2C-3

ブロックソート法を用いた 圧縮ループバックブロックデバイスの実装と評価

北川 健司† 丹 英之† 千葉 大作† 須崎 有康‡ 飯島 賢吾‡ 八木 豊志樹‡
株式会社 アルファシステムズ† 独立行政法人 産業技術総合研究所‡

■ はじめに

KNOPPIX[1]に代表されるライブ CD Linux(以下KNOPPIX)では,インストール無しで即座にPCを使用することができる.HDDへは全く影響を与えないことから,普段は別のOSがインストールされているPCでの,Linuxとの併用が安易に実現出来る.このようなライブCDでは,ファイル読み込み時におけるシーク時間の低減,そしてメディア容量を超えるサイズのファイルを格納するため,圧縮ループバックブロックデバイス(cloop)上に構築されたファイルシステムを用いている.しかし,現在の実装であるzlib[2]を用いたLZ77による圧縮では,更なるアプリケーションの追加を望むユーザにとって最善の方法であるとは言い難い.そこで,一般的にLZ法よりも圧縮率が高いとされる libbzip2[3]を用いたブロックソート法での圧縮ループバックブロックデバイスを実装した.本発表では,この実装,及びこれを組み込んだ KNOPPIX の評価について報告する.

■ cloop (Compressed Loopback Blockdevice) とは

ループバックブロックデバイス(loop)のドライバは,ファイルを HDD のようなブロックデバイスとして扱う機能を提供する.cloop とは,このブロックが圧縮されたものを扱えるように拡張されたループバックブロックデバイスのことである.

情報圧縮に於いて,ブロックサイズを大きくすると,圧縮率が向上することが経験的に分かっている.しかし,ブロックサイズが大きいほど伸張処理に時間を要するため,圧縮率と伸張時間の間にはトレードオフが存在する.

cloop では、現在ブロックの圧縮・伸張アルゴリズムにLZ77 を zlib(2.01-5 からは LZMA を AdvanceCOMP[4]の拡張)で実装している.この圧縮によって,KNOPPIXでは約 1.8GB のファイルを 700MB の CD メディアに収録している.

本研究では,限られた容量のメディアに更なるコンテンツを格納することを目的とするため,圧縮・伸張アルゴリズムに,一般的に zlib を用いた LZ77 よりも圧縮率の高いとされる libbzip2 を用いたブロックソート法に着目した.

Implement and estimate of a compression loopback block device using the block sorting method
† Alpha systems Inc.
‡ National Institute of Advanced Industrial Science and Technology (AIST)

■ 実装

libbzip2 を用いた圧縮・伸張アルゴリズムの実装は,libbzip2-1.0.2 を,カーネル空間で機能するように修正しcloop モジュールとリンクした.この cloop ドライバを産業技術総合研究所(AIST)で日本語化[5],及び公開[6]されている KNOPPIXv3.6 日本語版(knoppix_v3.6_20040816-20040914)へ kernel2.6.8.1 と共に組み込んだ.

KNOPPIX3.6 日本語版からは,cloop 内に格納されるファイルシステムが ISO9660 から ext2fs になった.そこで今回 CD を再構築する際に,まず 2,560MB の ext2fs イメージを用意し,そこにシステムを構成するファイルをコピーした.そして,そのイメージファイルを LZ 法とブロックソート法でそれぞれ圧縮を行った.

■ 実験

前項でのそれぞれの実装に対し,下記の作業についてCD からの読み込みにかかった時間,伸張にかかった時間を計測した.時間計測には,1ms (kernel2.6)のタイマー割り込みでインクリメントされる変数(jiffies)を利用した.

測定作業項目
①『KDE 起動』
KNOPPIX の起動から KDE が起動完了するまで
②『OpenOffice 起動』
CD に格納されている文書ファイルを OpenOffice で実行してから,表示完了するまで
③『mozilla 起動』
mozilla を実行してから,HP が表示完了するまで

KNOPPIX では,CD-ROM からの cloop ブロック参照に伴うシーク時間,ブロック読込時間,伸張時間を考慮し,ブロックサイズは 64KB となっている.一般的に情報圧縮では,データの大きさによって圧縮率が変化する.そのため,今回はブロックサイズの関係についても測定を行った.

実験では,CPU:PentiumIII 933MHz,RAM:256MB (PC100),CD-drive:24 倍速 の PC を使用した.

■ 評価

圧縮・伸張アルゴリズムと cloop ブロックサイズの変更に伴う読込時間,伸張時間,及びその合計時間の『KDE 起動』における測定結果を図 1 に示す.

また、それぞれの圧縮・伸張アルゴリズムとブロックサイズによる圧縮率を表 1 に示す。この値は,用意した

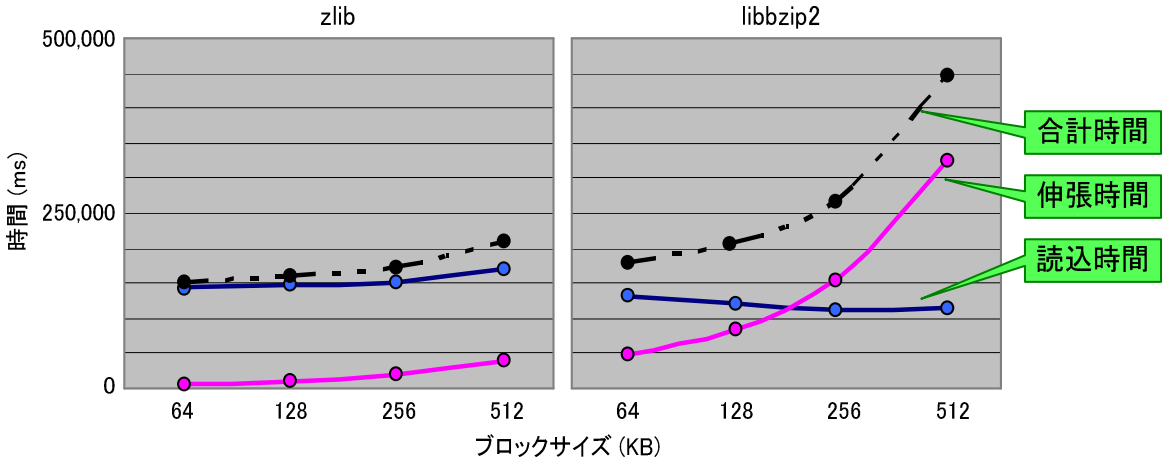

図１．圧縮･伸張アルゴリズムと cloop ブロックサイズの変化に伴う KDE の起動時間

表１．アルゴリズムとブロックサイズによる圧縮率

アルゴリズム	zlib				libbzip2			
ブロックサイズ (KB)	64	128	256	512	64	128	256	512
サイズ (MB)	669	661	657	655	646	628	614	604
圧縮率 (%)	36.46	36.02	35.81	35.70	35.20	34.24	33.48	32.89

ext2fs イメージサイズ(2,560MB)ではなく，システムを構成するファイルサイズ(ファイル数 93,485，合計サイズ 1,923MB)と圧縮後のイメージサイズから算出した．

LZ 法に対してブロックソート法では，圧縮率が向上しており，ブロックサイズ 512KB 時では，圧縮後のイメージサイズを約 51MB 削減することが出来た．また，圧縮率の向上により読込時間の短縮も見られた．しかし，それに相反して，伸張時間が増加しており，これがボトルネックとなったためにアプリケーション起動時間に影響を及ぼした．

この伸張時間の増加を緩和するため，LRU(Least Recently Used)方式のブロックキャッシュ 16MB を設けた．キャッシュにヒットしたブロックは伸張処理を必要としないため，ブロックサイズ 512KB 時ではキャッシュを利用しない場合に比べ、伸張時間を 45.73%(149,672ms)削減することに成功した．

■ まとめ

本論文では，ブロックソート法を用いた圧縮ループバックブロックデバイスの実装を行った．その結果，ファイルシステムを格納するメディアサイズの占有量を以前のものと比較して，最大 51MB (2.81%)削減(ブロックサイズ 512KB)することが出来た．これは，圧縮前の容量に換算すると約 150ＭＢに相当し，同じ容量のメディアを用いる場合，今までより多くのアプリケーションをインストールすることが出来ることになる．

一方，伸張時間はブロックサイズに応じて増加し，これが各アプリケーション起動のボトルネックとなった．しかし，ブロックキャッシュを設けることでその伸張時間を半分近く削減できたことも特筆すべき箇所である．

libbzip2 を用いたブロックソート法の伸張処理は CPU リソースを多く消費することから，CPU:Pentium4 3.2GHz(HT), RAM2.5GB(PC3300), CD-drive:48 倍速 の PC での実験も行った(キャッシュ無し)．その結果，伸張時間は最大 25%にまで短縮され，読込時間よりも伸張時間の方が短縮出来る割合が高かった．以上のことから，CPU 性能の更なる向上により，伸張時間には十分な速度が見込めると考えられる．

本論文の前半でも述べたが、圧縮率と伸張時間にはトレードオフの関係がある。そこで本研究のひとつの方向性として、使用率の高いアプリケーションやデータなどは従来の LZ 法による圧縮を行い、普段あまり利用しないものは圧縮率の高いブロックソート法による圧縮を行うことでユーザにとって満足のいく環境を提供できると考える。

今回、計測で用いたメディアは CD-ROM であったが，コンパクトフラッシュや USB メモリキーなどバイト当たりのコストが高いメディアを用いたシステム[7][8]に於いて、本研究は有用な方法であると考えられる．また，低帯域な通信環境で SFS-KNOPPIX[9]などのようなネットワーク先の cloop ファイルを参照する場合にも、通信帯域の細さを補完する有効な手段となる可能性が考えられる．

今後は，他のアルゴリズムやメディアについても検討していきたい．

参考文献 & URL

[1] Knopper,“Building a self-contained auto-configuring Linux system on an iso9660 filesystem”,2003
http://www.knopper.net/knoppix/
[2] zlib Home Site
http://www.gzip.org/zlib/
[3] The bzip2 and libbzip2 official home page
http://sources.redhat.com/bzip2/
[4] Advance Projects
http://advancemame.sourceforge.net/comp-readme.html
[5] 須崎,飯島, “デスクトップとしての日本語 KNOPPIX”, FIT2003,B-057,2003
[6] KNOPPIX_Japanese_edition
http://unit.aist.go.jp/itri/knoppix/index.html
[7] 小菅,丸山,“フラッシュメモリから起動する KNOPPIX”, FIT2004,N-026,2004
[8] USB-KNOPPIX
http://kserv.jec.ac.jp/news/release_2004-10-17.html
[9] SFS-KNOPPIX
http://unit.aist.go.jp/it/knoppix/sfs/