A-036

高速な LZ77 型圧縮アルゴリズム

A Fast LZ77 Data Compression Algorithm

井谷宣子†　小田切淳一†　吉田茂†
Noriko Itani　Junichi Odagiri　Shigeru Yoshida

１．まえがき

携帯電話、PDA、情報家電などの機器では、予め外部で作成したデータを転送・格納しておき、復元して使用することが多い。この用途では、完全復元型データ圧縮が使われ、特に復元性能（速度・メモリ量）が重視される。LZ77 型圧縮が最も向いているが、圧縮処理が遅いという課題があった。本稿では、LZ77 型圧縮において、圧縮処理の大半を占めるデータ列の繰返し探索について検討した。また、高速復元向きなシンプルな符号構成を併用して、高速な圧縮・復元処理結果が得られたので報告する。

２．従来技術と狙い

ソフトウェアやデータの容量圧縮に用いられる完全復元型圧縮には、確率統計型圧縮と辞書型圧縮の２方式がある。中でも辞書型圧縮に分類される LZ77 型圧縮は、復元が早く、ZLIB や GZIP, LHA に利用されている。LZ77 型圧縮は、繰返しを出現位置と長さに置き換えることによってデータ量を圧縮する。ZLIB や GZIP, LHA は、この一致位置と一致長を、さらに可変長符号で符号化して圧縮率をあげている。

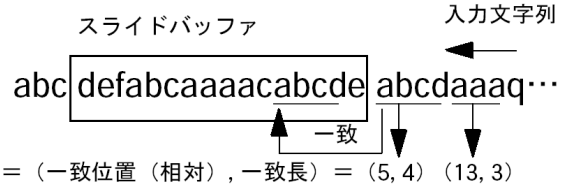


図１　LZ77型圧縮

LZ77 型圧縮は、圧縮符号自身が繰返し位置を示し、単純な複写で復元できるため、高速に復元できる。最も処理負荷が高いのは、圧縮時の繰返し探索で、実装上の一番の課題となる。従来では繰返しの先頭部分を探索テーブルに随時登録し、先頭が一致した位置を軸に一致を求めることによって高速化している。この探索テーブルには、ハッシュ表やトライ木などが使われている。どちらも復元処理の１０倍の時間がかかる。

本稿では、探索テーブルを随時更新するのではなく、最初に一括で作成する方法で、圧縮処理の高速化を図った。また、可変長符号を使用せず、固定長符号で高速な復元処理を保ちつつ、圧縮率向上を図った。

３．技術開発

3.1　最近出現位置テーブルの一括生成

入力バッファにおける各アドレスに対して、最も最近同じ３バイト列が出現した位置を格納した探索テーブルを最初に一括で作成する。この探索テーブルを「最近出現位置テーブル」と呼ぶ。

†（株）富士通研究所　ペリフェラルシステム研究所

最近出現位置テーブルは、次に示す２段階の処理を経て生成する。

(1)　３バイト列をコード順に並び替える

入力バッファから得られる全ての３バイト列をコード順に並びかえる。例えば図では、アドレス１「com」、アドレス２「omp」、アドレス３「mpr」… をコード順に並びかえる。並びかえは、通常のソート法でよいが、ここでは一番シンプルに構成できるラディックスソート（Radix sort）を用いる。

(2)　同じ３バイト列の１つ前の出現位置を求める

(1)で求めたリストについて頭から順に、隣接して格納している２つの３バイト列を比較し、最近出現位置を求める。例えば図では、「com」の出現位置が１、１５、２４と並び、アドレス１５「com」の一つ前の出現位置が１（相対位置は１５－１＝１４）、アドレス２４「com」の一つ前の出現位置が１５（相対位置は９）であることが求められる。

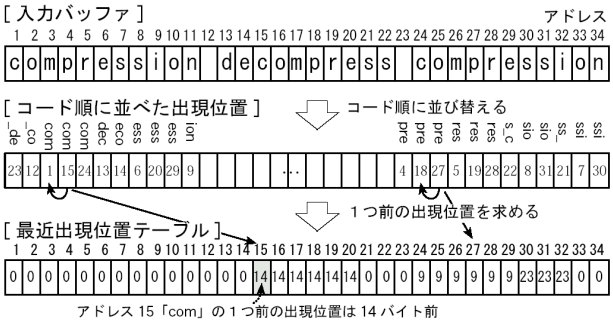


図２　最近出現位置テーブルの生成

3.2　最近出現位置テーブルによる繰返し探索

先に求めた最近出現位置テーブルを用いて、より長い繰返しを探すが、候補が増えると、より長い繰返しになるか否かの照合に時間がかかる。そこで、同じ接頭語を持つ全候補に対して照合する方法だけでなく、候補を絞って照合する方法も検討した。

3.2.1　全照合探索

符号化するバイト列の先頭３バイト列が過去に出現した位置を順に一つ一つ遡って比較する。最近出現位置テーブルを用いて、符号化位置から始まる３バイト列の一つ前の出現位置、さらにその一つ前の位置と前方に辿ると、同じ３バイト列が過去に出現した位置を全て求めることができる。こうして得られた候補から最も長く一致する位置を探す。照合においては、毎回４バイト目から照合するのではなく、従来に倣ってまず最初にそれまでの最長値＋１バイト目を照合する。

図３　全照合探索

3.2.2　選別照合探索

符号化位置から始まるバイト列と最も長く一致するバイト列の中にそのバイト列以外では現れない特徴的な３バイト列がある場合、その３バイト列を軸に探索すると候補を絞ることができる。この特徴を利用して符号化位置の３バイト列、符号化位置＋１の３バイト列、符号化位置＋２の３バイト列、…、符号化位置＋Nの３バイト列の最近出現位置を候補とし、一致の照合を行なう。照合においては、候補の位置がそれまでの最長値となった位置と比較して過去である場合に絞って照合する。

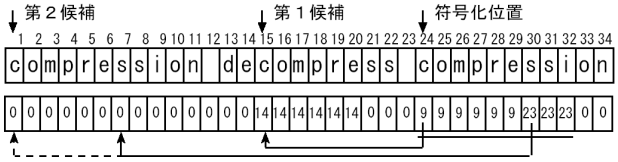


図４　選別照合探索

3.3　シンプルな符号構成

繰返し検索の高速化に加えて、LZ77型圧縮の高速復元の特徴を生かすため、シンプルな符号をベースに圧縮率を改善することを検討した。基本構成として、シンプルな１ビットフラグ符号、８ビット単位の固定長符号を用いた。従来に倣い、一致長に４ビット、一致位置に１２ビット割当てる。

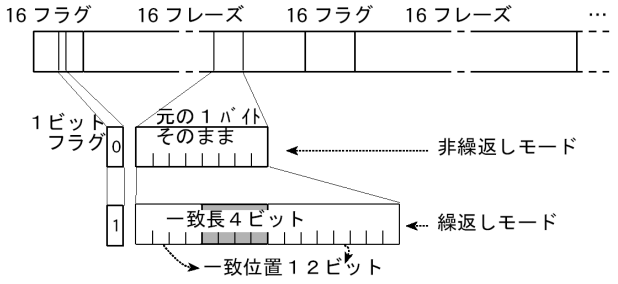


図５　固定長符号

圧縮率改善のため、基本構成をベースに、表現可能な一致長を長くするように拡張する。一致長に割当てたビットで表現可能な最大値（例えば４ビットの場合には１５）に達した場合、次に続く８ビットを一致長として用いる。拡張した８ビットも同様に表現可能な最大値（２５５）の場合は、さらに次に続く８ビットも一致長として用いる。残りの一致長が一致長拡張の値より小さくなるまで付加を行なう。一致長拡張導入にあわせて、基本構成のビット割当てを見直し、一致長に３ビット、一致位置に１３ビットを割当て、より広い領域で候補を得るようにする。

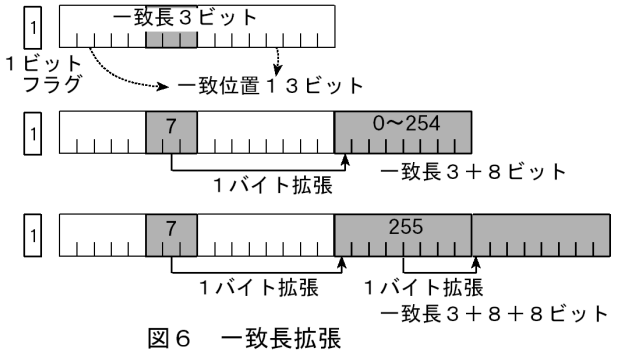


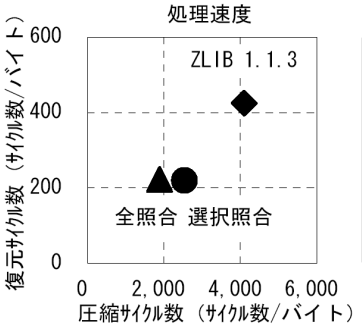
図６　一致長拡張

４．高速な圧縮・復元処理

最近一致位置テーブルについて、まず基本構成の符号を用い、全照合探索と選別照合探索とで速度と圧縮率を評価した。選別照合の方が３割早いが４％圧縮率が劣化した。復元では探索方法に関係なく同じ複写処理であるため、どちらも同じ速度となった。また、全照合探索について、基本構成の符号に一致長拡張を導入した場合の圧縮率を求めた。一致長拡張を導入する前と比べ７％圧縮率が改善された。

ZLIB と比較すると、どちらの探索方法でも、圧縮・復元処理共に ZLIB の半分のサイクル数（時間）となった。圧縮率については、一致長拡張を導入したものでも ZLIB に７％届かなかった。

ZLIB と本方式との圧縮速度差には、繰返し探索方法の違いの他に、可変長符号と固定長符号の違いと、4 ビット以上長い繰返しの探索の有無の違いが含まれている。ZLIB の圧縮処理の半分がこの処理が占めると見積もって、純粋に探索速度同士を比較した場合は、ZLIB で用いているハッシュ探索と同等の速度で探索できる。ZLIB と本方式との復元速度差については、可変長符号と固定長符号の差のみである。



圧縮率

圧縮後サイズ／元サイズ	
全照合	46.1%
選択照合	50.2%
全照合＋一致長拡張	39.5%
ZLIB 1.1.3	32.8%

図７　性能評価（calgary_corpusデータ）
PentiumⅢ 800MHz, Windows2000

５．むすび

LZ77 型圧縮において、データ列の繰返し探索に用いる探索テーブルを最初に一括で求める方法で、圧縮処理を高速化した。高速復元向きな固定長符号を併用し、ZLIB と比べて半分の圧縮・復元処理時間を得た。

今後の課題は、高速復元の特徴を保ったまま、圧縮率を改善することである。

６．参考文献

[1]Ross N. Williams,“An Extremely Fast ZIV-Lempel Data Compression Algorithm”, Data Compression Conf., 1991, pp.362-371

[2]http://www.gzip.org/zlib/

[3]奥村晴彦,“圧縮入門 Lharc と新 LH の作成”, CMAGAZINE, January 1991, pp.44-68

[4]福島荘之介,“ファイル圧縮ツール gzip のアルゴリズム”, bit, March 1996 Vol.28 No.3, pp.30-37

[5]山崎敏, 奥村晴彦,“LHA と ZIP－圧縮アルゴリズム×プログラミング入門”, December 2003, ISBN4797324287