Vol.2011-ARC-193 No.9
2011/1/21

SRAM/DRAM ハイブリッド・キャッシュにおける実行時動作モード決定法の提案

橋口慎哉[†1]　福本尚人[†1]
井上弘士[†2]　村上和彰[†2]

3 次元積層された大容量 DRAM をキャッシュとして利用する従来の DRAM スタック法では，アクセス時間の増加により性能が低下する場合があるという問題点があった．そこで我々は,「高速・小容量キャッシュモード」と「低速・大容量キャッシュモード」の 2 つの動作モードを，アプリケーションの特性に応じて使い分ける SRAM/DRAM ハイブリッド・キャッシュ・アーキテクチャを提案した[1)]．しかしながら，文献[1)] では動作モードの決定法に関しては議論はしていなかった．そこで本稿では，本方式におけるプログラム実行時の動作モード決定法を提案する．プログラム実行時に得た両動作モードの L2 キャシュミス率と，性能評価モデルに基づき，より高い性能となる動作モードを判定し、将来の動作モードを決定する．これにより，アプリケーションの特性の違いに応じた動作モード決定が可能となる．ベンチマーク・プログラムを用いた定量的評価を行った結果，最大で 3.01 倍，平均で 1.17 倍の性能向上を確認した．

Run-time Operation-Mode Management on SRAM/DRAM Hybrid Cache

SHINYA HASHIGUCHI,[†1] NAOTO FUKUMOTO,[†1]
KOJI INOUE[†2] and KAZUAKI MURAKAMI[†2]

3D stacked DRAM caches can dramatically reduce off-chip memory accesses. However, this approach degrades performance in some cases because increasing cache size makes access time longer. To solve this issue, we propose dynamically controlled SRAM/DRAM Hybrid Cache. Hybrid Cache supports two operation modes: fast, small SRAM cache mode, and slow, larage DRAM cache mode. The cache attempts to select an appropriate mode based on memory access behavior at run time. This paper proposes an algorithm for the mode selection. Our evaluation results show that we can achieve speed-up 3.01X in maximum and 1.17X in average.

1. はじめに

新しい半導体チップの実現法として 3 次元実装が注目されている．3 次元方向へ回路を集積することで，配線長を維持しつつ，回路を大規模化できるといった利点がある．また，たとえば DRAM とロジックのように異なる製造プロセスを経て作成した複数のダイを積層する事も比較的容易になる．このような背景の中，3 次元実装デバイスを前提としたプロセッサ構成法に関する研究が行われるようになってきた．その中でも特に，大容量な DRAM とプロセッサを積層し，これらの間を TSV（Through Silicon Via）で接続するアプローチが大きな注目を集めている[2)3)]．

これまでに我々は，3 次元実装デバイスを前提としたメモリ構成法として SRAM/DRAM ハイブリッド・キャッシュ・アーキテクチャ（以降，ハイブリッド・キャッシュと略す）を提案した[1)]．従来の単純な DRAM キャッシュ積層とは異なり,「高速かつ低容量な SRAM キャッシュモード」と「低速かつ大容量な DRAM キャッシュモード」を選択可能にする．これにより，高いメモリ性能を実現することができる．本方式においては，アプリケーションの特性に応じて適切な動作モードを選択する必要がある．しかしながら，文献[1)] でハイブリッド・キャッシュの潜在能力を明らかにする事を目的とするため，適切な動作モードは既知であると仮定していた．そのため，動作モード決定法の詳細は未だ議論されていない．

そこで本稿では，ハイブリッド・キャッシュにおける動的動作モード決定アルゴリズムを提案する．また，ベンチマーク・プログラムを用いた定量的評価を行い，その有効性を示す．本方式では，プログラム実行中に両動作モードでの L2 キャッシュミス率を測定または推定する．そして，メモリ性能モデルに基づきより高い性能となる動作モードを判定し，その結果を用いて将来の動作モードを決定する．これにより，アプリケーション特性の違いや変化（メモリ参照の振舞いなど），ならびに，ハードウェア特性の違い（SRAM や積層 DRAM のアクセス時間など）に応じた動的な動作モードの決定が可能となる．

以下，第 2 節ではハイブリッド・キャッシュの概要と，実行時での動作モード決定の必要性を述べる．次に，第 3 節では，動作モード決定法を提案し，その詳細を説明する．第 4 節ではベンチマーク・プログラムによる定量的評価を行い，最後に第 5 節で簡単にまとめる．

†1 九州大学大学院システム情報科学府
Graduate School of Information Science and Electrical Engineering, Kyushu University
†2 九州大学大学院システム情報科学研究院
Faculty of Information Science and Electrical Engineering, Kyushu University

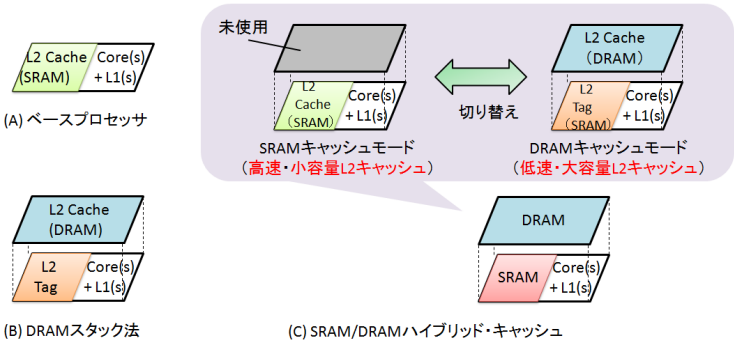


図 1 ベースプロセッサ／ DRAM スタック法／ SRAM/DRAM ハイブリッド・キャッシュ

2. SRAM/DRAM ハイブリッド・キャッシュの概要

近年，メモリウォール問題の解決を目的として，大容量 DRAM を LLC（Last Level Cache）として積層するアプローチが注目を集めている．従来の 2 次元実装プロセッサ（以降，ベースプロセッサと呼ぶ），ならびに，文献[2] で提案された DRAM スタック法の構成を図 1(A) と (B) に示す（本稿では LLC は L2 キャッシュと想定）．DRAM スタック法では，積層した DRAM を L2 キャッシュのデータ領域として使用し，タグ情報は下層の SRAM に格納される．このように大容量 DRAM を積層することで容量性ミスの大幅な削減を期待できる一方，アクセス時間が増加するといった問題が生じる．そのため，アクセス時間の増加を隠蔽できる程のキャッシュミス率の削減を達成できなければ，むしろ 3 次元積層により性能が低下する．

この問題を解決し，3 次元積層デバイスの潜在能力を高める方式として図 1 の (C) に示すハイブリッド・キャッシュを提案している[1]．本方式では以下 2 つの動作モードを有する．

- **SRAM キャッシュモード：**
  下層の SRAM は通常の L2 キャッシュメモリとして動作する．このとき，上層 DRAM は使用されない．つまり，L2 キャッシュは高速・小容量となる．
- **DRAM キャッシュモード：**
  上層 DRAM は L2 キャッシュのデータ記憶領域，下層 SRAM はタグ領域として使用される．L1 キャッシュ・ミスが発生した際，当該タグメモリを参照すると同時に，積層された大容量 DRAM のアクセスを開始する．つまり，L2 キャッシュは低速・大容量となる．

実行対象プログラムのワーキングセット・サイズが小さい場合には，ハイブリッド・キャッシュは通常の L2 キャッシュメモリとして動作し，上層 DRAM へのアクセスを禁止する．一方，より大きな L2 キャッシュが必要だと判断した場合には，上層 DRAM を L2 キャッシュとして使用するためにタグ RAM として動作する．このように，選択的に大容量 DRAM を活用することで，高速アクセスとキャッシュミス率改善の両立を可能にし，3 次元積層 DRAM の潜在能力を最大限に引き出す．

3. 動作モード決定法

3.1 設計選択肢

ハイブリッド・キャッシュでは，如何に適切な動作モードを選択できるかが極めて重要になる．そして，この適切な動作モードは，プログラム実行におけるメモリ参照の振舞い，ならびに，SRAM と積層 DRAM のハードウェア特性に大きく依存する．ハイブリッド・キャッシュでの動作モード決定においては，動作モード決定時期 (いつ動作モードを決定するか?)，ならびに，動作モード設定時期 (いつ動作モードを設定するか?) に関して，以下の選択肢が考えられる．

- **静的決定・静的切替え：**プログラム実行前にメモリ参照の振舞いを解析して動作モードを決定する．そして，実行開始時に 1 度だけ動作モードを設定する．実行プロファイルの事前取得が可能な場合や，ソースコード解析によりメモリ参照の振舞いを精度良く解析できる場合には有効である．ただし，プログラム実行中におけるメモリ参照の振舞いの変化には追従することができない．
- **静的決定・動的切替え：**プログラム実行前にメモリ参照の振舞いを解析して動作モードを決定する．そして，適宜，実行中に動作モードを設定する．上述した静的決定・静的切替えと同様，プロファイル取得やソースコード解析が必要となる．また，動作モードの再設定に関しては，例えばコンパイラ・サポートによる専用命令の実行などが挙げられる．静的決定・静的切替えとは異なり，入力の違い等によりメモリ参照の振舞いが変化する場合にも対応できる．一方，動作モードの再設定に伴うオーバヘッドが発生する．
- **動的決定・動的切替え：**プログラム実行中に将来の動作モードを決定し，それに基づきプログラム実行中に動作モードを再設定する．プログラムの事前解析やプロファイル取得は必要なく，バイナリの互換性を完全に保つことができる．上述した静的決定・動的

切替えと同様に，メモリ参照の振舞いの変化に追従できる一方，動作モードの再設定に伴うオーバヘッドが問題となる．

本稿では，サーバやデスクトップ PC といった汎用システムでの応用を前提とする．この場合，ソースコードが入手できない等の理由により再コンパイルできない状況が多く存在する．そのため，実行コードの互換性は極めて重要となる．また，組込みシステムとは異なり，実行アプリケーションと入力，ならびに，実行時の振舞いを事前に解析することは難しい．これらの理由により，本稿では動的決定・動的切替えのアプローチを採る．なお，組込みシステムのように事前解析が可能な場合には静的決定に基づくアプローチが有効であるため，静的決定・静的切替え，ならびに，静的決定・動的切替えアプローチに関しては今後検討を進める予定である．

3.2 動作モード切替えにおける性能オーバヘッド

ハイブリッド・キャッシュにおいて，プログラム実行中の動作モード切替えは以下の手順で行われる．

(1) L2 キャッシュへのアクセスを禁止する．プロセッサ側から L2 キャッシュへのアクセス要求があれば，動作モード切替えが終了するまでプロセッサはストールする．

(2) 現在使用している L2 キャッシュ（つまり，動作モード切替え前に使用している L2 キャッシュ）をフラッシュする．動作モードの切替えにより L2 キャッシュのデータ/タグ情報の記憶領域が変更されるために必要となる．

(3) データ/タグ情報記憶領域へのアクセスパスを変更する（詳細は文献[1] を参照）．これにより動作モードの切替えが完了する．

(4) L2 キャッシュへのアクセスを再開する．

動的に動作モードを切り替えることで大きな性能向上が期待できる一方で，上述したようにキャッシュをフラッシュする必要があるため，性能向上阻害要因として以下の 2 つの問題が生じる．

- **ライトバックに伴う性能オーバヘッド：**動作モード切替え前の L2 キャッシュ内のダーティ・ラインを全て主記憶へ書戻す必要がある．そのため，メモリバンド幅を圧迫し性能低下が発生する可能性がある．
- **初期参照ミスの増加：**動作モード切替え直後，L2 キャッシュは空の状態である．そのため，見かけ上の初期参照ミス (動作モード切替えが原因で新たに発生する初期参照ミス) が増加する．

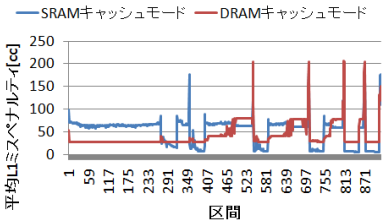


図 2 プログラム実行中の L1 ミスペナルティの変化：mcf

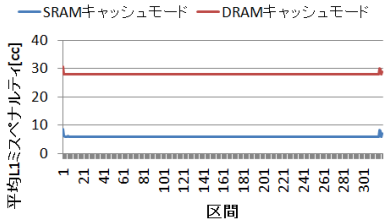


図 3 プログラム実行中の L1 ミスペナルティの変化：twolf

3.3 適切な動作モードの変化に関する解析

ハイブリッド・キャッシュでは，小容量かつ高速な SRAM キャッシュ・モードと，大容量かつ低速な DRAM キャッシュ・モードを選択できる．より高い性能を実現できる動作モード (以降，適切な動作モードと呼ぶ) を決定可能なアルゴリズムを考案するために，アプリケーション・プログラム実行に基づく定量的な解析を行った．図 2 ならびに図 3 は，SPEC CPU 2000 ベンチマークの *mcf* と *twolf* において，ハイブリッド・キャッシュの動作モードを固定して実行した場合の L1 ミスペナルティ(＝ L2 ヒット時間＋ L2 ミス率×主記憶アクセス時間) を示す．横軸は L2 アクセス 100 万回を 1 区間とした実行順の区間番号であり，縦軸は各区間における平均 L1 ミスペナルティである．

これらの図より，プログラム実行中，高性能な動作モードが切替わる頻度は低く，複数の区間で適切な動作モードが連続する傾向にあることが分かる．例えば *mcf* において，連続する 2 つの区間の適切な動作モードが同じである確率は約 0.97 と極めて高い．また，*twolf* においては実行を通して適切な動作モードが変化していない．

3.4 動的動作モード決定アルゴリズム

プログラム実行中の動作モード決定では，1) 適切な動作モードを如何に選択できるか，ならびに，2) 動作モード切替えオーバーヘッドを如何に低減するか，が重要となる．本節では，第 3.3 節で示した解析結果に基づき，ハイブリッド・キャッシュ向けの動的動作モード決定アルゴリズムを提案する．本方式では，プログラム実行を一定回数の L2 キャッシュアクセスで分割し (これを区間と呼ぶ)，各区間毎に適切な動作モードを決定する．具体的には，「**連続する 2 区間では適切な動作モードが同じになる確率が高い**」と想定する．そして，ある区間 N において，次区間 N+1 の動作モードを以下の判定式に基づき決定する．なお，

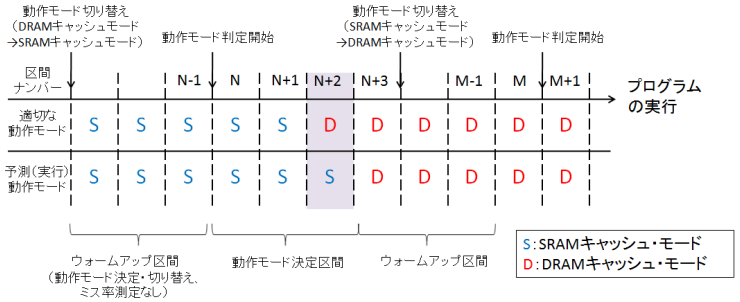


図 4 動作モード決定法の提案方式

1 区間の長さを 500 万 L2 アクセスとした．この理由については第 4.2 節で述べる．

$$if\ MR_{L2\_SRAM} - MR_{L2\_DRAM} > \frac{HT_{L2\_DRAM} - HT_{L2\_SRAM}}{MMAT}$$
$$then\ DramMode, else\ SramMode$$

- $HT_{L2\_SRAM}/HT_{L2\_DRAM}$：SRAM/DRAM キャッシュへのアクセス時間 [cycles]
- $MR_{L2\_SRAM}/MR_{L2\_DRAM}$：測定区間における SRAM/DRAM キャッシュ・モードでのミス率
- $MMAT$：主記憶アクセス時間 [cycles]

ここで，HT_{L2_DRAM}，HT_{L2_SRAM}，ならびに，$MMAT$ はハードウェア特性にのみに依存しており，チップ製造後には基本的に既知と考えることができる．一方，MR_{L2_SRAM} および MR_{L2_DRAM} はプログラム特性に依存するため，実行時にしか得ることができない．また，これらの値は各区間の実行において同時に得る必要がある．現在実行中の動作モードでの L2 ミス率に関しては単純なハードウェア・カウンタを搭載すれば良い．これに対し，現在の動作モードとは異なる動作モードの L2 ミス率を直接実測することは困難である．そこで，特別なハードウェア・サポートを用いたエミュレーションをサポートする．

一般に，キャッシュのミス率はタグ情報のみから得ることができる．そこで，DRAM キャッシュ・モード時には，下層 SRAM のタグメモリが未使用になることを利用し，ここに SRAM キャッシュ・モードを想定してタグ情報を格納する．これを用いて DRAM キャッシュ・アクセス時にも SRAM キャッシュでのヒット/ミス情報のみを取得し，SRAM キャッシュ・モードでの L2 ミス率を得る．同様に，SRAM キャッシュ・モード実行においても，DRAM キャッシュに格納可能なライン数分のタグ情報を保存するメモリを搭載すれば，DRAM キャッシュ・モードでの L2 ミス率を測定することができる．しかしながら，この場合には極めて大きな面積オーバヘッドが発生する．例えば，32MB の DRAM キャッシュにおいてラインサイズを 64B と仮定するとタグ領域は 2.5MB 程度必要となる．この問題を解決するため，幾つかのセットに対応する少数のタグ情報のみを保存するアプローチを採用する[4]．なお，これら L2 ミス率を測定もしくは推定するために必要となるハードウェア・サポートの詳細は現在検討中である．

3.5 動作モード切替え例

具体的な動作例を図 4 を用いて説明する．ここでは，区間 N からの実行に焦点を当て，初期動作モードは SRAM キャッシュ・モードであると仮定する．なお，図中の「適切な動作モード」とは，より高いメモリ性能を実現できる動作モードのことである．また，「予測動作モード」とは第 3.2 節で説明した提案アルゴリズムに基づき決定した動作モードを表す．

(1) 区間 N：予測動作モードは SRAM キャッシュ・モードのため，積層された DRAM は活性化されない．本区間の実行中，SRAM キャッシュ・アクセスにおける L2 ミス率 (つまり，MR_{L2_SRAM})，ならびに，DRAM キャッシュとして動作した場合の推定 L2 ミス率 (つまり，MR_{L2_DRAM}) を求める．また，本区間の実行の最後において，第 3.4 節で示した条件式に基づき区間 N+1 での動作モードを決定する．ここでは，条件式が不成立であった (SRAM キャッシュ・モードが選択された) と仮定する．

(2) 区間 N+1：予測動作モードは SRAM キャッシュ・モードであり，区間 N と同様に動作する．本区間の最後においてモード選択のための条件判定を行い，区間 N+2 での動作モードを決定する．ここでは，SRAM キャッシュ・モードが選択されたとする．

(3) 区間 N+2：適切な動作モードは DRAM キャッシュ・モードであるにも関わらず，予測動作モードは SRAM キャッシュ・モードを選択している．そのため，高いメモリ性能を実現できない．前の区間と同様に両動作モードにおける L2 ミス率を測定/推定し，本区間の最後において区間 N+3 の動作モードを決定する．ここでは，適切な動作モードは DRAM キャッシュ・モードであるため，次区間での適切な動作モードは DRAM キャッシュ・モードと予測される．また，動作モード切替えが発生するため，SRAM キャッシュの内容をフラッシュする．

(4) 区間 N+3：DRAM キャッシュ・モードで動作する．動作モード切替え直後に頻繁に初期参照ミスが生じた場合には，次区間の正しい動作モード選択が行えない可能性がある．そこで，ある一定区間は動作モードの変更を禁止する．以降，区間 M-1 まで同様．

(5) 区間 M：ウォームアップ期間が終了したため，再び L2 ミス率の測定/推定を再開し，区間 M+1 の動作モードを決定する．

4. 評価実験

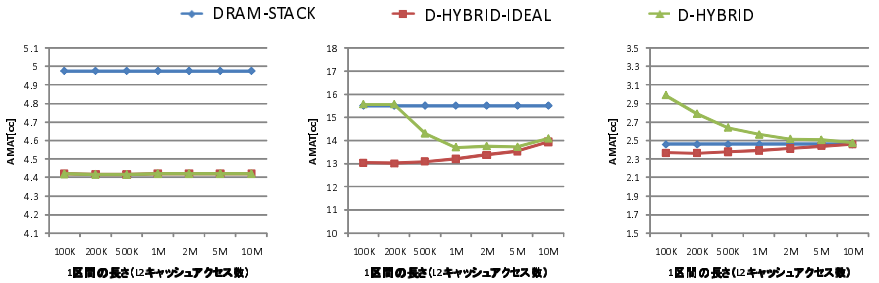
4.1 評価環境

本評価では，ミシガン大学で開発された M5 シミュレータ[5] を用いてトレースを取得し，メモリ性能値を算出した．評価指標は平均メモリアクセス時間 AMAT(=L1 ヒット時間＋ L2 ミス率×ミスペナルティ) である．インオーダ命令発行のシングルコア・プロセッサを想定し，L1 命令/データキャッシュは，それぞれ，容量 32KB，連想度 2，ラインサイズ 64B, アクセス時間は 2 クロックサイクルとした．また，主記憶アクセス時間は 181 クロックサイクルとした．これら，主記憶ならびにキャッシュ・アクセス時間に関しては関連研究を参考に決定した[2)6)7)]．ベンチマークプログラムは，SPEC-CPU2000[8]（入力は train）ならびに Splash2[9]（入力は表**??** *Cholesky*：*tk29.0*，*FFT*：が *4M data points*，*LU*：が *1024* × *1024 matrix*，*Barnes*：が *32K particles*，*FMM*：*64K particles*，*Ocean*：*258* × *258 ocean*，*WaterSpatial*：*4096 molecules*）から *18* 個選択した．評価対象モデルは以下の通りである．

- **DRAM-STACK**：大容量 *DRAM* をスタックする従来の *DRAM* スタック法 *(*図 *1(B)*．*L2* サイズは *32MB*，ブロックサイズは *64B*，連想度は *8*，*L2* アクセス時間は *28* クロックサイクルとする．
- **D-HYBRID**：ハイブリッド・キャッシュ*(*図 *1(C))*．第 *3.3* 節で示したアルゴリズムに基づき動的に動作モードを決定する提案型キャッシュ．*DRAM* の *L2* サイズは *32MB*，ブロックサイズは *64B*，連想度は *8*，アクセス時間は *28* クロックサイクルである．*SRAM* の *L2* サイズは *2MB*，ブロックサイズは *64B*，連想度は *8*，アクセス時間は *6* クロックサイクルとする．また，動作モード切替え後には，切替え先の *L2* キャッシュに格納されるラインと同数の後続メモリアクセスが *L2* ミスを発生すると仮定する *(*つまり，*SRAM* キャッシュ・モードに切替わる場合は *32K* アクセス，*DRAM* キャッシュ・モードに切替わる場合は *512K* アクセスがミスすると仮定する*)*．
- **D-HYBRID-IDEAL**：理想ハイブリッド・キャッシュ*(*図 *1(C))*．*D-HYBRID* において，各区間において常に適切な動作モードを選択でき，かつ，動作モード切替えに伴うオーバヘッドは発生しないと仮定．なお，*1* 区間は *L2* キャッシュ・アクセス *10* 万回とする．

4.2 1 区間の長さが性能へ与える影響

第 *3.4* 節で説明したように，提案する動作モード決定アルゴリズムでは実行のある区間毎に動作モードを決定する．この区間の長さは，メモリ参照の振舞いの変化に対する追従性，ならびに，モード切替えオーバヘッドの度合いに大きな影響を及ぼす．そこで，本節では適切な区間の長さを決定する．*3* つの代表的なプログラムに関して，*1* 区間の長さを変化させた場合の平均メモリアクセス時間 *(AMAT)* を図 *5* に示す．横軸は区間の長さ *(*つまり，*L2* キャッシュアクセス数*)* である．



(a) *FFT*　(b) *mcf*　(c) *bzip2*

図 5　1 区間の長さと性能

一般に，*D-HYBRID-IDEAL* では区間が短いほど高い性能を実現できる．これは，メモリ参照の振舞いの変化に対する追従性が向上するためである．これに対し，*D-HYBRID* では，動作モードの切替えが頻繁に発生しキャッシュミス率が高くなるため，*D-HYBRID-IDEAL* とは逆の傾向となる．このように，追従性の向上による性能向上と切替えオーバヘッドによる性能低下に関するトレードオフが存在し，*1* 区間の長さはこれを決定する重要な設計パラメータとなる．例えば，*mcf* ならびに *bzip2* においては区間が *100K*〜*200K* と短い場合には極めて大きな性能オーバヘッドが発生しており，*5M*〜*10M* といった比較的長い区間に設定することが適切である．一方，*FFT* に関しては，区間の長さに依存せず，*D-HYBRID-IDEAL* と *D-HYBRID* 共に平均メモリアクセス時間がほぼ一定となっている．これは，プログラム実行を通してほぼ全ての区間にて適切なキャッシュモードが同一であったためである．これら以外のプログラムに関しても，その殆どはこれら *3* つの場合に分類することができる．以上の結果を総合し，本評価では *D-HYBRID* における *1* 区間の長さは *5M* アクセスとした．

4.3 評価結果

性能評価結果を図 *6* に示す．横軸はベンチマークプログラム，縦軸は従来の *3* 次元積層法

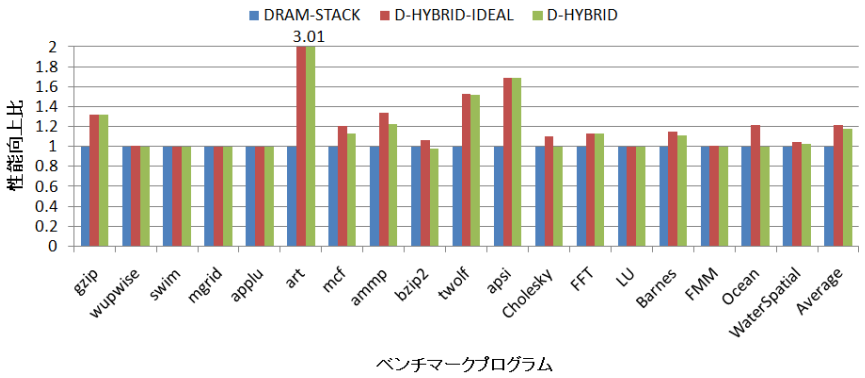


図 6 性能評価結果

である *DRAM-STACK* に対する性能向上比である．*gzip, art, twolf, apsi* といったプログラムにおいて，提案方式により高い性能を実現していることが分かる．特に，*art* では約 *3.3* 倍，*apsi* では *1.6* 倍の性能向上である．これらは理想系である *D-HYBRID-IDEAL* とほぼ同じ性能を達成しており，区間を十分に大きく設定することによる性能オーバヘッドの影響の削減し，かつ，適切な動作モードを正しく予測できた結果と考える．

5. おわりに

本稿では，我々が提案している，*3* 次元積層 *DRAM* を活用する *SRAM/DRAM* ハイブリッド・キャッシュの実行時動作モード決定法について提案を行った．評価実験の結果，本提案手法に基づいた *SRAM/DTAM* ハイブリッド・キャッシュは，従来手法と比較し平均約 *17* %高性能であり，有効であることを確認した．

今後は，本方式を実行可能なシミュレータを開発し，より詳細な評価を行う．また，消費エネルギーについての評価も行う予定である．

謝辞

日頃から御討論頂いております九州大学安浦・村上・松永・井上研究室ならびにシステム *LSI* 研究センターの諸氏に感謝します．本研究は主に九州大学情報基盤研究開発センターの研究用計算機システムを利用しました．なお，本研究は，独立行政法人新エネルギー・産業技術総合開発機構 *(NEDO)* 若手グラントの支援による．

参 考 文 献

1) 橋口慎哉, 小野貴継, 井上弘士, 村上和彰, ”*3* 次元 *DRAM* - プロセッサ積層実装を対象としたオンチップ・メモリ・アーキテクチャの提案と評価,”情報処理学会研究報告, *Vol. 2009-ARC-183, No.16, 2009* 年 *4* 月.
2) *Black, B., Annavaram, M., Brekelbau, N., DeVale, J., Jiang, L., Loh, G. H., McCauley, D., Morrow, P., Nelson, D. W., Pantuso, D., Reed, P., Rupley, J., Shankar, S., Shen, J. and Webb, C.: Die Stacking (3D) Microarchitecture,* MICRO 39: Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture, *IEEE Computer Society, pp.469–479 (2006).*
3) *Loh, G.H.: 3D-Stacked Memory Architectures for Multi-core Processors,* SIGARCH Comput. Archit. News, *Vol.36, No.3, pp.453–464 (2008).*
4) *Qureshi, M.K. and Patt, Y.N.: Utility-Based Cache Partitioning: A Low-Overhead, High-Performance, Runtime Mechanism to Partition Shared Caches,* Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture, *MICRO 39, Washington, DC, USA, IEEE Computer Society, pp.423–432 (2006).*
5) *Binkert, N.L., Dreslinski, R.G., Hsu, L.R., Lim, K.T., Saidi, A.G. and Reinhardt, S.K.: The M5 Simulator: Modeling Networked Systems,* IEEE Micro, *Vol.26, No.4, pp.52–60 (2006).*
6) *Loi, G. L., Agrawal, B., Srivastava, N., Lin, S.-C., Sherwood, T. and Banerjee, K.: A thermally-aware performance analysis of vertically integrated (3-D) processor-memory hierarchy,* DAC ’06: Proceedings of the 43rd annual Design Automation Conference, *New York, NY, USA, ACM, pp.991–996 (2006).*
7) *Thoziyoor, S., Muralimanohar, N., Ahn, J.H. and Jouppi, N.P.: CACTI5.1, Technical report, HP Lab (2008).*
8) *Henning, J. L.: SPEC CPU2000: Measuring CPU Performance in the New Millennium,* Computer, *Vol.33, pp.28–35 (2000).*
9) *Woo, S. C., Ohara, M., Torrie, E., Singh, J. P. and Gupta, A.: The SPLASH-2 Programs: Characterization and Methodological Considerations,* ISCA ’95: Proceedings of the 22nd annual international symposium on Computer architecture, *ACM, pp.24–36 (1995).*