HTTP-FUSE KNOPPIX

須崎有康，八木豊志樹，飯島賢吾[1)]，丹英之[2)]
産業技術総合研究所[1)]，アルファシステムズ[2)]

はじめに

- HTTP-FUSE KNOPPIX とは
- ”KNOPPIX” の仕組みのおさらい
- 分割圧縮ファイルによるループバックデバイス
- Internet 対応方法
- 関連研究、今後の予定

HTTP-FUSE KNOPPIX とは

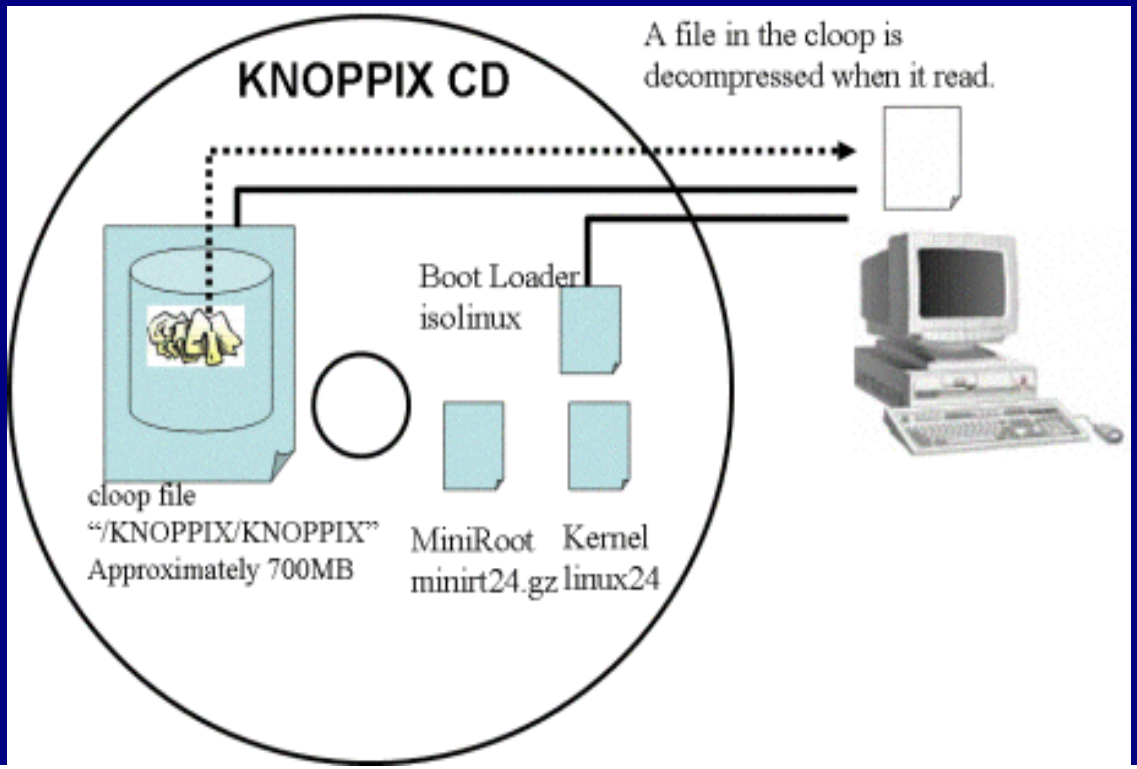
- 「Internet から起動する KNOPPIX」
- 大目的（予算要求用看板）
  - 現在の OS インストール／ブート方法の改革
    - OS インストールでは同じもののコピー
    - 内蔵デバイスに OS を固定するのは止めよう
    - Internet のどこかに OS はある
- 小目的（実利）
  - KNOPPIX のカスタマイズで CD を作り直すのは止めよう。

KNOPPIXのおさらい

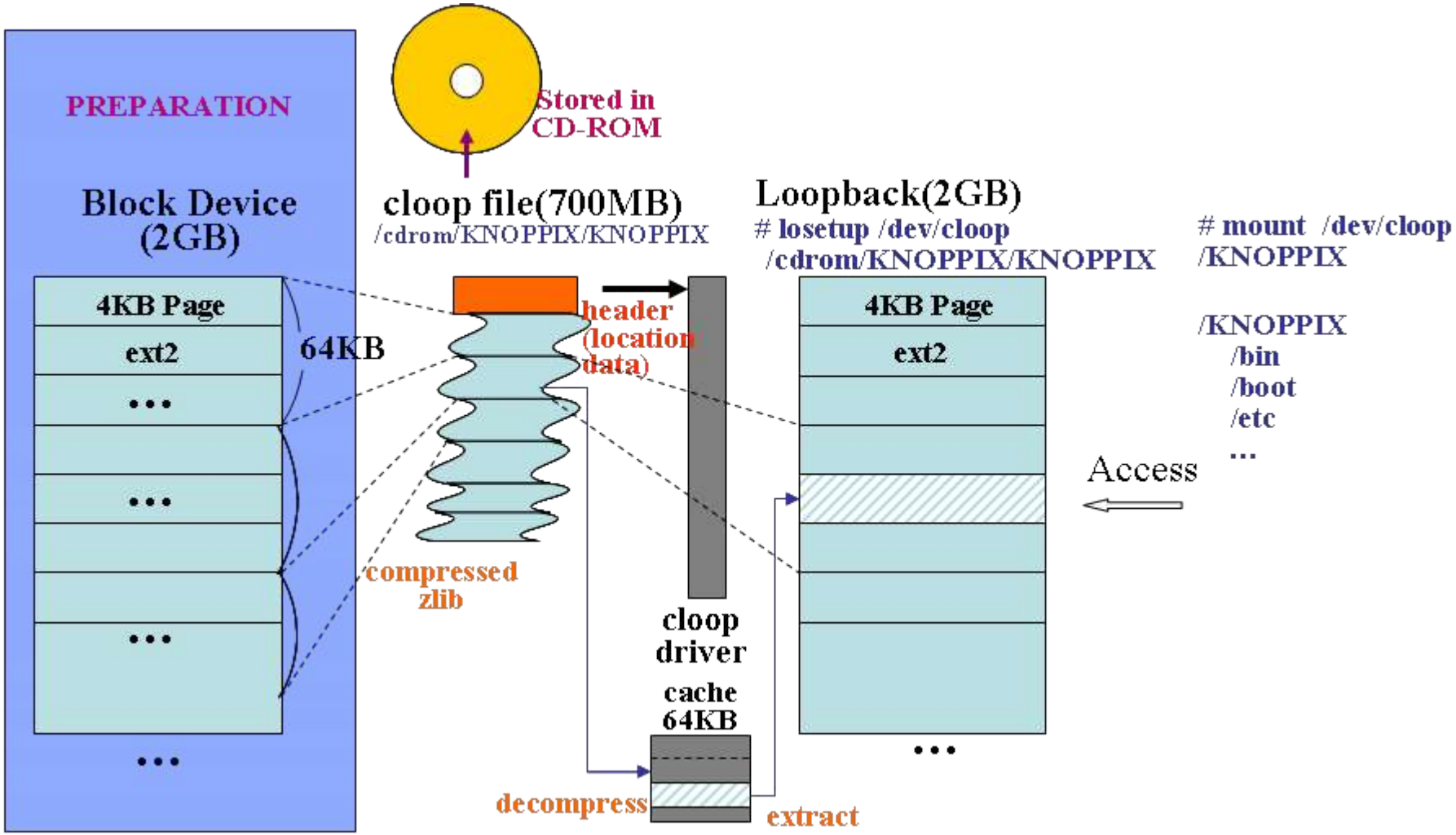
- ブート時のデバイスの自動認識/ドライバの組み込み機能(Autoconfig)に優れている
  - anonymous user がネットワークブートするための必要条件
- ルートファイルシステムが圧縮ループバックデバイス(cloop)に格納され、扱いやすい。
  - ただし１巨大ファイルで、更新が面倒臭い。

KNOPPIX の特徴（cloop）

- 700M CD-ROM に圧縮ループバックデバイス CLOOP を使い、 2G に拡張。
- 圧縮には zlib を利用。



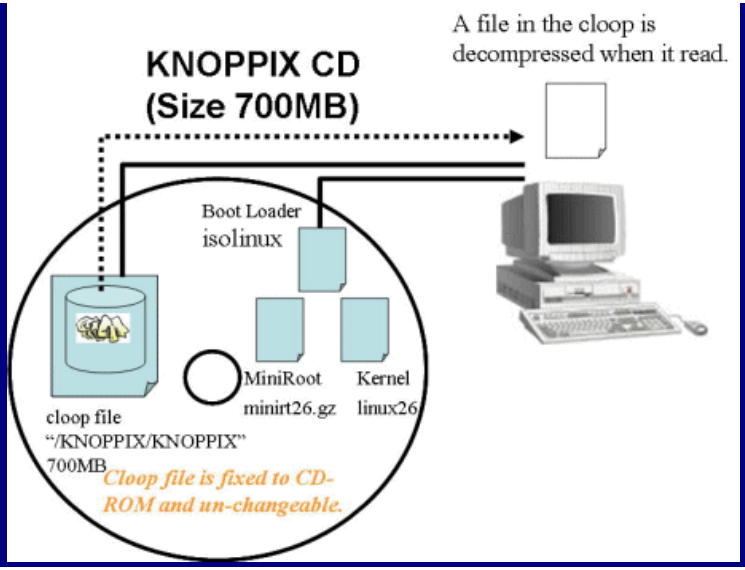
cloop

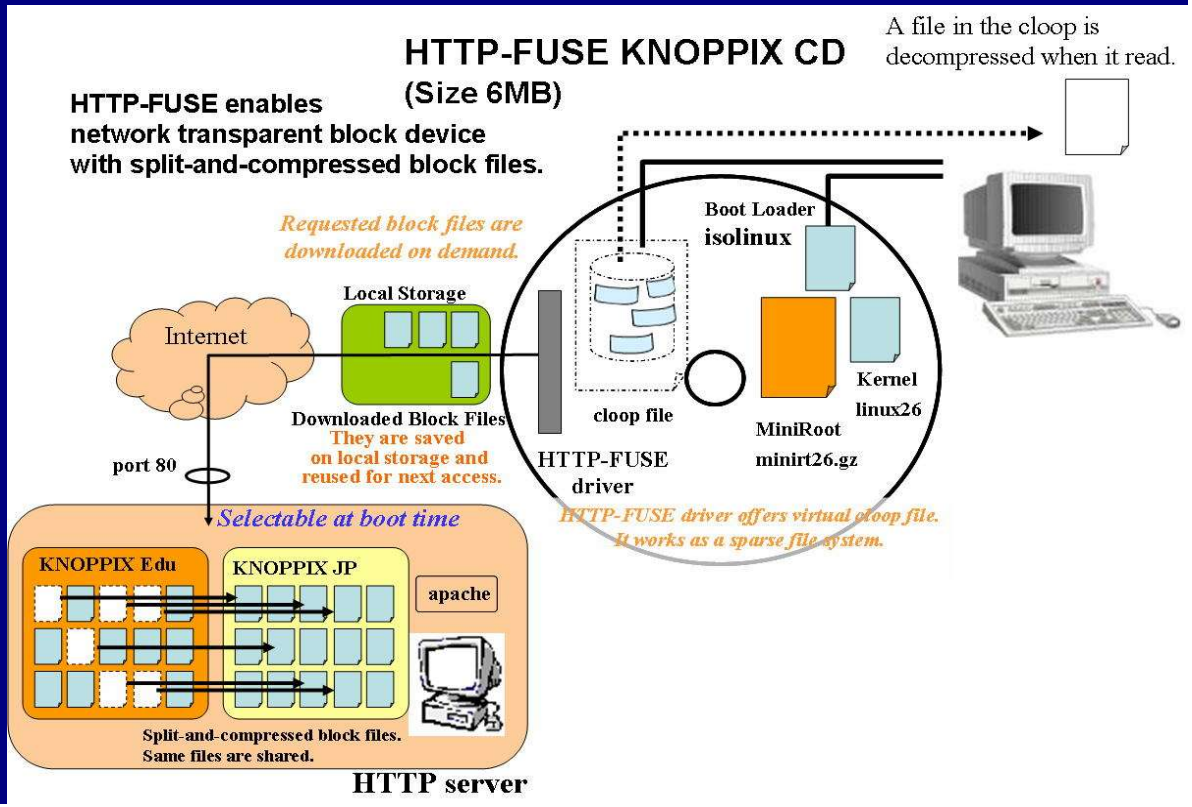

PREPARATION
Block Device
(2GB)
4KB Page
ext2
64KB
Stored in
CD-ROM
cloop file(700MB)
/cdrom/KNOPPIX/KNOPPIX
header
(location
data)
compressed
zlib
cloop
driver
cache
64KB
decompress
extract
Loopback(2GB)
# losetup /dev/cloop
/cdrom/KNOPPIX/KNOPPIX
4KB Page
ext2
Access
mount /dev/cloop
/KNOPPIX
/KNOPPIX
/bin
/boot
/etc

cloop の改良

- cloop 問題点
  - 1 巨大ファイル (700MB) のループバックマウント
  - 700MB ダウンロードは辛い。
  - 一部の変更でも 700MB の作りなし。
- Internet から直接 cloop を利用できないか
  - 利用した部分だけをダウンロードし保存。
    - リモートとローカルの透過的 cloop
  - 差分更新を可能にする。
    - 細かいブロックファイルのループバックマウント
  - 脱 Client/Server 。 P2P 的にばらまく。

- 現状の KNOPPIX
- 700MB CD にすべてを含む
  - bootloader,kernel,miniroot, RootFileSystem
- HTTP-FUSE KNOPPIX
- 6MB CD
  - bootloader,kernel, miniroot
- rootFS は Internet より取得する
  - Selectable!



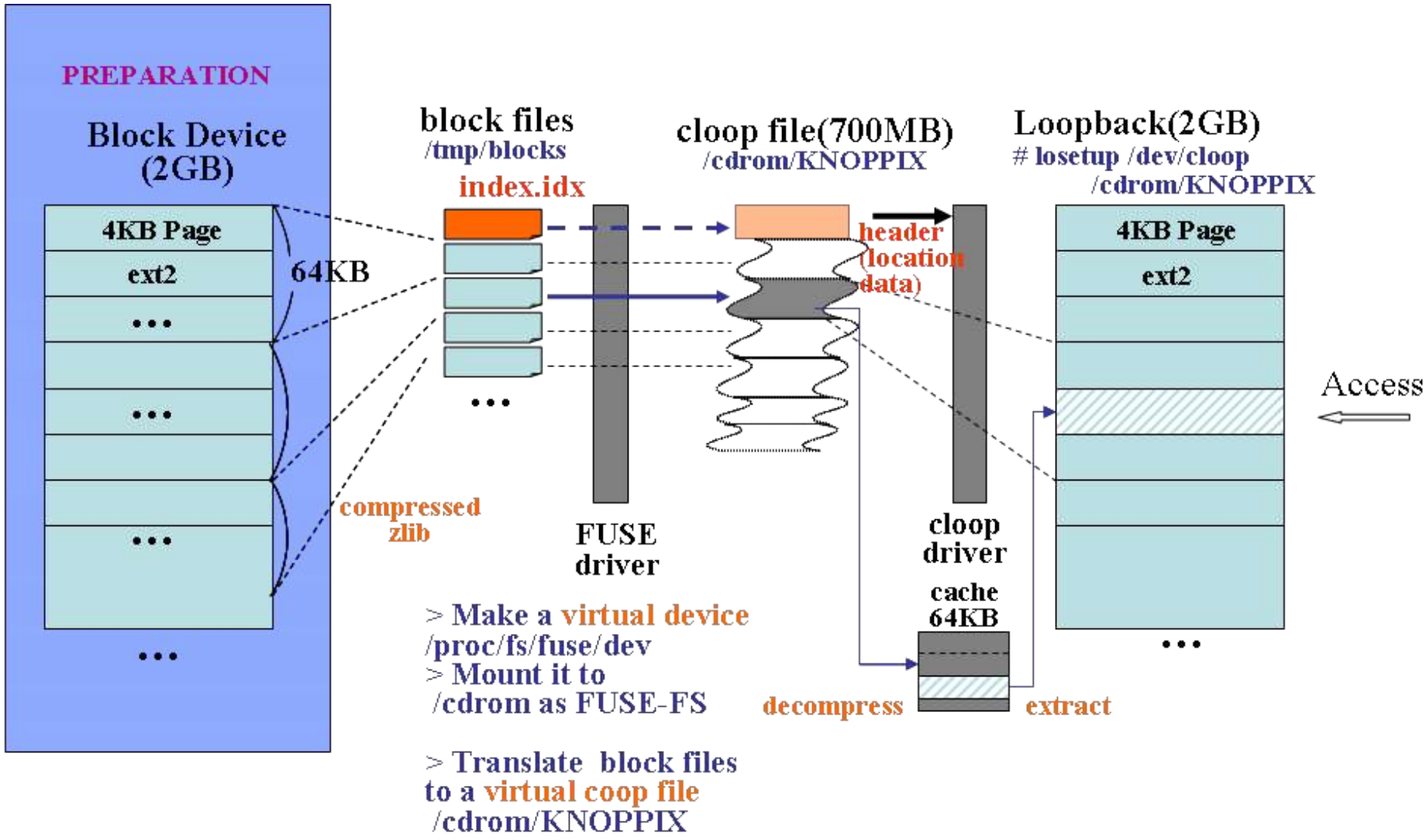


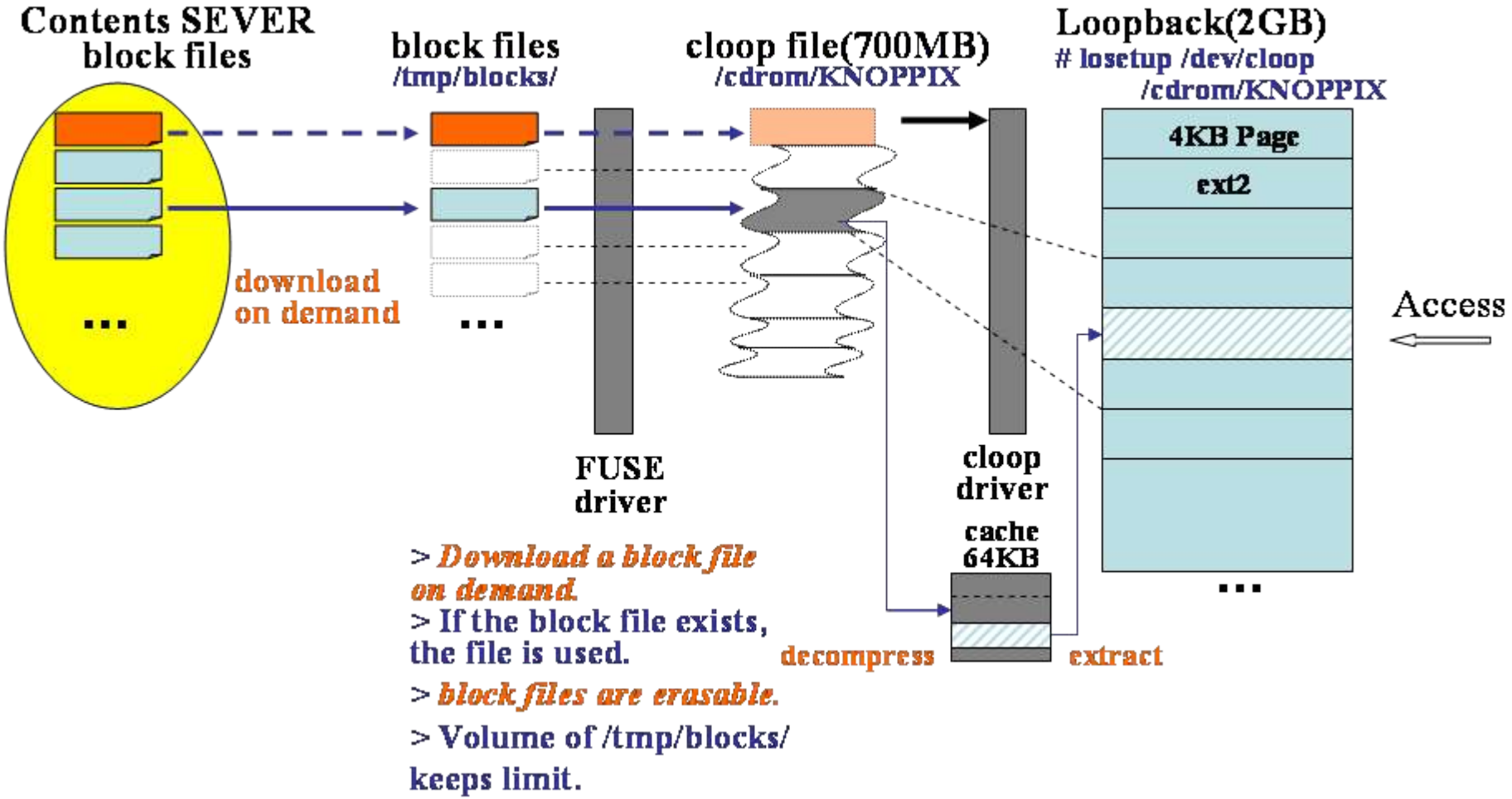
圧縮分割ファイルによるループバックデバイス

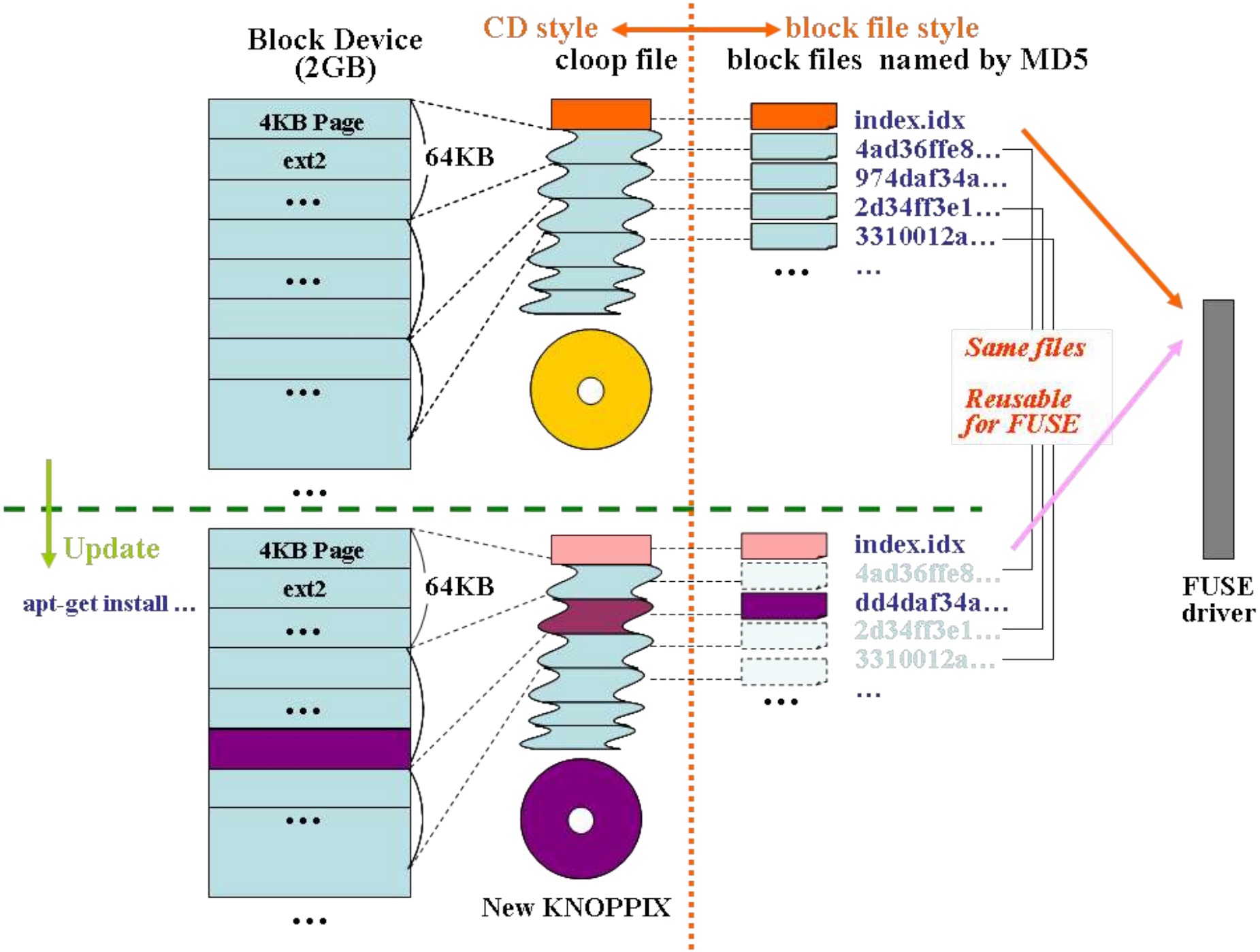
- cloopを改良してリモートとローカルで透過的に扱えるようにする。
  - cloopは64KB毎のブロックを切り出し、圧縮して一つのループバックデバイスファイルを作っていた
    - 大きなまま扱わなければならないことが問題
  - 64KB毎のブロックを切り出し、圧縮データを小さなブロックファイルとして扱い、それらをまとめてループバックデバイスとして扱えるようにする
  - ブロックファイルは必要に応じてdownloadする。

圧縮分割ファイルによる ループバックデバイス

- ファイル名はmd5sum　の値。
  - 内容確認ができる。
  - 同一内容は１ファイル。
- ブロックファイルをまとめる手段として仮想ファイルシステムFUSEを利用
  - 分割圧縮ファイルから構成できようにwrapperを作成
  - FUSEを通してブロックファイル群をcloopファイルとして構成する。


PREPARATION
Block Device
(2GB)
4KB Page
ext2
64KB
block files
/tmp/blocks
index.idx
cloop file(700MB)
/cdrom/KNOPPIX
header
(location
data)
Loopback(2GB)
# losetup /dev/cloop
/cdrom/KNOPPIX
4KB Page
ext2
Access
compressed
zlib
FUSE
driver
cloop
driver
cache
64KB
decompress
extract
> Make a virtual device
/proc/fs/fuse/dev
> Mount it to
/cdrom as FUSE-FS
> Translate block files
to a virtual coop file
/cdrom/KNOPPIX

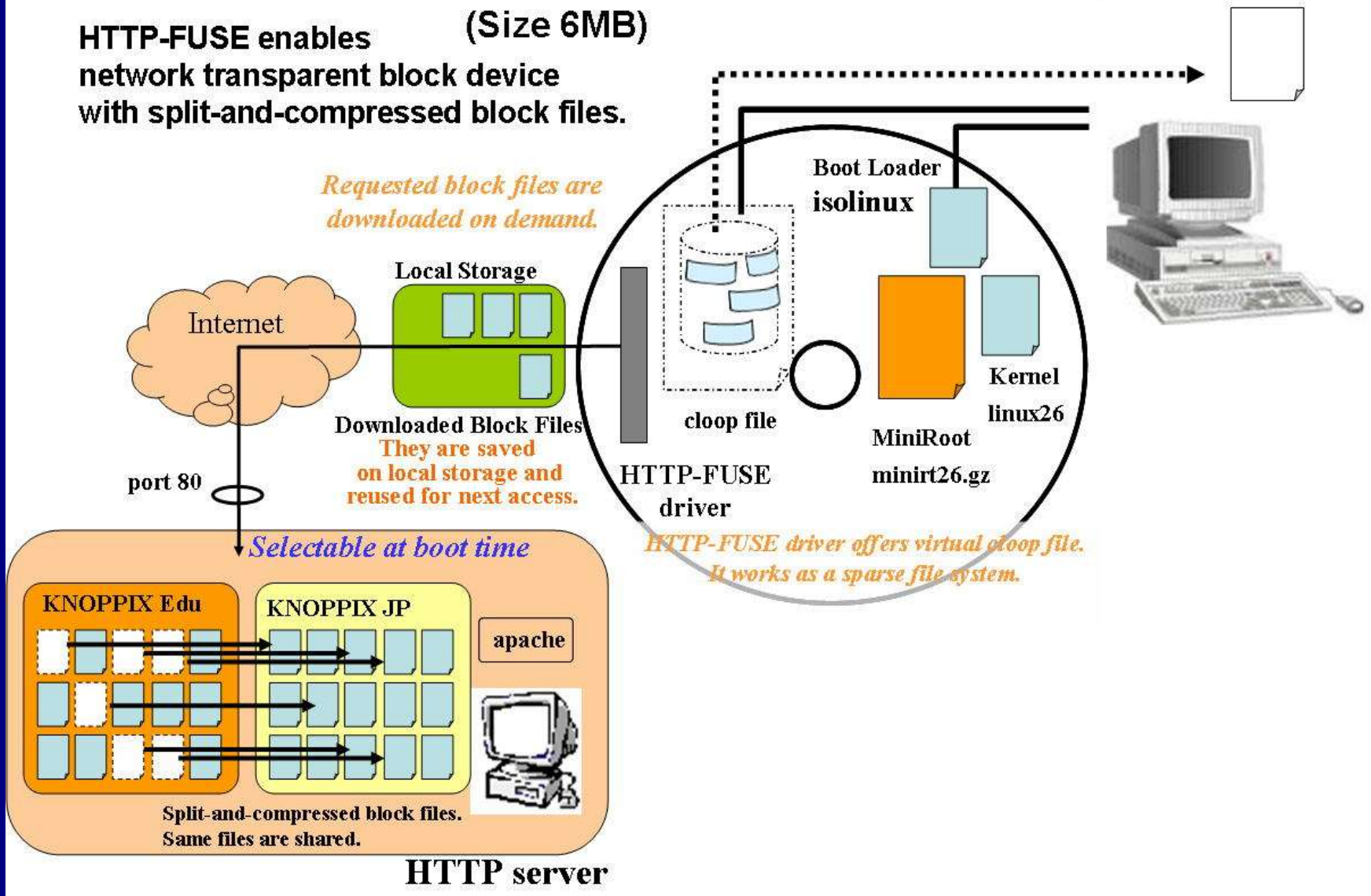

Contents SEVER
block files
block files
/tmp/blocks/
cloop file(700MB)
/cdrom/KNOPPIX
Loopback(2GB)
# losetup /dev/cloop
/cdrom/KNOPPIX
4KB Page
ext2
download
on demand
Access
FUSE
driver
cloop
driver
cache
64KB
decompress
extract
> Download a block file
on demand.
> If the block file exists,
the file is used.
> block files are erasable.
> Volume of /tmp/blocks/
keeps limit.


Block Device
(2GB)
CD style
block file style
cloop file
block files named by MD5
4KB Page
ext2
64KB
index.idx
4ad36ffe8…
974daf34a…
2d34ff3e1…
3310012a…
Same files
Reusable
for FUSE
FUSE
driver
Update
apt-get install …
dd4daf34a…
New KNOPPIX

HTTP-FUSE KNOPPIX CD
(Size 6MB)

A file in the cloop is decompressed when it read.

HTTP-FUSE enables network transparent block device with split-and-compressed block files.



FUSE+CLOOP の特徴

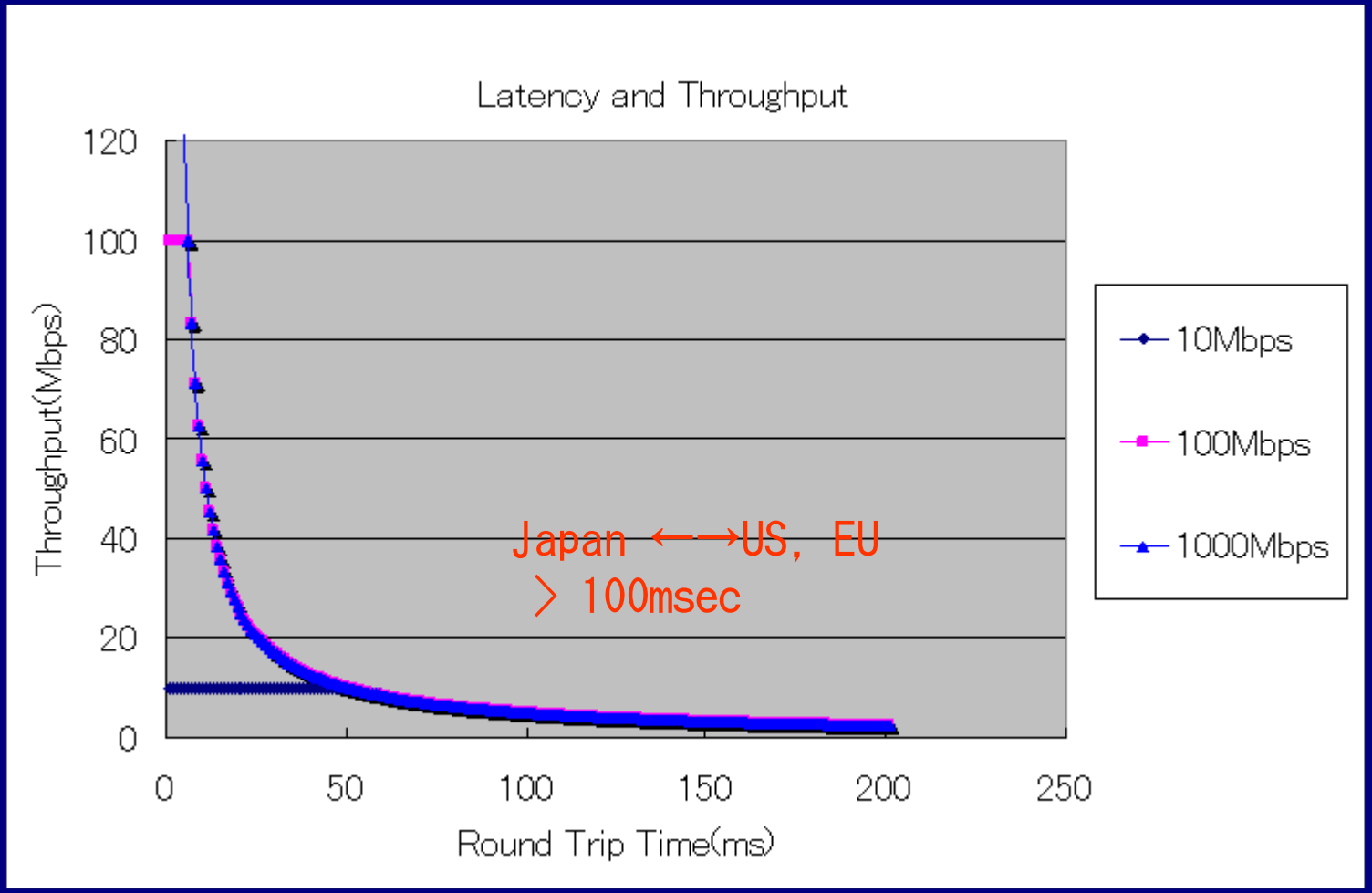
- Pros
  - リモート／ローカルの透過的に扱える
    - すべてダウンロードする必要なし
    - ローカルデバイスにダウンロードしたブロックファイルは再利用可能
  - 更新があった場合、変更があったブロックに対応する分割ファイルとヘッダ情報（index.idx）のみでよい。
- Cons
  - 性能は？

分割圧縮ブロックファイルの特徴

- 条件
  - (64KB cut)ブロックファイル
    - 31,206 files(Max:65,562 Min:84 Ave:21,801)
  - ブート時にダウンロードされるファイル
    - 4,424 files(87MB)
  - ダウンロードパターン(ext2へのアクセスパターン)
    - on demand(非連続)
- 要求
  - 小さいファイルを素早く、on demand 要求

どのようにしてブロックファイルを要求するか

- ダウンロード要求手順
  - ext2 -> cloop -> fuse -> (Ineternet) -> file
- cloop で access request が逐次化
  - コネクションが１つ。
    - NFS で使われているような多重 access request によるバンド幅拡張ができない。
  - レンテンシの影響を受けやすい。
    - Window size(64KB)以下より小さいファイルでは性能がでない。
      - 64KB 切出しでは(Max:65,562 Min:84 Ave:21,801)



Client のTCP windows size を64KB, 1 connection とした場合。
10Mbps : x＜ 50 y=10, x ＞= 50, y= (50/x)*10
100Mbps: x＜ 5 y=100, x＞=5, y=(5/x)*100
1000Mbps: x＜ 0.5 y=1000, x＞=0.5, y=(0.5/x)*1000

切り出しブロックサイズの変更

- FUSE+CLOOPでは切り出しブロックサイズの変更は自由にできる。

	Files	Max	Min	Ave	down(sizeMB)	Full
64KB	31,206	65,562	84	21,801	4424(87)	83%
128KB	15,611	131,118	149	43,041	2876(115)	72%
256KB	7,821	262,230	277	85,372	2089(171)	59%
512KB	3,927	524,454	531	169,501	1436(231)	45%

同一内容のファイルが存在する場合はファイル数が減る

full はブートに実際使われる ext2 ページ割合

切り出しブロックサイズの変更

- FUSEでの読み出し時間（オーバーヘッド）の影響
  - dd of=/cdrom/KNOPPIX if=/dev/null（680MB）
- bootchart時間（initの終からKDEの起動前まで）
  - HTTP down loadで起動。参考：CD起動の場合2:15程度

	RAM-Disk	HTTP	bootchart時間（データ/ファイル数）
64KB	9.96	1:58	0:43(87MB,4224)
128KB	10.59	1:42	0:57(115MB,2876)
256KB	20.91	1:43	1:01(171MB,2089)
512KB	54.34	2:12	1:16(231MB,1436)

#ThinkPADT42, PentiumM 1.8G,100MNIC,HTTPは直結RTT< 0.2msec

RTTの長い場合の影響は調べる必要あり。

ダウンロードの方法

- 高速にファイル（コピー）をばらまきたい。
- P2P? 特にBitTorrent
  - 通常のP2Pはファイル検索機能がある。該当ファイルを探すのには適するが非常に遅い。
  - 残念ながらBittorrentは大容量ファイルに適した設計。
    - ファイルをピースに分割し、複数のノードからピースを適当な順番でダウンロード（バンド幅拡張）し、最後に再構成することでバンド幅を稼いでいる。
- on demandに小さいファイルを多くダウンロードするのに適していない。

ダウンロードの方法

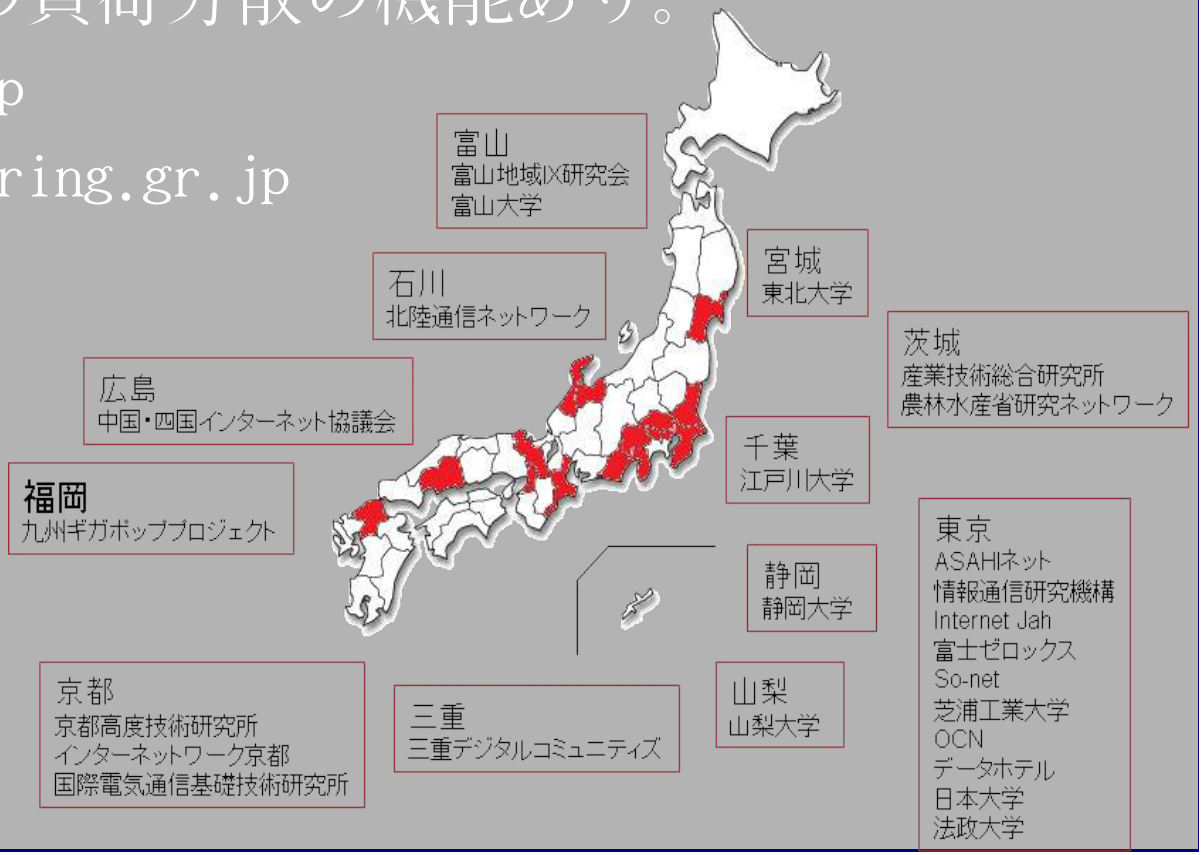
- CDN(Content Delivery Network)の利用。
  - 1つのURLで複数の配信法を有するもの。
  - ルーティング技術、キャッシュ、コンテンツ管理を行ない、効率的配信を実現する(仮想)ネットワーク。
    - 技術的蓄積や利用可能な資源がある。
      - RTT, レスポンスを短くする技術
        - BGPを利用したもの。サーバの負荷分散。
      - ミラーサーバ
      - proxy
        - P2P proxy(coral, P2P squid, dijjer,...)
        - reverse proxy
  - プロトコルは手軽で実績のあるhttp

CDN の候補

- ミラーサーバ群の ring プロジェクト
  - 国内に 20 以上配置。
- P2P proxy の coral プロジェクト
  - PlanetLab を使い、世界中に配置。

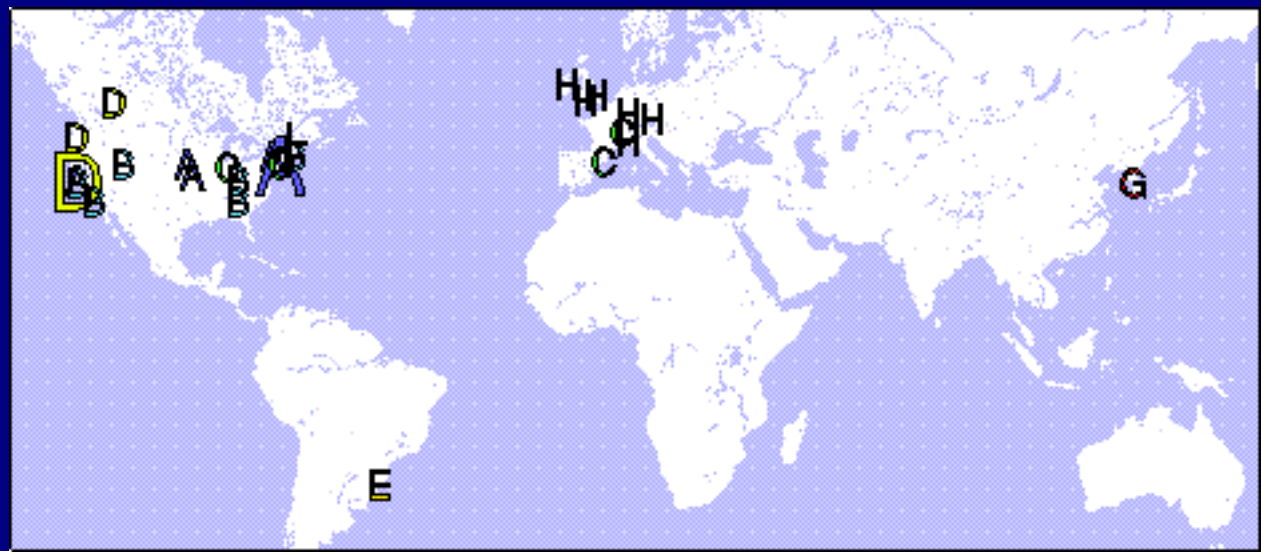
ringプロジェクト

- 日本のフリーソフト配信の最大ミラーサーバ
- tenbin，DNS balanceによるクライアントへ近いミラー選択、サーバの負荷分散の機能あり。
  - www.t.ring.gr.jp
  - www.dnsbalance.ring.gr.jp



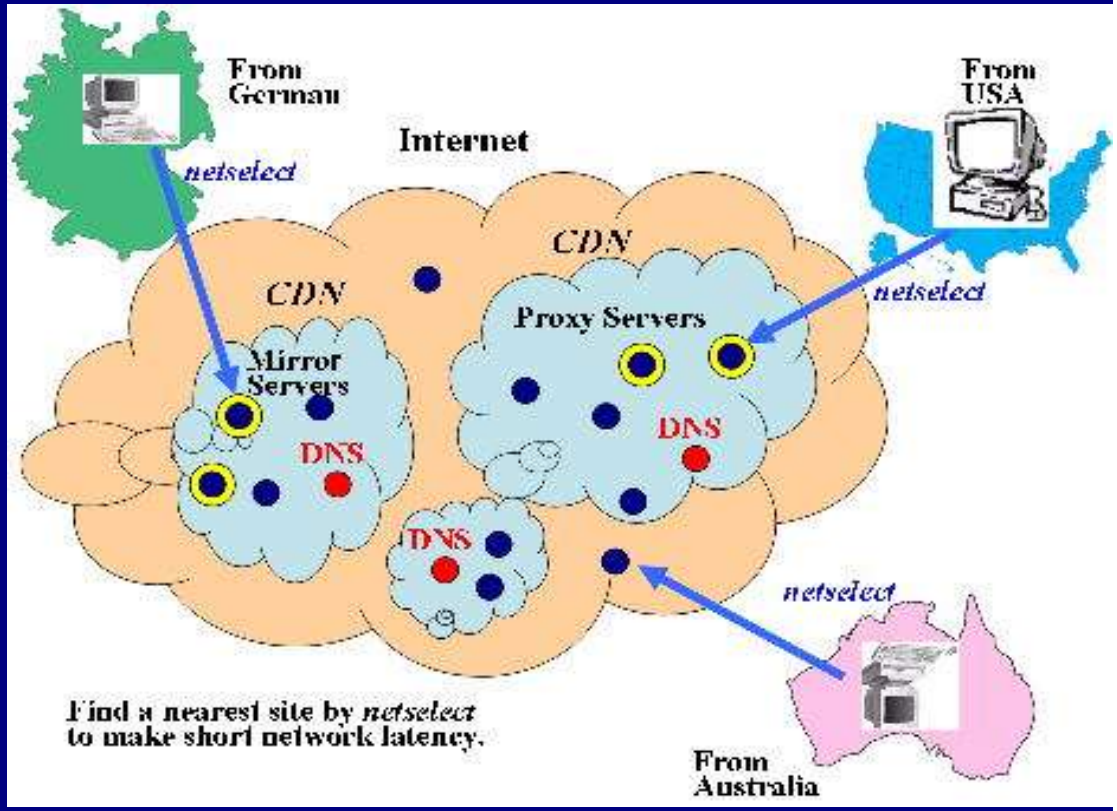
coral（P2P proxy）

- .nyud.net:8090　をURLに付けるのみ。
  - Ex: www.aist.go.jp.nyud.net:8090/index.html
  - nyud.netで名前解決
- DSHT(Distributed Sloppy Hash Table)によりホットスポットを回避
- coral　内にファイルがキャッシュされていればキャッシュを利用。



CDNとのネゴシエーション

- CDNのサーバに頼るだけでなく、クライアント側でもRTTが短くなるようなCDNを選択。
  - CDNとクライアントの相互のネゴシエーションによりダウンロードが決定。
    - 具体的にはCDNが返すIPアドレス群にnetselectをかけて選択。



現状

- 以上述べた機能をいれたHTTP-FUSE KNOPPIXを(04/14)公開。
  - ブート環境（6MB CD-ROM, USBブート）
  - ブートオプション
    - http_proxy=
      - proxyを指定可能
    - memcache
      - 二記憶に保存せず、 RAM DISKのみを使う

関連研究

- Plan9 の Venti[USENIX02]
  - アドレスではなく、書き込むブロック内容のハッシュで管理するストレージシステム
  - 内容が同じブロックは同一
  - ファイルシステムは別(fossil)
- Locality awareness がある分散ファイルシステム
  - AFS, Coda, Shark[NSDI05] など。ファイルシステムレベルで管理。メタデータのキャッシュなど機能分割が明確
  - 書き込み可能。ネットワークの離脱、再接続で一貫性管理あり。
    - Cons: サーバ/クライアント。特殊なプロトコル。

関連研究

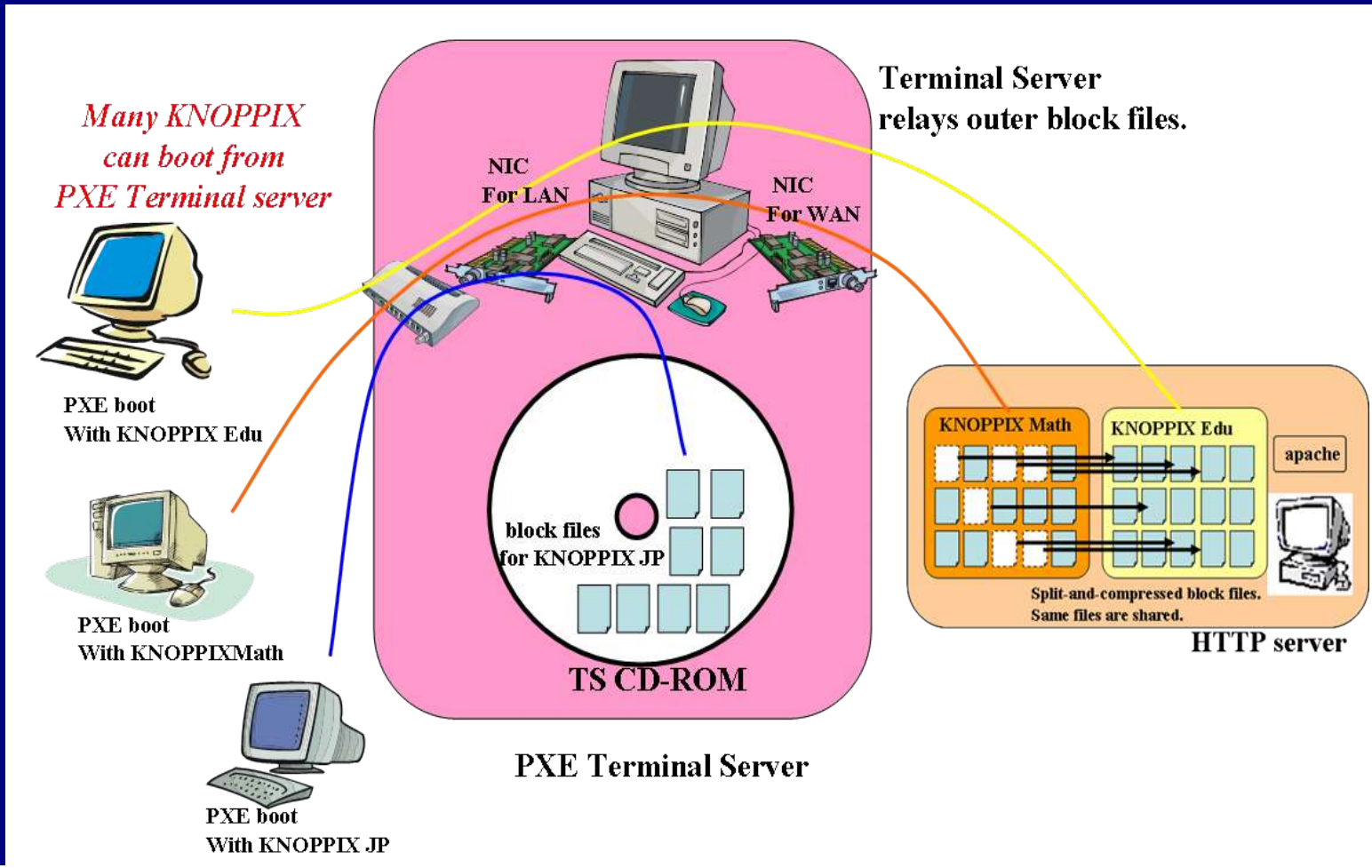
- UNIONFS
  - KNOPPIX3.8から導入された更新可能ファイルシステム
  - CD 上で apt-get が利用可能。
- CopyOnWrite を用いたオーバーレイブロックデバイス（丹、 Linux Conference'05）
  - ブロックレベルの差分更新記録システム
  - 分割ブロックファイルに変更できれば、カスタマイズが容易になる

今後の課題／展望

- ブートデバイス問題
  - 現状ではBIOSのブートデバイス(CD,HD)を使う必要あり
  - IntelがBIOS規格EFI(Extensible Firmware Interface)に期待
    - OSローダが自由に組み込めるようになり、今までの固定ブートデバイスから解放
- ファイルシステムの最適化
  - ブート時に使われるブロックを連続配置はできないか
    - フラグメンテーションの抑制。
- パッケージインストールにも対応したい。
  - 差分更新できないか。（関連：klik or zeroinstall）

PXE TerminalServer for HTTP-FUSE KNOPPIX

- N+I(6/8-10)で展示



まとめ

- 分割圧縮ブロックファイルによるループバックデバイスの作成。
- ネットワーク等価的にブロックファイルの利用が可能。
- ブロックファイルはコピー配布で構わない。現状ではCDNを使って配布。
  - この部分は改良の余地あり。
- この上で起動するKNOPPIXを作成。
  - HTTP-FUSE KNOPPIX