KNOPPIX

起動高速化適用マニュアル

2007/1/17 株式会社アルファシステムズ

著作権

2006 年 02 月 28 日 初版発行

著者:株式会社アルファシステムズ

発行:株式会社アルファシステムズ

Printed in Japan

更新履歴

年月日	更新内容
2006．02．28	・　初版
2007．01．17	・　第二版

用語定義

用語	意味
ライブ CD	CD もしくは DVD 一枚で提供されるオペレーティングシステムのことである．この文書では特に Linux システムを起動出来るものを指す.
KNOPPIX	CD のみでブート可能な Linux ディストリビューション．ドイツの Klaus Knopper 氏が Debian GNU/Linux をベースに開発している.
圧縮ループバックデバイス(cloop)	ブロックデバイスの圧縮イメージを扱うことが出来る仮想ブロックデバイスのことである.

表記について

◆UNIX のコマンド入力の指定は，下のような枠の中に記述しています．

例)

```
# cloopprofiler &
```

◆ファイル出力の例は，下のような枠の中に記述しています．

例)

```
cloop device : 0
total blocks : 36865
```

◆Linux の起動オプションの例は，次のような枠の中に記述しています．

例)

```
boot: kernel
```

◆ウィンドウの要素毎に，括る括弧を統一しています．

ウィンドウの要素	表記	表記例
ウィンドウのボタン	【】で括る	【Play】，【Next】
プルダウンメニュー，ポップアップメニューのボタン	『』で括る	『File』，『View』
選択ボタン	《》で括る	《block/cell》

◆マウスの操作は，以下のように記述しています．

左ボタンを押下　　→ 左クリック

左ボタンを二回連続で押下　　→ 左ダブルクリック

右ボタンを押下　　→ 右クリック

右ボタンを二回連続で押下　　→ 右ダブルクリック

◆ウィンドウの操作で，複数の操作を行う場合，以下のような箇条書きで示しています．

－操作 1 をします

－操作 2 をします

－操作 3 をします

目次

1.概要

この文書は,2005 年度上期オープンソースソフトウェア活用基盤整備事業「CD/DVD 起動 Linux の速度改善ドライバの開発」にて開発されたライブ CD 起動高速化ツール「Live CD Acceleration Tool kit(LCAT)」を利用して,既存のライブ CD の起動を高速化適用する際の手法を示すマニュアルです.

ここでは,ライブ CD の起動高速化手法の概要とライブ CD の起動高速化を行うツール LCAT の機能を簡単に説明します.

1.1 ライブ CD の起動速度問題

ライブ CD は一枚の CD 及び DVD-ROM から OS のシステム一式を立ち上げる事が出来るシステムであり、Linux や UNIX 系 OS の配布形態として近年注目を集めています.ライブ CD では CD-ROM 一枚にシステム環境が構築されており,起動の際は PC の主記憶領域とライブ CD の記憶領域のみを用いて実行されるので,他の OS がインストールされているマシンからでも簡単に Linux などのオープンソースオペレーティングシステムマシンとして利用することが出来ます.また,ライブ CD は書き込み不可能なメディアから起動するため,構築された環境がユーザの操作によって変更されることがありませんので,常に安定した環境を利用することが出来ます.これらに加えて,ライブ CD のメディアである CD-ROM は安価に大量生産が可能であることから広く配布を行うことに向いており,オープンソースソフトウェアの幅広い普及を実現するための基盤としての役割が期待されています.

ライブ CD はこれまで雑誌等に添付される試供版や,PC 不調などの緊急時用の臨時 OS として用いられることが大半でしたが,これらの特徴を生かし,高ロバスト性を要求される教育分野や,公衆向けのいわゆるキオスク端末などの用途に利用され始めています.また,様々な Linux ディストリビューションを元にしたライブ CD や,既存のライブ CD を元にした派生版の開発が進んでおり,ライブ CD は広く一般に認知されつつあります.

しかしながら,ライブ CD には起動時間がハードディスクドライブ(HDD)にインストールされた一般的な PC 環境と比較して長く,ユーザが PC を道具として利用できる状態になるまで時間がかかるという欠点があり,ライブ CD を常用環境として利用する妨げとなっています.そこで,起動時間の長さを短縮し,ライブ CD とライブ CD に搭載されるオープンソースソフトウェアの普及を

ライブ CD の起動速度が遅い原因は,大きく分けて 2 つあります.一つめは,起動時のイメージ読み込みに時間がかかること,もう一つは起動シーケンスが最適化されていないことです.以下に,これらの問題点をまとめます.

・イメージ読み込みの問題

ライブ CD では起動に必要となるデータを全て CD ドライブから読み出します.CD ドライブは HDD などの固定メディア装置と比較して,メディアへのアクセス速度が非常に遅いデバイスです.また,CD ドライブに搭載されている,CD-ROM から直接データ読み出しを行うピックアップレンズはシークが遅く,ランダムアクセスを不得手賭しています.

ライブ CD はライブ CD ではデータがディスクに焼き込まれる直前のファイルシステムイメージをそのままイメージファイルとして保持します.そのため,起動時にランダムアクセスが大量に発生し,CD ドライブのヘッドのシーク待ち時間が増大してしまい起動時間が増大します.

・初期化 RAM ディスクの問題

・起動シーケンスの問題

ライブ CD の起動シーケンスは周辺機器の自動検出，設定などの検出作業を行い，ユーザの手を煩わせること無く周辺機器を取り扱うことが出来るようにします．しかしながら，この自動検出のシーケンスはひとつひとつの機器について逐次実行しており，非効率的な動作となっています．

また，起動シーケンスは起動に必要になったデータを逐一 CD-ROM メディアから読み出そうとします．そのため，デスクトップ環境や日本語入力環境等大きなデータの読み出しと，それ以外の細かなデータの読み出しで I/O にかかる時間に差が生じ，さらにデータの読み出しが散漫に行われることになります．これにより，起動シーケンス内ではところどころに I/O 待ちの状態や，CPU 待ちの状態が発生してしまう状態となります．

1.2 ライブ CD 起動高速化手法

これまでに挙げた二つのライブ CD の起動時間問題を解決する手法として,それぞれについて解決策を提案します.

– ファイルシステムイメージの最適化

ライブ CD のイメージ読み込み問題の解決として,ライブ CD のファイルシステムイメージを最適化します.ライブ CD のファイルシステムイメージのデータブロックの並びをランダムアクセスが発生しないように連続に配置することで,CD-ROM ドライブのピックアップのシーク時間を減らします.これにより,ファイルシステムイメージの最適化が行われたディスクからは一気にデータを転送することが出来るようになります.

図 1 はファイルシステムイメージの最適化の概要を示したものです.「ブロック参照プロファイラ」でライブ CD のファイルシステムイメージが起動時にどのように読み込まれているかをプロファイリングし,「イメージ最適化ツール」でプロファイラにより解析されたデータブロック順にファイルシステムイメージのデータブロックを並び替えます.並び替えされた最適化済みファイルシステムイメージは専用に変更された「最適化イメージ対応ドライバ」を利用して読み出し,利用されます.

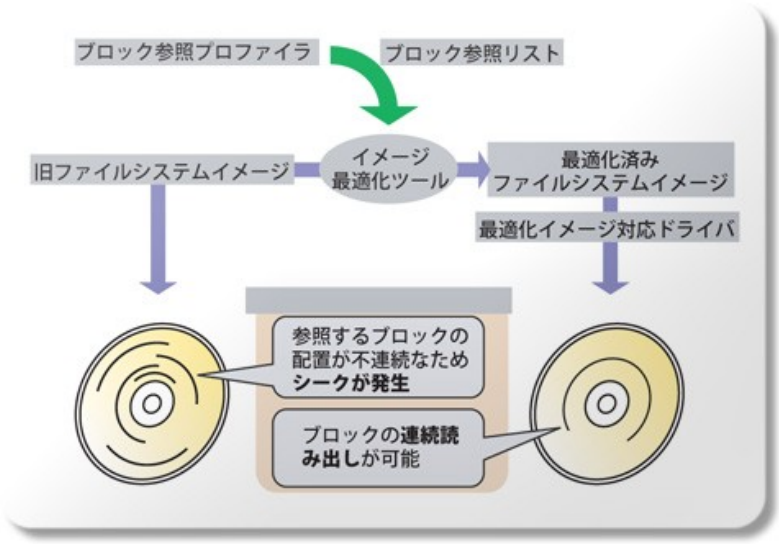

図 1:ファイルシステムイメージ最適化手法の概要

最適化されたファイルシステムイメージを元のファイルシステムイメージと置き換え,「最適化イメージ対応ドライバ」を初期化 RAM ディスク内の元のドライバと置き換えることで,ファイルシステムイメージの最適化は完了します.

– 起動シーケンスの最適化

起動シーケンス最適化を行うために,二つの変更をライブ CD の初期化 RAM ディスクに加えます.一つめは,初期化 RAM ディスクにライブ CD のブロック先読みプログラムを導入します.もう一つは,周辺機器の読み込みを逐次実行から並列実行に変更します.

ライブ CD のブロック先読みプログラムは,Linux 固有のシステムコール readahead()をライブ CD 向けに改良した新たな先読みプログラムです.一般的なライブ CD に用いられている cloop,squashfs などではファイルシステムイメージを圧縮した状態で保持しています.ライブ CD の先読みプログラムは圧縮状態のデータをキャッシュすることで,ライブ CD の使用に伴い不足がちになるメモリの消費を抑えつつ,データのキャッシュを行います.また,この先読みプログラムは起動時のプロセスを解析し,ディスク I/O が他の動作を待っているタイミングで動作するようにセットすることで,より効率的なデータ読み出しを行うことが可能となります.

さらに,周辺機器の並列検出を行うことで起動シーケンスの効率を向上させます.周辺機器は機器ごとに検出にかかる時間が異なります.長く時間のかかるシリアルポートなどの周辺機器の実行を並列化することで,その他の周辺機器が素早く検出することが出来るようにします.これは,起動 RAM ディスク内の起動スクリプト linuxrc の周辺機器検出部分を変更し,ハードウェア検出プログラム hwsetup,kudzu に改良を加えることで実装されます.

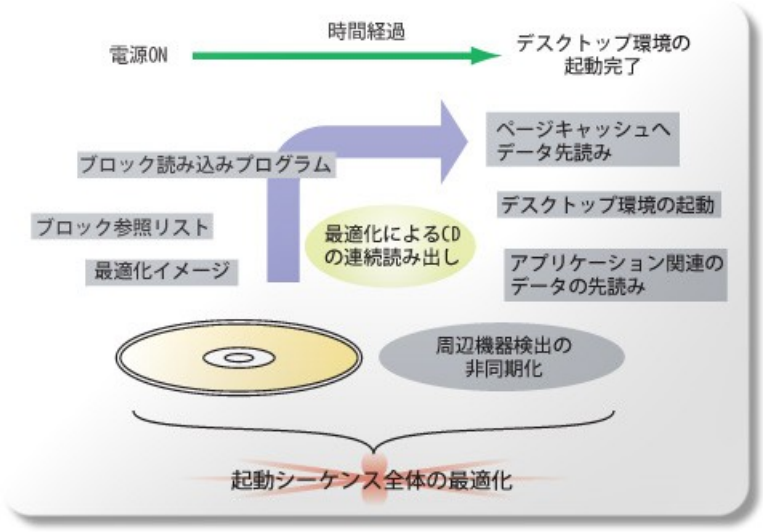

図 2:起動シーケンス最適化手法の概要

これらの起動シーケンス最適化処理は,ライブ CD の初期化 RAM ディスク内で行います.この最適化処理を導入した初期化 RAM ディスクと,最適化したファイルシステムイメージを合わせて,起動高速化を施したライブ CD は作成することが出来ます.

1.3 起動高速化の流れ

ライブ CD の起動高速化を行う流れは,以下のようになります.

I. 最適化イメージ対応ドライバを導入した ISO イメージの作成

II. ブロック参照状況の取得,解析

III. ブロック配置最適化

IV. 起動高速化 ISO イメージの作成

例えば,LCAT を利用して KNOPPIX に起動高速化を施す場合の具体的な流れは,下図 3 のようになります.流れの中で,色が変更されている部分が,起動高速化へむけて対応する部分となります.

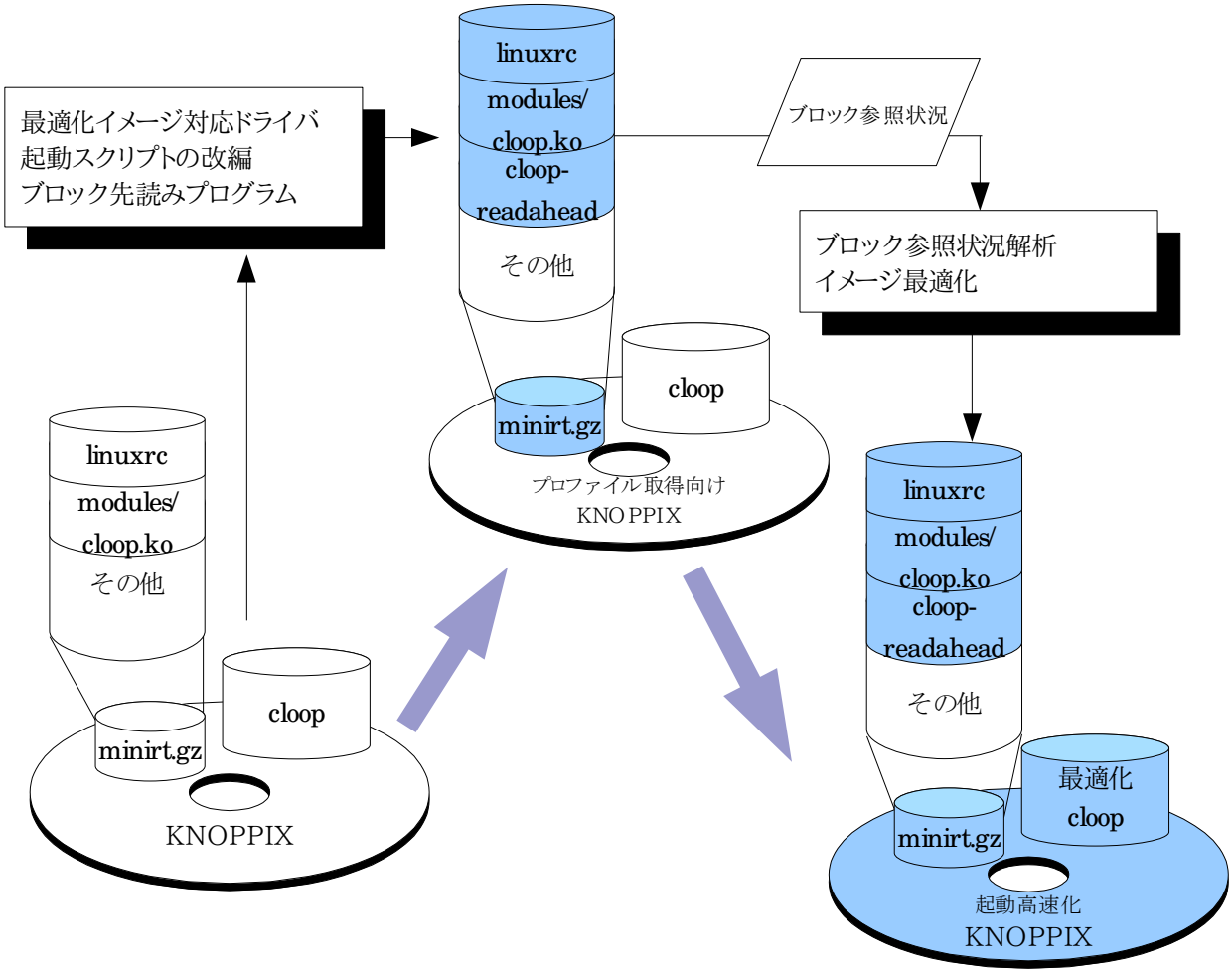

図 3:ライブ CD 最適化の手順

1.4 ライブ CD 起動高速化ツールキット"LCAT"

LCAT は,これまでに述べたライブ CD の起動高速化手法を実現するために開発されたツールキットです.LCAT は KNOPPIX 等のライブ CD が採用している圧縮ループバックブロックデバイス cloop にライブ CD の起動高速化手法を適用することが出来ます.また,LCAT のパッケージ内には KNOPPIX 向けの最適化起動スクリプトが同梱されています.LCAT 内のツールを cloop に適用し,起動 RAM ディスク内の起動スクリプトを差し替えることでライブ CD の起動高速化を実現することが出来ます.

1.5 この文書について

この文書は 2005 年度上期 オープンソースソフトウェア活用基盤整備事業「CD/DVD 起動 Linux の速度改善ドライバの開発」の成果であるライブ CD 起動高速化ツールキット LCAT を用いて,ライブ CD の起動高速化を行う方法を解説したものです.

この文書は,KNOPPIX 派生版の開発者がライブ CD に起動高速化を適用する際に利用することを想定したものであり,開発者はこの文書と LCAT のパッケージ内に収録された各 README ファイルを参考にライブ CD の起動高速化を行うことが推奨されます.

KNOPPIX 以外のライブ CD 開発者は,この文書と LCAT 内に含まれる各 README,各ツールのソースコードを参照し,各ライブ CD の起動高速化に利用することが可能です.

2.起動高速化向け開発環境

LCATを用いてCD版KNOPPIXの起動高速化を行う際に推奨する開発環境は,以下の通りです.

●ハードウェア

LCATを用いて円滑に開発を行う為の推奨環境を以下に示します.

表1:推奨動作環境

PC/AT互換機(i386 32bit)

CPU	1GHz以上のクロック周波数
メモリ	512MB以上
記憶領域	10GB以上
CDドライブ	24倍速以上のCD-Rドライブ

環境が充分でない場合,プログラムの挙動に若干の遅延が見られることがあります.

●ソフトウェア

LCATに含まれるプログラムを全て動作させるのに必要なソフトウェアを示します.

これらのソフトウェアは環境に直接インストールされているもの,仮想環境に構築されているもののどちらでも問題はありません.

表2:LCATに必要なソフトウェア一覧

OS	Linux kernel 2．6．12以降(2．6以降必須)
必要ライブラリ	libc6及び開発パッケージlibc6-dev(2．3．2以降) gtk2．0及び開発パッケージlibgtk2．0-dev(2．4以降) glib2．0及び開発パッケージlibglib2．0-dev(2．0以降

LCATは,以下の環境で動作確認をしています.

・KNOPPIX5.0DVD版(knoppix_v5.0.0DVD_20060225_cebit-20060406+IPAFont_AC20060412.iso)

・KNOPPIX5.0.1DVD版(KNOPPIX_V5.0.1DVD-2006-06-01-EN.iso)

・KNOPPIX5.0.1DVD版 HDDインストール

・Debian GNU/Linux Sarge 3.1

・Fedora Core 4(kernel module 除く)

3.開発環境の準備

(注: インストール作業は全てルート権限で行うものとします.)

最適化された cloop を開発するために,起動高速化ツールキット LCAT をシステムにインストールします.

開発環境が Debian 互換であれば,準備されている deb パッケージからそのままインストールを行うことが出来ます.

●deb パッケージからのインストール

deb パッケージからのインストールでは,次の二つのパッケージから選択することが出来ます.

- lcat_1.0.0_i386.deb
- lcat-cloop-module-2.6.12_1.0.0_i386.deb

lcat_1.0.0_i386.deb は,ユーティリティプログラムのみをインストールするパッケージです.最適化イメージ対応 cloop カーネルモジュールが必要な場合は,lcat-cloop-module-2.6.12_1.0.0_i386.deb をインストールするか,ソースパッケージから構築するかしなくてはなりません.lcat-cloop-module-2.6.12_1.0.0_i386.deb は,最適化イメージ対応 cloop カーネルモジュールを含むパッケージです.このカーネルモジュールは gcc3.3,kernel2.6.12 に対応したものです.これ以外の環境へ最適化イメージ対応 cloop カーネルモジュールを導入したい場合は,適合する環境下でソースパッケージから構築を行わなければなりません.

以下に,それぞれのパッケージのインストールの方法を示します.

・lcat_1.0.0_i386.deb

cloop カーネルモジュールを含まないパッケージです.このパッケージでインストールを行うと,LCAT の起動高速化ユーティリティのみがインストールされます.最適化イメージ対応モジュールが必要な場合は,ソースから最適化イメージ対応モジュールをビルドする必要があります.

```
# dpkg -i lcat-1.0.0_i386.deb
```

・lcat-cloop-module-2.6.12_1.0.0_i386.deb

cloop カーネルモジュールを含むパッケージです.このパッケージを利用してインストールを行う場合,先にパッケージ cloop-module がインストールされているとバージョンの衝突を起こしてしまうので,以下のようにインストールを行います.

```
# apt-get remove --purge cloop-module
# dpkg -i lcat-cloop-module-2.6.12_1.0.0_i386.deb
```

deb パッケージからインストールを行った場合,LCAT のユーティリティは/usr/local/bin に,cloop モジュールは/lib/modules/2.6.x/kernel/drivers/block/に配置されます.

●ソースアーカイブからのインストール

deb パッケージからではなく,ソースからインストールする場合は,ソースアーカイブ lcat_1.0.0.tar.gz を解凍して出来るパッケージディレクトリ$LCAT 上から,例えば次のように行います.

```
# ./configure
# make && make install
```

configure を実行する際に"./configure --prefix=(任意のディレクトリ)" とすると,任意のディレクトリに LCAT のユーティリティをインストールすることが出来ます.このとき指定を行わなかった場合のインストール先は, /usr/local/bin となります.

Linux システム上で make install からインストールを行った場合,cloop モジュールのインストール先は deb パッケージでインストールした場合と同じ場所にインストールされます.cloop モジュールをインストールしたくない場合は,ユーティリティディレクトリ$LCAT/util に移動した後,"make install"を行って下さい.カーネルモジュールは make を行えば,$LCAT/cloop-2.04-opt/以下に cloop.ko として保存されます.

4.Live CD Acceleration Tools (LCAT)

LCAT は,ライブ CD の起動高速化手法を実現するためのツールキットです.LCAT は起動高速化手法を KNOPPIX などで用いられている cloop に対して実装しています.LCAT が提供するツールを使用することで,cloop の最適化を実現することが可能になっています.また,LCAT は起動シーケンス最適化を施した起動 RAM ディスク内の起動スクリプト linuxrc を提供し,起動シーケンスの最適化を簡単に適用することが出来ます.

ここでは LCAT の概要と LCAT が提供するコマンドについて解説し,実際に LCAT を KNOPPIX に適用する手順を示します.

4.1 起動高速化ツール LCAT の概要

LCAT が提供する主な起動高速化ユーティリティプログラムとその機能を下に示します.

■cloopprofiler

「ブロック参照プロファイラ」を cloop に適用し実装したプログラムです.

■cloopoptimzier

「イメージ最適化ツール」を,cloop に適用し実装したプログラムです.このプログラムは cloop イメージの最適化を行います.

■cloop.ko

cloop のカーネルモジュールを最適化イメージを読み出せるように修正した「最適化イメージ対応ドライバ」です.このモジュールを使用することで,最適化した cloop イメージの読み出しやブロック参照状況の取得を行うことが出来るようになります.

■cloopreadahead

「ブロック先読みプログラム」を cloop イメージ向けに実装したものです.

LCAT はソースアーカイブ中に,起動シーケンス最適化を行った起動 RAM ディスク内の起動スクリプト linuxrc を同梱しています.このスクリプトを,起動高速化を行う KNOPPIX の minirt.gz 内に配置することで,起動シーケンスの最適化を行うことが出来ます.

また,周辺機器検出の並列化を目的として,KNOPPIX5.0.1 向けに改編したデバイス認識プログラム hwsetup, kudzu も LCAT に同梱されています.

4.2 コマンド一覧

ここでは LCAT に含まれている,起動高速化に利用するコマンド,カーネルモジュールの利用説明を行います.

●cloop.ko(最適化イメージ対応 cloop 用カーネルモジュール)

概要

LCAT のインストールを行うと,パッケージディレクトリ内のディレクトリ"cloop-2.04-opt"に最適化版 cloop カーネルモジュール(cloop.ko)が作成されます.このモジュールを KNOPPIX の起動 RAM ディスク minirt.gz に組み込むことで,最適化フォーマットの cloop イメージを読み込むことが出来るようになります.また,このカーネルモジュールを通常の Linux システムに組み込んで使用することも可能です.

最適化対応版の cloop.ko は以下の機能を持ちます.

* 最適化フォーマットされた cloop イメージのマウント

"cloopoptimzer"で最適化フォーマットを適用した cloop イメージをマウントし,読み出すことが出来ます.

* V2 フォーマットの cloop イメージのマウント

最適化されていない cloop イメージを読み出すことができる.LCAT は V2 フォーマットの cloop を対象としているので,それ以前のバージョンの cloop を組み込むことはできません.

* ブロック参照状況の調査

モジュールをロードする際,オプションに"chkblk=(調査する最大ブロック数)"とすると,/proc/cloop ディレクトリを作成し,ブロック参照状況を保存します./proc/cloop に含まれるファイルは次の通りです.

・/proc/cloop/read_blocks

cloop のブロック参照状況を記録するファイル.

・/proc/cloop/reset_read_blocks

/proc/cloop/read_blocks の内容を全てクリアするためのインターフェイスファイル.

・/proc/cloop/reset_profile

/proc/cloop/read_blocks のプロファイル(ヘッダ部分)のみをクリアするためのインターフェイスファイル.

使用方法

最適化イメージ対応 cloop.ko を組み込んで,cloop デバイスからデータを読み出したい場合は,cloop モジュールを insmod コマンドで読み込みます.このとき,読み出したい cloop イメージを"file=(cloop へのパス)"として指定します.

```
# insmod cloop-2.04-opt/cloop.ko file=/cdrom/KNOPPIX/KNOPPIX
```

insmod コマンドで読み込む際,オプションとして“chkblks=(記録するブロック数)"とすれば,/proc/cloop を作成し,ブロック参照状況を記録することが出来ます.

```
# insmod cloop-2.04-opt/cloop.ko  chkblks=10000 file=/cdrom \
/KNOPPIX/KNOPPIX
```

読み込みが出来れば,/dev 以下に新たに/dev/cloop が作成されています.このデバイスファイルを使用することで,insmod の際にターゲットとして指定した cloop イメージをマウントすることが可能です.cloop イメージのマウントの例は次のようになります.

```
# mount -o ro /dev/cloop  /cdrom/KNOPPIX/KNOPPIX
```

chkblks オプションを使用して cloop.ko を insmod した場合,ブロック参照状況を監視する/proc/cloop ディレクトリが作成されます.ブロック参照状況が記録されるファイル read_blocks は,次頁図 4 のような構成となります.

```
  cloop device : 0                          #  読み出されたcloopデバイスID
  total blocks : 36865                      #  総cloopブロック数
  file format  : optimized                  #  最適化されているか
  now jiffies  : 35549.0288                 #  計測時のuptime値
  cloop kernel thread mode.                 # threadモードで起動
  ------------------------------------------------------------------------
  call cloop_request    : 135551 times    # cloop読み込みリクエスト数
  call load_buffer()    : 485448 times    #  バッファ読み込みリクエスト数
  call read_from_file() : 42441 times     #  ファイルからの読み込み数
  read_time_max         : 0.0039 ms       #  最大読み込み時間
  read_time_min         : 0.0000 ms       #  最小読み込み時間
  read_time_sum         : 12.0818 ms      #  読み込み時間合計
  read length sum       : 820170874 bytes #  読み込みサイズ合計
  extract_time_max      : 0.0009 ms       #  最大伸張時間
  extract_time_min      : 0.0000 ms       #  最小伸張時間
  extract_time_sum      : 21.0215 ms      #  伸張時間合計
  extract length sum    : 2781413376 bytes#  伸張後サイズ合計

----------------------------------------------------------------------
no    blknum      size        access      read        extract
----------------------------------------------------------------------
###
# no        ->   読み込みID
# blknum    ->   ブロックID
# size      ->   ブロックサイズ
# access    ->   アクセス時のjiffies
# read      ->   読み込みに要した時間
# extract   ->   ブロックの伸張に要した時間
###
0     0           5435        25113.0688  0.0000      0.0001
1     32          26021       25137.0298  0.0001      0.0000
                                    :
```

図 4:/proc/cloop/read_blocksの内容

/proc/cloop/read_blocksのデータは,/proc/cloop/reset_read_blocksを以下のように使うことでリセットが可能です(ルート権限が必要です).

```
# echo 1 > /proc/cloop/reset_read_blocks
```

また,/proc/cloop/reset_profileで,/proc/cloop/read_blocksのヘッダ部分のみを全てゼロクリアすることが可能です(ルート権限が必要です).

```
# echo 1 > /proc/cloop/reset_profile
```

●cloopoptimizer

概要

cloopoptimizer は,標準フォーマットの cloop から最適化フォーマットの cloop を作成するユーティリティです. cloopoptimizer で作成した cloop は,LCAT に含まれる最適化イメージ対応 cloop モジュールからのみ読み込み可能となります.

cloopoptimizer は cloop の V2 以降に対応しており,それ以前のバージョンには適用できません.

使用方法

cloopoptimizer は以下の引数をとります.

```
# cloopoptimizer <original cloop file> <boot block list> \
[<application block list>]
```

最適化する cloop のパスを<original cloop file>に,ブート時のブロック参照状況を<boot block list>に,アプリケーションのブロック参照状況を<application block list>にそれぞれ指定します.<application block list>は無くても構いません.

cloopoptimizer の結果は標準出力に出力され,進捗状況は標準エラー出力に表示されます.cloopoptimizer の実行例を以下に示します.

```
# ./cloopoptimizer /cdrom/KNOPPIX/KNOPPIX /proc/cloop/read_blocks >
KNOPPIX.opt
block_size = 65536
num_blocks = 29859
header write.
start_offset = 477896
opt_blk_info_sum = 721645729, (704732KB)
block info table write.
block data write.
00001/29859: 0.000%, optidxno=00001, optidxsize=30220, write=30220
00002/29859: 0.003%, optidxno=00002, optidxsize=30491, write=30491
00003/29859: 0.007%, optidxno=00003, optidxsize=32705, write=32705
00004/29859: 0.010%, optidxno=00486, optidxsize=10347, write=10347
00005/29859: 0.013%, optidxno=00487, optidxsize=9412, write=9412
00006/29859: 0.017%, optidxno=00488, optidxsize=10374, write=10374
00007/29859: 0.020%, optidxno=00495, optidxsize=18319, write=18319
00008/29859: 0.023%, optidxno=00498, optidxsize=21719, write=21719
00009/29859: 0.027%, optidxno=00499, optidxsize=22415, write=22415
00010/29859: 0.030%, optidxno=00500, optidxsize=15652, write=15652
00011/29859: 0.033%, optidxno=00501, optidxsize=17766, write=17766
00012/29859: 0.037%, optidxno=00502, optidxsize=23801, write=23801
00013/29859: 0.040%, optidxno=00503, optidxsize=21791, write=21791
00014/29859: 0.044%, optidxno=00507, optidxsize=14744, write=14744
00015/29859: 0.047%, optidxno=00508, optidxsize=10267, write=10267
00016/29859: 0.050%, optidxno=00509, optidxsize=13166, write=13166
                                  :
```

図 5:cloopoptimzer の実行例

●cloopreadahead

概要

cloopreadahead は cloop ブロックの先読み機能を提供するユーティリティです.cloopreadahead を,最適化向けに変更した minirt.gz に組み込み,利用することで起動時の I/O の待ち時間を短縮することが出来ます.また,アプリケーションを起動する際にスクリプト等から呼び出すことで,I/O 時間の短縮を見込むことも出来ます.このプログラムでは,LCAT にて最適化した cloop にのみ対応しています.

使用方法

cloopreacahead は以下の引数を取ります.

```
# cloopreadahead <optimized cloop file> <block list file>
```

cloopoptimizer などで高速化された cloop のイメージへのパスを<optimized cloop file>に,最適化版 cloop.ko を利用して得たブロック参照状況 read_blocks を LCAT ユーティリティの中の rblk2bl で変換した先読みブロックリスト blklst を<block list file>にそれぞれ入力します.

cloopreadahead は読み込んだ cloop ブロックのブロック番号を標準出力に表示します.cloopreadahead の動作が正常な場合,この表示は読み込んだブロックリストの並ぶ順と同じとなります.

●rblk2bl/appblk2bl

概要

最適化対応カーネルモジュールを利用して作成したブロック参照状況ファイル read_blocks から,cloop 先読みリスト作成プログラム rblk2bl,appblk2bl を使用して cloopreadahead 等で使用される先読みリストを作成出来ます.先読みリストは cloop の高速化や,高速化したライブ CD 上から cloopreadahead を実行する際に利用します.

使用方法

rblk2bl は次のように引数を取ります.

```
# rblk2bl <block referencd data> <burst size> > <block list>
```

rlbk2bl は,cloop ブート時のブロック先読みリストを作成します.このブロック参照リストを使用して cloop のブロック配置最適化を行ったり,cloopreadahead でのブロック先読みを行ったりすることが出来ます.アプリケーション起動時に cloopreadahead を使用する場合には,アプリケーションのブロック参照状況を,このプログラムを使ってブロック先読みリストとしなければなりません.

<burst size>は cloopreadahead でのブロック先読みの粒度をバイト単位で設定します.この値は大きすぎても小さすぎても高速化の弊害となってしまいますので,適切な値を導入しなければなりません.

rblk2bl は標準出力にブロック先読みリストを表示します.rblk2bl が出力する先読みリストの例を,図 6 に示します.

```
# burst_size = 262144 (256 KB)
burst_start
0
32
34816
36352
4096
4480
20480
20992
12288
12928
22528
23296
16384
17408
24576
26112
26113
24578
24579
24583
24584
24585
# block = 22, size = 262346
burst_end
burst_start
26624
28160
              :
```

図 6:先読みブロックリストの例

appblk2bl は次のように引数を取ります.

```
# appblk2bl <boot read_blocks> <boot burst size> <aplication
read_blocks> ...
```

appblk2bl は,アプリケーション起動時のブロック先読みリストを作成します.このブロック参照リストは cloop-optimzier の第二引数に引きわたすことでアプリケーションのデータブロック配置を最適化した cloop を作成することが出来ます.

appblk2bl では,第一引数に cloop のブロック参照状況ファイルを指定しなければなりません.これは,ブート時のブロック参照リストとの衝突を避けるために使用されます.

appblk2bl には複数のアプリケーションの read_blocks を読み込ませる事が可能です.しかしながら,複数のアプリケーションを読み込ませて出来た先読みリストを cloopreadahead に利用することはできません.このブロックリストは最適化 cloop を作成するときの補助として利用します.

appblk2bl は標準出力にブロック先読みリストを表示します.appblk2bl の出力結果は,rblk2bl の出力結果と同じフォーマットで出力されます.

●cloopprofiler

概要

cloopprofiler は,cloop のブロック参照状況を可視化するユーティリティプログラムです.cloopprofiler はブート時やアプリケーション起動時のブロック参照状況を色付きのブロックで可視化し,ブロック読み出しの状況をアニメーション表示します.また,cloopoptimizer や rblk2bl,appblk2bl など LCAT のコマンドラインユーティリティの GUI フロントエンドや,これらのコマンドラインが提供する機能をまとめたライブ CD の起動高速化ウィザードともなります.

cloopprofiler は,次のような外観を持ちます.

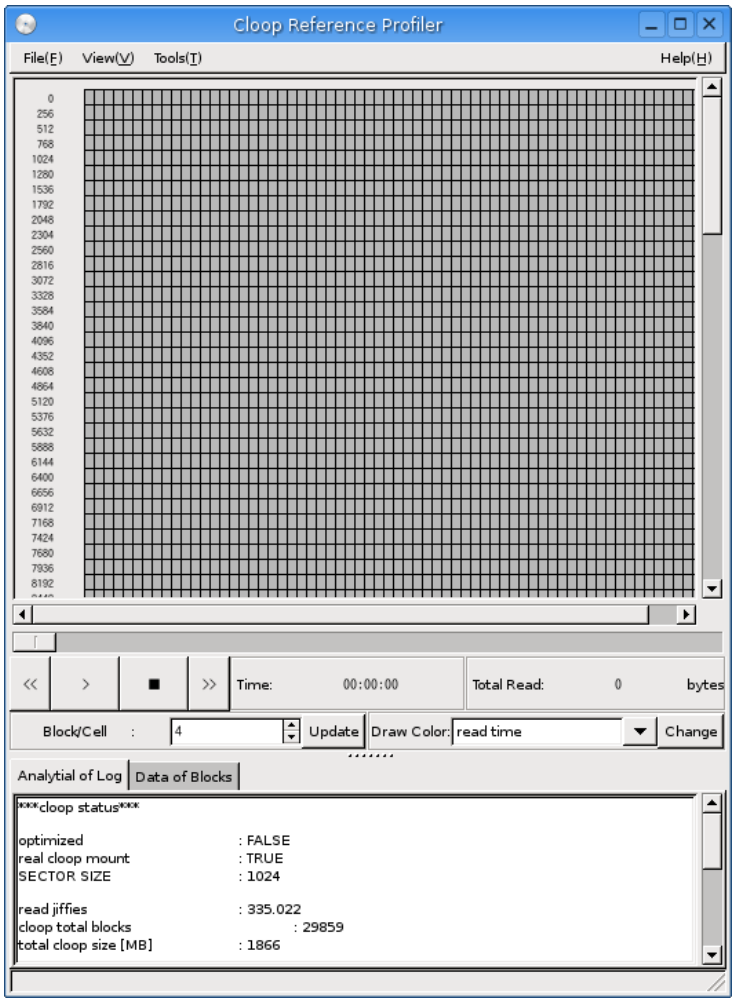

図 7:cloopprofiler の外観

画面上半分が cloop のブロック配置を表示する領域となります.ブロック参照状況を再生すると,この領域がアニメーションしながら変化してゆきます.

cloopprofilerが出来ることを以下に示します.

– 可視化機能

1. ブロック参照状況を可視化してアニメーション表示します.
2. ブロック参照状況を解析して表示します.
3. cloop ヘッダを可視化してアニメーション表示します.
4. 再生,一時停止,早送り,巻き戻しなどアニメーションの制御を行います.
5. ブロック読み出し時間の大小で色分けして表示します.
6. ブロック読み出し回数の大小で色分けして表示します.
7. 表示するブロックのデータ単位を cloop ブロック量とデータオフセット間で切替えます.
8. 表示するブロックに含まれる cloop ブロック量を調節します.
9. 表示するブロックに含まれる cloop データ量を 2 の指定数乗バイトで調節します.
10. 表示領域のブロックを左クリックすることでポップアップウィンドウを表示します.
11. ブロック参照状況をリストとして整理,表示します.
12. ブロックリストの要素を指定することで,描画領域の対象となるブロックを強調表示します.
13. 時間の進捗をスライダで表示します.
14. スライダを左クリックしドラッグすることで,描画時間を移動します.

– 起動高速化機能

15. 最適化ウィザードを利用して,先読みブロックリストと最適化 cloop を作成します.
16. 先読みブロックリストウィザードを利用して,先読みブロックリストのみを作成します.
17. 最適化 cloop 作成ウィザードを利用して,最適化 cloop のみを作成します.
18. 最適化対応ドライバ搭載 KNOPPIX から立ち上げたとき,read_blocks の初期化を実行します.

使用方法

cloopprofilerの主な使用方法は説明します.

・ブロック参照状況の可視化

『File』→『Open read_blocks』からブロック参照状況を読み出すと,ブロック参照状況をアニメーション出来ます.

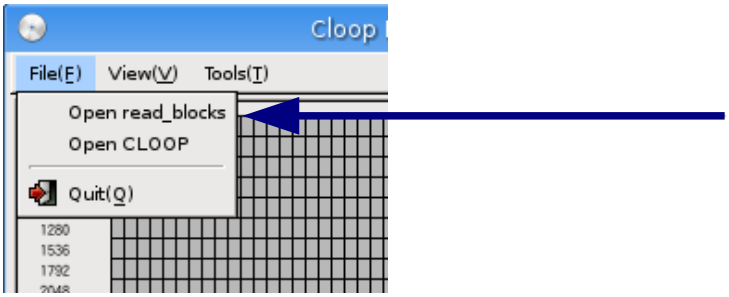

図 8:Open read_blocks

アニメーションの制御はウィンドウ中段のボタンやタイムスケール等で行います.

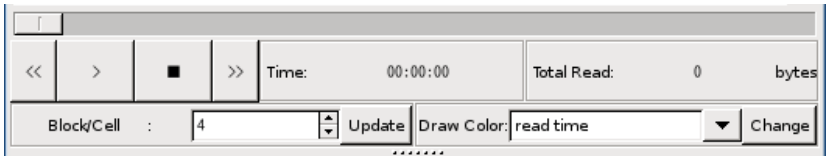

図 9:cloopprofiler のアニメーション制御ボタン

・cloop イメージの可視化

ブロック参照状況の読み出しを行った後,『File』→『Open CLOOP』で cloop のイメージを読み出すと,cloop のどの部分からデータが読み出されているかを可視化,アニメーション表示出来ます.アニメーションの制御はブロック参照状況の場合と同じです.

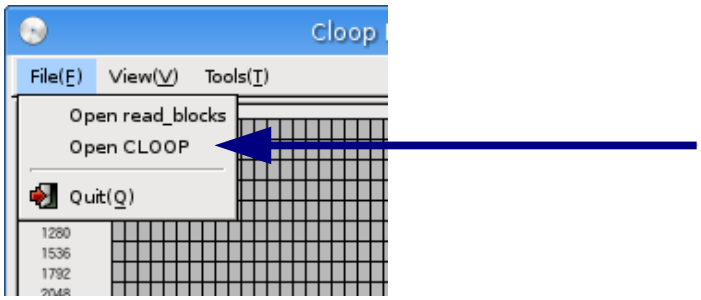

図 10:Open CLOOP

cloop イメージ,ブロック参照状況の表示切替えは,『View』→『draw resource』メニュー内の『read blocks log』,『cloop header』で切替えることが出来ます.

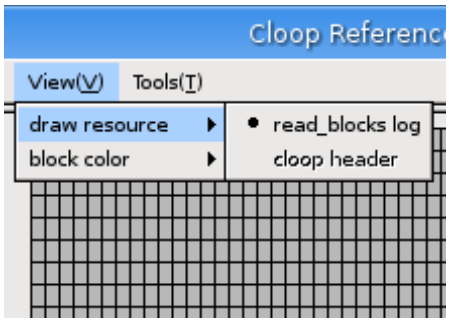

図 11:描画モードの切替え

・cloop のイメージ最適化

cloop のイメージ最適化は,『Tools』→『Accelerate Wizard』から呼び出すダイアログから行うことが出来ます.

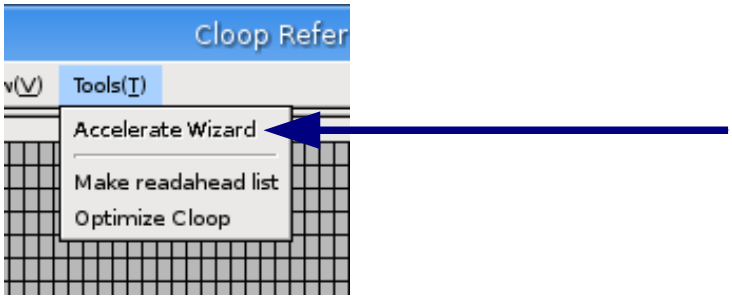

図 12:最適化ウィザードの呼び出し

cloop 先読みリスト,最適化 cloop の保存先を指定して【Next】ボタンを押下することで最適化を自動的に実行出来ます.

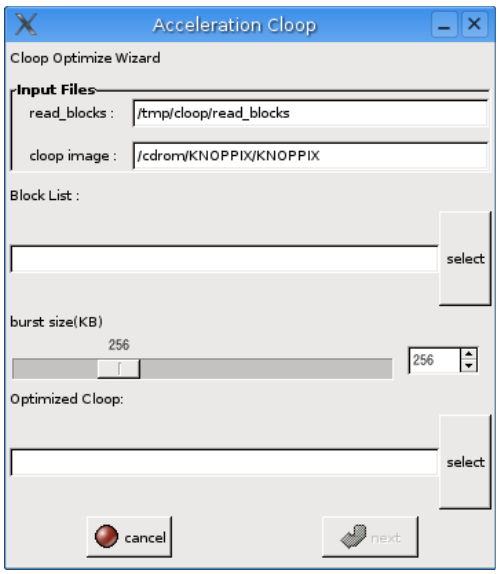

図 13:最適化ウィザード

“Block List"で指定した先に,cloopreadahead 用のブロック先読みファイルが,"Optimized Cloop"で指定した先にイメージ最適化された cloop がそれぞれ保存されます.ISO イメージとするには,これらのファイルを利用して mkisofs などのコマンドを使用します.

メニュー一覧

ここでは,cloopprofiler のメニューから実行できる機能を説明します.

–『File』

『File』メニューは,ブロック参照リスト read_blocks や cloop のファイルを開き,可視化させる事が出来ます.また,cloopprofiler を終了させることも出来ます.

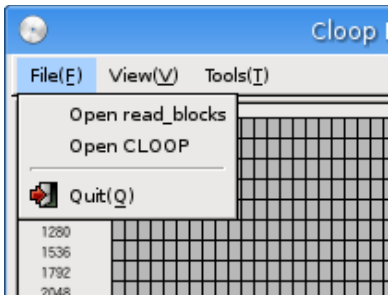

図 14:『File』メニュー

『Open read_blocks』

『Open read_blocks』メニューは,ブロック参照リスト read_blocks のデータを読み込み,可視化するコマンドです.『Open CLOOP』メニューを使用するためにはまずこのメニューで cloop に対応する read_blocks を読み込む必要があります.

このメニューで可視化できる read_blocks は,LCAT1.0 以降の cloop ドライバで出力された read_blocks のみであり,それ以前のトライアルバージョンのものは読み込み不可能となっています.

『Open CLOOP』

『Open CLOOP』メニューは,cloop の実体ファイルを読み込んでデータを可視化します.『Open CLOOP』メニューを実行することで,『View』→『draw resource』メニューから描画するリソースタイプを選択することが出来るようになります.また,『Open CLOOP』メニューを実行することで,『Tool』→『Accelerate Wizard』メニューを実行することが出来るようになります.

もしも別の cloop を読み込ませたいのなら,もう一度『Open read_blocks』メニューを実行するところからやり直さなければなりません.

『Quit』

『Quit』メニューは,このプログラムを終了します.

–『View』

『View』メニューは,ブロック参照状況の表示の調整を行います.

『draw resource』

『draw resource』コンテナは,表示させるブロックを cloop のデータと CD 上のオフセットとのどちらかを選択することが出来ます.

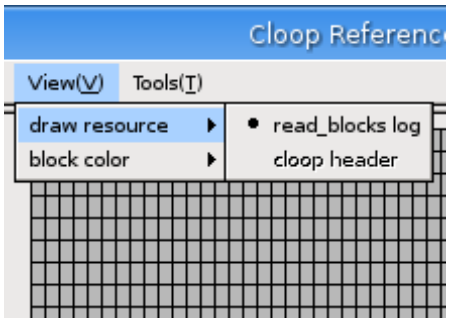

図 15:『draw resource』

『read_blocks log』

『read_blocks log』を選択した場合,描画の表示を read_blocks から読み出した cloop のデータ上でのブロック配置でデータの描画を行います.

『cloop header』

『cloop header』メニューでは,描画の表示を cloop のヘッダから読み出した物理的なブロック配置でデータの描画を行います.

『block color』

『block color』メニューコンテナは,cloop のデータブロックを表す描画ブロックの色の付け方を変更します.これにより,見た目でどこが読み込み時間が長いか,どこが頻繁に読み込まれているのかを一目で判別

できるようになります.

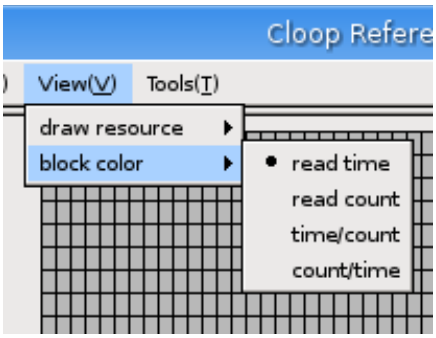

図 16:『block color』

『read time』

『read time』は,ブロックの色を読み込みにかかった時間で分けて表示するモードであり,読み込み時間が長ければ長いほど色が深くなります.

『read freq』

『read freq』は,ブロックの色を読み込み回数によって表示する.読み込み回数が多ければ多いほど色が深くなるモードです.読み込み回数が多いブロックは次第に色が深くなっていきます.

『time/freq』

『time/freq』は,ブロックを対角線で半分に切り,上の三角を『read time』形式,下の三角を『read freq』形式で描画します.

『freq/time』

『freq/time』は,ブロックを対角線で半分に切り,上の三角を『read freq』形式,下の三角を『read time』形式で描画します.

–『Tools』

『Tools』メニューは,cloop の起動高速化に役立ついろいろなユーティリティダイアログを呼び出します.

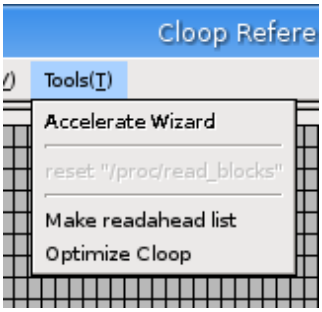

図 17:『Tools』メニュー

『Accelerate Wizard』

『Accelerate Wizard』は,『Open read_blocks』,『Open CLOOP』で開いた cloop イメージを最適化するウィザードを表示します.これらのメニューからファイルが読み出されていない状態では使用出来ません.詳しくは"dialog"の項を参照してください.

『Make readahead list』

『Make readahead list』はブロック参照リストひとつをブロック先読みリストに変換するダイアログを表示します.詳しくは"dialog"の項を参照してください.

『Optimize Cloop』

『Optimize Cloop』は cloop を最適化するダイアログを表示します.高速化するには対応したブロック参照リスト化,ブロック先読みリストが必要となります.詳しくは"dialog"の項を参照してください.

–『Help』

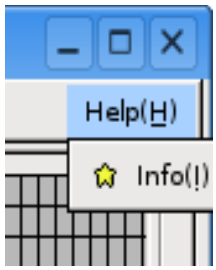

図 18:『Help』メニュー

『info』

『info』はこのプログラムの情報を表示します.

インターフェイス一覧

ボタン等のインターフェイスが出来ることを示します.主なボタンの配置は図 9 を参照してください.

* 【>】(【|| 】) ボタン

表示可能なデータが読み込まれているとき,アニメーションを再生します.アニメーションの再生中は一時停止ボタンとなります.

* 【■】 ボタン

アニメーションの再生をストップし,タイムカウンターを初期状態に戻します.

* 【>>】 ボタン

アニメーションを早送りします.

* 【<<】 ボタン

アニメーションを巻戻しします.

* 描画領域

アニメーションが描画される領域です.アニメーション中,現在読み込んでいるブロックは,青い枠で囲まれて表示されます.描画済みの領域をマウスでクリックすることで,その領域に書き込まれているブロックの詳細をポップアップして表示されます.ウィンドウ下部のリストをクリックすることで,リストに表示されているブロックが強調表示されます.

* 時間進捗バー

描画領域の真下に表示されているバーは,時間の進み具合を表します.トグルをつまんで動かすことで,再生を好きな場所にスキップさせる事が出来ます.

* 【Block/Cell】 選択ボタン

ひとつのブロックに含まれるデータ量を変更出来ます.『read_blocks』を選択したモードではブロックの数が,『cloop header』を選択したモードではブロックのデータサイズ(2 の選択数乗バイト)がデータ量の単位になります.データ量を変更したあとは,【>】(【|| 】) ボタン,【■】 ボタン,【change】ボタンを押すことで変更を適用出来ます.この変更を行うとアニメーションは一旦停止し,また再生を待つ状態になります.

*《Draw Color》 リスト

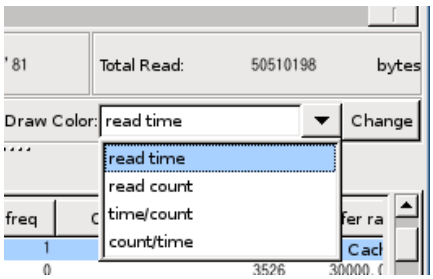

図 19:プルダウンした《Draw Color》

『View』→『draw color』メニューと同じ効果を持ちます.また,メニューの選択とこのリストの選択は同期して動きます.詳しい内容については,『draw color』メニューの項を参照して下さい.

*"Analytial of Log"

read_blocks の内容を読み込み,分析した結果を表示します.

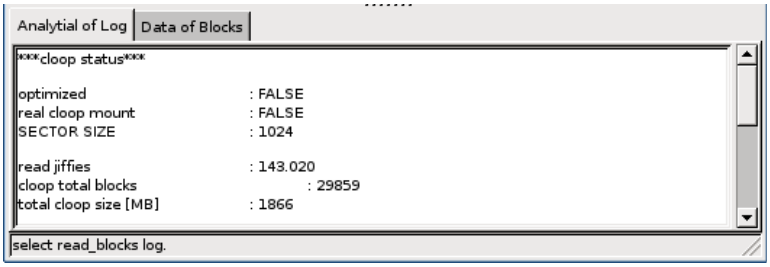

図 20:Analytial of Log

*《Data of Blocks》リスト

描画ブロックの情報をまとめてリストとして表示します.リストの内容をクリックすると,クリックしたリストが指す描画領域のブロックが強調表示されます.

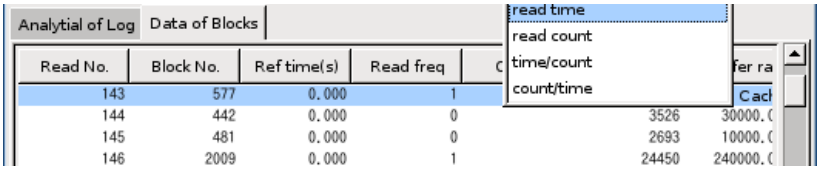

図 21:Data of Blocks

ダイアログ一覧

ここでは,『Tools』メニューから起動する各種のダイアログの説明を行う.

『Accelerate Wizard』

『Accelerate Wizard』は,『Open read_blocks』,『Open CLOOP』で読み込んだ CD のデータを元に,cloop の最適化を行います.最適化はファイルの指定をすれば自動的に行われ,最適化された cloop とブロック先読みリストを生成します.

『Accelerate Wizard』の外観については,図 13 を参照してください.

以下は,"Accelerate Wizard"ダイアログ内の説明です.

- read_blocks

『Open read_blocks』で読み込んだ read_blocks のパスを表示します.編集は出来ません.

- cloop image

『Open CLOOP』で読み込んだ cloop のパスを表示します.編集は出来ません.

- Block List

ブロック先読みリストの出力先の指定を行います.

- burst size

cloopreadahead での読み込みブロックの粒度をキロバイト単位で指定します.

- Optimized Cloop

最適化した cloop の出力先を指定します.

『Make readahead list』

『Make readahead list』では,任意の read_blocks からブロック先読みリストを作成します.これは rblk2bl と同様の動作をします.

以下は,このダイアログ内の説明です.

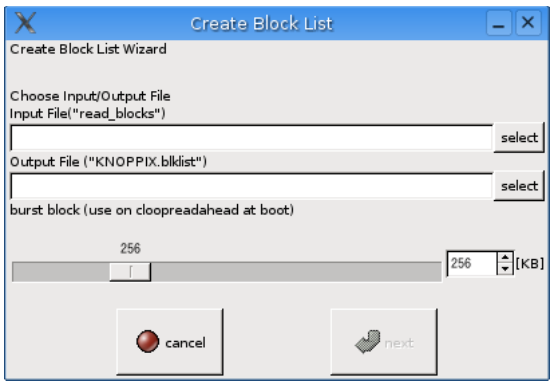

図 22:"Make readahead list"

- Input File

入力のブロック参照リストの指定を行います.

- Output File

ブロック先読みリストの出力先の指定を行います.

- burst size

cloopreadahead での読み込みブロックの粒度をキロバイト単位で指定します.

『Optimize Cloop』

『Optimize Cloop』では,任意の cloop を最適化します.最適化には対象となる cloop のブロック参照状況先読みファイルが必要となります.これは,cloopoptimizer と同様の動作をしますが,アプリケーションのリストを取ることは出来ません.

以下は,このダイアログ内の説明です.

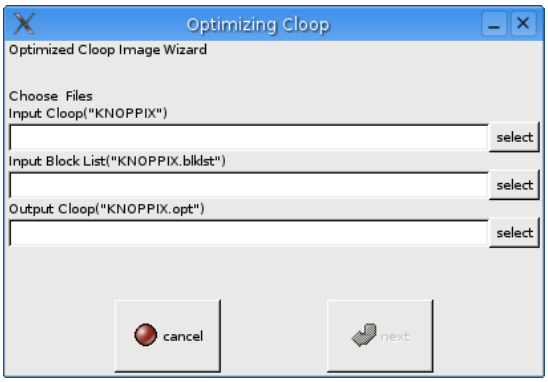

図 23:Optimized Wizard

- Input cloop

  入力の cloop の指定を行います.

- Input Block List

  入力の cloop に対応したブロック先読みリストの指定を行います.

- Output Cloop

  最適化した cloop の出力先を指定します.

5.KNOPPIXの起動高速化

ここでは,ライブCDであるKNOPPIXを対象として実際に起動高速化を行う手順について述べます.ライブCDの起動高速化では,CDに格納するcloopイメージファイルの最適化を行うことが主な作業となります.

cloopイメージ最適化には,前章で述べた通り実際にシステムを起動させ,どのcloopイメージ中のデータブロックを参照したかについてのデータを必要とします.この,cloopイメージ中のデータブロックの参照状況を把握するため,プロファイルの機能がついた,最適化イメージ対応cloopドライバを導入します.

次に最適化イメージ対応cloopドライバを導入したKNOPPIXをプロファイリングモードで実際に動かし,cloopイメージ中のデータブロックの参照状況を取得します.

そして,この取得したデータブロックの参照状況を元にcloopイメージを最適化し,再びISO9660のイメージファイルにまとめることで,KNOPPIXの起動高速化が完了します.

- 注意

- 以下の手順を実行する前に,LCATがインストールされている必要があります.インストール方法は,第X章を参照してください.

-- 以下の手順は全てroot権限で行います.操作ミスにより,開発環境を破壊する恐れもありますので,細心の注意を払って作業してください.

-- 比較的大きいファイルを扱います.ハードディスクの空き容量を確認してください.

5.1 プロファイル機能の付いた最適化イメージ対応cloopドライバの導入

まず,適用前KNOPPIXのISOイメージファイル(ここでは,knoppix_5.0.1cd_j.iso) に格納されているファイル群を作業用ディレクトリにコピーします.

```
# mkdir tmp_mount
# mount -t iso9660 -o loop,ro ./knoppix_5.0.1cd_j.iso ./tmp_mour
# cp -pRv ./tmp_mount ./MASTER
# umount tmp_mount
```

これで,"./MASTER"配下に適用前KNOPPIXのISOイメージファイルが揃いました.

KNOPPIXではcloopドライバが仮のrootファイルシステムであるinitrdに格納されています.そこで次は,initrdであるminirt.gzを差し替えます.

```
# mv ./MASTER/boot/isolinux/minirt.gz \
./MASTER/boot/isolinux/minirt.gz.org
# gzip -cd ./MASTER/boot/isolinux/minirt.gz.org > miniroot
# mount -t ext2 -o loop ./miniroot tmp_mount
```

これで,ループバックイメージ miniroot が./tmp_mount にマウントされました.

まず,最適化イメージ対応 cloop ドライバを差し替えます.バイナリインストールの場合は,
/lib/modules/2.6.X/kernel/drivers/block/cloop.ko ,ソースからコンパイルした場合は,コンパイルを実行したディレクトリ配下${compile_root}/lcat_1.0/cloop-2.04-opt/cloop.ko を,./tmp_mount 配下にコピーします.

```
  # mv ./tmp_mount/modules/cloop.ko \
./tmp_mount/modules/cloop.ko.org
  # cp <コピー元>/cloop.ko ./tmp_mount/modules
```

次に,最適化イメージ対応 cloop ドライバのプロファイリングモードを利用できるよう,linuxrc を修正します.具体的には,insmod で cloop のドライバをロードする処理の部分に,引数処理の追加を行います.行番号は,
KNOPPIX5.0.1CD 日本語版の linuxrc です.

```
  # cp -a ./tmp_mount/linuxrc ./tmp_mount/linuxrc.bak
  # emacs -nw ./tmp_mount/linuxrc

     mount_knoppix()
     {
      if test -n "$FOUND_KNOPPIX" -a -f
"$1/$KNOPPIX_DIR/$KNOPPIX_NAME"; then
----  ここから：  コメントアウト
     #  # DEBUG
     #  # echo "6" > /proc/sys/kernel/printk
     #  $INSMOD -f /modules/cloop.ko file="$1/$KNOPPIX_DIR \
/$KNOPPIX_NAME"
----  ここまで：  コメントアウト
----  ここから：  追加
          #  accelerated-knoppix : cloop-opt.ko
          #
          echo "6" > /proc/sys/kernel/printk
          THREAD="thread_mode=1"
          CHKBLK="0"
          for i in $CMDLINE; do
           case "$i" in chkblk=*|CHKBLK=*) eval $i; [ -n
"$chkblk" ] && CHKBLK="$chkblk" ;; esac
           case "$i" in *noclpthread*|*NOCLPTHREAD*) THREAD=""
;; esac
          done
          echo ""
          $INSMOD -f /modules/cloop.ko chkblks=$CHKBLK "$THREAD"
file="$1/$KNOPPIX_DIR/$KNOPPIX_NAME"
          echo "0" > /proc/sys/kernel/printk
----  ここまで：  追加
       mountit /dev/cloop /KNOPPIX "-o ro" || FOUND_KNOPPIX=""
       # Allow multi-image KNOPPIX mounts
       if [ -n "$FOUND_KNOPPIX" -a -x "$DYNLOADER" -a -x
/KNOPPIX/sbin/losetup ]; then
----  ここから：  コメントアウト
     #   echo ""
----  ここまで：  コメントアウト
#  次頁へ続く
```

```
# 前頁からの続き
       echo -n "${CRE} ${GREEN}Found primary KNOPPIX compressed
image at\\ ← 一行
 ${MAGENTA}$1/$KNOPPIX_DIR/$KNOPPIX_NAME${GREEN}.${NORMAL}"
       for c in 1 2 3 4 5 6 7; do
```

また,cloop イメージ中のデータブロックを先読みする処理を追加しておきます.KNOPPIX5.0.1CD 日本語版では,"toram", "tohd", "bootfrom"のブートオプションを処理する部分(line:577-586) を先に実行し,その結果を用いてデー

タブロックの先読み処理の実行を判断します.また,データブロックの先読みを行いながらcloopイメージファイルをマウント(line:575-576)します.

```
      copy_to()
      {
      |
      }

---- ここから： 移動
     COPYTO=""
     BOOTFROM=""
     DO_REMOUNT=""
     REAL_TARGET=""
     UNIONFS=""

     case "$CMDLINE" in *toram*) DO_REMOUNT="yes";
     COPYTO="ram"; ;; esac
     case "$CMDLINE" in *tohd=*) DO_REMOUNT="yes";
     COPYTO="hd"; ;; esac
     case "$CMDLINE" in *bootfrom=*) DO_REMOUNT="yes";
     BOOTFROM="yes" ;; esac

---- ここまで： 移動
---- ここから： 追加
#
# accelerated-knoppix : cloop blocks readahead
#
if test -z "$DO_REMOUNT" -a -n "$FOUND_KNOPPIX" ; then
  if test -x /accel/cloopreadahead ; then
    CLOOPREADAHEAD="yes";
    case "$CMDLINE" in *nocbr*|*NOCBR*) CLOOPREADAHEAD=""; ;;
     esac
    if test -n "$CLOOPREADAHEAD" -a -f
     /cdrom/${KNOPPIX_DIR}/${KNOPPIX_NAME}.boot.lst ; then
      echo ""
      echo -n " ${GREEN}Reading cloop
     blocks....${BLUE}(Backgrounding)${NORMAL}"
      /accel/cloopreadahead /cdrom/$KNOPPIX_DIR/$KNOPPIX_NAME
     \
        /cdrom/${KNOPPIX_DIR}/${KNOPPIX_NAME}.boot.lst >
     /.cloopreadahead.log 2>&1 &
    fi
  fi
fi
---- ここまで： 追加
# 次頁へ続く
```

```
# 前頁からの続き
  ----  ここから：  移動
        mount_knoppix /cdrom

  ----  ここまで：  移動
        # Remount later after copying/isoloading/driverloading?
        # pre-test if everything succeeded
        if  test -n "$DO_REMOUNT" -a -n "$FOUND_KNOPPIX"
```

linuxrc の編集が終了したら,cloop イメージ中のデータブロックを先読みするコマンド cloopreadahead を導入します.バイナリインストールの場合は,/usr/local/bin/cloopreadahead,ソースからコンパイルした場合は,コンパイルを実行したディレクトリ配下${compile_root}/lcat_1.0/util/cloopreadahead を./tmp_mount 配下にコピーします.

```
# mkdir ./tmp_mount/accel
# cp <コピー元>/cloop.ko ./tmp_mount/accel
```

上記作業の全てが終わったらループバックイメージ miniroot のマウント解除を行い,minirt.gz として配置します.

```
# umount tmp_mount
# gzip -c ./miniroot > ./MASTER/boot/isolinux/minirt.gz
```

これで,プロファイル用 KNOPPIX のファイル一式が./MASTER 配下に揃いました.これを ISO イメージファイル(knoppix_5.0.1cd_j_prof.iso)にまとめます.

```
  # mkisofs -l -r -J -V "KNX501CDJ_PROF" -hide-rr-moved -v \
      -b boot/isolinux/isolinux.bin -c boot/isolinux/boot.cat \
      -no-emul-boot -boot-load-size 4 -boot-info-table \
      -o ./knoppix_4.0.2cd_j_prof.iso ./MASTER
```

これで,プロファイル機能の付いた最適化イメージ対応 cloop ドライバを導入したプロファイル用 KNOPPIX の ISO イメージファイル(knoppix_5.0.1cd_j_prof.iso)が作業ディレクトリ下に準備できました.

5.2 プロファイリングモードでの起動によるブロック参照状況データの取得

第 1 節,プロファイル機能の付いた最適化イメージ対応 cloop ドライバの導入で作成したプロファイル用 KNOPPIX を,実機もしくは QEMU などの仮想 PC を用い起動します.その際,次のブートオプションを指定することでプロファイリングモードで起動します.

```
boot: knoppix chkblk=10000 nocbr
```

デスクトップの起動完了後コンソールを開き,下記操作でシステム起動時のブロック参照状況データを採取します.

```
# cat /proc/cloop/read_blocks > /tmp/boot.read_blocks
```

これで,システム起動時のブロック参照状況データが/tmp/boot.read_blocks に保存されました.

KNOPPIX に収録されているアプリケーションの中で,利用頻度の高いものがあればそのアプリケーションも高速化出来ます.ここでは,OpenOffice.org の起動高速化を行います.

アプリケーション起動時のブロック参照状況データ採取の前に,今までのブロック参照状況を下記操作でリセットします.

```
# echo 1 > /proc/cloop/reset_read_blocks
```

次に,ランチャアイコン,メニューなど,なんらかの方法で対象とするアプリケーションである,OpenOffice.org を起動します.起動の完了を確認した後,コンソールでの下記操作でアプリケーション起動時のブロック参照状況データを採取します.

```
# cat /proc/cloop/read_blocks > /tmp/appli.read_blocks
```

これで,アプリケーション起動時のブロック参照状況データが/tmp/appli.read_blocks に保存されました.数種類のアプリケーションが対象となる場合は,それぞれのアプリケーションに対して起動・終了を行った後,ブロック参照状況データを採取してください.

このプロファイルモードで採取した,システム起動時のブロック参照状況データファイル(boot.read_blocks),アプリケーション起動時のブロック参照状況データファイル(appli.read_blocks)を,USB メモリーキーなどのリムーバブルメディア,もしくはネットワークを用い,開発環境である,

開発環境である,第 X 節,プロファイル機能の付いた最適化イメージ対応 cloop ドライバの導入で参照した参照した,./MASTER のある作業ディレクトリ配下に転送します.

5.3 ブロック参照状況の解析

第 2 節,プロファイリングモードでの起動によるブロック参照状況データの取得で採集したシステム起動時,及びアプリケーション起動時の各ブロック参照状況データファイル(boot.read_blocks, appli.read_blocks)を用い,ブロッ

ク参照状況の解析を行います.転送したファイルはテキストファイルですのでそのまま閲覧出来ますが,ブロック参照状況可視化ツールである cloopprofiler を用います.

cloopprofiler は,下記のコマンドので実行します.

```
# cloopprofiler
```

cloopprofielr の起動完了後,メニューバー『File』→『Openread_blocks』を選択し,ファイルダイアログを用いブロック参照状況データファイルを指定します.データの読み込み後,【>】ボタンを押下することによって,実時間で cloop 中のブロック参照をアニメーションで表示します.

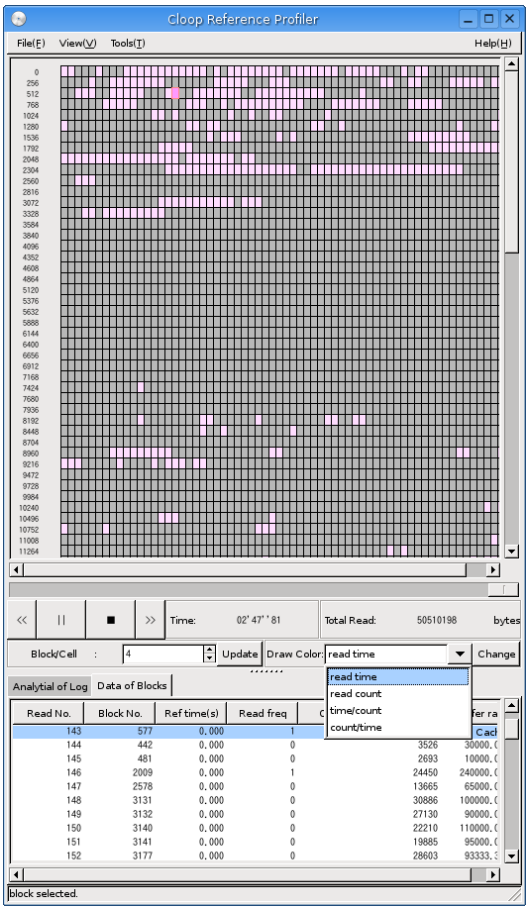

図 24:cloopprofiler で

データ描画を行ったときの様子

表示領域のブロック解像度の変更は,【block/Cell】スピンボタンで行います.

ブロック参照状況のアニメーションは,デフォルトではブロック番号で描画しています.この表示を実際の cloop イメージファイル中の位置で描画するには,ブロック参照状況データを採取する際に対象とした cloop イメージファイルが必要になります.

cloop イメージファイルの指定は,メニューバー『File』→『Open CLOOP』を選択し,ファイルダイアログを用い cloop イメージファイルを指定します.cloop イメージファイルは,作業用ディレクトリ配下,./MASTER/KNOPPIX/KNOPPIX にあります.

cloop イメージの読み込み後,メニューバー『View』→『draw resource』→『cloopheader』を選択することにより,cloop イメージファイル中の位置で描画するモードとなります.

5.4 既存 cloop イメージファイルの最適化

第 3 節,プロファイリングモードでの起動によるブロック参照状況データの取得で採集したシステム起動時,及びアプリケーション起動時の各ブロック参照状況データファイル(boot.read_blocks, appli.read_blocks)を用い,既存 cloop イメージ中のブロック配置を最適化します.

cloop イメージのブロック配置を最適化するには,コマンド cloopoptimizer を用いる方法,もしくは cloopprofiler の GUI で操作する方法があります.ここでは,コマンド cloopoptimizer を用いる方法について述べます.cloopprofier を用いる場合は,第 X 章,XXX を参照してください.

以下の操作で,既存 cloop イメージファイルを別名(KNOPPIX.normal)に変更した後,cloopoptimizer を実行し,最適化 cloop イメージファイルを MASTER/KNOPPIX/KNOPPIX に作成します.

```
# mv ./MASTER/KNOPPIX/KNOPPIX ./KNOPPIX.normal
# cloopoptimizer ./KNOPPIX.normal ./boot.read_blocks \
      ./appli_read_blocks > ./MASTER/KNOPPIX/KNOPPIX
```

これで,最適化済み cloop イメージファイルを./MASTER 配下に用意できました.

5.5 先読み cloop イメージファイル中データブロックリストの作成

第 4 節,プロファイリングモードでの起動によるブロック参照状況データの取得で採集したシステム起動時のブロック参照状況データファイル(boot.read_blocks) から,先読みする cloop イメージファイル中のデータブロックリストを作成します.

先読みする cloop イメージファイル中のデータブロックリストは,コマンド rblk2bl を用いる方法,もしくは cloopprofiler の GUI で操作する方法があります.ここでは,コマンド rblk2bl を用いる方法について述べます. cloopprofiler を用いる場合は,第 4 章コマンド一覧の cloopprofiler を参照してください.

以下の操作で,システム起動時のブロック参照状況データファイルから起動時に先読みする cloop イメージファイル中のデータブロックリスト(./MASTER/KNOPPIX/KNOPPIX.boot.lst)を作成します.

```
# rblk2bl ./boot.read_blocks > ./MASTER/KNOPPIX/KNOPPIX.boot.lst
```

これで,起動時の先読み cloop イメージファイル中データブロックリストを./MASTER 配下に用意できました.

5.6 最適化 cloop イメージファイルを格納した ISO イメージファイルの作成

第 1 節,プロファイル機能の付いた最適化イメージ対応 cloop ドライバの導入,第 4 節,既存 cloop イメージファイルの最適化,及び,第 5 節先読み cloop イメージファイル中データブロックリストの作成によって,最適化イメージ対応 cloop ドライバ,最適化済み cloop イメージファイル,起動時の cloop イメージファイル中データブロックリストが,./MASTER 配下に揃いました.

これを ISO イメージファイル(knoppix_5.0.1cd_j_AC.iso)にまとめます.

```
# mkisofs -l -r -J -V "KNX402CDJ_PROF" -hide-rr-moved -v \
    -b boot/isolinux/isolinux.bin -c boot/isolinux/boot.cat \
    -no-emul-boot -boot-load-size 4 -boot-info-table \
    -o ./knoppix_4.0.2cd_j_AC.iso ./MASTER
```

これで,cloop イメージを最適化した KNOPPIX の ISO イメージファイル(./knoppix_5.0.1cd_j_AC.iso)が完成しました.

5.7 更なる高速化

本章では,cloop イメージの最適化を中心に述べましたが,ライブ CD の起動高速化には,周辺機器検出の非同期化など起動シーケンスを最適化する手法もあります.これは,ライブ CD の種類やバージョンによって対処が異なります.

KNOPPIX5.0.1CD 日本語版での適用例を LCAT アーカイブ lcat_1.0/KNOPPIX/配下に納めました.参考にしてください.

6.謝辞

LCATの開発は，独立行政法人情報処理推進機構(IPA)2005 年度「オープンソースソフトウェア活用基板整備事業『CD/DVD起動Linuxの速度改善ドライバの開発』」プロジェクトの一部として行われました.IPAからは,本プロジェクトの計画,開発,検収などの進行について幅広い御支援を頂きました.この御支援が無ければ本プロジェクトの完了は無かったと言っても過言ではありません.

厚い御支援に感謝いたします.