem})\} \quad (3)$$

$$AMAT_{dedicate} = HT_{L1} + MR_{L1} \times \{C_{L2dedicate} + HT_{L2dedicate} + MR_{L2dedicate} \times (C_{memdedicate} + MP_{mem})\} \quad (4)$$

ここで，各項の定義は以下の通りである．

- $AMAT_{share}$: キャッシュ共有型の平均メモリアクセス時間
- $AMAT_{dedicate}$: キャッシュ非共有型の平均メモリアクセス時間
- HT_{L1}: L1 キャッシュアクセス時間
- MR_{L1}: L1 キャッシュミス率
- $C_{L2share}$: 共有キャッシュにおける L2 キャッシュポートおよびバス競合による平均待ち時間
- $C_{L2dedicate}$: 非共有キャッシュにおける L2 キャッシュポートおよびバス競合による平均待ち時間
- $HT_{L2dedicate}$: 分割サイズでの L2 アクセス時間
- $HT_{L2share}$: 共有キャッシュの L2 アクセス時間
- P_{L2}: L2 キャッシュ領域の競合により増加するキャッシュミス率
- $C_{memshare}$: キャッシュ共有型における主記憶読出し時のバス競合による平均待ち時間
- $C_{memdedicate}$: キャッシュ非共有型における主記憶読出し時のバス競合による平均待ち時間
- $MR_{L2dedicate}$: 非共有キャッシュにおける L2 キャッシュミス率
- $MR_{L2share}$: 共有キャッシュを占有したときの L2 キャッシュミス率
- MP_{mem}: 主記憶読出しに要する時間とデータ転送時間

3. 定性的評価

本節では，キャッシュ共有型/非共有型 CMP のメモリ性能を式 (3)，(4) に基づいて定性的に評価する．ここで，プロセッサコア A ならびに B で実行されるスレッドを，それぞれ便宜上“メインスレッド”ならびに“サブスレッド”と呼ぶ．そして，メインスレッドの実行性能に対し，サブスレッドが与える影響を考察する．メインスレッドの $AMAT$ において，サブスレッドにより影響を受ける項は $C_{L2share}$，P_{L2}，$C_{memshare}$，$C_{memdedicate}$ の 4 つである．

まず，キャッシュ共有型において，サブスレッドにより影響を受ける各項を以下に列挙する．

- $C_{L2share}$: メインスレッドによる L2 キャッシュへのアクセス頻度が高いとき，ならびにサブスレッドによる L2 キャッシュへのアクセス頻度が高いときに大きくなる．
- P_{L2}: 共有キャッシュ領域の競合により増加する．
- $C_{memshare}$: オフチップアクセス頻度に伴って増加する．したがって，増加ミス率 P_{L2} に付随して，$C_{memshare}$ は大きくなる．

次に，キャッシュ非共有型において，サブスレッドにより影響を受ける項は $C_{memshare}$ である．$C_{memshare}$ は，L2 キャッシュ-主記憶間のバス競合による待ち時間であり，オフチップアクセス頻度に伴って増加する．

以上を元に，キャッシュ共有型とキャッシュ非共有型，それぞれにおける $AMAT$ を比較する．一般的に，主記憶はオンチップ・キャッシュに比べてアクセス時間が長い．また，L2 キャッシュ-主記憶間のバス帯域が制限されているため，L2 キャッシュ-主記憶間のバス競合が発生すると，それにより発生する待ち時間はオンチップの共有バスに比べて大きくなる．したがって，$AMAT_{share}$ と $AMAT_{dedicate}$ の差へ支配的な影響を及ぼすのは，キャッシュ共有型における $P_{L2} + MR_{L2share}$ およびキャッシュ非共有型における $MR_{L2dedicate}$ であることが予想される．

キャッシュ共有型において，各スレッドの最大使用可能領域は非共有型の 2 倍である．したがって，メインスレッドを単独実行した場合，容量性ミスが改善されるため $MR_{L2dedicate} > MR_{L2share}$ となる．しかしながら，キャッシュ共有型ではスレッド間で競合性ミスが発生し，増加ミス率 P_{L2} が生じる．以上を踏まえると，$AMAT_{share}$ が $AMAT_{dedicate}$ より短くなるのは，$MR_{L2dedicate} > MR_{L2share} + P_{L2}$ のときである．つまり，キャッシュを共有することによる容量の増加によりミス率が改善され，かつ，その改善ミス率がキャッシュ領域の競合による増加ミス率 P_{L2} を上回る場合となる．逆に，$AMAT_{dedicate}$ が $AMAT_{share}$ より短くなるのは，$MR_{L2dedicate}$ のキャッシュ容量増加による改善を，増加ミス率 P_{L2} が上回る場合である．

4. 定量的評価

本節では，キャッシュ共有型/非共有型 CMP の $AMAT$ を評価する．

4.1 実験の目的

サブスレッドを性質の異なるものに変更することで，メインスレッドの $AMAT$ がどのように変化するかを実験により解析する．

4.2 実験環境

シミュレータとして，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 [1] 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り，OS(Linux,FreeBSD) を含めたプログラムの実行をサポートしている．本実験では，プロセッサコアにどのプログラムが割り当てるかを指定するため，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 に対応している．

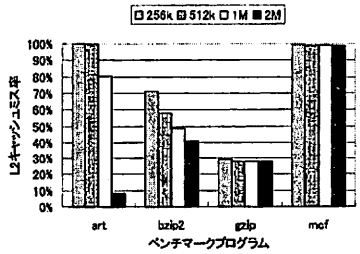

図3 各ベンチマークのキャッシュ容量と L2 ミス率変化

表1 シミュレータ設定

(cc: clock cycle(s))

命令発行方式	アウト・オブ・オーダ
オンチップ動作周波数	3.2GHz
分岐予測器	
種類	bybrid
BTB サイズ	4k エントリ,4 ウェイ
RAS	16
命令発行幅	4 命令/cc
命令デコード幅	4 命令/cc
IQ サイズ	64 エントリ
rob サイズ	192 エントリ
storebuffer サイズ	32 エントリ
IFQ サイズ	32 エントリ
LSQ サイズ	32 エントリ
ソフトウェア・プリフェッチ	squash
整数演算器 (ユニット数，実行，発行レイテンシ)	
ALU	4, 1 cc, 1 cc
Mult, Div	1,(Mult: 3 cc,1 cc) (Div: 20 cc, 19 cc)
浮動小数点演算器 (ユニット数，実行，発行レイテンシ)	
ALU	4, 2 cc, 1 cc
Mult,	2, 4 cc, 1 cc
Div.	12 cc, 12 cc
SQRT	24 cc, 24 cc
オンチップ・キャッシュ構成	
L1 データキャッシュ	32KB (64B/エントリ, 4 ウェイ, 512 エントリ)
L1 命令キャッシュ	32KB (64B/エントリ, 4 ウェイ, 512 エントリ)
L2 キャッシュ(命令・データ統合) 共有型	1MB (64B/エントリ, 8 ウェイ, 4096 エントリ)
L2 キャッシュ(命令・データ統合) 非共有型	512kB (64B/エントリ, 8 ウェイ, 2048 エントリ)
レイテンシ	
L1 キャッシュ	2 cc
L2 キャッシュ(1MB)	14 cc
L2 キャッシュ(512kB)	12 cc
主記憶	250 cc
置換アルゴリズム	LRU
L2-主記憶間バス幅	8B
L2-主記憶間バス周波数	800MHz
オンチップ共有バス幅	64B

また，実験で用いるベンチマークプログラムとして SPEC2000 CPU ベンチマークセット [12] を用いた．ここで，キャッシュの共有/非共有による利用可能な最大容量の違いで，L2 キャッシュミス率改善が有るものと無いものを含んでいることが適切である．そこで，L2 キャッシュ容量を変化させて L2 キャッシュミス率がどのように変わるか調査し (図 3)，特性の異なるものを実験対象として選択した．art および bzip2 ではキャッシュ容量の増加に伴いミス率が改善されるが，gzip および mcf ではキャッシュ容量の変化に対してミス率の変化が見られない．なお，入力データには reference を用いた．シミュレータの主な設定は，表 1 の通りである．メインスレッドで 1 つのベンチマークプログラムを実行し，サブスレッドで実行するベンチマークプログラムを変化させて実験を行う．メインスレッドで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．

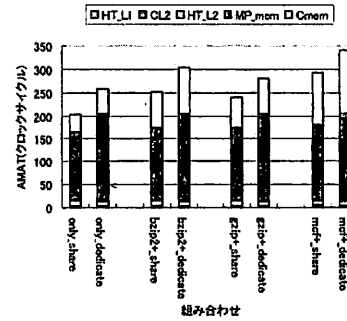

図4 art の $AMAT$ 変化

4.3 実験結果

図 4，図 5，図 6，図 7 はそれぞれ art，bzip2，gzip，mcf をメインスレッドとし，サブスレッドを変更したときの各ベンチマークプログラムの $AMAT_{share}$ もしくは $AMAT_{dedicate}$ を表している．グラフの縦軸は $AMAT$ であり，横軸はプログラム実行条件を示す．横軸の添え字 share ならびに dedicate は，それぞれ，共有キャッシュ型ならびに非共有キャッシュ型における実行を表しており，ベンチマークプログラム名はサブスレッドを表す．なお，only はメインスレッド単独実行時の結果である．凡例にある HT_L1，CL2，HT_L2，MP_mem，Cmem はそれぞれ，第 2.3 節で定義した HT_{L1}，C_{L2}，HT_{L2}，MP_{mem}，C_{mem} に対応しており，$AMAT$ の内訳を理解するため，以下の計算式により算出した．キャッシュ共有型では，

- CL2= $C_{L2share} \times MR_{L1}$
- HT_L2= $HT_{L2share} \times MR_{L1}$
- MP_mem= $MP_{mem} \times MR_{L1} \times (MR_{L2share} + P_{L2})$
- Cmem= $C_{memshare} \times MR_{L1} \times (MR_{L2share} + P_{L2})$

となる．また，キャッシュ非共有型では，

- CL2= $C_{L2dedicate} \times MR_{L1}$
- HT_L2= $HT_{L2dedicate} \times MR_{L1}$
- MP_mem= $MP_{mem} \times MR_{L1} \times MR_{L2dedicate}$
- Cmem= $C_{memdedicate} \times MR_{L1} \times MR_{L2dedicate}$

となる．なお，MP_{mem} に含まれるデータ転送時間は，キャッシュブロックサイズ，バス幅，バス周波数，チップ動作周波数から算出した．

全体を通じて，キャッシュ共有型/非共有型における $AMAT$ の差は C_{L2} の影響は小さく，主に L2 キャッシュミスペナルティにより生じている．

$AMAT_{share}$ と $AMAT_{dedicate}$ の優劣にみられる特徴はメインスレッドによって異なる．まず，art では，どの組み合わせにおいても，$AMAT_{share}$ が $AMAT_{dedicate}$ より短い．次に，bzip2 は，art との組み合わせにおいてのみ，$AMAT_{share}$ が長く，その他 2 つの組み合わせでは，$AMAT_{dedicate}$ が長い．また，gzip では，全ての組み合わせにおいて，$AMAT_{share}$ が $AMAT_{dedicate}$ より長い．最後に，mcf では，どの組み合わせにおいても共有/非共有の差がほとんどみられない．$AMAT$ の内訳を見ると，その大半を L1 キャッシュのアクセス時間が占めている．

4.4 考　察

L1-L2 キャッシュ間のバス競合が $AMAT$ の差に寄与しなかったのは，L2 キャッシュアクセス頻度に対してバンド幅が十分で

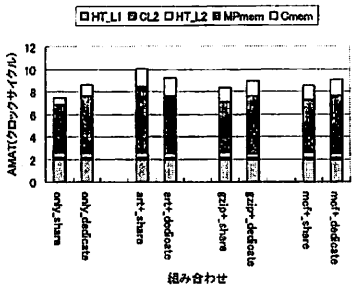

図 5 bzip2 の $AMAT$ 変化

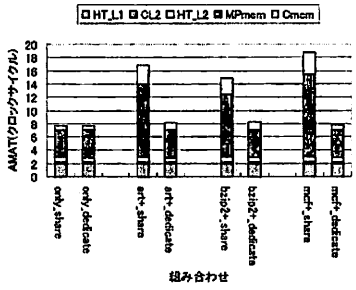

図 6 gzip の $AMAT$ 変化

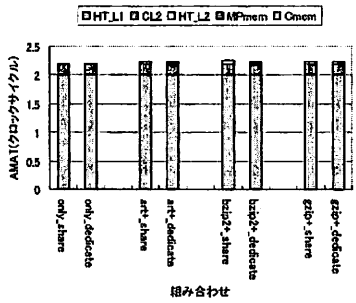

図 7 mcf の $AMAT$ 変化

あったためであると考える．

また，各組み合わせにおける $AMAT_{share}$ と $AMAT_{dedicate}$ の比較結果は以下の 3 つに分かれる．なお，括弧内にサブスレッドを示す．

- $AMAT_{share}$ が短い
  - art(+bizp2, +gzip, +mcf)
  - bzip2(+gzip, +mcf)
- $AMAT_{dedicate}$ が短い
  - bzip2(+art)
  - gzip(+art, +bizp2, +mcf)
- $AMAT_{share}$ と $AMAT_{dedicate}$ に差が無い
  - mcf(+art, +bizp2, +gzip)

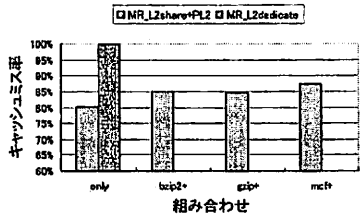

図 8 art の L2 ミス率変化

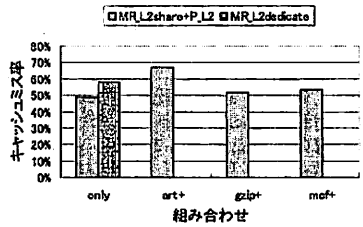

図 9 bzip2 の L2 ミス率変化

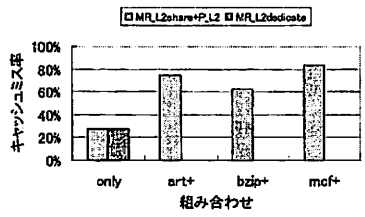

図 10 gzip の L2 ミス率変化

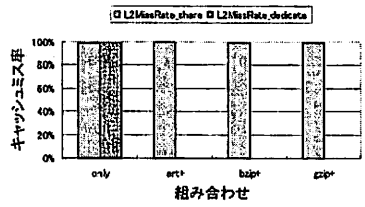

図 11 mcf の L2 ミス率変化

これらの $AMAT_{share}$ と $AMAT_{dedicate}$ との優劣は，$MR_{L2dedicate}$ と $MR_{L2share} + P_{L2}$ の大小により説明できる．図 8，図 9，図 10，図 11 は，それぞれ art，bzip2，gzip，mcf をメインスレッドとし，サブスレッドを変更したときの L2 ミス率変化を示している．

まず，$AMAT_{share}$ が短くなる art と bzip2 ではキャッシュ容量増加によりミスの改善が見られる．その改善ミス率がキャッシュ領域の競合により発生する P_{L2} を上回ったとき，$AMAT_{share}$ が短くなっている．

また，$AMAT_{dedicate}$ が短くなるのは，P_{L2} が改善ミス率を上回るときであり，その原因として以下の 2 つの場合がある．1) キャッシュ容量増加によりミス率が改善されるが，それを P_{L2}

表 2 ベンチマークプログラムの L1 キャッシュミス率

	art	bzip2	gzip	mcf
L1 キャッシュミス率	66.95%	3.09%	5.32%	0.06%

が上回っている．2) キャッシュ容量増加によりミス率が改善されず，P_{L2} だけが生じている．bzip2 では，art との組み合わせにおいて，1) に該当し，gzip は単独で実行した場合，キャッシュ容量増加によるミスの改善がみられず，2) に該当する．

さらに，$AMAT_{share}$ と $AMAT_{dedicate}$ に差が無い mcf では，キャッシュ容量増加によるミスの改善がみられない．ただし，単独で実行した際，ほとんどのアクセスが L2 キャッシュミスとなっており，キャッシュ共有型において P_{L2} が増加する余地がない．そのため，全ての組み合わせにおいて，共有型/非共有型でキャッシュミス率に差が無く，$AMAT_{dedicate}$ と $AMAT_{share}$ に差が生じない．

メインスレッドの違いによって，$AMAT$ の長さに顕著な違いが見られる．例えば，art では最短でも 200 クロックサイクル要するのに対して，mcf では 2.5 クロックサイクル以内である．この差の原因は，式 (3)，(4) における MR_{L1} であると考える．表 2 は各ベンチマークプログラムの L1 キャッシュミス率である．$AMAT$ が大きな art では，L1 キャッシュミス率が高く，総アクセスに占める L2 キャッシュアクセスの割合が大きい．そのため，高い L2 キャッシュミス率のとき，主記憶アクセスのペナルティの影響を大きく受け，$AMAT$ は長くなる．また，mcf では L1 キャッシュミス率が極めて小さい．そのため，総アクセスに占める L2 キャッシュアクセスの割合が小さく，高い L2 キャッシュミス率に関わらず，$AMAT$ のほとんどを L1 ヒット時間が占める結果になっている．

5. 関連研究

キャッシュ共有型 CMP においてキャッシュ領域の競合を抑制する手法として，動的なキャッシュ領域の分割手法が挙がる [6][7][8]．参考文献 [6] では，共有キャッシュにおいて増加するキャッシュミス (率) をスレッド間で均一となるように，キャッシュ領域を割り当てる．参考文献 [7] では，同時実行のスレッドをデータの再利用性で特徴付け，再利用性の高いスレッドにより多くの領域を割り当てる．具体的には，キャッシュから追い出されたデータに対してのアクセス回数を追跡し，キャッシュ上に残っていた場合の性能の利得でスレッドの重み付けを行う．参考文献 [8] では，マージナルゲインカウンタ (スタックディスタンスプロファイル [9] と同一のもの) を元に，CMP 全体でキャッシュミスが最小となるように動的にキャッシュ領域を割り当てる．スタックディスタンスプロファイルは，キャッシュにデータが読込まれたとき，もしくは，最後にアクセスされたときから再度アクセスするまでの間に，同一セット内の幾つの異なるブロックにアクセスしたかによって，キャッシュヒットをインデックス付けする．少数の異なるブロックにしかアクセスしないスレッドは，参照の局所性が高くヒットしやすい．そのため，参考文献 [8] では優先してブロックを割り当てる．

参考文献 [10] は，特定のキャッシュセットに集中してアクセスするスレッドの組み合わせを検出し，OS が実行スレッドを変更することでキャッシュミスを削減する手法を提案している．

参考文献 [6][7][8][10] では，L2 キャッシュミスを削減する手法の提案を行なっており，メモリアクセス時間の詳細な解析を行なっていない．

参考文献 [11] では，スタックディスタンスプロファイルにアクセス回数の情報を付加し詳細化した，サーキュラーシーケンスプロファイルを入力とするキャッシュミス解析モデルを構築している．解析モデルの入力として，各スレッドの単独実行時のサーキュラーシーケンスプロファイルを与えると，出力として，スレッドを共有キャッシュ型 CMP で同時実行したときの L2 キャッシュミス率が得られる．しかしながら，参考文献 [11] では，L2 キャッシュミス率のみの解析を行なっており，L1 キャッシュ，オフチップ・メモリを含めた $AMAT$ の解析は行なっていない．

6. まとめ

本稿では，CMP における設計選択肢である共有/非共有キャッシュ型に着目し，対象のモデル化，および定性的，定量的評価を行った．単一のスレッドにおける平均メモリアクセス時間について解析を行った結果，共有キャッシュ型の利用可能容量増加による利得を受け，他のスレッドによる競合性キャッシュミスの悪影響が利得を下回る場合，メモリ性能が高まることが明らかになった．

謝辞

本論文をまとめるにあたり，多大なるご協力を頂いた九州大学大学院システム情報科学府の三輪英樹氏に深く感謝します．また，共にご議論頂いた九州大学システム LSI 研究センターならびに安浦・村上・松永・井上研究室の皆様に感謝します．なお，本研究は一部，科学研究費補助金 (学術創成研究費:課題番号 14GS0218，若手研究 A:課題番号 17680005)，ならびに，松下電器産業株式会社との協同研究による．

文献

[1] Nathan L. Binkert, Erik G. Hallnor, and Steven K. Reinhardt. "Network-Oriented Full-System Simulation using M5", Proceedings of the Sixth Workshop on Computer Architecture Evaluation using Commercial Workloads (CAECW), Feb 2003.

[2] IBM,"Advanced POWER Virtualization on IBMeserver p5 Servers: Architecture and Performance Considerations",November 2005.

[3] Keltcher,C.N. McGrath,K.J. Ahmed,A. Conway,P,"The AMD Opteron processor for multiprocessor servers," Micro,IEEE,23(2),pp.66-76, March-April 2003.

[4] Intel,"Dual-Core Update to the Intel Itanium 2 Processor Reference Manual For Software Development and Optimization", January 2006.

[5] D.Pham .et al,"Design and Implemantation of a First Generation CELL Processor",ISSCC 2005.

[6] Seongbeom Kim, Dhruba Chandra, Yan Solihin, " Fair Cache Sharing and Partitioning in a Chip Multiprocessor Architecture " , 2004 PACT

[7] Alex Settle, Dan Connors, Enric Gibert, Antonio Gonzalez , " A Dynamically Reconfigurable Cache for Multithreaded Processors",2005 Journal of Embedded Computing

[8] G. Edward Suh ,Srinivas Devadas, Larry Rudolph, " A New Memory Monitoring Scheme for Memory-Aware Scheduling and Partitioning " , 2002HPCA

[9] R.L.Mattoson, J.Gecsei, D.Slutz, I.Traiger,"Evaluation Techniques for Storage Hierarchies",IBM Systems Journal,9(2),1970.

[10] Joshua Kihm, Alex Settle, Andrew Janiszewski, Dan Connors, "Understanding the Impact of Inter-Thread Cache Interference on ILP in Modern SMT Processors ",2005 Journal of Instruction-Level Parallelism

[11] Dhruba Chandra, Fei Guo, Seongbeom Kim, Yan Solihin, "Predicting Inter-Thread Cache Contention on a Chip Multi-Processor Architecture" ,2005 HPCA

[12] SPEC -Standard Performance Evaluation Corporation, http://www.spec.org/

[13] John.L.Hennessy, David.A.Patterson, "Computer Architecture:A Quantitative Approach", Morgan Kaufmann Publishers,Inc.,3rd edition, 2002.

[14] SimpleScalar LLC,http://simplescalar.com/