

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。

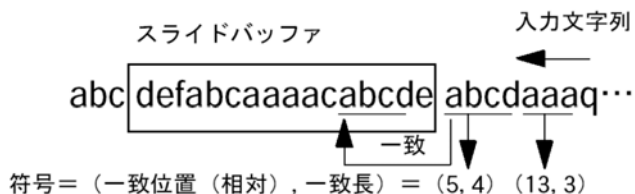


図1 LZ77 型圧縮

例えば、図1の入力文字列「abcdaaaq…」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。

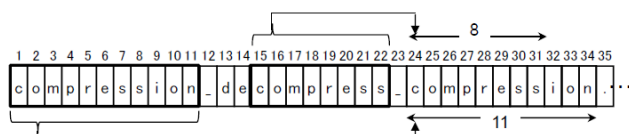


図2 複数回出現する繰返し

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。

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倍にすることを旨とし、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

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	{非繰返しのバイト列}		
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、LZ0^[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。LZ0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。

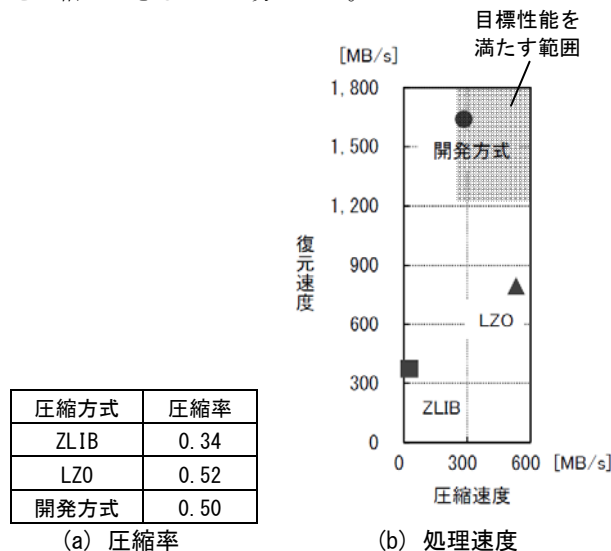


図5 圧縮性能

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倍にすることを旨とし、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, Vol. 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