A-012

転送速度を 2 倍にするデータ圧縮技術

A Data Compression Method for Doubling Data Transmission Rate

井谷宣子†
Noriko Itani

1　まえがき

クラウドの普及やデータセンターへのサーバ集約により通信データ量が急増している。また、膨大なデータが収集され分析処理されており、ネットワーク速度およびストレージアクセス速度の改善が望まれている。例えば Web 検索エンジンでは、全体で数百テラバイト～数ペタバイトのデータを扱うため、多数のマシンをネットワークに繋いだ分散ファイルシステムを利用している。このようなシステムでは、ネットワークを介してのデータ送受信、ストレージからのデータ読み出しが処理速度のボトルネックとなっている。一度取得したデータを高速なメモリに保持するキャッシュ技術が進んでいるが、初めて利用するデータには効果がない。そこで、データ圧縮技術を用いてデータ量を減らすことで、ネットワークを介したデータ送受信、ストレージからのデータ読み出し時間を短縮することを狙う。

2　従来技術と狙い

バイナリデータやテキストデータなどの圧縮ではビットのロスなしに圧縮できるロスレスデータ圧縮が用いられている。ロスレスデータ圧縮でよく知られている ZLIB[1]では、LZ77 型圧縮[2]-[4]を用いており、繰返し出現するデータ列を過去の位置と長さで表現することでデータ量を削減する。

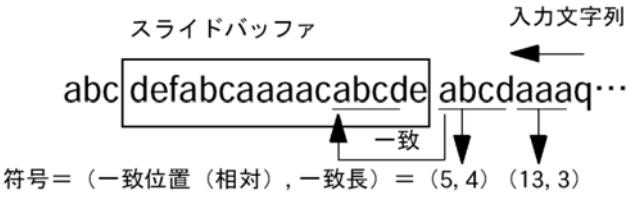


図 1　LZ77 型圧縮

例えば、図 1 の入力文字列「abcdaaq…」の先頭 4 文字「abcd」は、過去の文字列「defabcaaaacabcde」の最後から 5 文字目と一致する。この「abcd」を(5,4)と表現することで、そのままの文字列よりも小さいデータに圧縮できる。復元時には、5 文字前から 4 文字コピーすることで元のデータが復元できる。

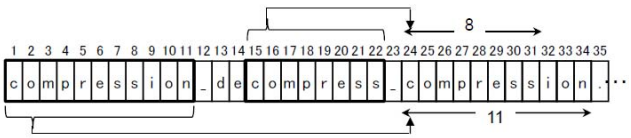


図 2　複数回出現する繰返し

このような繰返しは、対象データ中に複数の組合せが存在する場合がある。例えば、図 2 では、アドレス 24 から始まるバイト列は、アドレス 1 およびアドレス 15 から始まるバイト列と一致する。前者とは 11 バイト、後者とは 8 バイト一致するため、前者を選択した方が圧縮率を高くできる。そのため ZLIB では、複数の繰返し候補から最も長く一致する繰返しを、探索処理により求めている。さらにデータ量を減らすため、繰返しの位置や長さなどを、ハフマン符号で符号化している。圧縮時は、最長一致探索およびハフマン符号による符号化の処理時間が大部分を占める。また、復元時は、ハフマン符号の復号および繰返しのコピー処理の処理時間が大部分を占める。

一方、データ圧縮を利用してデータ転送時間を短縮するには、圧縮率と圧縮・復元の処理速度の二つの要素を考慮しなければならない。ZLIB では、標準データに対する圧縮率は 1/3 程度であり、処理速度は、圧縮時が 30 メガバイト／秒、復元時が 400 メガバイト／秒程度（Intel Core i7-3770 3.4GHz 使用時）である。この処理速度は、ネットワークおよびストレージで広く使われているギガビットイーサ（1Gbps）や SATA3（600 メガバイト／秒）の転送速度より低く、データ量を 1/3 に減らせても、転送速度を高速化できない。

本稿では、ギガビットイーサおよび SATA3 の転送速度を、体感的に効果のある 2 倍にすることを目指し、圧縮率 1/2 で、2Gbps（256 メガバイト／秒）以上の圧縮速度、1200 メガバイト／秒以上の復元速度を実現する圧縮方式の開発を目標とした。

3　開発方式

データの特性及び処理系の特徴を利用することで、LZ77 型圧縮をベースに、圧縮率 1/2 で、圧縮・復元処理を高速化する方式を開発した。具体的には、処理時間の大部分を占める圧縮時の繰返し探索、符号化、復元時の繰返しコピー処理の 3 つの処理を高速化した。

3.1　繰返し探索における比較候補の削減

繰返し探索では、繰返し候補の数に比例して処理時間が増加する。圧縮率低下を抑えつつ繰返し候補を 1 つに絞ることで処理速度を向上した。

まず、圧縮率への寄与が少ない 3 バイト列以下の短い繰返しを探索対象から除外することで繰返し候補を削減した。さらに、テキストやプログラムなどでは繰返しが近くにある傾向があることから、最も近くに存在する 4 バイト列の繰返しのみに候補を絞った。この一つの候補についてのみ 5 バイト目以降の一致を比較し、最も長く一致するバイト数を求めた。

3.2　バイト単位の符号

高速処理できるバイト単位の符号を使用した。符号をバイト単位で区切り、個々の符号のビットの長さおよび位置

†（株）富士通研究所

を固定することで、少ない命令で符号化・復号できるように符号を構成した。

具体的には、図 3 に示す 1 バイト～4 バイトの符号を使用した。圧縮率を上げるため、最短の 1 バイト、2 バイトの符号を、出現回数の多い非繰返しと、近接した繰返しに割当てた。復元時には、先頭の 1～3 ビットで 1 バイト～4 バイトの符号のいずれかを識別可能とした。

例えば、5 バイト前の 4 バイトの繰返しは、2 バイトで符号化する。位置 5 は 10 ビット「00 0000 0101」、長さ 4 は 5 ビット「0 0100」で表現し、それぞれ d および t に対応する位置に格納することで 2 バイト「0000 0100 0000 0101」に符号化する。

	第 1 バイト	第 2 バイト	第 3 バイト	第 4 バイト
1 バイト（非繰返し）	110t tttt	{非繰返のバイト列}		
2 バイト（繰返し）	0ddt tttt	dddd dddd		
3 バイト（繰返し）	10dt tttt	dddd dddd	0ddd dddd	
4 バイト（繰返し）	111t tttt	tttt tttt	dddd dddd	dddd dddd

先頭 1～3 ビットで 1 バイト～4 バイト符号のいずれであるかを区別する。
t：長さのビット（l と1 が見分けにくいのでt で記載）
d：位置のビット（相対）

図 3　バイト単位符号のビット割当て

3.3　多バイトコピーに適した符号化

昨今の CPU に装備されている多バイトコピー命令を活用することでコピー処理を高速化し、次に示す多バイトコピーに不向きなコピー元とコピー先の重なりを除外することで高速化を行った。

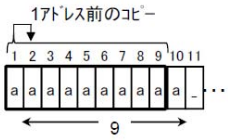


図 4　コピー元とコピー先が重なる例

図 4 の例では、第 1 バイトを第 2 バイトにコピーし(ステップ 1)、第 2 バイトを第 3 バイトにコピーする(ステップ 2)。そのため、ステップ 1 の処理が終わるまで、ステップ 2 の処理が行えず、1 バイトずつのコピーとなる。

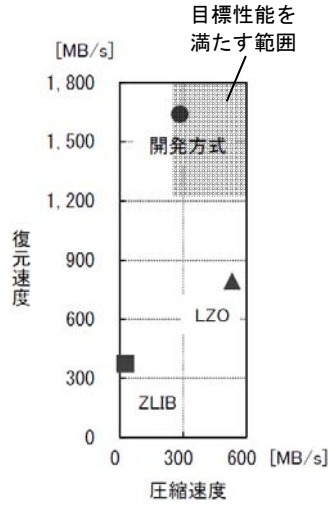
符号生成において、コピー元とコピー先との重なりをなくした。コピー元とコピー先が重なるのは、繰返しの位置 d が長さ t よりも小さいときであるため、d<t の繰返し(d, t)を(d, d), (2d, 2d), (3d, 3d)…のように分割した。

4　評価

ロスレスデータ圧縮の評価データとして標準的に使われている Silesia コーパス[5]を用いて、開発方式の性能評価を行った。但し、従来技術との比較のため、LZO[6]が圧縮できない x-ray のデータは、評価データから除外した。処理速度は、CPU に Intel Core i7-3770 3.4GHz、メモリに PC10600 の DDR3 8GB を搭載した PC で計測した。結果を図 5 に示す。

評価の結果、開発方式は、圧縮率 1/2、圧縮速度 280 メガバイト／秒、復元速度 1600 メガバイト／秒であった。一方、ZLIB は、開発方式よりも圧縮率が 1/3 と高いが、圧縮速度、復元速度ともに大幅に低い。LZO は、圧縮速度が約 2 倍高速だが、復元速度は 1/2、圧縮率も低い。

この結果、開発方式により、従来方式では実現できなかった、圧縮率 1/2 以下、圧縮速度 256 メガバイト／秒以上、復元速度目標値 1200 メガバイト／秒以上を実現し、ギガビットイーサでの転送速度および SATA3 での読み出し速度を 2 倍にできることが分かった。



圧縮方式	圧縮率
ZLIB	0.34
LZO	0.52
開発方式	0.50

(a) 圧縮率　　(b) 処理速度

図 5　圧縮性能

5　むすび

データ圧縮技術を用いてデータ量を減らすことで、ギガビットイーサおよび SATA3 の転送速度を、体感的に効果のある 2 倍にすることを目指し、高速な圧縮方式を開発した。LZ77 型圧縮をベースに、繰返し探索における比較候補削減、バイト単位の符号、多バイトコピー命令の活用による高速化を行うことで、2 倍速で圧縮率 1/2 のデータ圧縮方式を実現した。

今後の課題は、高速処理を保ったまま、圧縮率を 1/3 にすることである。ネットワークやストレージシステムへの適用[7]と併せて検討する。

6　参考文献

[1] http://www.gzip.org/zlib/
[2] J. Ziv and A. Lempel, “A Universal Algorithm for Sequential Data Compression,” IEEE Transactions on Information Theory, Vol. 23, no. 3, pp. 337–343, 1977.
[3] E. Fiala and D. Greene, “Data Compression with Finite Windows,” Communications of the ACM, Vo. 32, no. 4, pp. 490–505, 1989.
[4] 井谷宣子,小田切淳一,吉田茂, “高速な LZ77 型圧縮アルゴリズム”, FIT2004, A-036
[5] http://sun.aei.polsl.pl/~sdeor/index.php?page=silesia
[6] http://www.oberhumer.com/opensource/lzo/
[7] 亀山裕亮, 佐沢真一, 橋間正芳, “多様な通信環境に適用可能なデータ転送高速化技術", DICOMO2014