計算機アーキテクチャ 124－6
（1997. 6. 27）

ディスクアレイのキャッシュ上圧縮データ管理方式

小原 清弘† 高本 良史† 小田原 宏明†

ディスクシステムのキャッシュ上データを圧縮することにより、実効的なキャッシュ容量・ヒット率を増加させ平均レスポンスタイムの短縮が期待できる。一方、圧縮データは可変長のため、空き領域検索やガーベージコレクション等の処理が新たに必要となり、このオーバヘッドがレスポンスタイム短縮効果を相殺してしまう。この問題に対し従来の管理方式と上位互換かつ低コストの圧縮データ管理方式を提案する。この方式は30%～100%のキャッシュ増加率が期待でき、管理によるオーバヘッド増加はレスポンスタイムの1%以下である。これによりキャッシュ容量・ヒット率向上による平均レスポンスタイム向上効果を十分に享受でき、システム全体のコスト性能比を向上できる。

Improving Disk Cache Performance Via Low-Overhead Compressed Data Managemant Algorithm

Kiyohiro OBARA† Yoshifumi TAKAMOTO† Hiroaki ODAWARA†

Compressing cached data in disk array systems increases effective cache size and cache hit rate, and thus improves system response time. On the other hand, compressed data requires variable-length record managements, free space search and garbage collection. These overhead cancels performance improvements achieved by increasing effective cache size. This paper proposes a new cached data managing method with low overhead, upper compatible with conventional data managing method. With this method, the effective cache size increases by 30% to 100%, but the overhead is reduced to less than 1% of the response time. This method improves system cost-performance effectively.

1. はじめに

近年、複数のディスクドライブにデータを分割して格納するディスクアレイシステムが、メインフレームおよびサーバマシン用の大規模ストレージとして一般的になってきている[1)2)]。ディスクアレイシステムは、スループットと信頼性において、単一のディスクドライブ以上のレベルが達成可能である。

一方、レスポンスタイム（応答性能）は、ディスクアレイの部品となるディスクドライブ個々の性能に強く依存するが、キャッシュメモリの搭載により向上することが可能である。しかし、ディスクドライブと比較して半導体メモリは高価なため、大容量のキャッシュメモリの搭載は、コストを押し上げる原因になる。

ところで、記録データ量の削減を行う手段として、データ圧縮がある[3)]。一般にデータ圧縮により、データ量は約半分程度に削減可能である。したがって、データ圧縮をキャッシュメモリ上のデータに応用した場合、実効的にキャッシュメモリの容量が二倍に増えたとみることができ、キャッシュヒット率の増加によりレスポンスタイムの向上が期待できる。一方デメリットとして、圧縮後のデータサイズが不定なため可変長データの管理が必要になり、オーバヘッドとなる。

したがって、キャッシュ上のデータ圧縮により、キャッシュ容量増大効果およびレスポンスタイムの向上効果を得るには、可変長の圧縮データの管理コストを削減することが必要である。この問題に対し本報告は、ブロック単位圧縮方式と呼ぶ、キャッシュ上の低オーバーヘッドな圧縮データ管理技法を示す。

これまでのディスクシステムにおけるデータ圧縮は、ディスクドライブ上のデータを対象としており、キャッシュ上のデータに関する研究は少ない。ディスクドライブ上のデータ圧縮と、

†㈱日立製作所 中央研究所
Hitachi, Ltd., Central Research Laboratory

本報告のキャッシュデータ圧縮は、共存も可能である。

2. キャッシュデータ圧縮の効果と問題点

2.1. ディスクアレイにおけるキャッシュの効果

プロセッサシステムにおけるキャッシュメモリと同様に、ディスクアレイにおいてもキャッシュ容量を増やすにつれキャッシュヒット率が向上し、その結果、平均レスポンスタイムが向上する。この向上効果はホストで動作するアプリケーションに依存する。例えばメインフレーム向けディスクアレイシステムでは、キャッシュ容量を二倍に増やした場合、キャッシュヒット率が十数ポイント、平均レスポンンスタイムが数十%向上した実例がある。

ディスクシステムの大容量化に伴い、搭載されるキャッシュ容量も大容量化している。しかし、キャッシュはビット単価の高い半導体メモリを用いるため、キャッシュの大容量化はシステムコストに大きな影響を与える。一般に、キャッシュメモリのコストはディスクアレイシステム全体のコストの大きな割合を占める。

このような高価なキャッシュメモリのコストを削減する手段として、キャッシュ上データの圧縮が考えられる。一般にデータ圧縮により、データ量は元の半分以下に削減できる。したがって、データ圧縮をキャッシュメモリ上のデータに適用した場合、非圧縮時と比較して、二倍のキャッシュメモリを搭載した場合と実効的に等価になる。また、半分のキャッシュ容量でも実効的なキャッシュヒット率は変わらないため、実効的に非圧縮システムと等価な性能が期待でき、システムのコスト削減に大きく寄与できる。

2.2. キャッシュデータ圧縮の問題点

データ圧縮には明らかなデメリットも存在する。その中でも最も重要なことは、データ圧縮の導入により、可変長データ管理式が必要になる点である。例えば、キャッシュ上のデータを更新し再圧縮した場合、元のサイズと同じになることは希である。このため一般的な領域管理を導入すると、格納可能な空き領域検索と、領域の断片化防止のためのゴミ集等の処理が必要になる。

空き領域検索やゴミ集めは、対象とする領域全体を検索する手間がかかり、キャッシュ容量に比例してそのコストは増大していく。例えば最近のメインフレーム向けディスクアレイは、数 GB のキャッシュメモリを搭載している。したがってキャッシュメモリの 1/10 の領域の走査でさえ、アクセスは数百M(メガ)回のオーダになる。一方ディスクアレイのコントローラに用いられるプロセッサは数十 MIPS 程度の性能であり、上記処理は数十秒オーダの高負荷な処理となる。

このような問題を解決し、キャッシュ圧縮によりレスポンスタイム向上効果を得るには、キャッシュ容量に依存しない低処理負荷な圧縮データ管理方式が必要である。本報告は、この問題に対処する、ブロック単位圧縮方式と呼ぶ、ディスクアレイシステム向きの低オーバーヘッドなキャッシュデータ圧縮管理技法について報告する。第3章では、このキャッシュデータ圧縮管理方式の詳細を述べる。第4章では、この管理方式の評価を行う。

3. ブロック単位圧縮方式

3.1. 従来のキャッシュ管理方式

ディスクアレイを例として、従来の非圧縮データに対するキャッシュ管理方式、および本報告で述べるブロック単位圧縮方式を説明する。図 1 にディスクアレイの内部構成例を示す。このディスクアレイシステムは独立したプロセッサを有するチャネルアダプタ(CHA)およびディスクアダプタ(DKA)と、この二つのアダプタから共有されるキャッシュメモリから構成されるマルチプロセッサシステムとなっている。

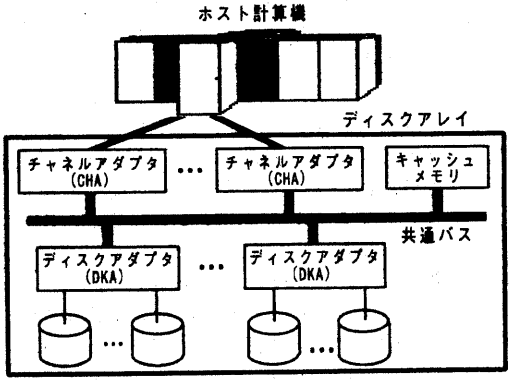

図 1 ディスクアレイの内部構成例

キャッシュ管理方式はディレクトリ方式を採用している。すなわち、システム内の全てのドライブ、トラックやレコードに対応するエントリテーブルが存在し、それがポインタでツリー状にリンクされている。目的のレコードがキャッシュ上に存在するかの判定、すなわちヒットミス判定は、このディレクトリ・ツリーを順に辿ることにより行われる。

キャッシュメモリの全ての領域は、キャッシュセグメントと呼ばれる固定長の領域単位で管理されている。個々のキャッシュセグメントの状態を管理するのが、セグメントコントロールブロック(SCB)で、ディレクトリ・ツリーの終端（葉）に当たる部分である。キャッシュセグメント一つに、SCB 中の一つのエントリが対応する。

図 3(a)に示すように、一つのキャッシュセグメントは、複数個のキャッシュブロックにより構成されている。また、SCB の一つのエントリは、キャッシュセグメント先頭アドレスと、トラック内の位置、キャッシュブロックのダーティー/クリーンのビットマップを持っている。つまりダーティー/クリーン情報、すなわちその内容が更新されたか否かの情報は、キャッシュブロック単位で管理している。

3.2. ブロック単位圧縮方式の基本概念

キャッシュ上データ圧縮に伴うオーバヘッド増加は圧縮後の可変長データの管理に起因する。この問題に対し、従来の固定長管理のオーバヘッドと同等のコストで可変長圧縮データを管理する方式が、ブロック単位圧縮方式である。ブロック単位圧縮方式の基本概念を図 2 に示す。

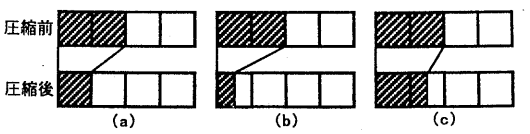

図 2 ブロック単位圧縮方式の基本概念

ブロック単位圧縮方式は、圧縮率*1 50%を前提とし、従来二つのキャッシュブロックを占めていたデータを圧縮して、一つのキャッシュブロックに格納することを基本とする（図 2(a)）。圧縮率が 50%以下の場合、圧縮データは一つのブロックサイズ以下となり余りの領域が生じるが、この領域は利用しない（図 2(b)）。逆に圧縮率が 50 %以上の場合、一つのキャッシュブロックでは格納しきれずあふれが生じる。この場合、あふれたデータは新たなキャッシュブロックに格納する（図 2(c)）。あふれを格納したブロックに対しても、余りの領域は利用しない。

*1 圧縮率は。「圧縮後ファイルサイズ/圧縮前ファイルサイズ」で表現する。
例えば、圧縮率 25%の場合、圧縮前と比較して、1/4 のサイズになる。

3.3. ブロック単位圧縮方式の詳細

従来方式とブロック単位圧縮方式における、SCB とキャッシュセグメント/キャッシュブロックの構成の比較を図 3 に示す。圧縮データを取り扱う場合、図 3(b)に示すような管理構造にする。ここでは、非圧縮時に図 3(a)のように格納されるデータを、本方式により圧縮して格納した状態を示している。あふれ格納 SCB へのポインタ（後述）とブロック内利用サイズリストが加わった事を除けば、SCB の構造は非圧縮データを管理する従来方式と同じである。ブロック内利用サイズリストは、その SCB が管理するキャッシュブロックの数だけ存在するキャッシュブロック内での利用バイト数のリストである。

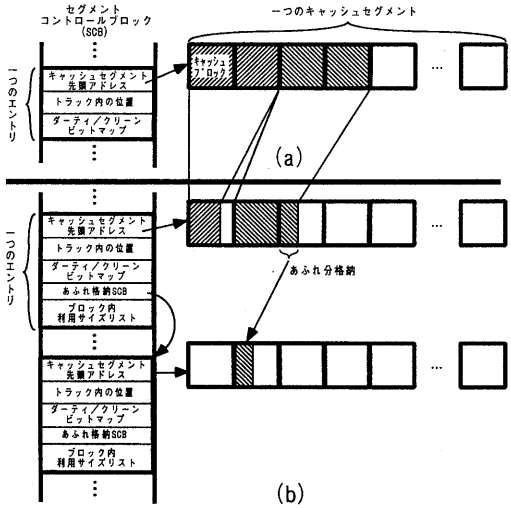

図 3 SCB とキャッシュセグメントの構成
従来方式(a)、ブロック単位圧縮方式(b)

圧縮単位は、従来のキャッシュブロック二個

分とする。つまり二個分のデータを圧縮して一つのキャッシュブロックに格納する。次の二個分のデータは、圧縮して次のキャッシュブロックに格納する。圧縮アルゴリズムやデータに依存するが、概して元データの半分以下(50%以下)に圧縮できる。したがって大部分の場合、圧縮データは一つのキャッシュブロック中に格納できる。

一般に 50%以下の圧縮率は達成可能であるが、圧縮率が 50%以上となり、圧縮しても一つのキャッシュブロックに格納しきれないケースも生じる。この場合あふれ部分を、他のキャッシュブロックに格納する。あふれ部分を格納するキャッシュブロックの決め方はいくつかある。図 3(b)では、新たに未使用のキャッシュセグメント持って来て、非あふれ部分が格納されたキャッシュブロックと同一の位置のキャッシュブロックにあふれ部分を格納している。すなわち、ここではキャッシュセグメントの二番目のキャッシュブロックがあふれたため、新たな未使用のキャッシュセグメントの二番目のキャッシュブロックにあふれを格納している。

また、今回述べたキャッシュ管理方式では、ダーティ/クリーン判定を行うブロックを複数個まとめて、キャッシュの領域管理単位（キャッシュセグメント）としている。しかし本方式は、ダーティ/クリーン判定単位と、キャッシュの領域管理単位が同一の場合に対しても適用可能である。

これらの方式では、あふれ部分を格納するキャッシュセグメントの構造、サイズ及びその管理方式は、非あふれデータを格納するキャッシュセグメントと共用である。つまり、あふれデータ専用の格納領域はない。このため本方式は、キャッシュブロックのあふれる割合、すなわち圧縮率に依存せず適用可能である。補足ながら全てのキャッシュブロックにあふれが生じたとしても、キャッシュメモリの利用効率は、非圧縮データを格納した場合と同じである。非圧縮時と比較して、キャッシュメモリの利用効率の低下はない。即ち本方式は、圧縮により二倍以下の範囲で、実効的なキャッシュメモリの容量を増やす方式を提供している。

さらに本方式は、圧縮後のデータサイズを格納するブロック内利用サイズリストのフィールドが加わっただけで、SCB のエントリの構造は非圧縮データを扱う従来の構造と上位互換である。このため圧縮/伸張処理を除き、管理方式への追加処理は、圧縮/伸張によって生じたデータの長さを管理する処理、およびあふれが生じた場合の処理だけである。キャッシュのヒットミス判定方式も非圧縮時と同じでよい。このため、本処理方式による圧縮管理コストは非常に小さく、実効的なキャッシュ容量増加とキャッシュヒット率向上によるレスポンスタイム向上を十分に享受できる。

4. ブロック単位圧縮方式の評価

4.1 キャッシュ増加率

通常、キャッシュ増加率は圧縮率の逆数である。しかし本方式は、固定長のキャッシュブロック二つを圧縮単位として圧縮し、一つのキャッシュブロックに格納するため、圧縮率が 50%以下の場合でも圧縮率は 50%に丸められる。また圧縮率が 50%以上になり、一つのキャッシュブロックに収まらない場合は、もう一つキャッシュブロックを利用する。この場合、（これらのキャッシュブロックに関しては）圧縮効果がなかったことになる。

このように本方式では、圧縮単位を考慮した圧縮率（キャッシュ増加率）の計算が必要となる。この関係を示したのが図 4 である。すなわち、対象とする入出力ファイルを圧縮単位に分割し、各圧縮単位毎に圧縮を行う。圧縮率は、各圧縮単位の圧縮率が 50%以下なら 50%、50%以上なら 100%に丸める。この丸めた圧縮率の平均を、そのファイルの「実効圧縮率」と呼び、その逆数が「キャッシュ増加率」である。キャッシュ増加率とは、キャッシュ上の全てのデータがそのファイルの場合の、キャッシュの実効的な容量増加率である。

圧縮率は対象とするファイルによって変化する。本方式の評価では、T.C.Bell らが用いたファイルセットを対象とする。これは、圧縮アルゴリズムの比較評価に広く用いられているファイルセット[3]である。このファイルセット中には、11 種類 18 個のファイルが示されているが、ディスクアレイが主に利用される用途を考慮し、7 個のファイルを選択した。また最近のディスクアレイは、インターネットの Web サーバのストレージになる場合も多い。このため評価対象のファイルに二種類の HTML 文書も加えた（表 1）。

表1 評価対象ファイル一覧

名前	内容
news	News batch file (Internet New Contents)
obj1	Compiled code for Vax
obj2	Compiled code for Apple Macintosh
pic	Picture 5 from the CCITT Fax test file
progc	C source code: compress version 4.0
progl	Lisp source code: system software
progp	Pascal source code
html1	日立の WWW サーバの新着情報ページ (JIS コード、'96 11/8)
html2	米 NetScape 社ホームページ (JAVA Script 含む、'96 11/8)

本方式においては、圧縮単位すなわちキャッシュブロックサイズの選定も重要なパラメタである。圧縮単位を大きくした方が冗長性が増し圧縮率が上がるが、同時にキャッシュブロックサイズの拡大ともなり性能に悪影響を与える。例えば、転送サイズの増加によるレスポンスタイムの低下などである。したがって、本方式が有効な効果を達成できる条件の下で、最も小さい圧縮単位を選定する必要がある。ここでは、キャッシュブロックサイズ、2KB、4KB、8KB（圧縮単位としては、4KB、8KB、16KB）に対して評価した。

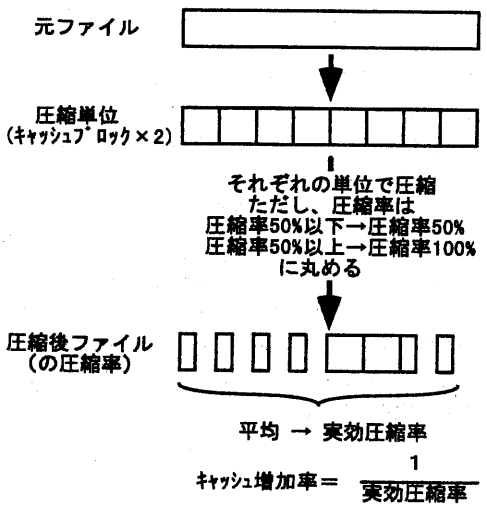

図4 実効圧縮率の計算方法

また、圧縮アルゴリズムの選択も同様に重要である。本方式の原理は圧縮アルゴリズムに依存しないが、圧縮率が悪ければキャッシュ増加率が悪くなる。ここでは、圧縮アルゴリズムとして、compress、gzip に対して評価を行った。

これまで述べた評価指標に基づいて、キャッシュ増加率を求めたのが図5から図7である

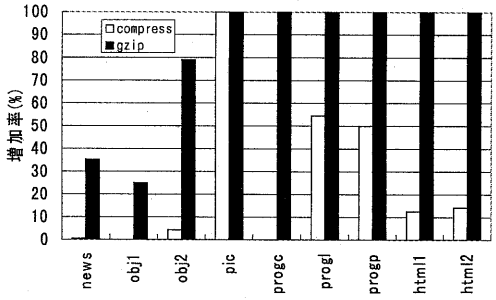

図5 キャッシュ増加率
(キャッシュブロックサイズ 2KB の場合)

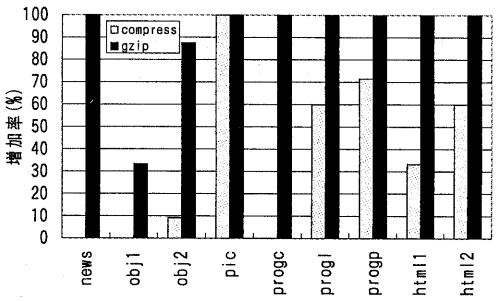

図6 キャッシュ増加率
(キャッシュブロックサイズ 4KB の場合)

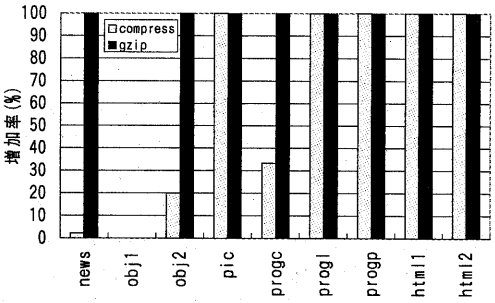

図7 キャッシュ増加率
(キャッシュブロックサイズ 8KB の場合)

compress での news ファイルおよび obj ファイルに対しては、キャッシュ増加率すなわち実効

圧縮率が悪い。これは、ファイルをキャッシュブロックサイズ（圧縮単位）に区切ったため、圧縮対象となる冗長データが、一つの圧縮単位内に出現しにくくなったためである。これらのファイルは、ファイル全体の圧縮も 50%をやや越える程度の圧縮率であり、圧縮率が 50%以下にならなければ効果が出ない本方式では不得手の性質を持つファイルである。

上記以外のファイルに対しては総じて良い結果が出ている。キャッシュブロックサイズに関しては、8KB で、ほぼ全ての圧縮アルゴリズムに対してキャッシュ増加率が 100%（二倍に増える）となっている。また、キャッシュブロックサイズが 2KB や 4KB でも比較的良い結果を出している。このため、キャッシュブロックサイズの選択は、実際に適用するシステムにおいて、サイズ増加によるデータ転送時間増加による影響等を考慮して個別に決定できる。

総覧すると、極端に増加率が悪い news、obj のケースを除いて、本方式によりキャッシュ増加率 30 ～ 100%の性能が達成されており、本方式によるキャッシュ増加効果の有効性が確認できた。

キャッシュの増加と、キャッシュヒット率の向上効果およびレスポンスタイムの向上効果の関係は、ホスト上で実行されるアプリケーションの性質に強く依存する。したがって、これらの関係を一般的定量的に示すのは困難である。しかし一般的な傾向として、システムの総容量に対しキャッシュの容量が小さい場合、キャッシュの増加によりシステムのレスポンスタイムは向上するのは自明である。

4.2 管理オーバヘッドとレスポンスタイム

前章で議論したように、ブロック単位圧縮方式は従来のキャッシュ管理方式と上位互換性があり、付加される処理内容も少ないため、オーバヘッドも小さい。この付加される管理ルーチンのオーバヘッドの値をシミュレーションにより算出した。この結果、圧縮/伸張時間を除いたオーバヘッドの増加は、4KB 入出力時において、10 μs程度であった。これは、システム全体のレスポンスタイムの 1%以下であり非常に小さい。このため、レスポンスタイムに大きく影響するのは圧縮/伸張時間となる。これに対しては、圧縮/伸張をハードウエア化すれば、見かけ上の圧縮/伸張時間を、システムのレスポンスタイムのオーダで見て、ほぼゼロにすることができる。

5. おわりに

ディスクアレイシステム向きのキャッシュ上圧縮データ管理方式について報告した。一般にキャッシュデータの圧縮により、実効的なキャッシュ容量/ヒット率が増加し、平均レスポンスタイムを短縮できる。平均レスポンスタイムが従来と同じシステムを想定した場合、キャッシュの実容量を削減でき、コスト低減に寄与できる。一方、圧縮データ管理には空き領域検索やガーベージコレクション等の可変長データ管理コストがかかり、このオーバヘッドが前記効果を相殺する。

これに対し、ブロック単位圧縮方式と呼ぶキャッシュ上圧縮データ管理方式を提案した。この方式は、キャッシュの管理単位のブロック二つをまとめて圧縮し、一つのブロックに格納する。管理単位を固定長とすることにより従来の管理方式と上位互換とし、可変長データ管理を不要化した。これにより、本方式のオーバヘッドによるレスポンスタイム増加は、1%以下に抑えられた。また、サーバ等に良く格納されるファイルに対し、30%～ 100%のキャッシュ増加率を得た。

本方式は、キャッシュ上データの管理方式であるため、RAID 方式およびディスクドライブ内への格納方式には依存しない。しかし、ディスク上へも圧縮して格納した方がよりコスト低減に寄与できる。このため、ディスク上データの圧縮管理方式との連携が今後の課題である。

参考文献

1) 菊池他:ファイル装置の信頼性向上技術，情報処理, Vol.37, No.11, pp.995-1000(1996).
2) 出揃った並列汎用機とディスク・アレイ，日経ウオッチャー IBM 版別冊，日経 BP 社(1995).
3) 植松友彦:文書データ圧縮アルゴリズム入門，CQ 出版社(1995).