マルチメディア通信と分散処理 70−26
（1995. 5. 26）

UNIX分散環境における可逆圧縮技術自動選択法に関する研究

小山 高明

慶應義塾大学 大学院 政策・メディア研究科

マルチメディア化やコンピュータネットワークの普及で、音声・画像データなど我々の扱うデータのサイズが巨大化し、ハードウェアの性能向上だけでは回避できなくなってきた。

この問題を解決させるため、様々な圧縮技術の開発とそれら技術のネットワークやファイルシステムへの実装が行なわれている。しかし、各圧縮技術がファイルの種類によって圧縮性能に偏りがあるにも関わらず、圧縮技術を実装した既存のシステムは、単一圧縮技術を利用したものやユーザーのコマンドによる手動切替え型、また圧縮性能を一定と考えて他の要因から圧縮技術を選択するものだけであり、圧縮する各データによって圧縮技術の性能が変化することを考慮に入れて選択するものはない。

本稿では、まず、UNIX分散環境における問題点、可逆圧縮技術と圧縮技術を採り入れた既存システムについての問題性を述べ、データ毎の各圧縮技術の性能予測方法とUNIX分散環境における圧縮性能による自動選択法を設計し、評価、検討した。

A research on an automatic selection of lossless compression algorithms in the UNIX distributed environment

Takaaki Koyama

Graduate School of Media and Governance, Keio University
5322, Endo, Fujisawa-shi, Kanagawa, 252 Japan
geige@mag.keio.ac.jp

The size of disk space we need is increasing and our necessary network transfer rate is high. Because many people have communications in computer network, and we have or send not only text files but also music and video files. So our demand is not satisfied by only the improvement of computer hardware.

To solve this problem, various compression algorithms were designed and various systems with those algorithms are released. Although the compression ratio and the compression speed of each compression algorithm is not fixed, all systems have a fixed table of the ratio and the speed.

In this paper, at first the issues of distributed environment, lossless compression algorithms and the system with compression algorithms are discussed. Then the automatic selection for lossless compression algorithms with the forecast of compression ratio and compression speed are designed.

1 はじめに

最近のマルチメディア化やネットワーク化は、ファイル使用量の膨大な増加と、ネットワーク上でのデータ転送量やアクセス回数の増加を促し、この増加量が、ハードディスクなどの2次記憶装置の供給量や、ATM LANといった新しいネットワークハードウェアの開発による転送能力の増加分を上回っている為、現在、2次記憶装置の不足やネットワークの転送時間の増大が問題となっている。この問題を解決させるために、単一圧縮技術を採り入れたシステムや複数圧縮技術手動切替型のシステム、また音声・画像通信においては、通信状況に応じて複数の非可逆圧縮技術から自動選択するシステムなどが開発されているが、どの圧縮技術にも圧縮するデータによって向き不向きが存在するため、データ毎には圧縮技術の性能を考慮していないこれらのシステムでは、各圧縮技術の圧縮性能を十分に発揮できない。

しかし、データ上の1byte文字の分布にはファイルの種類毎に偏りがあり、ファイルの種類によって各圧縮技術の圧縮性能も偏るため、圧縮したいデータからファイルの種類をわりだすことができれば、各圧縮技術の圧縮性能を予測可能である。

よって、圧縮技術の圧縮性能向上を考えれば、単一圧縮技術や、通信状況に応じた圧縮技術自動選択より、圧縮するデータ毎にファイルの種類から予測した各圧縮技術の圧縮性能も選択基準に加える、複数圧縮技術自動選択法の方が有効といえる。

本稿では、可逆圧縮技術にしぼって、現状と問題点、また、圧縮するデータ毎に各圧縮技術の圧縮性能を考えた複数可逆圧縮技術自動選択方法について提案し、模擬実験による評価も行なう。

2 現状と問題点

ここでは、圧縮技術を用いていないシステムの限界、既存の可逆圧縮技術の抱える問題点、そして、圧縮技術を採り入れた既存システムの問題性について述べる。

2.1 圧縮技術を用いていないシステムの限界

ネットワーク化とマルチメディア化の促進は、計算機資源の浪費を促し、ネットワークハードウェアの転送能力の限界と2次記憶装置の不足、という問題を引き起こしている。

2.1.1 ネットワークハードウェアの転送能力の限界

ファイル転送量は、ネットワークハードウェアの転送能力を越えて増加している。ここ10年間でネットワークハードウェアはEtherからATMへと発展したが、物理層レベルでの転送能力で考えると、Etherの転送速度が10Mbps、ATMの転送速度がOC3で155Mbpsであり、15倍しか伸びていない。無論、UDP/IPレベルやマシンのオーバーヘッドによっては転送速度はさらに落ちる。

しかしその間にインターネット社会では、電子メール、電子ニュース、ftp、NFSを初め、nv、vatによる音声画像通信、WWWによる音声・画像ファイルの転送までネットワークを利用するようになった。転送するデータ量を考えると、メールテキストサイズが4Kbyte程度*に対し、音声ファイルサイズは8bit周波数8 KHzによる1分間のサンプリングで480Kbyteになる。これだけ見ても、48倍増加する計算になる。

さらに、ネットワークの普及やWWWの登場によりネットワークアクセス回数も増加する一方であり、現状のネットワークハードウェアの転送能力ではすでに限界にきている。

2.1.2 2次記憶装置の不足

ファイルサイズの巨大化により、2次記憶装置は不足している。慶應義塾大学藤沢キャンパスを例にとると、現在、1.8Gbyteのディスク1台につき学生135名分のホームディレクトリを割り振り、ディスク使用量は全ディスク空間の70%に達している。よって、学生135名分に残されたディスク空間は1.8Gbyteの3割分、すなわち540Mbyteだけである。これは、学生1人当たり平均4Mbyteしか残っていない。

この原因も、前項で述べた内容と同じく、WWW等のシステムによる画像情報通信の普及により、各自でいくつも画像データなどの大きなサイズのファイルを保持するようになったためである。

2.2 可逆圧縮技術の抱える問題点

可逆圧縮技術の主なものは、ハフマン符号[2]、算術圧縮[7][1]、LZ78[9]を母体としたLZW[6]や、LZ77[8]を母体としたLZSSなどであり、一番性能の良い圧縮技術の圧縮率は平均50%以上を達成していると言われている。

しかし、表1†でわかるように、圧縮技術には、圧縮性能がファイルの種類別に偏っていることと、圧縮したデータを再度同じ圧縮技術で圧縮しようとしても広がってしまう圧縮限界という2つの問題点が存在する。よって、場合によっては圧縮性能が平均圧縮率を下回ることもあり得る。

圧縮性能がファイルの種類別に偏るのは、ファイルの種類毎に使用する文字コードと使用するデータフォー

*3.1.1項参照

†Mark Nelson著のData Compression Book[3]上に記載のプログラムを利用

マットが決っているため、ファイルの種類毎に1byte文字の散らばりに偏りがあり、現在の可逆圧縮技術がこの1byte文字の散らばり、すなわち、1byte文字の出現頻度や出現サイクルに応じて統計的モデル化や辞書の構築を行なうからである。

2.3 圧縮技術を採り入れた既存システムの問題性

計算機資源の浪費を回避するため、圧縮技術を採り入れたファイルシステムやネットワークプロトコルが開発されている。しかし現時点では、圧縮ファイルシステムであるStackerやSunのPowerDrive[5]を初め、圧縮技術を採り入れたほとんどのシステムで、1つの圧縮技術しか使用していない。また、複数の圧縮技術を採り入れたシステムでも、手動切替え型[‡]か、ネットワークプロトコルなどでは、実際に圧縮したいファイルやデータとは関係なく、予め各圧縮技術の平均圧縮率や平均圧縮速度などの圧縮性能を計測しておき、その計測結果を圧縮性能の尺度として圧縮技術選択を行なうものしかない。よって、データ毎に圧縮技術の性能が変化するにも関わらず、このことを考慮に入れたシステムは1つもない。

3 自動選択法の設計

今回提案する自動選択法は、圧縮技術の性能という尺度を考慮した自動選択法である。よって、圧縮したいデータ毎に各圧縮技術の性能を予測し、その予測性能と他の要求事項を考慮して、圧縮技術を選択する。

3.1 データ毎の各圧縮技術の性能予測方法

ここで予測する圧縮技術の性能とは、次に示す4点のことである。

- 圧縮率(%)…$(1 - \frac{\text{圧縮データサイズ}}{\text{元データサイズ}}) * 100$
- 圧縮時間…圧縮するデータ1byte当たりの圧縮にかかる平均時間
- 解凍時間…解凍されたデータ1byte当たりの解凍にかかった平均時間
- メモリ消費量…圧縮を実行するのに必要なメモリ量

2.2項の通り、同じファイルの種類である1byte文字のばらつきの近いファイルをいくつ圧縮しても、各圧縮技術の圧縮性能は変わらない。

よって、1byte文字のばらつき度合の似ているファイルの集合を「ファイルの種類」と定義し、「ファイルの種類」で各圧縮技術の圧縮性能が同じもの同士を1つにまとめてファイル特性と定義した。

そして、定義されたファイル特性別に各圧縮技術の圧縮性能表を予め作成し、圧縮するデータからファイル特性を予測することで、各圧縮技術の圧縮性能を予測することにした。

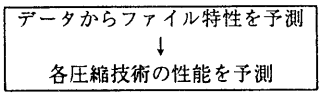

図1: 各圧縮技術の性能予測

3.1.1 ファイル特性予測方法

ファイル特性予測方法は、速くて正確でメモリ消費量も少なく、圧縮するデータそのものを使って予測しなくてはいけないことが条件となる。

現在、ファイル特性予測方法としては、以下の3つが考えられるが、本ファイル特性予測方法では、この3つのファイル特性予測方法を複合利用することで、より正確なものとした。

1. キーワードや指定されたマークの抽出による方法
2. データサイズによる方法
3. 静的ハフマン符号を用いた方法

よって、これら3つの方法の説明と検討、そして、本自動選択における複合利用法について説明する。

キーワードや指定されたマークの抽出による方法

キーワードや指定されたマークによって、ファイル特性を判別する方法であり、/etc/magicファイルなどを利用するfileコマンドやnkfコマンドなどがこれにあたる。fileコマンドは、ファイルの先頭3byteにつけられたマークを利用するため、メモリの浪費も少なく処理速度も速い。しかし、これら抽出法の問題点は、マークを抽出するだけであり勘違いしやすいことと、全く新しいファイルについては、判別不能に陥りやすい点である。また、もしも間違えてしまった場合の対処ができない。

失敗することを回避する策としては、他にチャック機能を置くことである。

データサイズによる方法

ホームディレクトリなど、あるディレクトリ以下に存在するすべてのファイルサイズを調べると、ファイル特性毎にあるサイズに近いことが分かる。

[‡]Linux上の圧縮ファイルシステムDouble File Systemでは、5つの圧縮技術からコマンドによる手動で切替えができる

例えば、ある人のホームディレクトリ以下に存在するファイルを調べた所、シェルスクリプト、TeX のファイル、アスキーテキスト、メールテキスト、GIF ファイルが平均 4Kbyte であり、ニューステキストが平均 6Kbyte、C 言語のソースファイルが平均 6Kbyte、DVI ファイルが平均 9Kbyte、ポストスクリプトファイルが平均 81Kbyte、gzip の圧縮したファイルが平均 200Kbyte、TIFF ファイルが 5732Kbyte であった。

特に、メールなどのテキストファイルなどはほとんどが 4Kbyte であり、ファイルサイズからファイルの種類も、ある程度判別できることがわかる。

静的ハフマン符号を用いた方法

2.2 項でも述べた通り、ファイルの種類は 1byte 文字のばらつきと深い関係がある。そこで、ファイルの先頭 2.5Kbyte をとってきて、それを静的ハフマン符号化で圧縮し、その結果得られる圧縮率でファイル特性 (文字のばらつき性) を予測する方法を考案した。この方法が可能である理由を以下に述べる。

1. 静的ハフマン符号化を使用する理由

   ここで静的ハフマン符号化を使用する理由は、この方法が以下の 2 点を満足するからである。

   (a) 静的モデル化により、1 つのファイルの完全なモデル化の可能。

   (b) ハフマン符号のオーダー値 0 の統計的モデル化により、1byte 文字の出現確率に応じた圧縮率を出すことができる。

   例として、以下の図 2のようにファイル特性を定めた時に、各ファイル特性毎に、すべてのファイルの先頭 2.5Kbyte をそれぞれ静的ハフマン符号化で圧縮し、その圧縮率とそのファイルの数のグラフの一例を図 3,4,5に示す。

- 圧縮ファイル (GIFF、gzip ファイル、compress ファイル)
- バイナリファイル (実行可能ファイルすべて)
- テキストファイル (character で表現されているものすべて)
- 音声画像 raw data ファイル
- その他

図 2: ファイル特性の 1 例

これらのグラフから、先頭 2.5Kbyte を圧縮するだけでも、圧縮率に偏りがあることが分かる。実行ファイルの場合は 20%を中心として一つの山に、またテキストファイルの場合は、20%と 34%の 2 つを頂点とした双子の山ができている。ちなみに、圧縮ファイルについては、ほとんどが-2%で、それ以外は皆-2%以下であった。

また、おもしろいことに、実行ファイルでも、BSD/386 1.1(図 4参照) と Sun OS 4.1.3 はグラフが似た形になったが、Linux1.1.73(図 5参照) は違う形であった。これも、ファイル形式の違いから起きたものと考えられる。

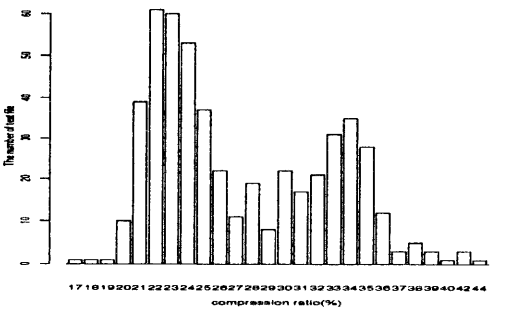

図 3: テキストファイルの先頭 2.5Kbyte を静的ハフマン符号化で圧縮

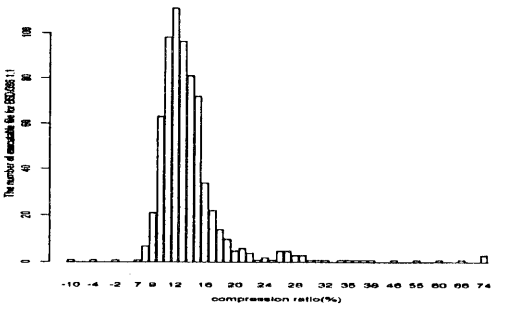

図 4: BSD/386 1.1 用の実行ファイルの先頭 2.5Kbyte を静的ハフマン符号化で圧縮

2. 先頭 2.5Kbyte を使用できる理由

   図 345のグラフの通り圧縮率に偏りがあるため、使用可能である。よって、ここでは時間的コストメモリなどの物理的コストについて述べる。

   (a) 時間的コスト

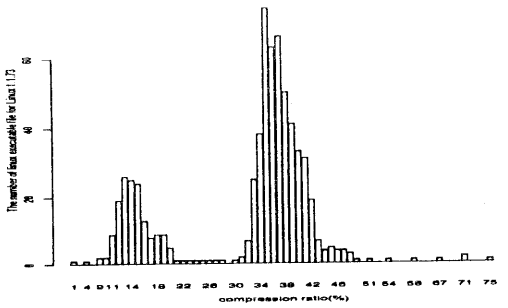

図 5: Linux 用の実行ファイルの先頭 2.5Kbyte を静的ハフマン符号化で圧縮

データの先頭 2.5Kbyte 分を read してから静的ハフマン符号により圧縮し write はせずに圧縮率だけ表示させるプログラムを作り、IBM PC AT 互換機†を用いて、その local の SCSI ディスク上にあるファイルを使って計測した。それによると、read 時間も含めて平均 0.22 秒であり、特に圧縮時間のかかる gzip で圧縮したファイルに対する場合でも、0.4 秒以上するものはなかった。また、file コマンドも同じファイルに対して行なうと、平均 0.20 秒であり、よって、ハフマン符号による圧縮自体は、大した時間差ではないことが分かる。

(b) 物理的コスト

物理的コストとは、主にメモリ使用量のことである。

この方法のメモリ使用量についてだが、きちっとしたメモリ使用量の平均値は計測できなかった。しかし、上記と同じマシンで gzip ファイルに対して行なった場合、18Kbyte であり、物理的コストも大した量ではないことが分かる。

3.1.2 複合利用法

本自動選択法のファイル特性予測方法は、以上の 3 つの方法を複合的に使用する。方法は、図 6の流れ図の通りである。

まず、キーワード抽出による方法を行ない、確認のためデータのサイズによる方法も利用する。もしも、キーワード抽出による方法で判断不能、または、サイズによる方法との結果、相違が生じた場合は、静的ハフマン符号化の圧縮を用いて割りだす。

†Intel i486DX2/66MHz を搭載した IBM PC AT 互換機, 主記憶 20Mbyte,OS は Linux 1.2.5 で計測

```
キーワード抽出による分析;
データサイズから分析;

if((キーワード抽出による分析結果 == 判断不能) ||
    (データサイズからの分析結果 != キーワード抽出
による分析結果)){
    静的ハフマン符号化による分析;
      if(静的ハフマン符号化による分析結果 == デー
タサイズからの分析結果){
        return 静的ハフマン符号化による分析結果;
      }else
        return キーワード抽出による分析結果;
}else{
return キーワード抽出による分析結果;
}

if(a == ''data''){return c;
}else
if(b == c){ return c;
}else return a;
```

図 6: 複合利用法

3.2 可逆圧縮技術決定方法

圧縮技術の圧縮技術を選ぶ際に重要な判断条件は、圧縮技術の圧縮性能と、各システムの要求事項である。よって、各システムの要求事項に沿いながら、本自動選択法に基づくファイル特性分析で得られた各圧縮技術の圧縮性能分析を用いることで、より確実な圧縮実行が可能となる。

4 評価

本自動選択法を用いた最大圧縮コマンドを作り、実際に使用してみた。

このコマンドは、圧縮性能表 1を考慮して、GIF JPEG gzip compress などのファイルを圧縮ファイル系、ascii テキスト、メール、ニュース english テキスト、C 言語ソースファイルとシェルスクリプトファイルをテキストファイル系、TIFF PostScript ファイルをグラフィックスファイル系、残りすべてを実行ファイル系というファイル特性にし、テキストファイル系なら動的算術圧縮オーダー値 3 と実行ファイル系なら gzip -9、グラフィックスファイル系なら算術圧縮オーダー値 1 を使用する。ファイル特性分析は、file コマンド Version 3.13 を利用して、図 6の流れのように、実行した。

ある一人のホームディレクトリのファイルすべてを圧縮した実行結果は、表2の通りである。

この結果からもわかる通り圧縮率は向上する。

しかし今回は、圧縮性能表にメモリ消費量について調査しなかった。また、ファイル特性の定義において、「ファイルの全種類」を file コマンドで判別できる全種類と定義した後、圧縮率がだいたい近い4つに分けてしまった。この定義の段階で圧縮時間についても考慮する必要がある。

また、file コマンドでは間違えてしまうものに対してどう対処すべきか、と今回は実行ファイルとした tar ファイルについての圧縮法選択はどうしたら良いかなどは、今後の研究課題とする。

そして、さらに新しいファイルの種類であるデータ形式が見つかった場合は、速やかにファイル特性への分類と追加を行ない、ファイル特性の予測方法においても、file コマンド以外にもう一つ、チェック機能を持たせる予定である。

表 2: 圧縮実行結果

圧縮コマンド	総 1Kbyte ブロック数	圧縮実行後の 1Kbyte ブロック数	圧縮率
圧縮法自動選択	43688	27383	36.51%
LZSS のみ	43688	30876	29.32%
gzip -9 のみ	43688	28401	34.9%

5 まとめと今後の課題

まず、現状とその問題点として、計算機資源の浪費によるシステムの限界、可逆圧縮技術の性質、圧縮技術を採り入れた既存システムの圧縮技術の非有効利用性について述べ、有効的な圧縮技術利用のための複数可逆圧縮自動選択方法として、ファイル特性予測法による圧縮技術性能予測方法を提案し、評価を行ない、この方法の有効性を述べた。

しかし、実際の評価の所で、ファイル特性の定義に圧縮時間による分類と圧縮中のメモリ消費量についての分類は行なえなかったため、圧縮率の計測だけに留まった。また、tar ファイルなどに対する対処をどうすべきかは、今後の課題とする。

また、今後、実際にネットワークプロトコルやファイルシステムへの実装を行ない、その性能を評価したい。

参考文献

[1] Fiala, Edward R., and Greene, Daniel H., "Data compression with finite windows, " *Communications of the ACM*, Volume 32, Number 4, April 1989, pp 490-505

[2] Huffman, D. A., "A method for the construction of minimum-redundancy codes, " *Proceedings of the IRE*, Volume 40, Number 9, Sep. 1952, pp 1098-1101

[3] Nelson, Mark, The Data Compression Book, M&T Publishing, Inc., 1992

[4] Storer, James A., IMAGE AND TEXT COMPRESSION, Kluwer Academic Publishers, 1992

[5] Taylor, Dave, "PowerDrive, " *SunWorld*, May 1993, pp 94-99

[6] Welch, Terry, "A Technique for High-Performance Data Compression, " *IEEE Computer*, Volume 17, Number 6, June 1984, pp 8-19

[7] Witten, Ian H., Neal, Radford M., and Cleary, John G., "Arithmetic Coding for Data Compression, " *Communications of the ACM*, Volume 30, Number 6, June 1987, pp 520-540

[8] Ziv, J., and Lempel, A., "A universal algorithm for sequential data compression, " *IEEE Transactions on Information Theory*, Volume 23, Number 3, May 1977, pp 337-343

[9] Ziv, J., and Lempel, A., " A., Compression of individual sequences via variable-rate coding, " *IEEE Transactions on Information Theory*, Volume 24, Number 5, Sep. 1978, pp 530-536

[10] 小山高明, "圧縮ファイルシステム (CFS) における圧縮アルゴリズム自動選択機構," 1993 年度徳田・村井研究室 卒業論文, Feb. 28, 1994

表 1: 各圧縮技術の圧縮率 単位は%

ファイルの種類	静的算術	動的ハフマン	動的算術オーダー1	動的算術オーダー2	動的算術オーダー3	lzss	gzip -1	gzip -6	gzip -9
Bourne Shell script	15	21	42	44	49	41	49	52	53
C Shell script	30	31	43	45	54	43	48	50	55
English text	25	27	36	37	37	19	33	36	36
English text with escape sequeces	30	30	49	60	61	46	55	58	60
FIG file	49	51	61	58	57	54	57	59	60
GIF file	-21	-9	-12	-13	-12	-8	-3	1	2
JPEG file	0	0	-22	-22	-22	-10	-2	1	2
PostScript documment	36	36	71	76	75	62	65	68	70
PostScript File	42	42	61	73	75	69	65	68	69
TIFF File	34	36	47	44	36	34	51	58	59
DVI	19	20	44	53	53	42	51	59	59
ascii text	23	31	59	68	71	52	63	65	70
ascii text with escape sequences	22	27	43	54	54	38	50	52	52
c program text	26	29	61	73	73	52	62	69	70
gzip file	0	0	-25	-9	-8	-11	-10	-11	-15
mail text	28	26	40	40	45	35	41	44	45
mail text with escape sequences	21	21	34	42	42	29	35	39	40
news text	4	22	31	33	33	16	30	30	32
smtp mail text	23	24	35	44	44	30	35	41	40
BSD/386 1.1 executable file	25	26	44	47	50	42	44	48	66