圧縮ブロックデバイスにおける ファイルシステム最適化

北川 健司† 丹 英之† 阿部 大将† 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

Agenda

- **研究背景と目的**
- **関連研究**
- Ext2optimizer**とは**
- **評価**
  - KNOPPIX
  - HTTP-FUSE KNOPPIX
- **まとめと今後の展開**

研究背景

一つの大きなイメージをブロック単位で圧縮を行う機構
(ex. cloop, squashfs, zisofs)

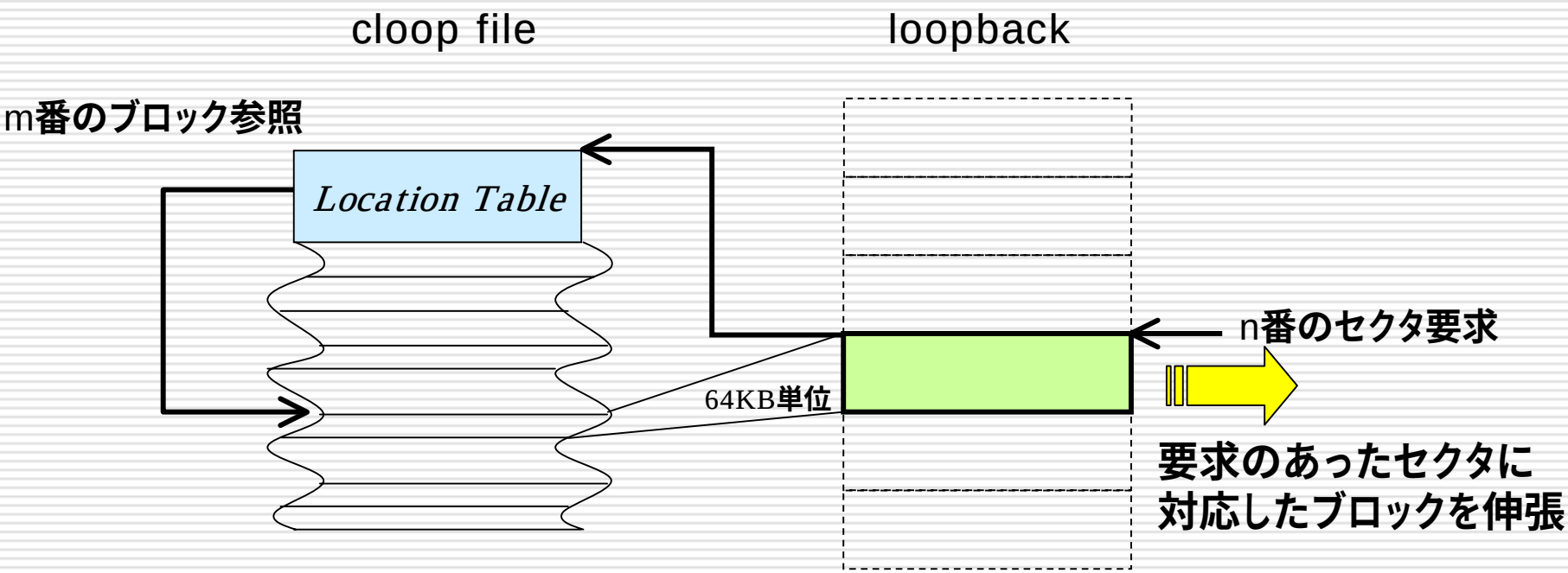
ライブCDを中心に広く普及

- 限られた容量制限への対応
- 起動デバイスとCPUとの速度差緩和

ライブCD ”KNOPPIX”

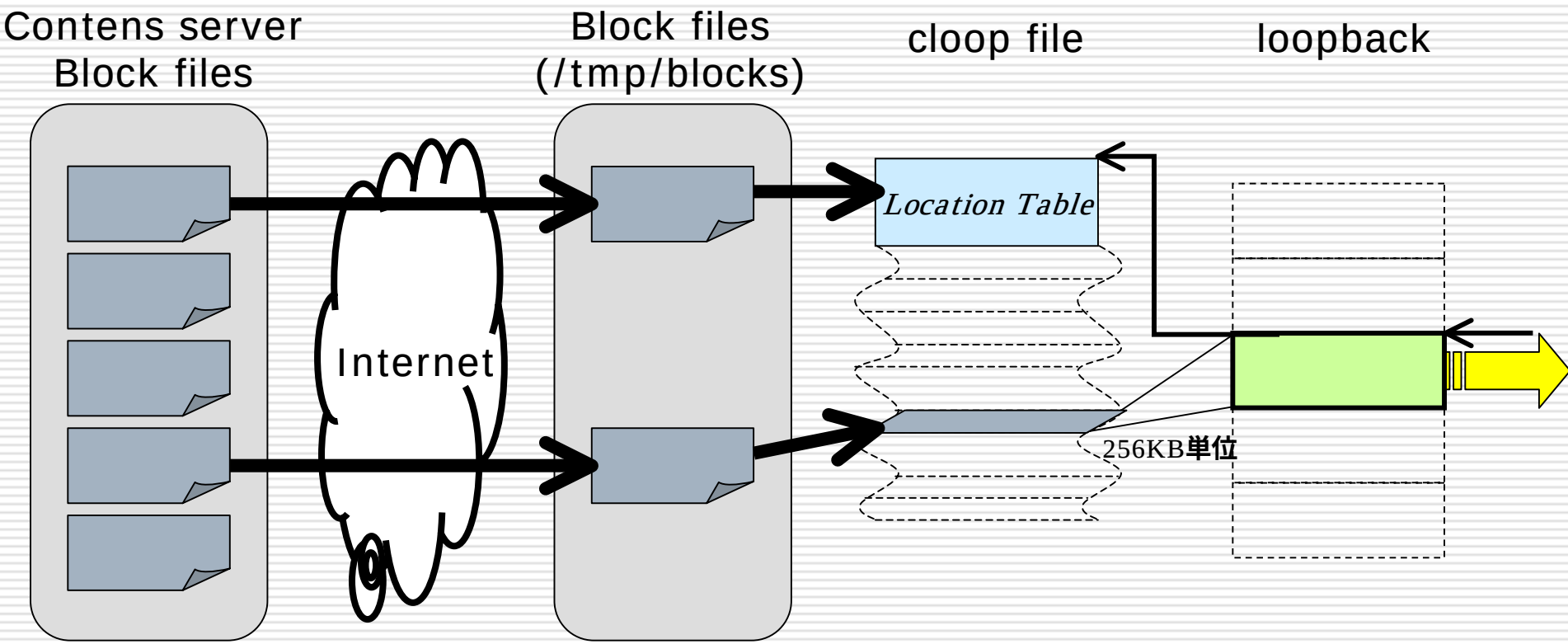
- CD/DVD起動のDebian GNU/Linux
  - Knopper氏(独)が開発、AISTで日本語化
  - 日本語化されたライブCDとして認知度が高い
  - 様々な派生物が存在
- AutoConfigによる強力なデバイス自動認識
  - Linuxで使用可能なデバイスは，ほぼ自動設定
- cloop/zlibを用いた圧縮ループデバイス
  - 700MB CDメディアに2GBのコンテンツを収録可能

圧縮ループデバイス(cloop)



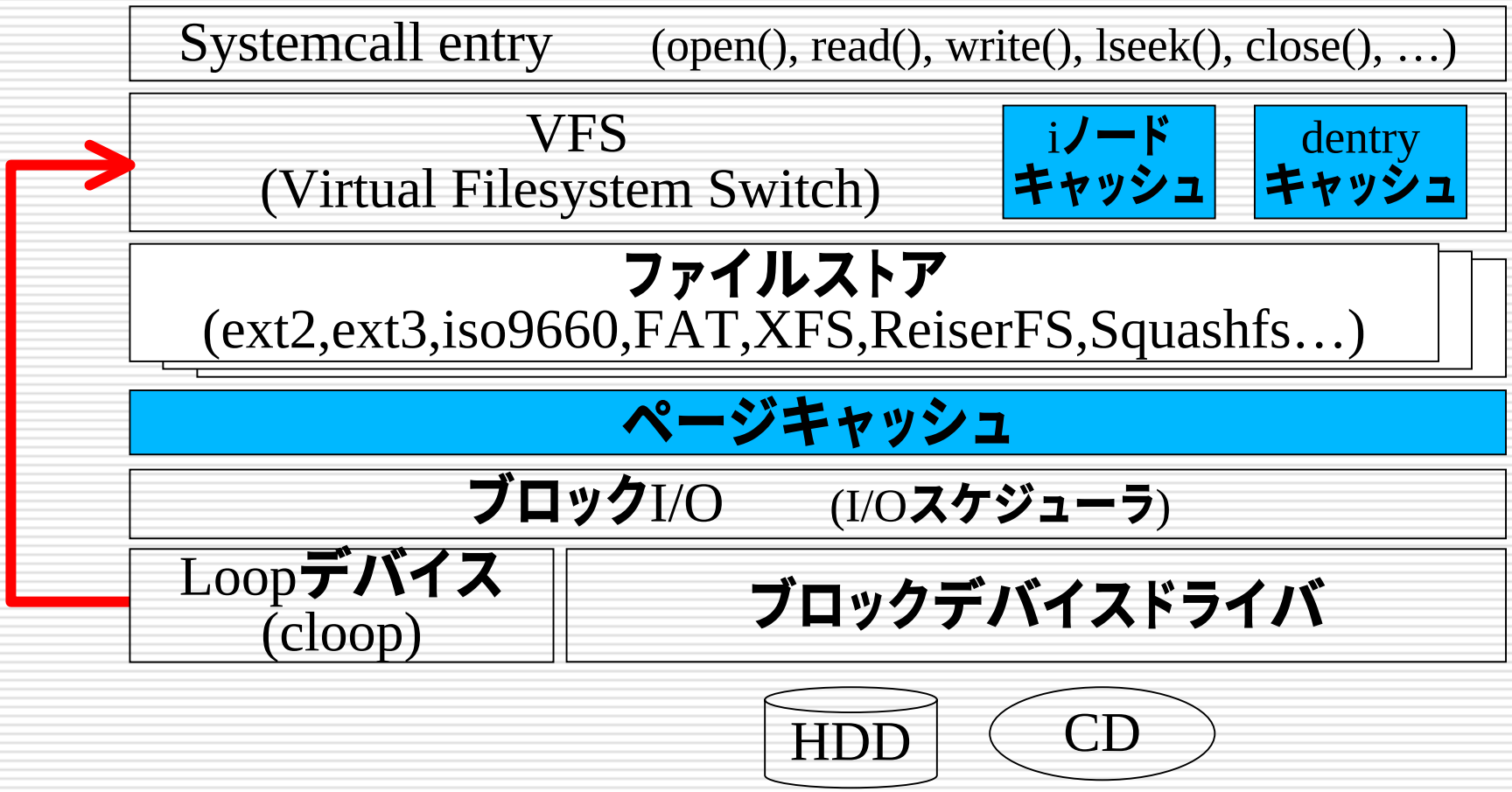
圧縮ファイルの中のファイルシステムにアクセスできるデバイス

分割圧縮ループデバイス(http-fuse cloop)



一つのファイルが, 一つのcloopブロックに対応
オンデマンドにダウンロードして利用

ブロックデバイスの階層構造



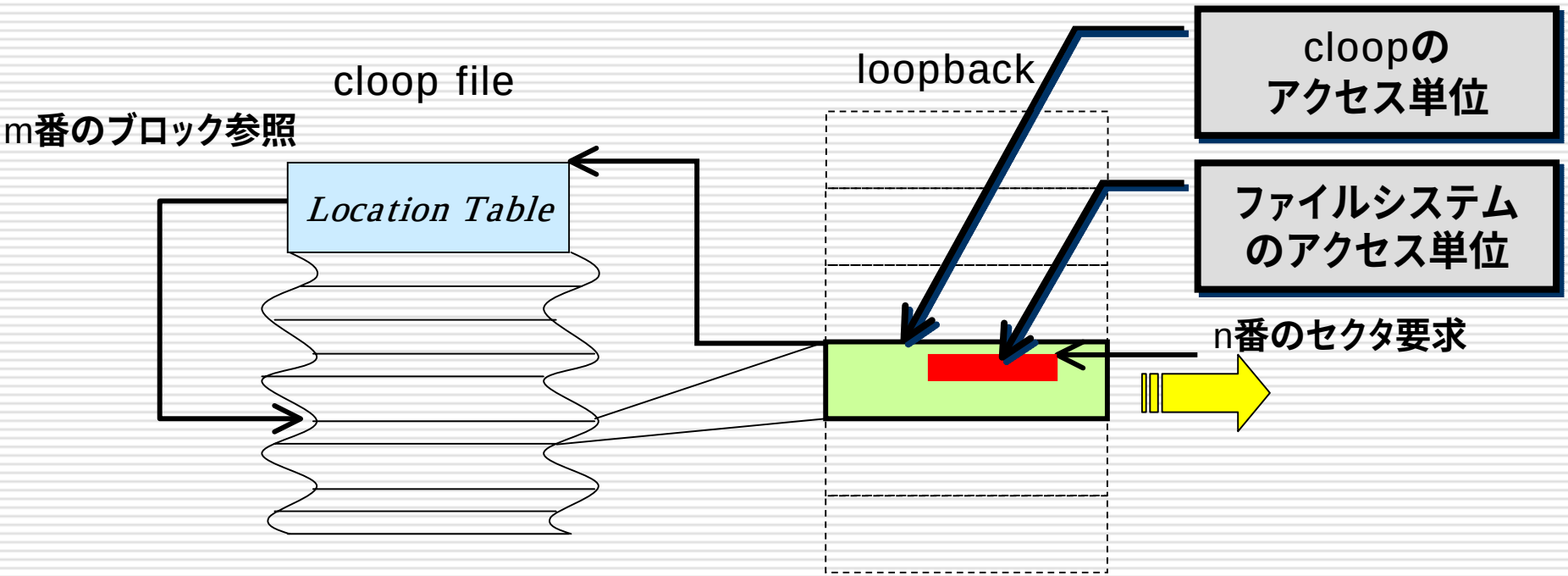
cloop上のファイルシステムは，何でも良い

cloopの役割とは

- 限られた容量に多くのコンテンツを格納可能
  - CD700MBに約2GBのコンテンツを格納
- ストレージとの転送速度差緩和
  - 低速な二次記憶装置との転送速度緩和
  - 狭いネットワーク帯域を緩和
- フレームワークを提供
  - 上層のファイルシステムは任意

cloopの問題点(1/2)

- 通常, cloopファイルは, cloop上のファイルシステムのアクセス単位より大きな単位で圧縮を行う



実際に必要なのはファイルシステムで要求された部分

cloopの問題点(2/2)

- アクセスがcloopブロック単位なので...
  - ほんの一部の情報参照でも，その部分を含む大きな範囲の読み込み･伸張が必要となる
    - KNOPPIX:64KB, HTTP-FUSE KNOPPIX:256KB
    - Ext2:4KB, iso9660:2KB
  - システム起動時には多くのファイルを必要とするため，その非効率性が無視できない

ファイルシステムの必要とする部分のみを
cloopブロックに詰めることができないか？

研究目的

cloop上のファイルシステムを最適化

cloopブロックの有効バイト率の向上

最も負荷の掛かる処理であるシステム起動時に
効率的な読み込み･伸張処理を行う

最適化するファイルシステム

ファイルシステムの条件

- cloop上のファイルシステムは読み込み専用
  - 耐障害性は考慮しない
- 最適化の単位はファイルより細粒度
  - システム起動のアクセス単位はファイルより小さい
- ファイルシステムの一部の変更が他に影響を及ぼさない
  - HTTP-FUSE サーバへの親和性

Ext2ファイルシステムを選択

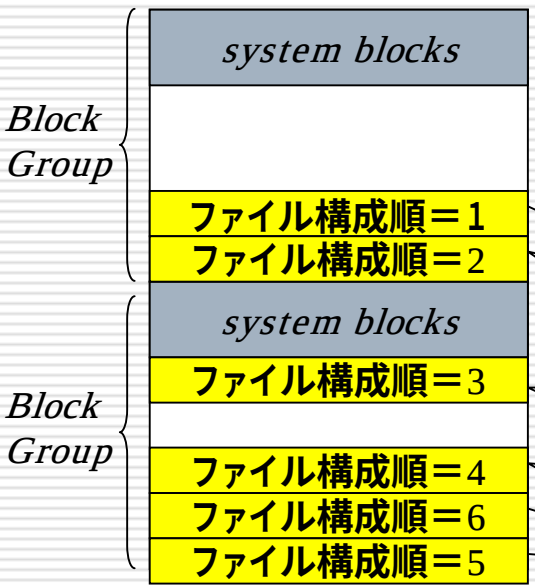
関連研究

効率的に情報を取得するために，
ファイルシステムに対し最適化処理を施す研究

- e2defrag
  Ext2**ファイルシステム用のデフラグツール**
- Ext2kai
  ライブCD**用のライトワンスファイルシステム**
- LCAT(LiveCD Acceleration Toolkit)
  cloop**ファイルを読み出し順に再配置**

DAV(Disk Allocation Viewer)

Ext2/Ext3ファイルシステムの断片化状態を可視化



□ フラグメンテーションの定義

(1) データブロックが、システムブロックを挟んでいる
(2) データブロックが、物理的に連続していない
(3) データブロックの順番が逆転している

＊ただし、(1)に関しては不可避であるため、他と区別

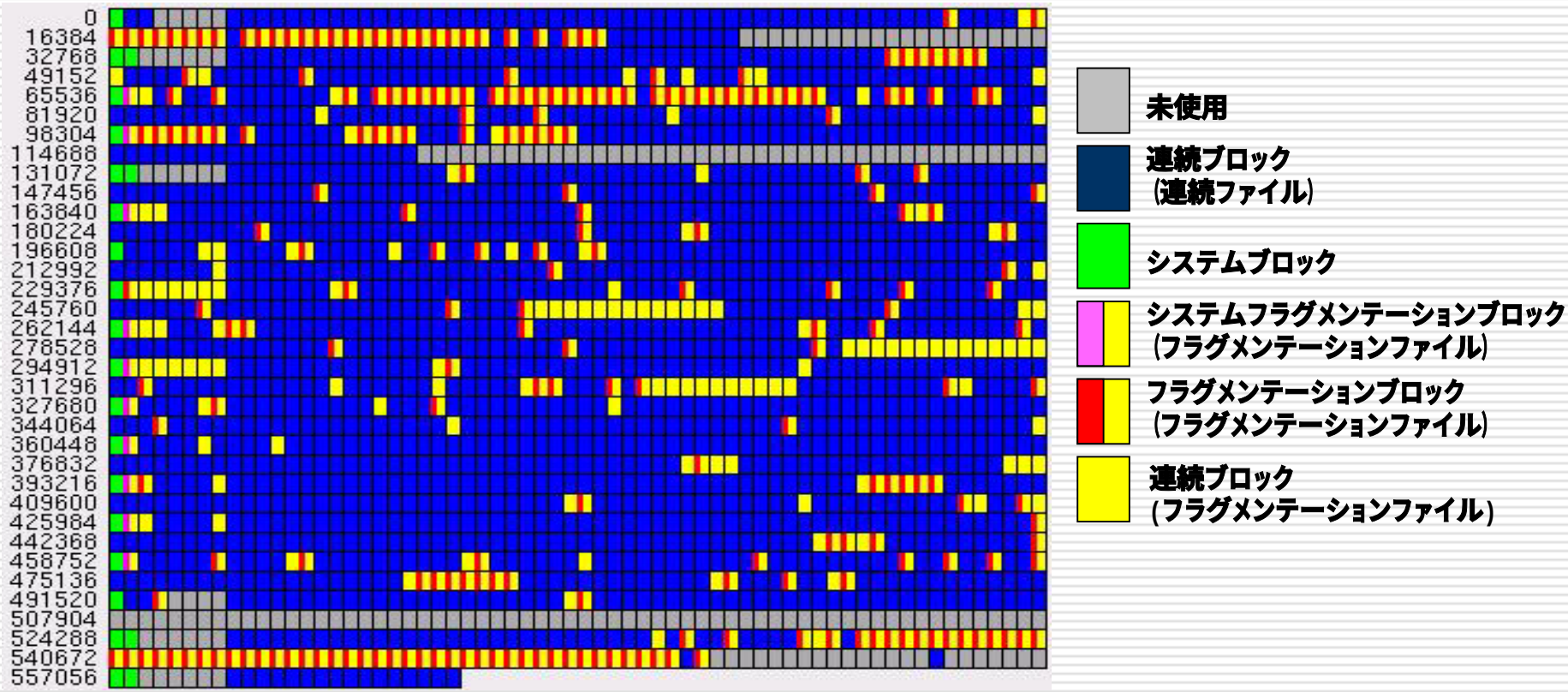
フラグメンテーションなしと定義

フラグメンテーションなし
フラグメンテーションあり(1)
フラグメンテーションあり(2)
フラグメンテーションあり(3)

DAVによる可視化

cloopファイルを伸張しイメージを可視化

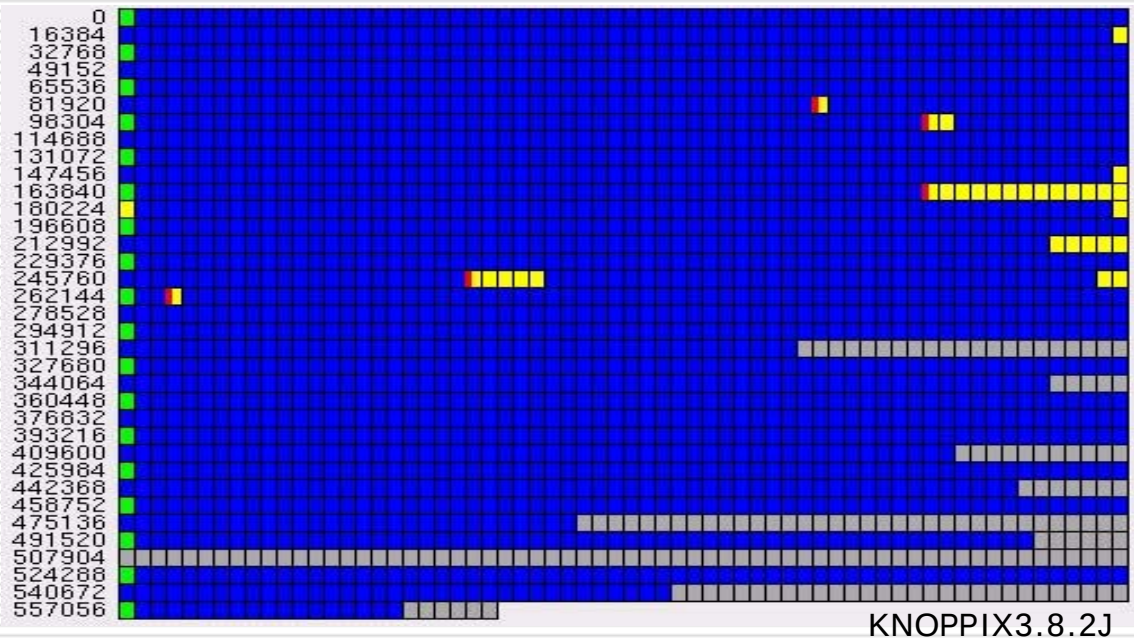
KNOPPIX3.8.2日本語版



関連研究

e2defrag※

- デフラグメンテーションを行うツール
- ファイル分断によるcloopブロック参照数を抑制

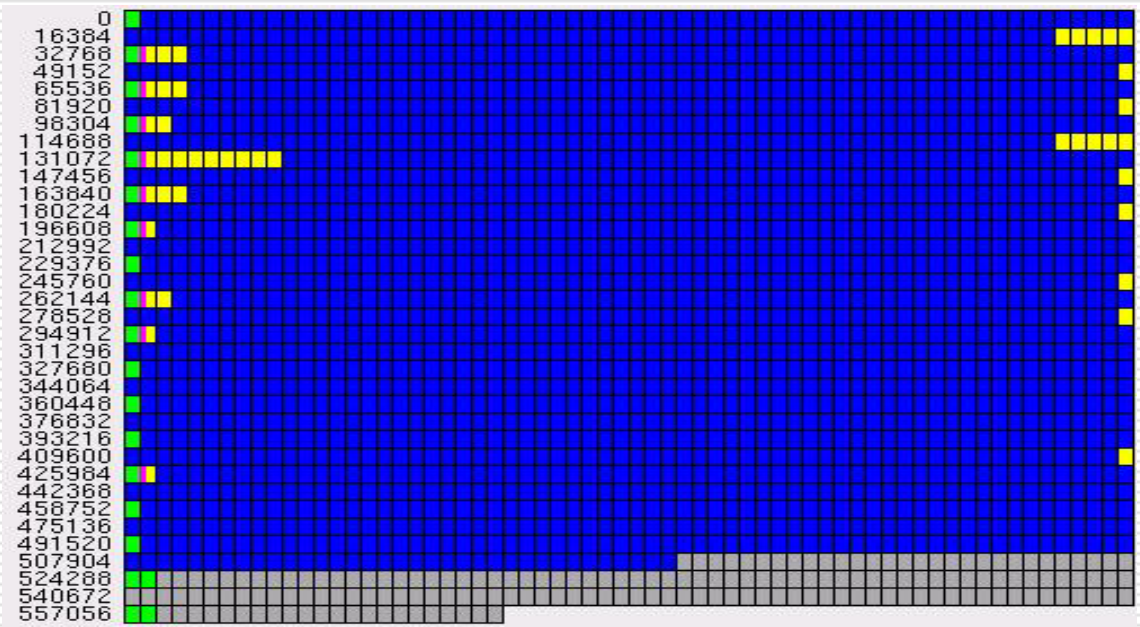


※http://e2compr.sourceforge.net/defrag.html

関連研究

Ext2kai※

- ファイル単位で最適化
- 指定順に，断片化無く，スパース無しで配置

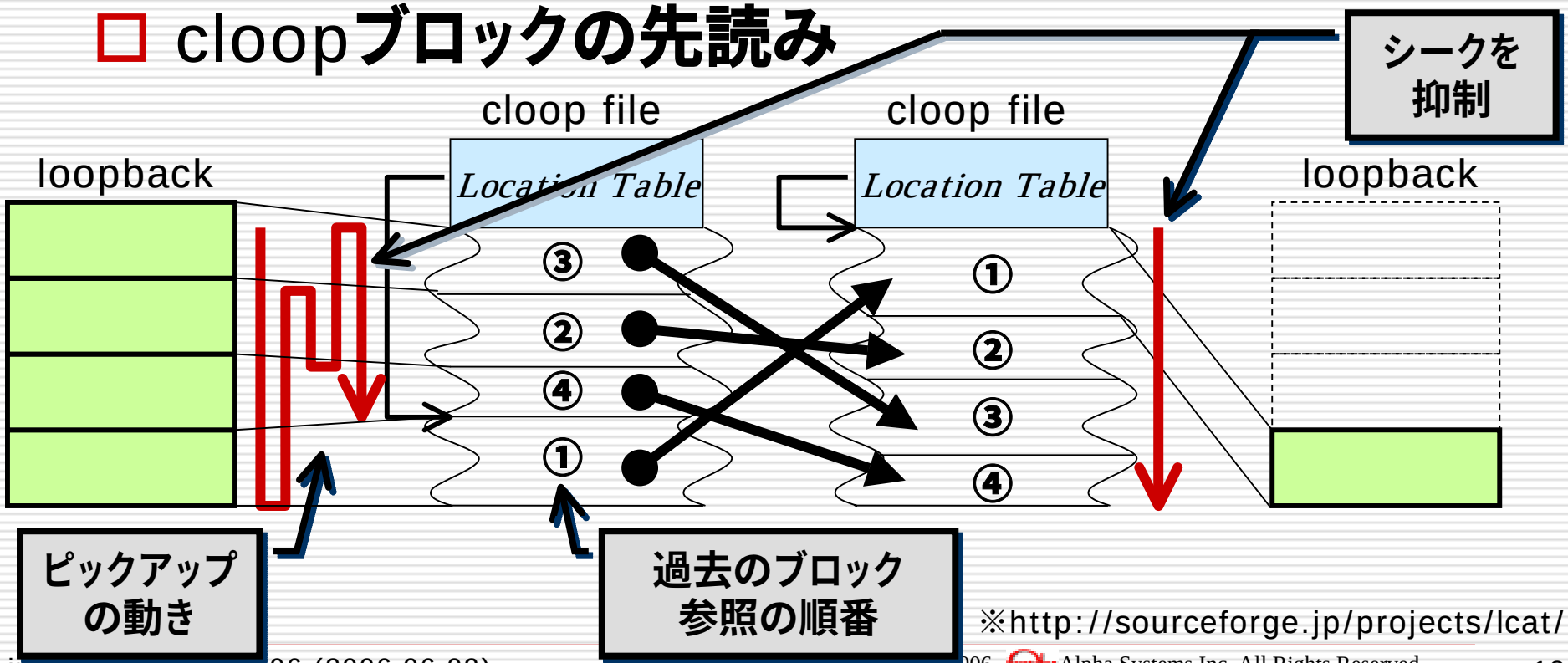


KNOPPIX3.8.2J

※http://www.alpha.co.jp/knoppix/download/paper/FIT2005_9_paper.pdf

関連研究
LCAT※

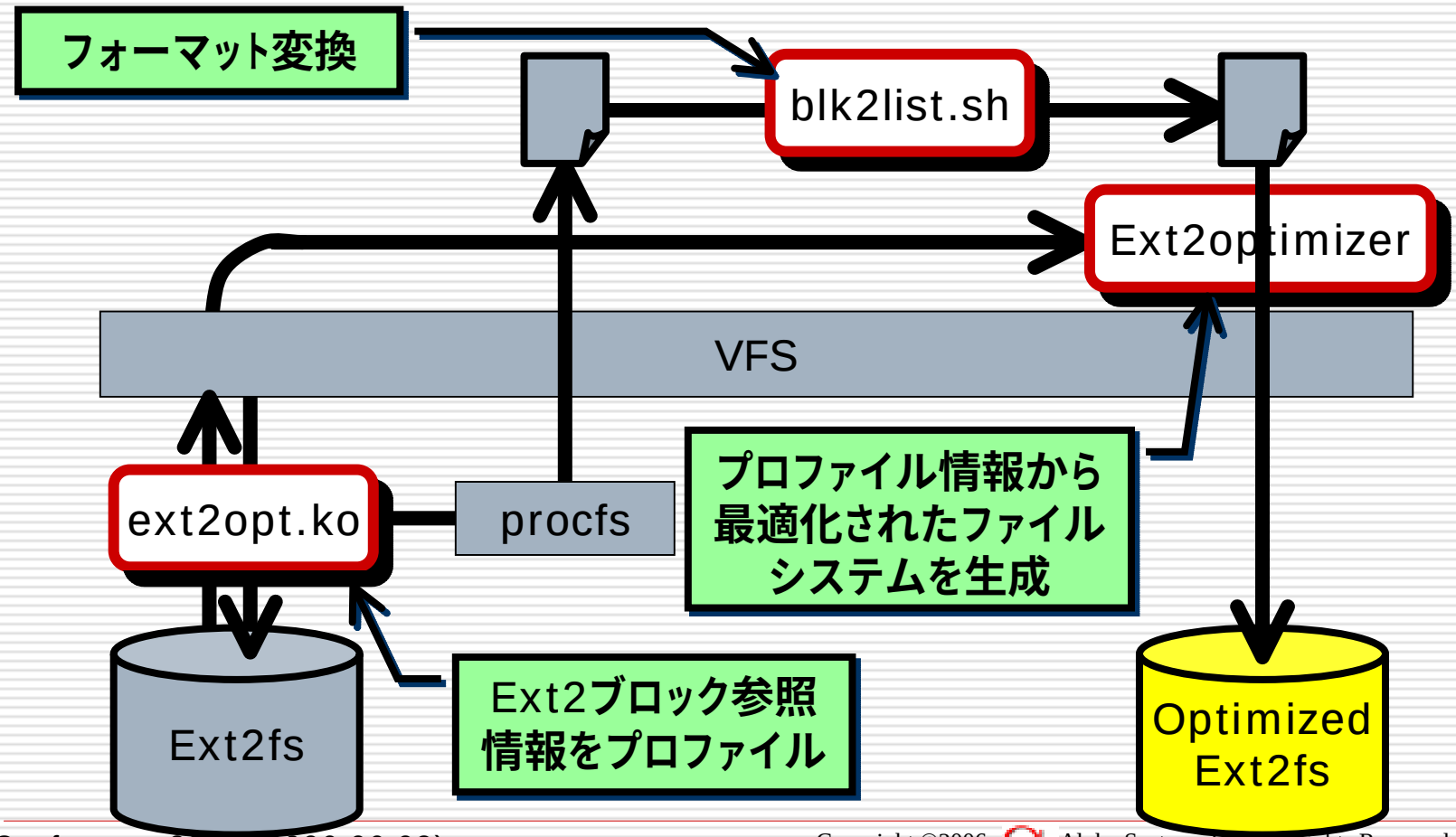
- cloopファイルをcloopブロック単位で最適化
- CDドライブの特性から外周配置
- cloopブロックの先読み



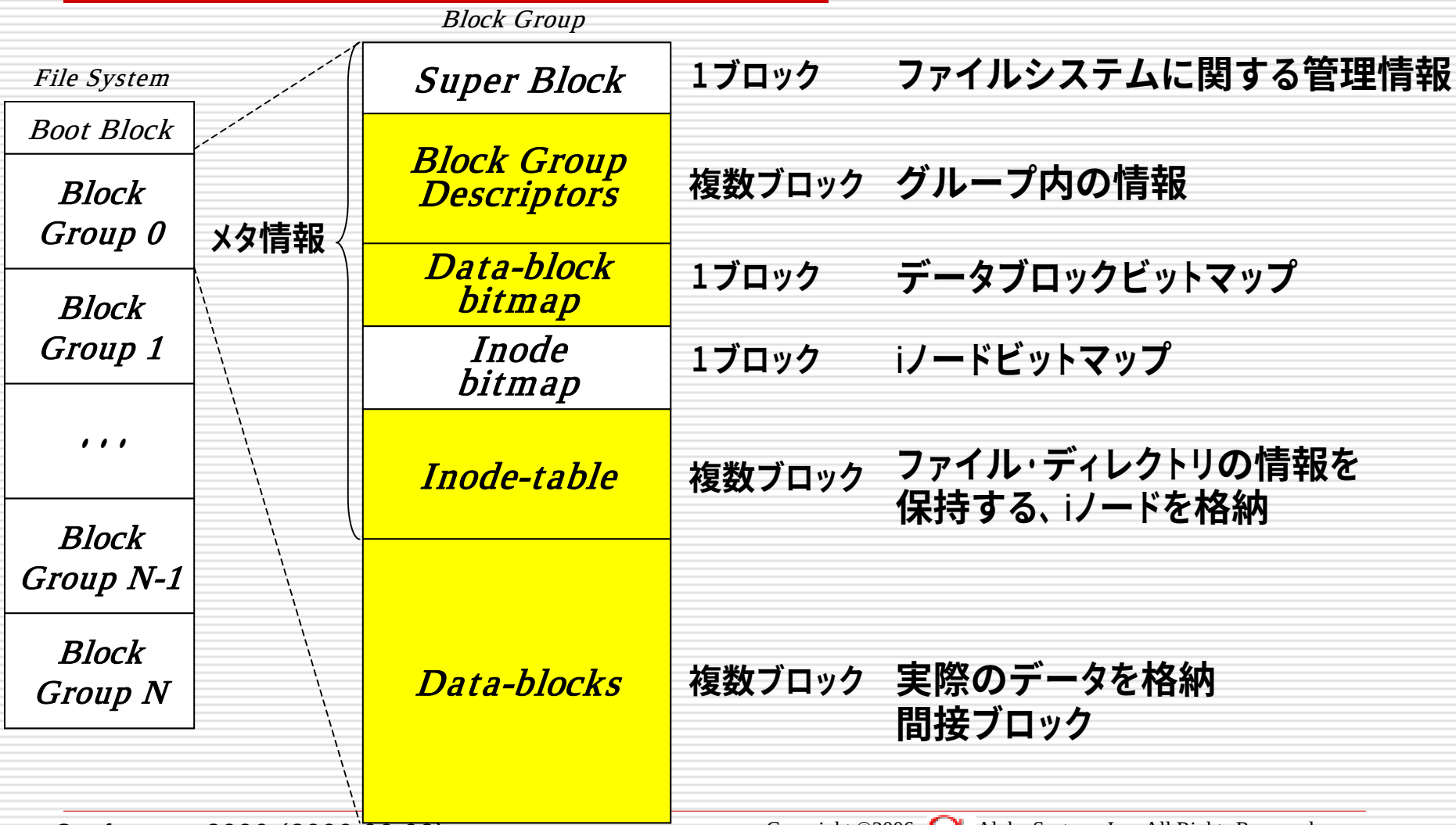
※http://sourceforge.jp/projects/lcat/

Ext2optimizerの概要

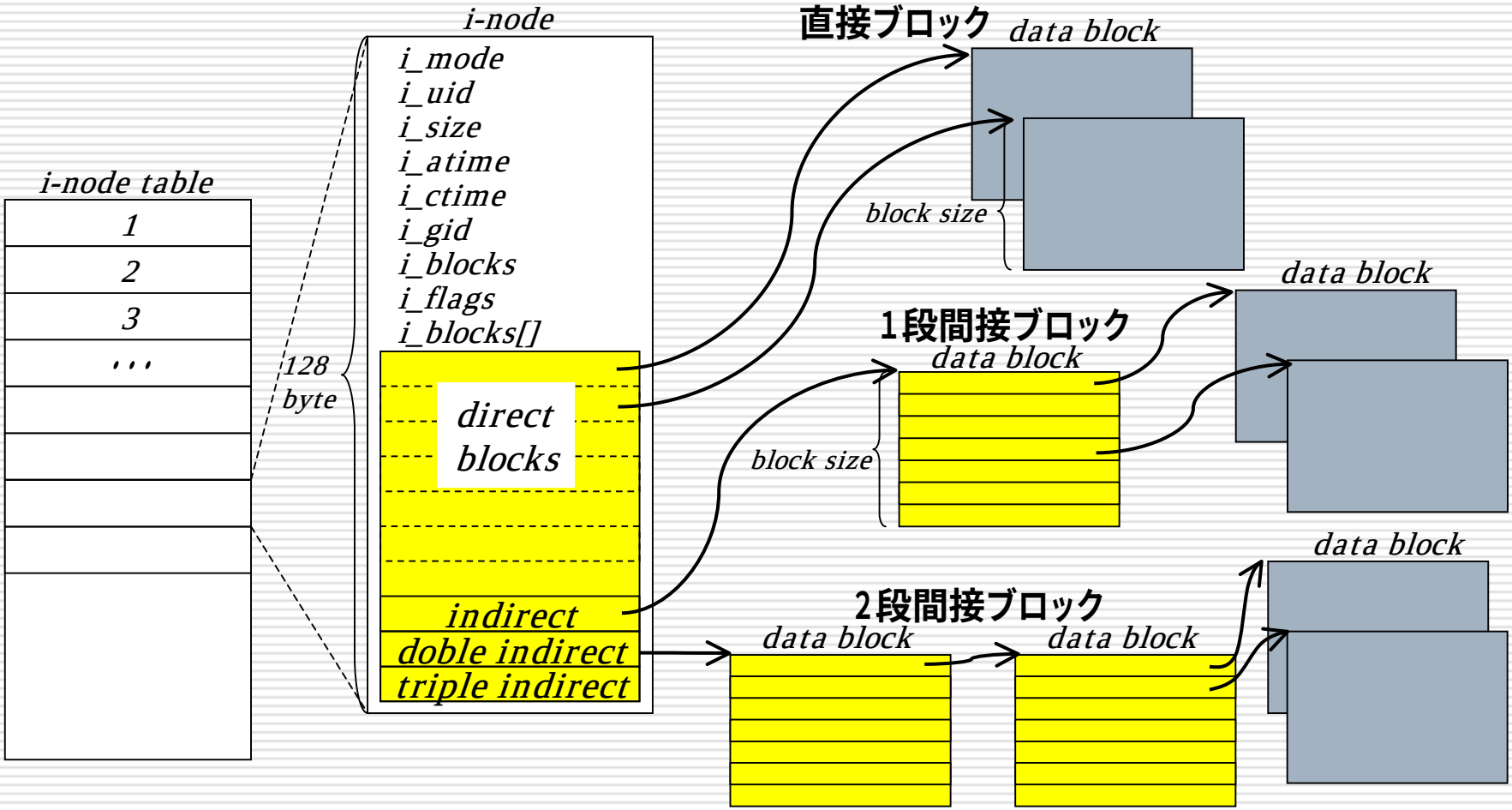
Ext2ファイルシステムの最小単位(以後Ext2ブロック)で最適化



Ext2ファイルシステム構成(1/2)



Ext2ファイルシステム構成(2/2)



メタ情報の書き換えと，データブロックの移動による最適化

Ext2optimizerの実装(1/3)

- Ext2opt.ko
  - ext2_get_block(), ext2_get_branch()から**それぞれ直接ブロック番号, 間接ブロック番号を深度別, 時系列に蓄積**(bh->b_blocknr)
  - procfs**経由で情報を取得**

```
dev = dfed2880
readblock = 24828 / 50000
-------------------------------------------------------------
   dev         no      direct   indict1  indict2  indict3
-------------------------------------------------------------
dfed2880         0         383
dfed2880         1       80202
dfed2880         2      118202
dfed2880         3       80379
dfed2880         4       8037a
dfed2880         5       8037b
dfed2880         6       8037c
dfed2880         7       8037d
dfed2880         8       8037e
dfed2880         9       8037f
dfed2880        10       80380
dfed2880        11       80381
dfed2880        12       80382
dfed2880        13       80383
dfed2880        14       80384
dfed2880        15       80386     80385
dfed2880        16       80387     80385
dfed2880        17       80388     80385
```

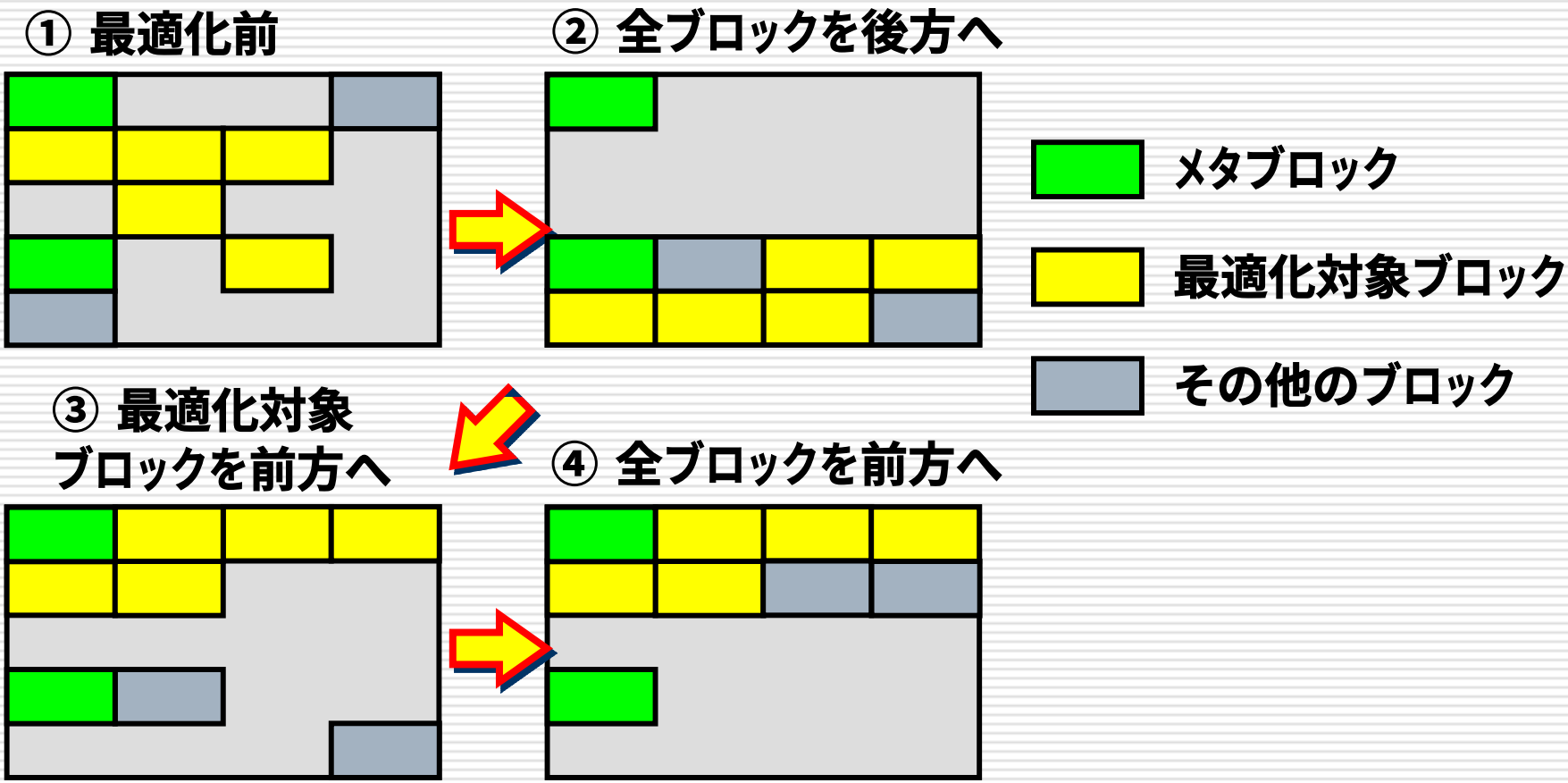
Ext2optimizerの実装(2/3)

- Blk2list.sh
  - 二次元情報を一次元情報に変換
- Ext2optimizer
  - 一次元情報に基づきファイルシステムを最適化
  - メタ情報と間接ブロックを操作
  - 操作対象は,
    - 通常ファイル
    - ディレクトリ
    - 60byte以上のシンボリックリンク
    - それ以外はiノードが情報を保持
  - 制御情報を最適化後,
    最適化済みファイルシステムを出力

ファイル種別	説明
0	未定義
1	通常ファイル
2	ディレクトリ
3	キャラクタ型デバイス
4	ブロック型デバイス
5	名前付きパイプ
6	ソケット
7	シンボリックリンク

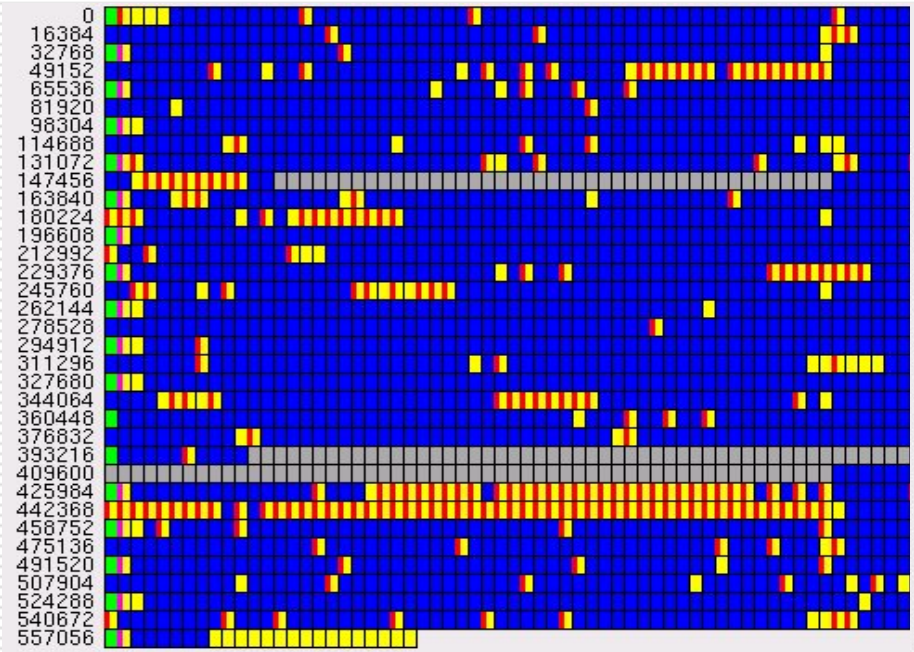
Ext2optimizerの実装(3/3)

Ext2optimizerの最適化の様子

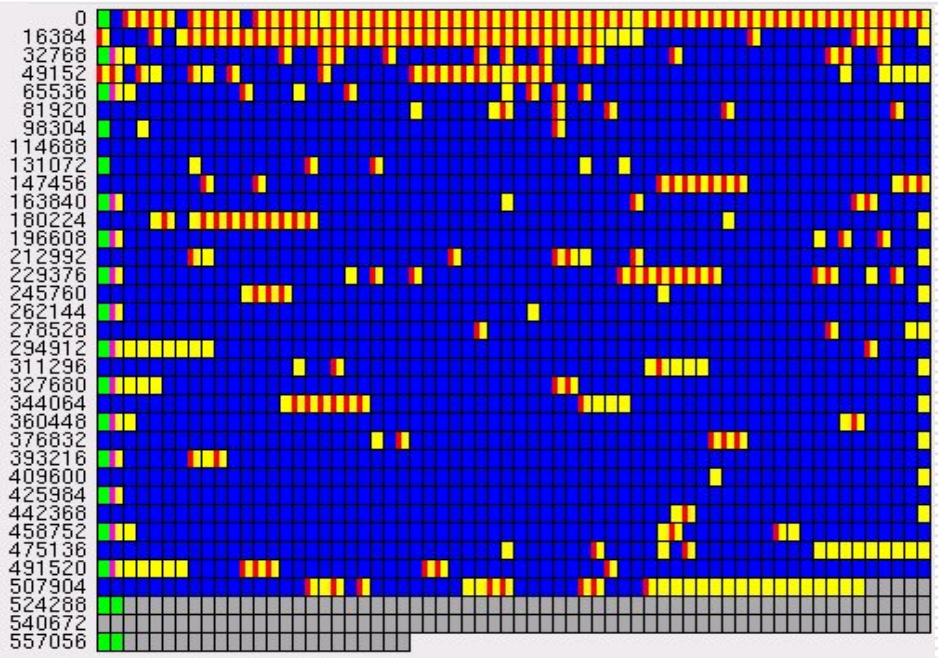


Ext2optimizerによる最適化前後のイメージ

最適化前



最適化後



システム起動に必要なブロックがイメージ前半に集約

関連研究と本研究の相違点

- 最適化の単位がExt2ブロック単位
  - Ext2kaiがファイル単位(LCATがcloopブロック単位)で最適化を行うのに対し，より粒度の小さいExt2ブロック単位で行う
- 最適化のためのフラグメンテーションを発生させる
  - ディスクアクセスの効率化は一般的にファイル断片化を防ぐ傾向にあり，e2defrag，Ext2kaiでは，フラグメンテーションさせないようにしている
- 必要な部分を可能な限り集約配置
  - Ext2kaiでは，必要な部分を含んだファイル単位で配置していたが，ファイルの本当に必要な部分のみで集約配置できる

評価

- 評価は，bootプロンプトからKDEスプラッシュが消えるまでのシステム起動について行った

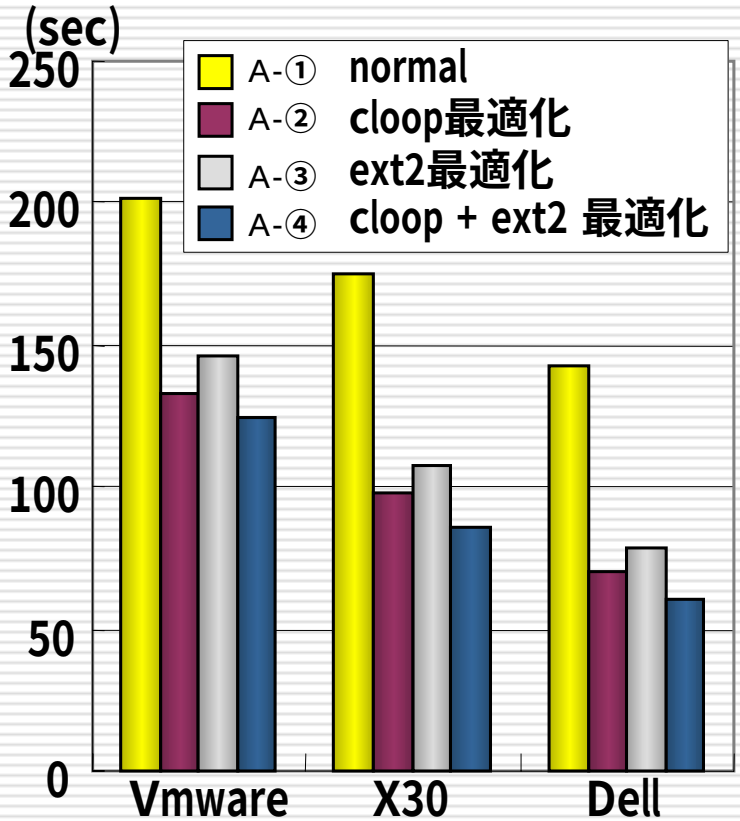
- Ext2optimizerにより最適化されたイメージファイルをライブCDに適用
  - KNOPPIX
  - HTTP-FUSE KNOPPIX

評価1　KNOPPIX

- KNOPPIX3.8.2日本語版をベース
- Kernel-2.6.11にExt2optパッチを適用
- VMware5.0(build-13124)にてプロファイル
  (Ext2ブロック，及びcloopブロック)

ID	説　明
A-①	KNOPPIX3.8.2日本語版
A-②	cloop最適化
A-③	Ext2最適化
A-④	Ext2最適化 + cloop最適化

システム起動時間(KNOPPIX編)



CPU	P4 3.2GHz	P3 1.06GHz	P4 3.2GHz
MEM	384MB	768MB	1.5GB
CD-Drive	40x	24x	40x

- Cloopブロックを60%に抑制
- メタ情報以外のCloopブロックはシーケンシャル配置
- LCATでメタ情報を最適配置
- 更に約10秒の高速化

実装レイヤが異なるため，
相性の良い最適化処理を行えた

評価2　HTTP-FUSE KNOPPIX

- HTTP-FUSE-KNOPPIX4.0.2をベース
- cloopファイルはKNOPPIX4.0.2を最適化
- 分割サイズは, 64,128,256,512KB
- VMware5.0(build-13124)にてプロファイル
- サーバ･クライアント間は, ハブで接続

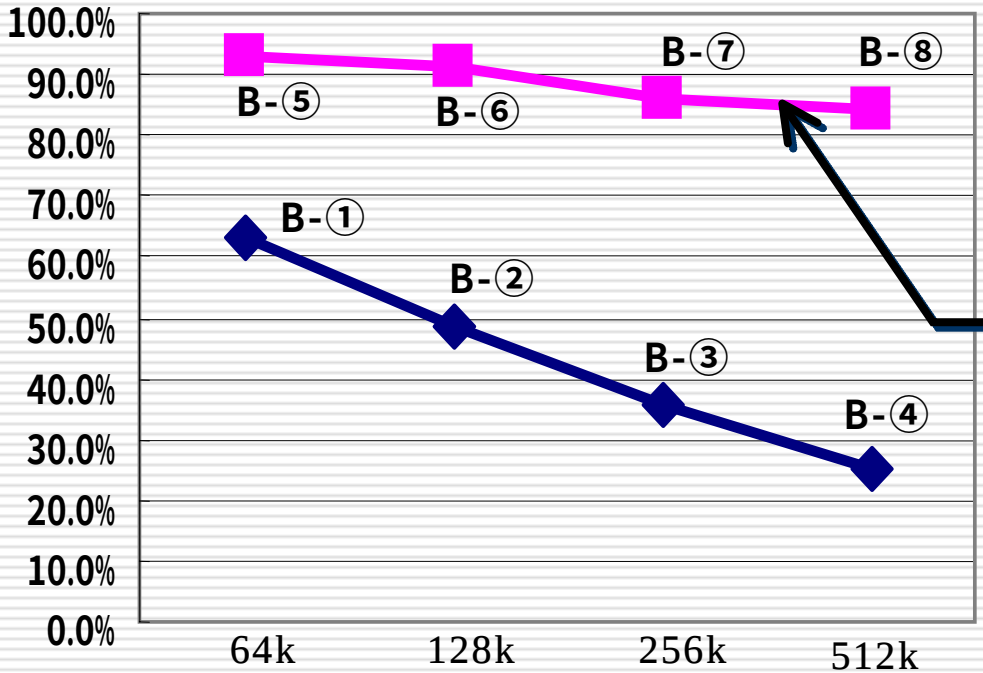
サーバに配置したHTTP-FUSE環境

Block Size	ファイルシステム	
	Normal	optimize
64	B-①	B-⑤
128	B-②	B-⑥
256	B-③	B-⑦
512	B-④	B-⑧

Webサーバ:
CPU:P4　2GHz　MEM:512MB　NIC:100Mbps
Kernel-2.6.11　　Apache-1.3.33-8

クライアントPC:
CPU:P3　1.06GHz　MEM:768MB

cloopブロックの平均有効バイト率



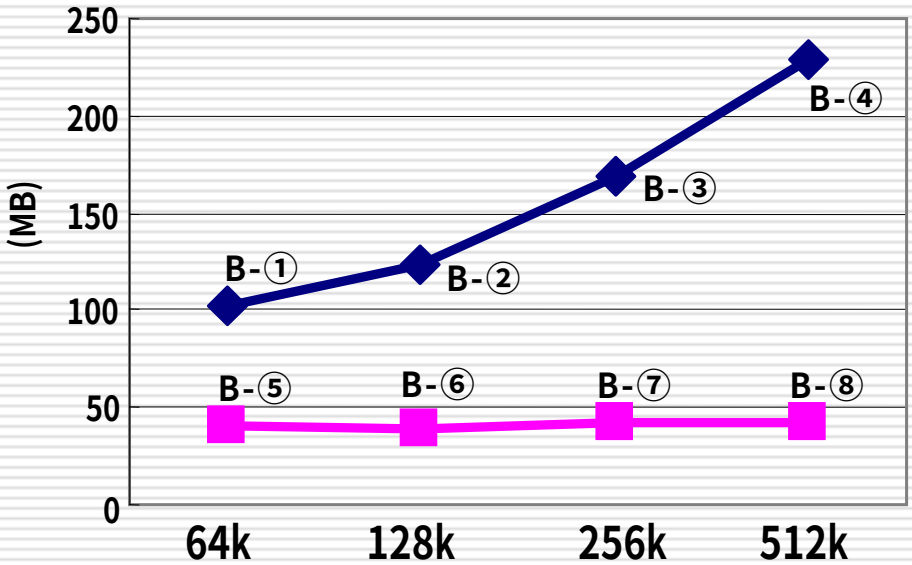
$$\frac{\text{ユニークなext2ブロックの総容量}}{\text{ユニークなcloopブロックの総容量}} \times 100$$

メタ情報は最適配置されないため，cloopブロックの有効バイト率は100%とならない

切り出しサイズの増大の影響を殆ど受けず，高い平均有効バイト率を維持している

ダウンロードファイルの合計サイズ

/tmp/blocksにダウンロードされた圧縮ブロックファイルの総容量

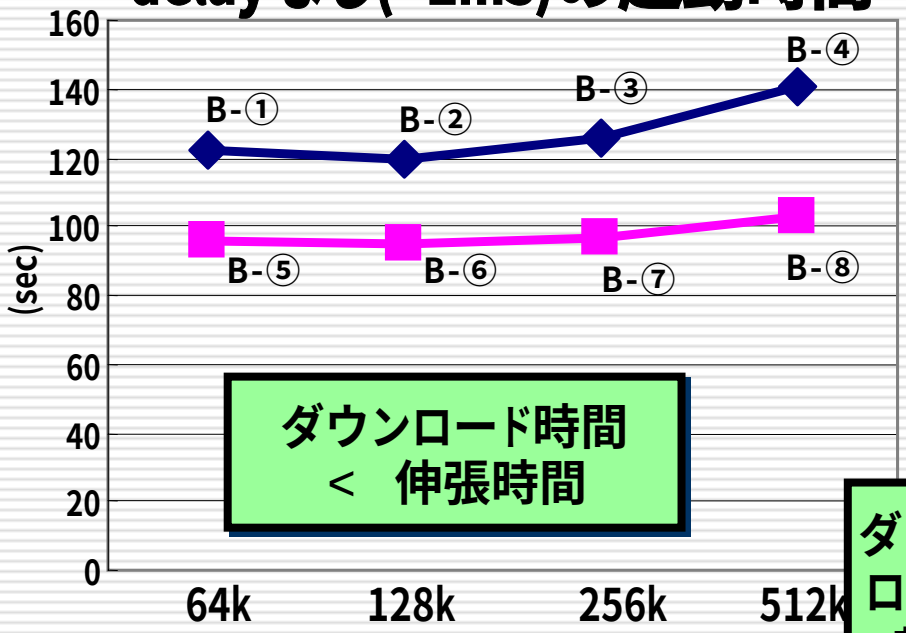


ファイルシステムが最適化されているため，
切り出しサイズの増大に影響を受けにくい

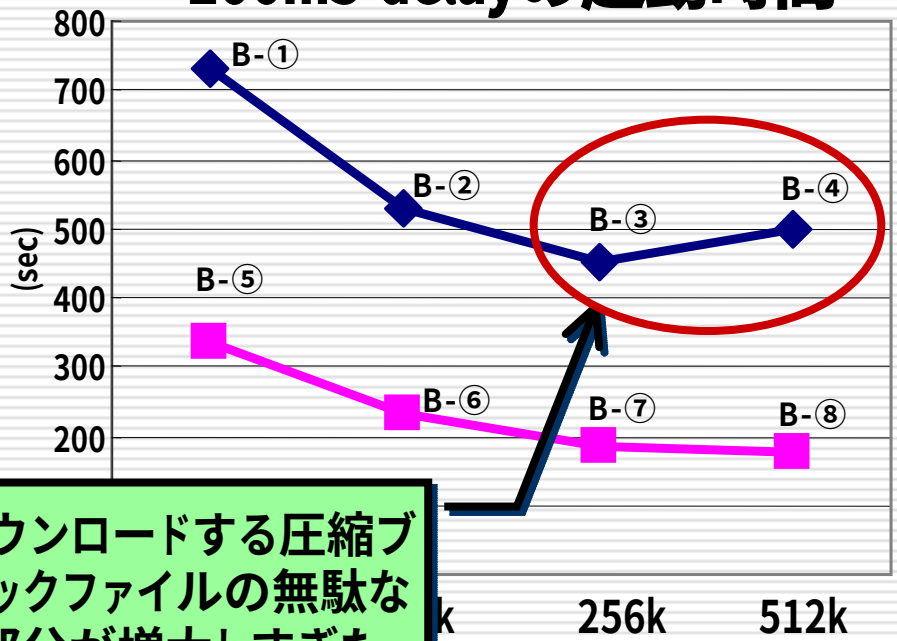
システム起動時間(HTTP-FUSE編)

netem※によるレイテンシの影響についても調査

delayなし(<1ms)の起動時間



100ms delayの起動時間



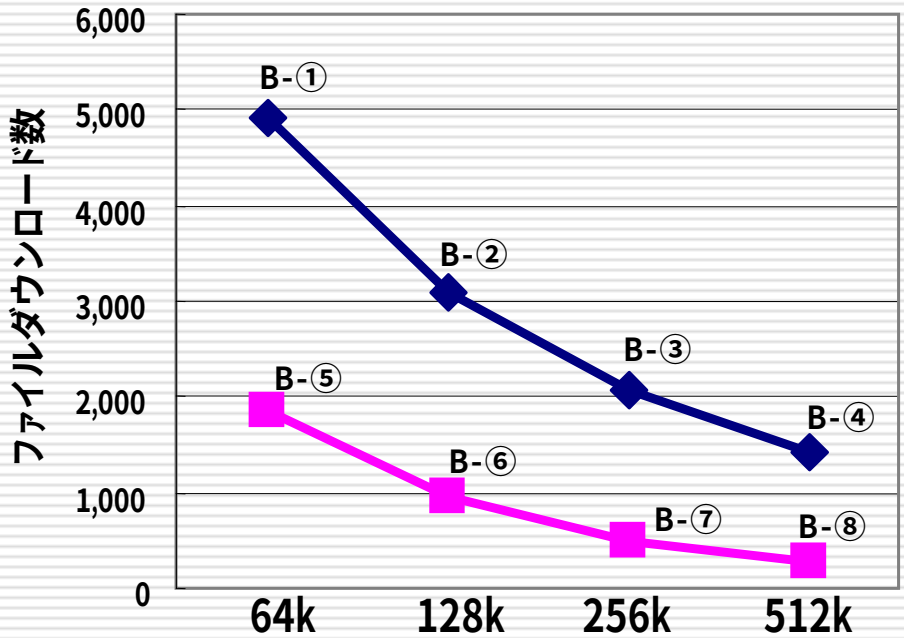
ダウンロードするファイルサイズが大きい程，レイテンシの影響は少ない

切り出しブロックサイズを増大しても無駄な部分はダウンロードしなくてよい

コネクション数：1

スケーラビリティについて

HTTPサーバへのGETリクエスト数は最大80％削減



ネットワークトラフィック，HTTPサーバ負荷の低減

まとめ

- Ext2ファイルシステムを最適化するExt2optimizerを開発した．また，それにより最適化されたファイルシステムをKNOPPIX，HTTP-FUSE KNOPPIXに適用し，簡単な実験を行った
- 最適化されたファイルシステムは既存のExt2.koでアクセス可
- システム起動に必要なcloopブロックの絶対数を抑制
- LCATとは適用レイヤの違いにより最適化処理が補完
- ネットワークブートでは，HTTPサーバの負荷やネットワークトラフィックを抑制し，スケーラビリティの確保が期待できる

今後の展開

- iノードなどのメタ情報の最適化を可能にすることで，更なる効率化が期待できる
- デフラグ的な最適化手順を踏むことで，HDDインストールされたシステムで起動に最適化されたイメージが構築できる

ご静聴ありがとうございました.

KNOPPIX カスタマイズサービス

- 個人情報漏洩対策をご検討の方、Winnyにお悩みの方、必見！

モバイル向けシンクライアントシステム「あんしんモバイル」を展示！
株式会社リクルートエージェント様の導入事例を詳しく紹介します！

情報漏洩対策 ご相談会を開催をいたしております。情報漏洩対策をご検討中の方。担当者がご相談を承ります。

- KNOPPIXを無料配布！

「Accelerated-KNOPPIX」
最新版1.1のCD-ROMを限定無料配布

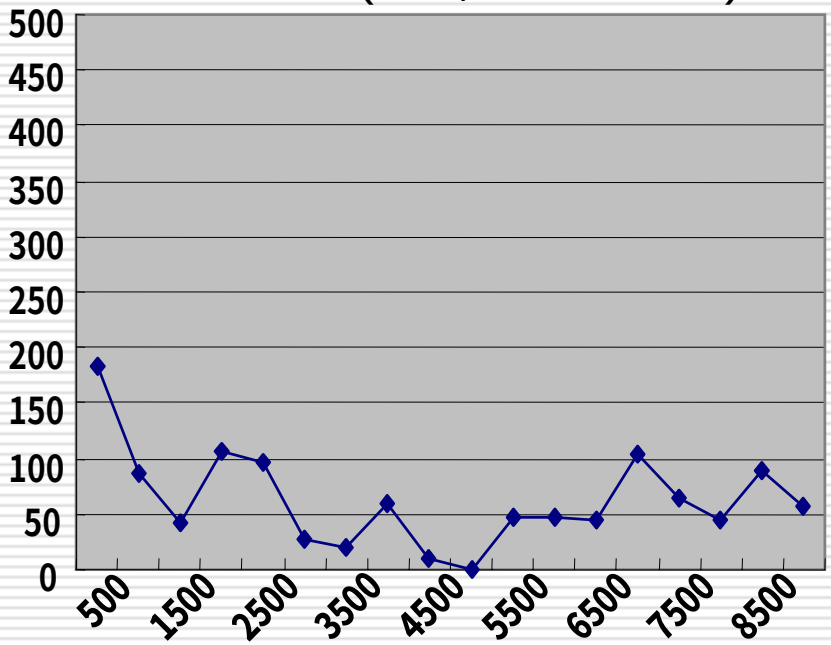
株式会社アルファシステムズ

cloopブロックの集約状況

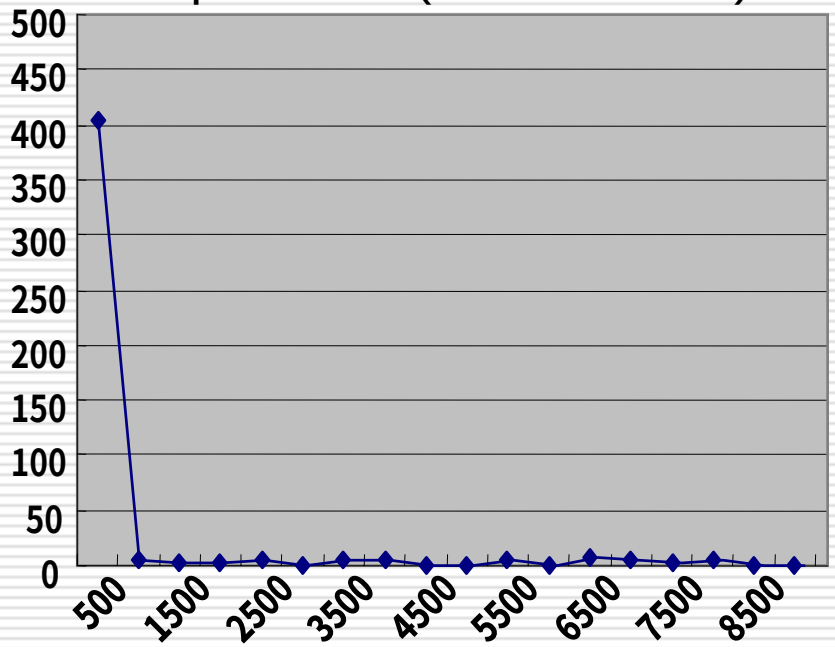
システム起動に必要なcloopブロックの集約状況

KNOPPIX4.0.2　256KB　8,800ブロック中

Normal (計1,125ブロック)



Optimized (計461ブロック)



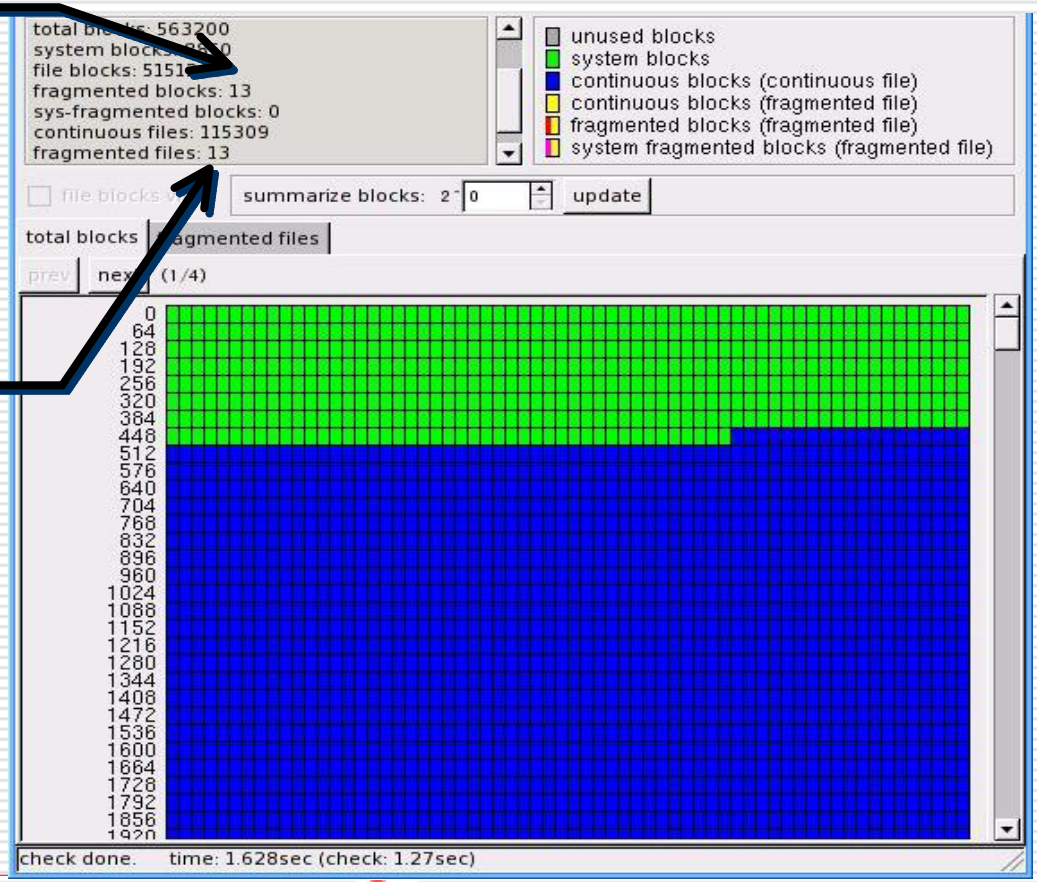
500ブロック目以降は，メタ情報を含むcloopブロックのみ

Ext2optimizerによる最適化(1/3)

e2defragにてデフラグを行った

フラグメンテーション ブロック数
13

フラグメンテーション ファイル数
13

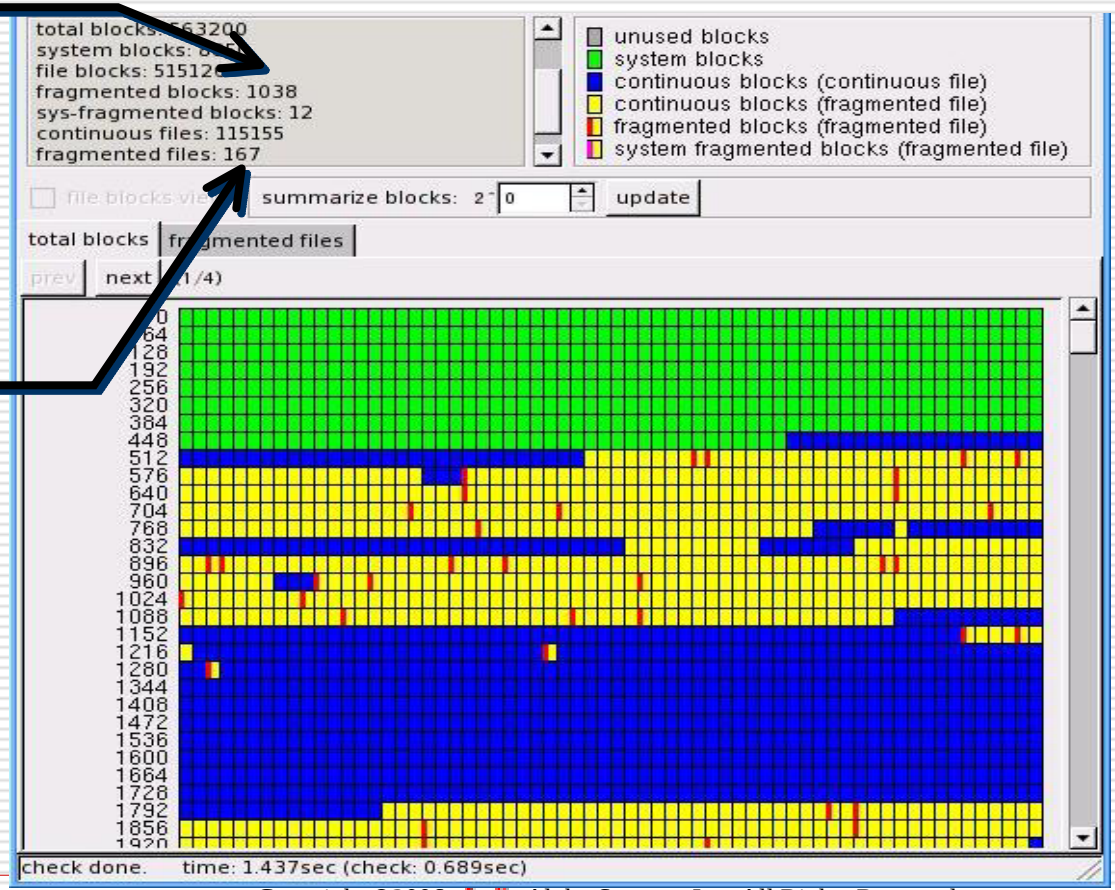


Ext2optimizerによる最適化(2/3)

Ext2optimizerにより最適化

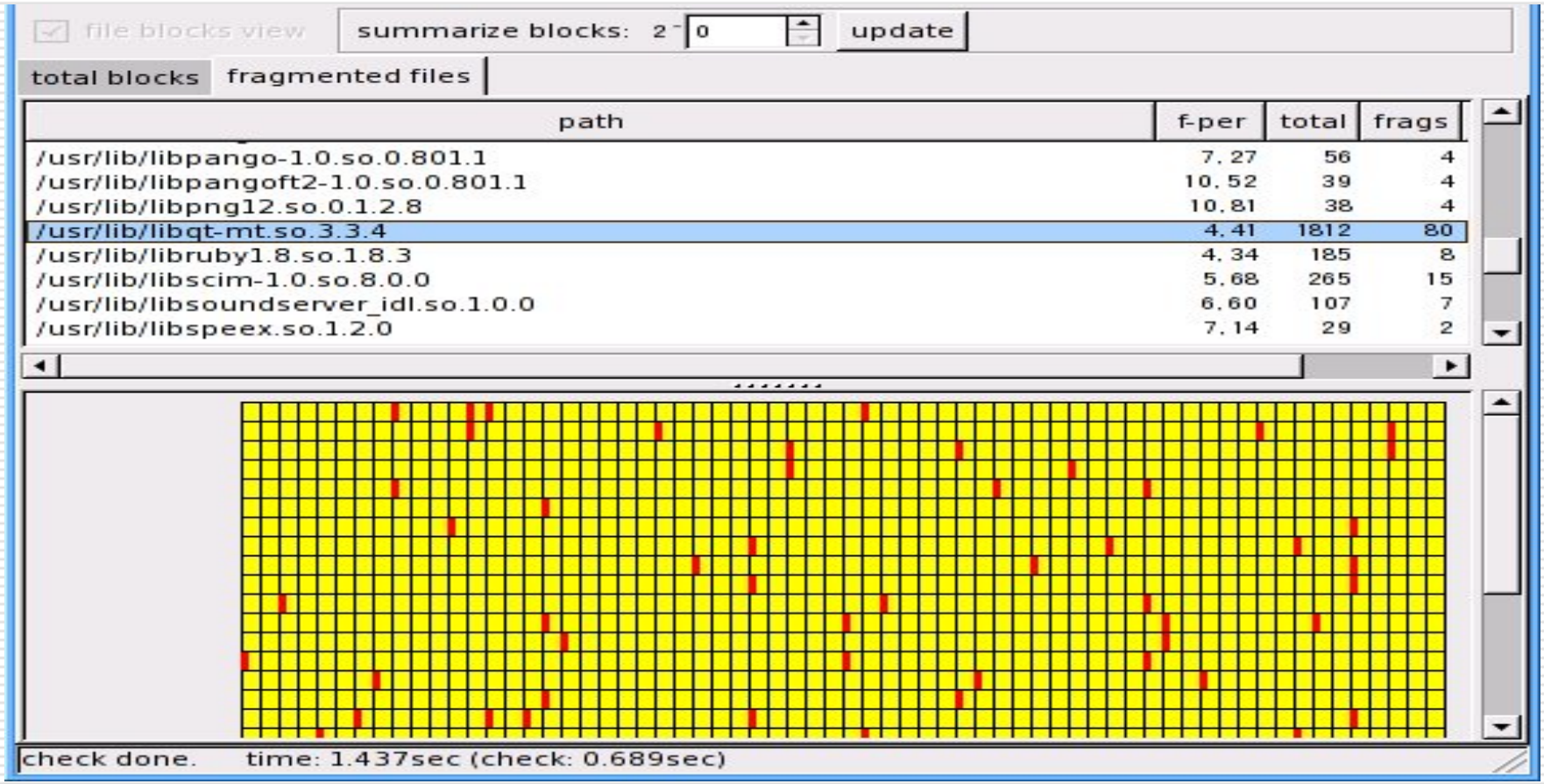
フラグメンテーション
ブロック数
1038

フラグメンテーション
ファイル数
167



Ext2optimizerによる最適化(3/3)

最適化後にフラグメンテーションするファイルの大半はライブラリ



Ext2fs ディレクトリ構造

```
#define EXT2_NAME_LEN 255

struct ext2_dir_entry_2 {
        __le32  inode;                  /* Inode number */
        __le16  rec_len;                /* Directory entry length */
        __u8    name_len;               /* Name length */
        __u8    file_type;
        char    name[EXT2_NAME_LEN];    /* File name */
};
```

ファイル種別	説明
0	未定義
1	通常ファイル
2	ディレクトリ
3	キャラクタ型デバイス
4	ブロック型デバイス
5	名前付きパイプ
6	ソケット
7	シンボリックリンク

inode	rec_len	name_len	file_type	name
21	12	1	2	. ¥0 ¥0 ¥0
22	12	2	2	. . ¥0 ¥0
35	28	4	2	h o m e ¥n ¥n ¥n ¥n
57	12	3	2	u s r ¥n
61	12	3	1	v o o ¥n