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

2007－ARC－174 (4)
2007／8／1

エントロピーによるメモリアクセス特性の表現とキャッシュ性能

横田隆史† 大津金光† 馬場敬信†

メモリ階層の中で性能・コストの両面から大きな役割を担うキャッシュメモリは，プログラム実行時のメモリアクセスに時間的・空間的な局所性があることが前提となっている．偏りが大きいほど高い性能を期待できる一方で，偏りのないプログラムでは高性能を見込めない．本稿では，メモリアクセスの偏りが分布関数で表せるものとし基礎的な検討を行う．また，メモリアクセスに係わる偏りを数値的に表現するためにエントロピーの概念を導入する．関連するプログラムおよびキャッシュの動作に対してエントロピーを定義し，ヒット率との関係を調べた予備評価の結果を示す．

Entropy Representation of Memory Access Characteristics and Cache Performance

Takashi Yokota,† Kanemitsu Ootsu† and Takanobu Baba†

Cache memories play important roles in modern computer architecture. Performance of a cache memory largely depends on spatial and temporal locality in memory accesses. Highly local accesses result good cache performance. In this paper, we first discuss distribution functions of memory accesses, and then, we introduce an entropy concept in cache mechanisms. We define new entropies based on memory access behavior and cache structure. Preliminary evaluation results of cache performance and entropy values are reported.

1. はじめに

計算機システムでは，なんらかの**「偏り」**を前提として成立している部分が非常に多い[1]．計算機アーキテクチャから基盤ソフトウェア，アプリケーションに至るまで，偏りの存在を前提として性能・コストのバランスがとられている．

その典型的な例が記憶階層である．レジスタに始まり，1次キャッシュからn次キャッシュ，主記憶，二次記憶，更には時としてオフラインストレージまで，壮大な階層が築き上げられている．計算機システムは，対象とする処理に内包されるすべての「偏り」を用いることにより「最適化」が進められてきており，その結果，現在の姿になっている．すなわち，現在の計算機システムの性能や効果は基本的に「偏り」に大きく依存している．

計算機アーキテクチャの分野では，これまで，処理性能に代表される評価指標を用い，様々な制約条件を勘案しながら，評価指標を最大化させるための方式を議論してきた．無論こうした方式の議論に際しても，(時として暗黙のうちに)「偏り」が前提にされている場合がほとんどである．

しかし，我々は，偏りの存在を定性的には理解していても，それが具体的にどのような姿をしているのか，あまり理解していない．往々にして，ベンチマークプログラムにより方式の効果に差が生じるが，その差を正確に説明するに至らない．さらに，「偏り」を数値的に表現しようという試みもあまりなされていない．

これまでに我々は，分岐予測器に着目し，プログラムの実行に伴う条件分岐命令での条件の成否の偏り(ないし規則性)と分岐予測性能について議論してきた[2)~4)]．本稿では，やや視点を変え，メモリ階層，特にキャッシュメモリに着目し，プログラム実行に伴い発生する「偏り」について議論する．

2. モデル化

議論を簡単にするため，できるだけ簡素なモデルを考える．ここでは，プロセッサの命令実行部に対して，命令とデータとをそれぞれ別の機構・経路により供給するハーバードアーキテクチャを前提とし，命令・データで別個のキャッシュ(1次キャッシュ：IL1, DL1)を備えるものとする．

次にアクセス頻度の偏りに着目し，メモリアクセス挙動をモデル化することを考える．メモリアドレスごとにアクセス頻度をグラフ化すると，たとえば**図1**(a)のようになろう．命令・データのどちらでも良いし，両者を混合したものでも良い．このように，アドレスによってアクセス頻度が著しく異なるものと推測される．ただしアドレス値とアクセス頻度の関係をグラフにしても得られる情報は少ない．

† 宇都宮大学工学部
Utsunomiya University, Faculty of Engineering

そこでさらに議論を進めるため，アクセス頻度の高い順にアドレスを並べ替える．すると，図 1(b) のように分布曲線を描くはずである．このようにして，アクセス頻度に対して分布関数を当てはめられるようにすれば，議論がしやすくなる．たとえば，高頻度でアクセスされる部分が必ずキャッシュに乗り，はみ出た部分 (spill) がキャッシュミスを引き起こす，といった非常に単純化したモデルを考えることも可能である．

3. 分布関数とキャッシュヒット率

メモリアクセスの頻度の偏りが，分布関数により表せるものとして議論を進める．実際のプログラムでは様々なアクセスの偏りのしかたが考えられるが，本稿では，正規分布，指数分布，べき分布，コーシー分布を考え，メモリアクセスがこれらの分布関数で近似できるものとする．以下，キャッシュ容量を C，メモリアクセス範囲の上限を DR として議論する（図 1(b) 参照）．

3.1 分布関数モデル

任意の分布関数 $f(x)$ を考える．ここで，$f(x)$ はアドレス x に相当するメモリブロックのアクセス回数を表すものとする．$f(x)$ は $x \geq 0$ で定義される単調減少関数であり，$\lim_{x\to\infty} f(x) = 0$ である．現実的には，$x > DR$ で $f(x) = 0$ となる．メモリアクセス回数の総計を N_{acc} とすれば，

$$N_{acc} = \int_0^{\infty} f(x)dx = \int_0^{DR} f(x)dx \qquad (1)$$

である．

3.2 単純モデル

ここで，分布関数のうち $0 \leq x < C$ の部分がキャッシュブロックに均等に（相互に干渉せず）割り当てられるものとする．この場合，スピル部分のみがキャッシュミスを起こす．スピル部分の個々のアクセスは，キャッシュ内のどのブロックと競合するのか確定できないため，ミス率の期待値を求める．すなわち，スピル部分の面積 $S_{spil} = \int_C^{\infty} f(x)dx = \int_C^{DR} f(x)dx$ を求め，キャッシュ容量 C 内の $f(x)$ に対して，S_{spil}/C をミスヒット率として計上する．その様子を**図 2** に示す．ここで，メモリアドレス x_i でのミスヒット率は

$$m_i = (S_{spil}/C)/(f(x_i) + S_{spil}/C) \qquad (2)$$

となる．キャッシュメモリ全体でのミスヒット率は，すべての m_i についての調和平均を求めれば良く，S_{spil}/N_{acc} となる．

上記では暗黙のうちに $DR > C$ とした．これが成り立たない場合は，スピルが発生しないことになり，ミスヒット率はゼロとなる．

3.3 精密モデル

次に，メモリブロックがキャッシュブロックにマップされる任意の組合せに対して，その生起確率とそのときのミス率により全体のミス率の期待値を求めることを考える．これを精密なモデルと呼ぶことにする．

アドレス上限 DR までの各メモリブロックは，キャッシュ内のいずれかのブロックにマップされる．簡単のために 2 つのメモリブロックのみをアクセスする場合を考えよう．ブロック 1 (b_1) とブロック 2 (b_2) がキャッシュ上で競合しなければ，ミス率は 0%となる．唯一競合する場合は，b_1 と b_2 がキャッシュの同一ブロックにマップされた場合である．ここでさらに，当該キャッシュブロックでのミス率を考える．ある時点で当該ブロックに b_1 が入っていたとする．次にヒットする確率は b_1 の出現確率 $p(b_1)$ に等しく，ミスする確率は $(1 - p(x_1))$ である．また，当該ブロックに b_2 が入っていた場合のミス発生確率も同様に $(1 - p(x_2))$ である．これらのミス発生確率に，b_1, b_2 の出現確率を乗じれば，トータルのミス発生確率の期待値が求められる．したがって，あるキャッシュブロック b_i に n 個のメモリブロックがマップされる状況では，ミスの発生確率の期待値は

$$P_{bm} = \sum_{j=1}^{n} p(x_j) \cdot (1 - p(x_j)) \qquad (3)$$

で求められる．

キャッシュ全体でのミス率を求めるには，キャッシュブロック 1 個について行った上述の議論を拡大すれば良い．b_1 から b_{DR} のメモリブロックを要素数 C のキャッシュにマップする全組合せについて，その生起確率と式 (3) によるミス率の期待値とを乗じ，それらの総和を求めれば良い．しかし，全組合せについて上述の計算を行うことは現実的とは言えない．このために以下で近似的な手法を検討する．

3.4 近似モデル

図 3 にそのモデルを示す．アクセス頻度順に並べられたメモリブロックは，その順にキャッシュブロックにマップされる．キャッシュ容量 C を越える部分（スピル）も同様にしてサイクリックに重畳していく．その結果，i 番目のキャッシュブロックには，メモリブロック $x_i, x_{(i+C)}, x_{(i+2C)}, \ldots$ がマップされる．これが平均的な状態と考える．キャッシュブロックひとつあたりのメモリブロックの重畳数は $\lceil DR/C \rceil$ であるから式 (3) よりキャッシュブロックあたりのミス率を求めることができる．

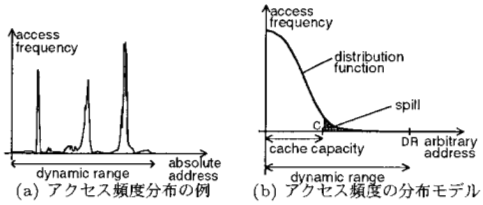
上記の分布関数 $f(x)$ を用い，メモリブロックの出現確率を $p(x) = f(x)/N_{acc}$ とすれば，キャッシュ全体のミスヒット率の期待値は

$$\sum_{i=1}^{C} \sum_{j=1}^{\lceil DR/C \rceil} p(x_{((j-1)\cdot DR+i)}) \cdot (1 - p(x_{((j-1)\cdot DR+i)}))$$

となる．

3.5 人為パターンによる試行

分布関数に従う乱数生成器を作成し，その数値をアドレスを入力としてキャッシュをシミュレートし



(a) アクセス頻度分布の例　(b) アクセス頻度の分布モデル

図 1　メモリアクセスのモデル

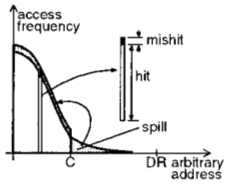


図 2　ミス率の単純モデル

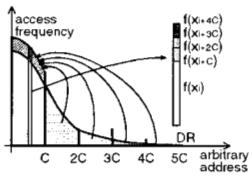


図 3　ミス率の近似モデル

た．発生アドレス数は 1,000 万，キャッシュは容量 2^n $(2 \leq n \leq 16)$ のダイレクトマップ方式である．乱数生成には Mersenne Twister を使用した．使用した分布関数は，本節冒頭に示したものである．分布関数ごとに適当なパラメータを数通り設定し，キャッシュサイズに対するヒット率を求めた．その結果を**図 4** に示す．

分布関数による差が明確に現れることを期待したが，べき分布以外は類似した形状をしている．このため，グラフ形状だけで分布関数を推定するのは困難である．(ただし，べき分布かそれ以外かはわかる)．

4. エントロピーの導入

メモリアクセスはプログラムの実行挙動に従った偏りや規則性を示すはずである．代表的なアルゴリズムからプログラムの実行挙動，さらにメモリアクセス挙動を解析する手法も考えられるが，本稿では敢えて統計的なアプローチを試みる．前節では，プログラムの実行挙動を分布関数の形式で抽象化し議論した．本節では，情報量の観点からの議論を試みる．

メモリへのアクセスアドレスは，命令・データそれぞれに対して逐次的に発生するものとする．我々は文献 2)~4) において，Shannon による古典情報理論を時系列に発生する条件分岐命令の実行結果に適用し，その情報量☆を求めた．これに対して本稿では，メモリアドレスの分布の様子（図 1）そのものをエントロピーとして定量的に表現することを試みる．

4.1 アクセス分布エントロピー

キャッシュブロックのサイズを一定とする．メモリアドレスを仮想的にキャッシュブロックと同一のサイズで分割し，このブロックを単位として考える．(仮にデータブロックと呼ぶ)．

全体のアクセス数を N_{acc}，i 番目のデータブロック B_i のアクセス数を n_i としたとき，$p_i = n_i/N_{acc}$ として，

$$H(B) = -\sum_i p_i \log_2 p_i \qquad (4)$$

によりメモリアクセスアドレスの分布に対するエントロピー値が求められる．これを以降では，**アクセス分布エントロピー** (distribution entropy) と呼ぶ．

☆ 次に実行する分岐命令の実行結果の確からしさ．

4.2 キャッシュエントロピー

次に，視点を変え，キャッシュエントリの使用頻度に着目しよう．前節でアクセス分布エントロピーを定義したが，これは，メモリアクセス自身の分布状態を表すものであり，キャッシュメモリの使用状況まで表すことはできない．すなわち，アクセス分布エントロピーは，あくまでキャッシュメモリの入力の状態量であり，その入力の結果もたらされたキャッシュ内部の状態を表すものではない．

そこで，キャッシュの各エントリの使用頻度をもとにエントロピーを定義することを考える．前節で用いたデータブロックの代わりに，キャッシュブロックを単位として用いる．全体のアクセス数 N_{acc} は上記と同じである．i 番目のキャッシュブロック C_i の使用回数を c_i，$q_i = c_i/N_{acc}$ として，

$$H(C) = -\sum_i q_i \log_2 q_i \qquad (5)$$

によりエントロピーを定義する．これを，**キャッシュエントロピー** (cache entropy) と呼ぶことにする．

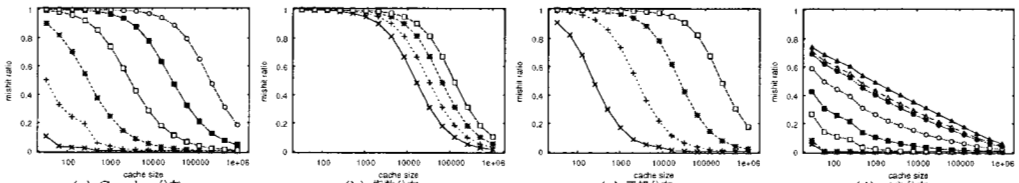
4.3 $H(B)$, $H(C)$ の比較

3.4 節において図 3 をもとに議論したように，元のアクセス頻度の分布に従い単純に重畳してキャッシュブロックにマップした場合のエントロピーを考える．アクセス分布エントロピー $H(B)$ は式 (4) のとおりである．単純な重畳によってマップした場合のキャッシュエントロピー $H(C')$ は，式 (5) をもとに $H(C') = -\sum_i q'_i \log_2 q'_i$ で与えられる．ここで $q'_i = \sum_{j=1}^{\lceil DR/C \rceil} p(x_{((j-1)\cdot DR+i)})$ である．

もし $H(B) \approx H(C')$ の場合，アクセス分布は $0 \leq x \leq C$ で急峻な（偏りの著しい）曲線であることが予測される．むろんこの場合のヒット率は高い．逆に $H(B) >> H(C')$ の場合は分布曲線がなだらかであることが推測される．この場合高いヒット率は期待できない．これに関しては 5.5 節において議論する．

5. SPEC CPU2000 による評価

SimpleScalar toolset の `sim-cache` をベースとし，アクセス分布エントロピー，キャッシュエントロピーの測定機能を追加した．キャッシュのヒット率，両エントロピー値は，10,000,000 命令によるウインドウ時間を単位として測定した（$N_{acc} = 10^7$）．キャッシュ



(a) Cauchy 分布　(b) 指数分布　(c) 正規分布　(d) べき分布

図 4　アクセスアドレスの分布関数の違いによるキャッシュヒット率の違い

構成は，sim-cache の設定に従い，sim-cache でのキャッシュエンティティごとに各値を測定する．(本稿の評価では，一次命令キャッシュ，一次データキャッシュ，二次命令・データ統合キャッシュについてそれぞれ測定した．ただし結果は一次キャッシュのみを示す)．

5.1　キャッシュサイズとヒット率の関係

まず，メモリアクセスの傾向が時間の経過に伴ってどのように変化するか（あるいはしないのか）を調べた．キャッシュサイズごとにヒット率の時系列変化を求めることにより，キャッシュサイズ—時間（ウインドウ時間）—ヒット率の 3 次元グラフを求めた．結果を**図 5** に示す．左奥から手前に向かう軸がキャッシュサイズを log スケールで表している．左から右に向かう軸は経過時間を表しており，表示単位はウインドウ時間（10,000,000 命令）である．開始時間は，プログラムによって任意に選んでいる．そして縦軸がデータキャッシュ(DL1) のミス率を表している．

キャッシュサイズに対するキャッシュヒット率の変化の様子を見ることで，そのウインドウ時間におけるメモリアクセスの挙動状況（アクセス分布）を知ることができる．プログラムによって，アクセス分布が目まぐるしく変わるものと，ほとんど変化しないものとがある．また，短いウインドウ時間でナイフの刃のように突出している分布が多く見られる．その持続時間（ナイフの刃の厚さに相当）は，多くの場合，その持続時間は 1 ウインドウ時間内である．

キャッシュサイズに対するミスヒット率の変化，すなわちメモリアクセスの分布のグラフ形状は，大きく，台形状と三角形状に分類できよう．台形状の分布は，キャッシュサイズを一定以上大きくしなければヒット率が得られない状況を示す．これは，図 4 における cauchy 分布，指数分布，正規分布の分布形状に近い．また，三角形状の分布は，図 4 におけるべき分布のそれに近いものと推察される．ただし，図 5 の各グラフのミス率は，図 4 のそれと比較して，かなり低い．この理由として以下が考えられる．

- 実際のメモリアクセスの分布関数の性質によるため．
- ミス率が極めて低いアクセス分布の中に，別の分布関数を持つアクセスが混在するため．

これ以上の解析は今後の課題とする．

図 5 から，プログラムの実行経過に伴い，メモリアクセスの挙動（アクセス分布）も時としてまったく異なる分布形状のものに変化していることがわかる．

5.2　エントロピーの時系列変化

SPEC CPU2000 の各プログラムについて，ウインドウ時間ごとのヒット率，アクセス分布エントロピー，キャッシュエントロピーを求めた．結果を**図 6** に示す．キャッシュは容量 1024 ブロックのダイレクトマップ方式による一次データキャッシュである．図の各グラフ中では，ヒット率（最上段），アクセス分布エントロピー（中段），キャッシュエントロピー（下段）を示している．

5.3　アクセス分布エントロピーとヒット率

メモリアクセスが薄く広く分散している場合，アクセス分布エントロピーは大きくなる．逆に，分布範囲自身（DR で定義）は広くても，大多数のアクセスが特定の狭い範囲にとどまっていれば，アクセス分布エントロピーは小さくなる．このため，アクセス分布エントロピーが大きいほどキャッシュはヒットしにくく，同エントロピーが小さいほどヒットしやすいものと予想される．ただし，アクセス分布の大きさがキャッシュ容量以下である場合は，ほとんどヒットするものと考えられるから，アクセス分布エントロピー値が一定以下の領域でヒット率が飽和するはずである．図 6 の各グラフから，こうした傾向を読み取ることができる．

図 6 の各ウインドウ時間での測定結果をもとに，アクセス分布エントロピーとヒット率の関係を求めた．**図 7**(a) が命令キャッシュ(IL1) での関係であり，同図 (b) がデータキャッシュ(DL1) での関係である．

上述の通り，アクセス分布エントロピーが小さいとき，大多数のメモリアクセスが狭い範囲に集中していることを表しているから，ヒット率は高くなる．逆に，アクセス分布エントロピーが大きいとき，メモリアクセスは広く薄く分布することからヒット率が落ちる．

一方で，アクセス分布エントロピー値とヒット率の関係は単純な相関関係で表されるものでもないことがわかる．エントロピー値が大きいほどヒット率が広い範囲で分散しており，相関関係そのものは認められるがエントロピー値からヒット率を推定することは，(特にエントロピー値が大きい領域で) 困難である．

5.4　アクセス分布エントロピーとキャッシュエントロピー

次に，アクセス分布エントロピーとキャッシュエン

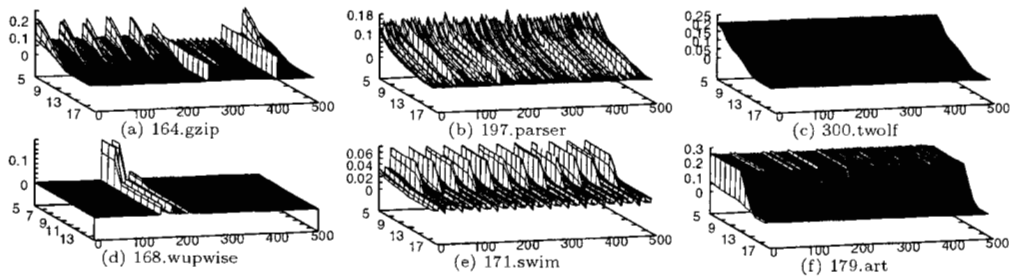


図 5　キャッシュサイズとヒット率の関係の時間変化の様子 (DL1)

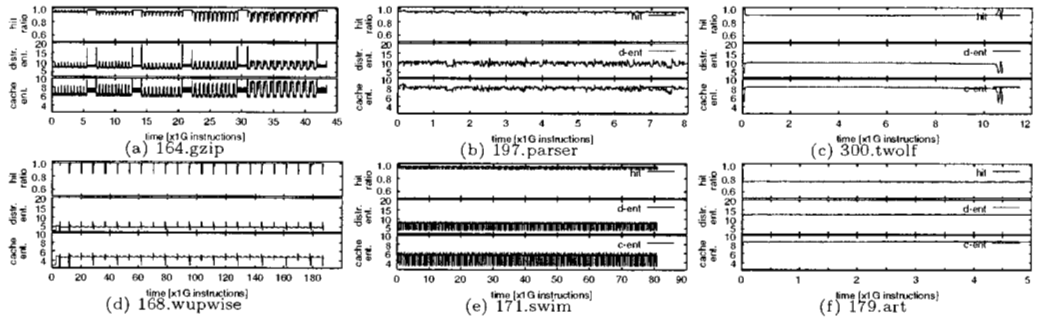


図 6　キャッシュのヒット率とアクセス分布エントロピー，キャッシュエントロピーの時間変化 (DL1)

トロピーの関係を検証する．図 6 では両エントロピーが同期した変動を示しており，良好な相関関係が推測される．実際，メモリのアクセス分布に大きな偏りがある場合，その特徴は，キャッシュブロックの使用頻度にも大きな偏りとして表れてくることが容易に考えられる．

前節と同様にして，測定した全ウインドウ時間でのアクセス分布エントロピーとキャッシュエントロピーの組を 2 次元のグラフ状にプロットし，両者の相関関係を調べた．その結果を**図 8** に示す．

元になるメモリアクセス分布が広い（すなわちアクセス分布エントロピーが大きい）ほど，キャッシュエントリの使用の偏りが少なくなる（エントリが満遍なく使われる）傾向がある．

5.5　アクセス分布／キャッシュエントロピーの差とヒット率

各エントロピーが「分布の広さ加減」を表しているものと考えるなら，アクセス分布エントロピーと，キャッシュエントロピーの差がヒット率と相関関係にあるはず．CPU からのメモリアクセスは，キャッシュのいずれかのブロックにマップされる．アクセス分布エントロピーで表されるメモリアクセス分布が，キャッシュブロックへのマップによって，別の形状の分布へと変化する．むろん，変化後の分布はキャッシュエントロピーで表される．したがって単純に考えれば，両エントロピーの差がキャッシュブロックへのマップによってアクセスが重畳された実質的な分量を表すことになる．

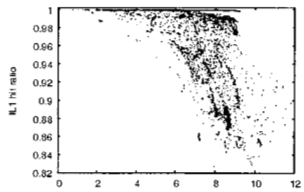
この仮説を検証するため，前節と同様にして，アクセス分布エントロピー／キャッシュエントロピーの差とヒット率の相関関係を求めた．結果を**図 9** に示す．横軸は両エントロピーの差であり，縦軸がヒット率を表す．命令キャッシュは 256 エントリ，データキャッシュは 4096 エントリである．

5.6　考　　察

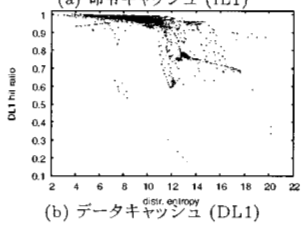
図 9 では期待したほどの相関関係が得られていない（ただしある程度の弱い相関は認められる）．単純に差をとったのではダメであり，元の分布の大きさ ($H(B)$) を考慮する必要があるのか，更に検討の余地がある．

図 8 に示されているようにアクセス分布エントロピーとキャッシュエントロピーの間にはきわめて高い相関関係がある．しかし図 7 によれば，アクセス分布エントロピーとヒット率とは十分に高い相関があるわけではない．こうした状況をさらに解析し関連を明らかにしていく必要がある．

図 6 によれば，キャッシュ容量に対するヒット率の関係が，1 測定ウインドウ時間だけ特異な形状になる場合があることが観測されている．このことから，プログラムのメモリアクセス挙動に対して測定に用いたウインドウ時間が長すぎることが考えられる．この場合，ある分布パターンから別の分布パターンへの遷移が，同一のウインドウ時間内に発生したものと考えられる．アクセスパターンが変化する場合は，前後のアクセスパターンの差異の分が新たなキャッシュミスとなる．

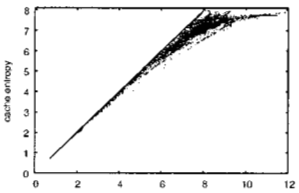


(a) 命令キャッシュ (IL1)

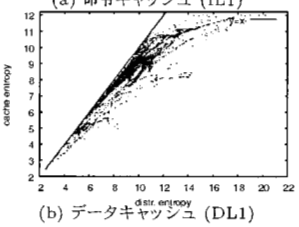


(b) データキャッシュ (DL1)

図 7 $H(B)$ とヒット率

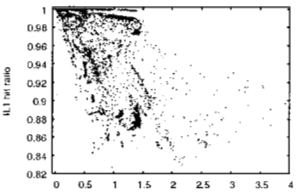


(a) 命令キャッシュ (IL1)

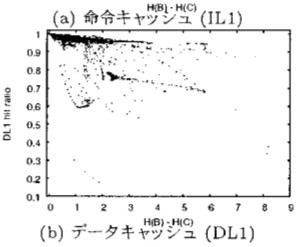


(b) データキャッシュ (DL1)

図 8 $H(B)$ と $H(C)$



(a) 命令キャッシュ (IL1)



(b) データキャッシュ (DL1)

図 9 $(H(B) - H(C))$ とヒット率

また，本稿で行った両エントロピーの定義が妥当であったのか，あるいは，エントロピーの表す値の解釈，および評価の方法が妥当であったのか，検証が必要であろう．本稿では主として空間的局所性に関した議論を行ったが，キャッシュメモリの場合は，時間的・空間的局所性のほか，ミスヒットの発生によるリプレースも考慮する必要があり，今後これらに対しても議論を展開していく必要があろう．

6. 関連研究

Chow はキャッシュのミス率がキャッシュ容量のべき乗に比例するとの結果を示している[5]．また，Voldman らはフラクタルを用いてキャッシュの振舞をモデル化している[6]．Phalke らは同一メモリアドレスへのアクセス間隔とミス率をもとに，情報圧縮の考えを適用した手法を提案している[7]．また，Kharbutli らはアクセス分布のパラメータ (balance, concentration) の計算式を定義し，キャッシュアクセスの分散手法を検討している[8]．また，Jaleel らは SPEC ベンチマークのメモリ・ワークロードを示している[9]．文献 10) では，キャッシュ周りのシミュレーションの高速化手法の一環として，メモリアクセスの特徴量を検討し，複数の統計値を組み合わせて用いている．

7. おわりに

プログラム実行時におけるメモリアクセス挙動は，キャッシュの性能に大きく影響し，実行性能を大きく左右することになる．キャッシュの有効性は，言うまでもなくメモリアクセスの時間的・空間的局所性を大前提としている．我々はこうした定性的な事実認識から進め，メモリアクセス挙動に対して統計的・情報理論的な観点から定量的な理解ができないか議論した．

実プログラムでのキャッシュ特性を，代表的な分布関数をもとに人為的に生成したパターンでの特性とを比較した．また，メモリアドレスの分布のしかたや，キャッシュブロックの使用回数の偏りから独自のエントロピーを定義して，キャッシュ性能と比較した．本稿での評価は現在のところ予備実験的な位置づけであり，得られている現象を十分に説明できるだけの材料は得られていないが，各プログラムでのメモリアクセス挙動に関して，今後の研究につながる知見を得られた．

謝辞 本研究は，一部日本学術振興会科学研究費補助金（基盤研究 (B)18300014，同 (C)19500037，若手研究 (B) 17700047），および宇都宮大学重点推進研究の援助による．

参考文献

1) J. L. Hennessy, D. A. Patterson, “Computer Architecture,” Morgan Kaufmann, 2003.
2) 横田ほか “分岐予測器における予測成功率とエントロピー”，情報処理学会研究報告，2005–ARC–165, 2005.
3) T. Yokota, et al., “Entropy Properties in Program Behaviors and Branch Predictors,” Proc. IASTED PDCS 2006, pp.448–453, Nov. 2006.
4) T. Yokota, et al., “Introducing Entropies for Representing Program Bahavior and Branch Prediction Performance,” Proc. Workshop on Experimental Computer Science,June 2007.
5) C. K. Chow, “On Optimization of Storage Hierarchies,” IBM J. Res. and Dev., Vol.18, No.3, pp.194–203, May 1974.
6) J. Voldman, et al., “Fractal Nature of Software-Cache Interaction,” IBM J. Res. and Dev., Vol.27, No.2, pp.164-170, May 1983.
7) V. Phalke, B. Gopinath, “Compression-Based Program Characterization for Improving Cache Memory Performance,” IEEE Trans. Comput., Vol.48, No.11, pp.1174–1186, Nov. 1997.
8) M. Kharbutli, et al., “Using Prime Numbers for Cache Indexing to Eliminate Conflict Mises,” Proc. HPCA 04.
9) A. Jaleel, et al., “CMP$im: A Binary Instrumentation Approach to Modeling Memory Behavior of Workloads on CMPs,” Univ Maryland, Tech. Report UMD–SCA–2006–01.
10) 小野ほか，“メモリアクセスの特徴を活用した高速かつ正確なメモリアーキテクチャ・シミュレーション法”，SACSIS2007 論文集，pp.19–26, May 2007.