1A-03

ストレージシステムにおける高性能可逆データ圧縮方式の開発

水島 永雅　新井 政弘　小関 英通　河村 篤志†

株式会社 日立製作所 研究開発グループ 情報通信イノベーションセンタ†

1.研究背景と課題

フラッシュメモリを媒体とする SSD の普及を加速するには、依然低価格な HDD と競合させるべく SSD の容量単価の削減が必要である。その実現技術の 1 つとして可逆データ圧縮がある。SSD はデータアクセスの応答時間が HDD に比べて短いことが利点であり、圧縮機能を搭載した場合でも応答性能を極力悪化させないことが重要である。圧縮機能を搭載する SSD の構成を図 1 に示す。データライトではホストからの受信データを可逆圧縮してフラッシュメモリに格納するが、データをキャッシュした直後にホストに完了応答し、圧縮と格納を後回しにするため直接的な性能影響はない。一方、データリードではフラッシュメモリから読んだデータを伸張してホストに送信するが、応答時間にデータの伸張時間が含まれるため、伸張に時間がかかるほど圧縮機能がない場合と比べて応答性能が悪化する。ゆえに圧縮機能を搭載する SSD は伸張処理をハードウェアで高速化する必要がある。

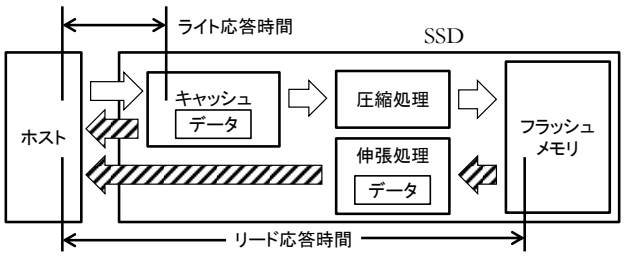

図 1　圧縮機能を搭載する SSD の構成

しかし従来の可逆圧縮アルゴリズムはハードウェアで伸張しても高速化に限界がある。最も一般的な Deflate の圧縮方式[1](図 2)は、スライド辞書圧縮 LZ77[2]に従い、平文データ(図上)を先頭からスキャンし、直前までの所定長の文字列の中から最も長く一致する文字列を発見してコピー記号に置換する(図中)。例えば 2 度目に現れる 4 文字の”bcde”は 6 文字前と一致するため[4,6]へ置換する。そして置換しなかった文字とコピー記号をハフマン符号で符号化する(図下)。

圧縮データはビット長が一定でない符号の連結で構成される。伸張処理はこれを元の平文データに逆変換する処理だが、それを高速に行うには圧縮データに含まれる複数の符号を同時に解釈して復号すること(並列化)が望ましい。

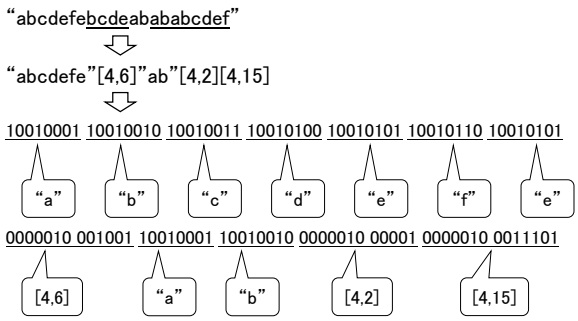

図 2　Deflate の圧縮方式

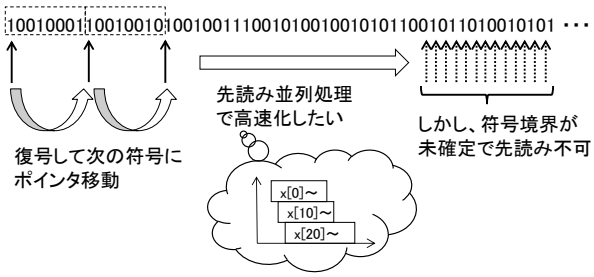

図 3　先読み並列処理の困難性

しかし各符号の長さは多様であるため、図 3 のように離れた場所の符号を先読みして並列的な復号を試みようにも符号間の境界が不明であり、直前までの符号を全て復号するまでその境界が確定しない。すなわち、伸張処理は一般に直列化せざるを得ない。実際 333MHz 駆動のハードウェアは 1 サイクルで高々2 符号しか伸張できず、高速化限界は 666MB/s となる。例えば 8KB リードでは、圧縮の効かないデータで最悪 12.3μs の伸張時間を費やす。圧縮機能のない SSD の一般的なリード応答時間を 150μs と想定すると 8.2%の応答性能悪化となる。これを常に 1%未満に抑えたいなら、例えばデータを 16 個に分割して個別に圧縮し、伸張を 16 並列化すれば 0.51%(0.77μs)に改善するが、辞書が分断されて圧縮率が大きく悪化する。(これら従来方式を、それぞれ単純圧縮、分離圧縮と呼ぶ。)

2.開発した圧縮方式

我々は伸張の高速化困難性から、圧縮率の悪い分離圧縮を選択せざるを得ない状況を回避したいと考え、先読み並列処理が可能な圧縮方式(図 4)とそれに対応した伸張回路(図 5)を開発した。

圧縮手順は、まず平文データを N(B)単位で等分割し(図は N=8)、各部分の LZ77 圧縮と符号化を行

Development of the High Performance Lossless Data Compression Method for Storage Systems

† Nagamasa Mizushima, Masahiro Arai, Hideyuki Koseki and Atsushi Kawamura,
Hitachi, Ltd., Center for Technology Innovation – Information and Telecommunications

Copyright ©2016 Information Processing Society of Japan.
All Rights Reserved.

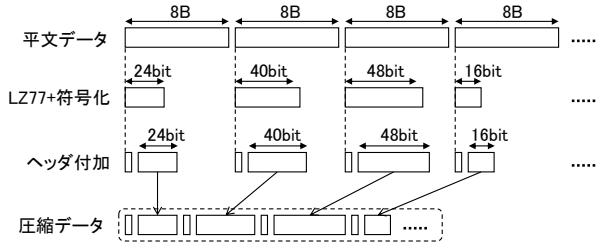

図 4　開発した圧縮方式

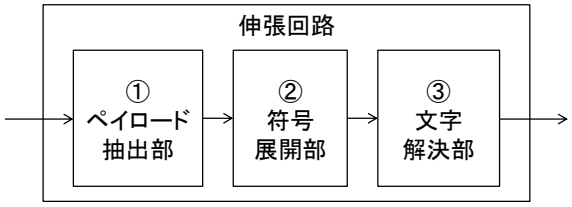

図 5　開発した伸張回路

う。圧縮率の悪化を抑えるため、LZ77 の辞書は分割境界を超えて参照可能とするが、コピー記号へ置換可能な文字列は分割境界を跨がないように置換する。圧縮後の各符号列(ペイロード)の先頭にその長さが分かるヘッダを付加する。最後にそれらを連結して圧縮データを構成する。

伸張回路は 3 段の処理部で構成される。最初のペイロード抽出部(図 6)は、1 サイクル毎にヘッダを解析し、後続のペイロードを抽出して次段に送出すると同時に解析位置を次ヘッダに移す。次段の符号展開部(図 7)は、各ペイロードを 1 符号ずつ復号するパイプラインであり、コピー符号[L, J]を L 個の空箱[J]に、他の符号を元の文字に変換する。空箱とは辞書参照するまで未解決な部分を意味する。最終の文字解決部(図 8)は、分割単位 N の幅を持つシフトレジスタとセレクタで構成した辞書参照回路であり、前段の空箱[J]にそこから距離 J の位置の文字を引いて代入し、平文を全て復元する。本伸張回路は 1 サイクル毎に N(B)の平文出力が可能であり、駆動周波数 F(MHz)での伸張速度は常に N・F(MB/s)となる。N は応答時間の許容増加率に基づき設計する。例えば F=333、N=32 とすれば 8KB リード応答時間は上記例と同じ 0.51%増となる。

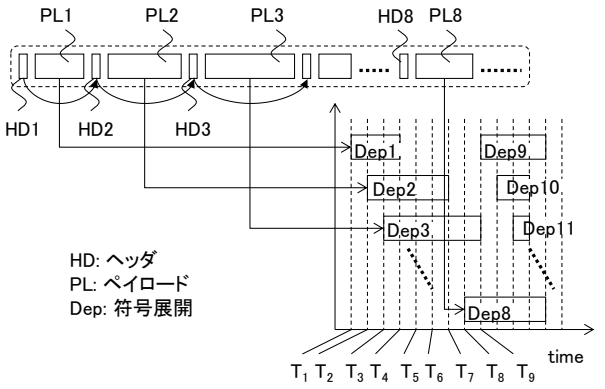

図 6　ペイロード抽出部

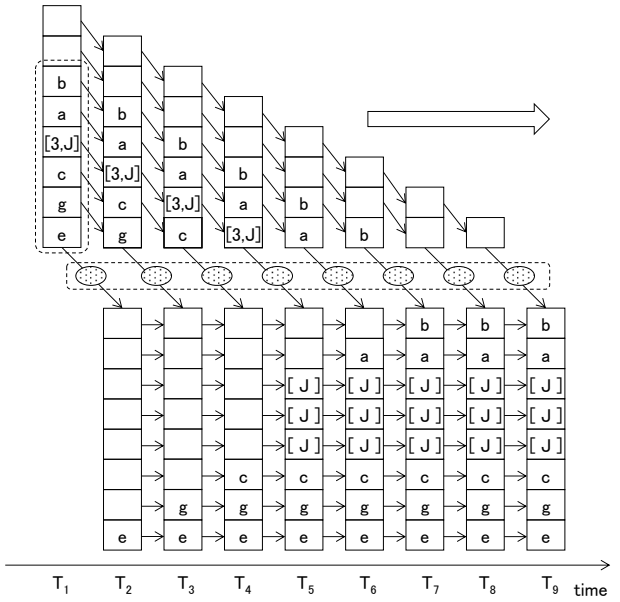

図 7　符号展開部

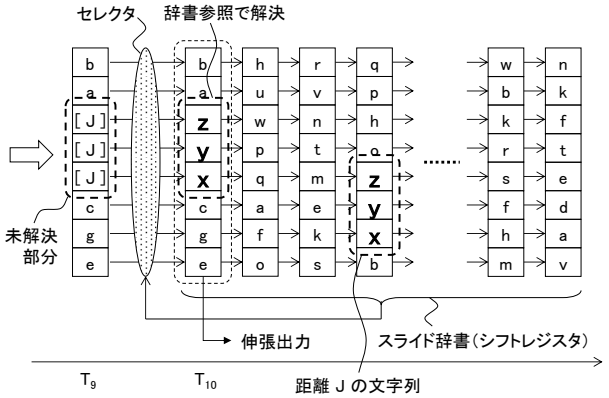

図 8　文字解決部

3. 効果

開発方式(辞書 8KB)による圧縮率を、圧縮ベンチマーク[3]の”E.coli”の先頭 8KB を用いて従来方式と比較した(表 1)。本方式は単純圧縮からの圧縮率悪化を分離圧縮の約 6 割に抑えることができた。

表 1　圧縮方式比較

方式	単純	分離	本開発
伸張時間	5.3 μs	0.43 μs	0.77 μs
圧縮結果	3528B	4544B	4125B
圧縮率	43.1%	55.5% (+12.4)	50.4% (+7.3)

参考文献

[1] RFC1951 DEFLATE Compressed Data Format Specification version 1.3, 1996.
[2] Jacob Ziv and Abraham Lempel; A Universal Algorithm for Sequential Data Compression , IEEE Transactions on Information Theory, May 1977
[3] http://corpus.canterbury.ac.nz/descriptions/#large

Copyright ©2016 Information Processing Society of Japan.
All Rights Reserved.