Reduction of RAM consumption by SquashFS
SquashFSによるRAM使用量削減

– 起動時間の問題とその解決方法の紹介 –

TOSHIBA Corp.
Keijiro Yano
2006/01/20

Contents

- Cramfs vs. SquashFS
- SquashFS適用事例紹介と問題点
- SquashFSへの拡張
- SquashFSのブロックサイズについて

環境

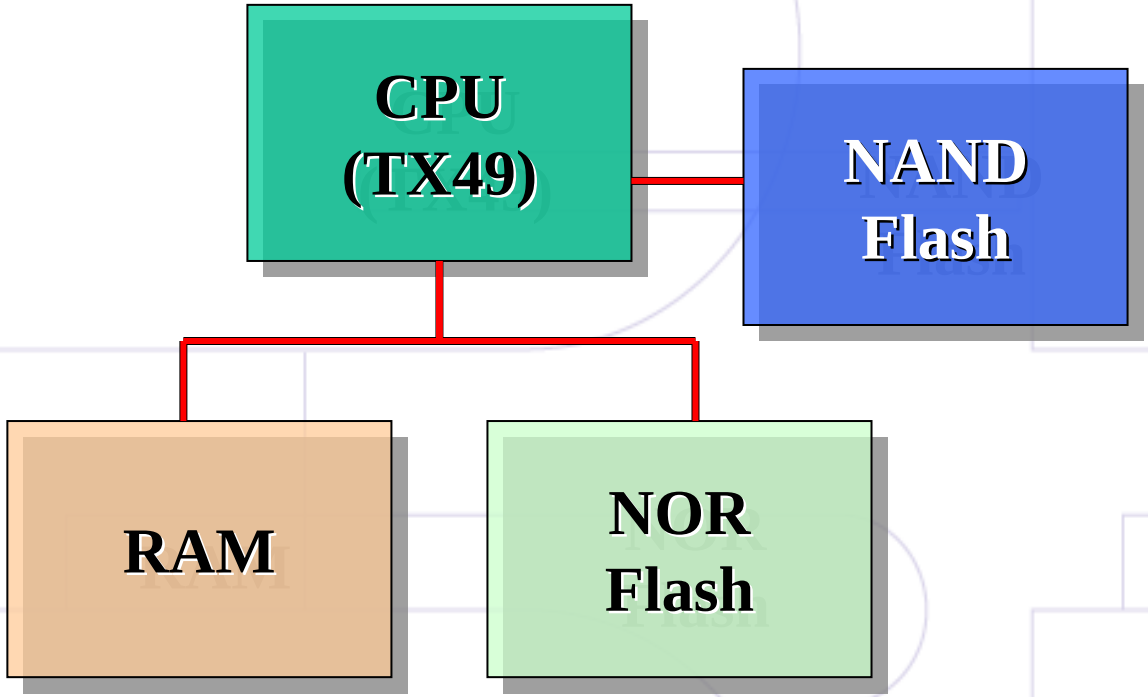
- Kernel
  - Linux 2.6.10 for MIPS
- SquashFS Patch
  - SquashFS 2.2
  - http://sourceforge.net/project/showfiles.php?group_id=63835

Cramfs vs. SquashFS

	Cramfs	SquashFS
圧縮アルゴリズム	ZLIB	ZLIB
圧縮単位	4KB 固定	0.5 ～ 64KB
メタデータ圧縮	×	○
フラグメント・ブロック対応	×	○
XIPサポート	○	×

**Cramfs では XIP サポート共に、
Linearアクセス対応がサポートされており、
Block Device なしで直接マウントすることが可能。**

評価システムの構成



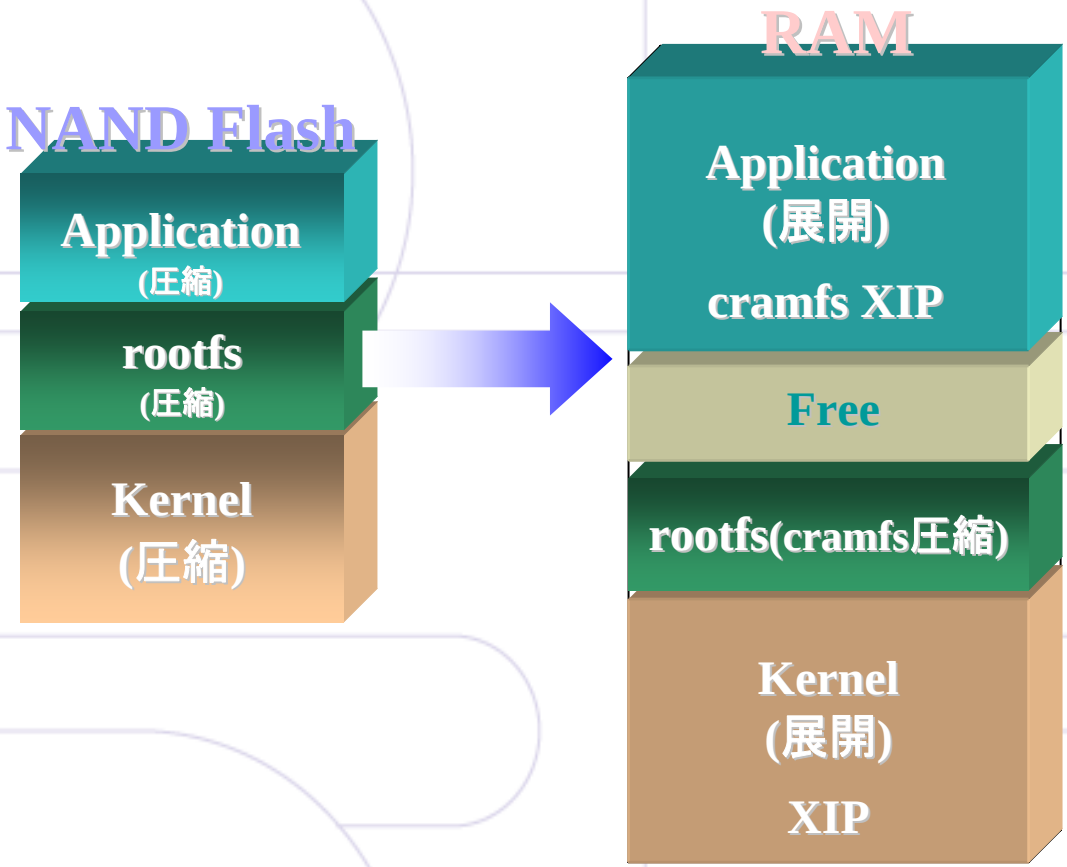
① Bootloader 起動

② NAND Flashから
Kernel/rootfs読み込み

③ RAMに展開

④ RAMからKernel実行

評価システムの構成

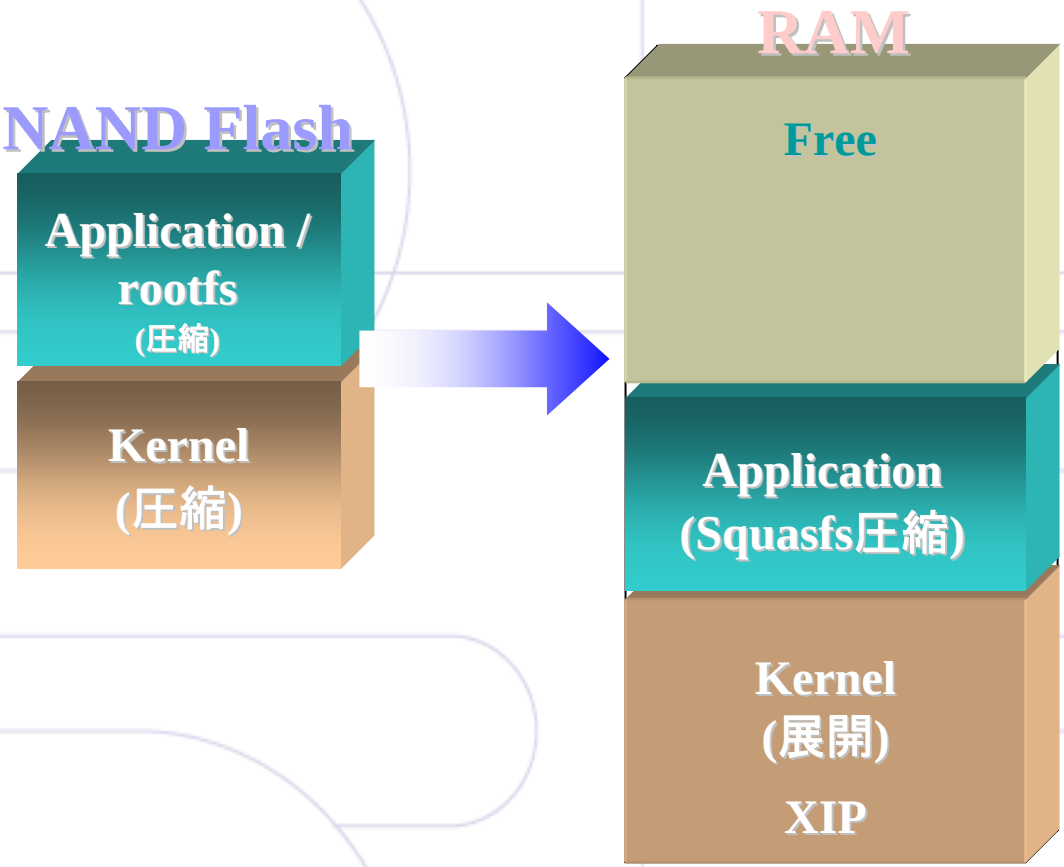

NAND Flash
Application
(圧縮)
rootfs
(圧縮)
Kernel
(圧縮)
RAM
Application
(展開)
cramfs XIP
Free
rootfs(cramfs圧縮)
Kernel
(展開)
XIP

評価システムで使用しているファイルシステムについて

cramfs	cramfsの圧縮を使用。 NAND Flashには、mkcramfsで作成されたバイナリをそのまま保存。
cramfs(XIP)	cramfsのXIPを使用（非圧縮）。 NAND Flashにはmkcramfsで作成されたバイナリをZLIBで圧縮して保存。

- 評価システム仕様の制約により、NAND Flashをmountし、アプリケーションをそこから実行する形態は取れない。
- NAND Flash容量の制限から、アプリケーションをcramfs(XIP)で持つ必要があった。

SquashFS適用後の評価ボードのシステム構成



SquashFS適用後の評価ボードのシステム構成

Demand Pagingにより展開＆実行。
メモリ不足時には不要部分を破棄することにより、RAM使用量削減を実現。

アプリケーションのパフォーマンス、特に起動時間への影響が大きい。

SquashFSの適用事例と問題点

- SquashFSを評価システム上に適用

メリット

NAND Flash上のバイナリサイズ削減

→ 評価システムの事例で

cramfs(noXIP)との比較　約1MB削減。

cramfs(XIP)との比較　ほぼ同等。

→ NAND Flashからの読み込み時間短縮により、起動時間の高速化をはかれる。

RAM使用量削減

→ 評価システムの事例でcramfs(XIP)と比較して約5MB

※削減量はアプリケーションに大きく依存する。

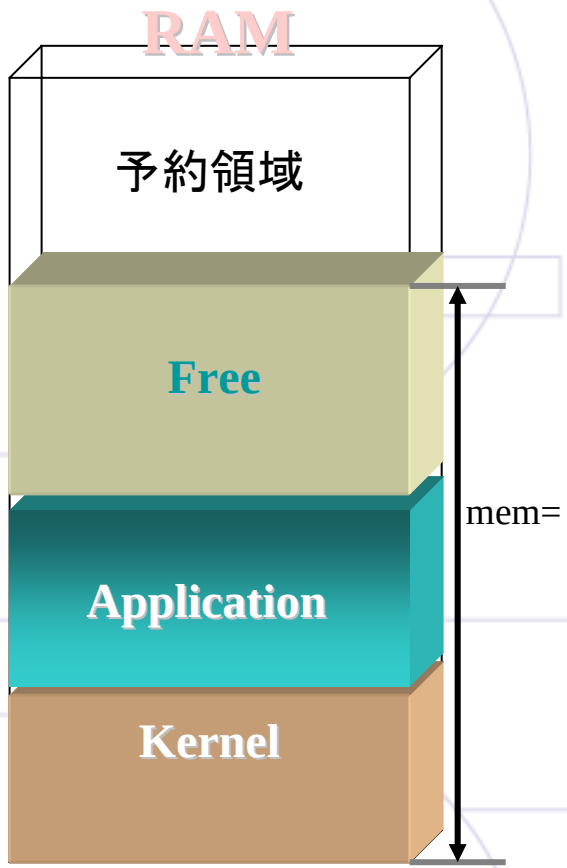
デメリット

Demand Paging のOverheadによる
パフォーマンス低下

RAM使用量削減について

- 評価システムでは、Linuxで管理するRAM以外に、固定領域を予約して使用する必要がある。
  → “mem=“ オプションを使用して、区別している。

- RAM使用量削減は、“mem=“で指定するサイズをより小さくする方向で検討。
  → 1MBずつ、mallocして適当なデータを書き込むアプリケーションを実行し、どこまで確保できるかで削減量の見積もりを行った。



SquashFSの拡張

SquashFSのmountのために、

RAM Disk

Or

MTD Uncached system RAM Driver

を使用する必要があった。

rootfs mountの際に
RAM Diskへの
展開処理が入り
起動時間が遅い

Readから実際の
RAMアクセスまでの
S/W構成が間接的で
アクセス速度が遅い

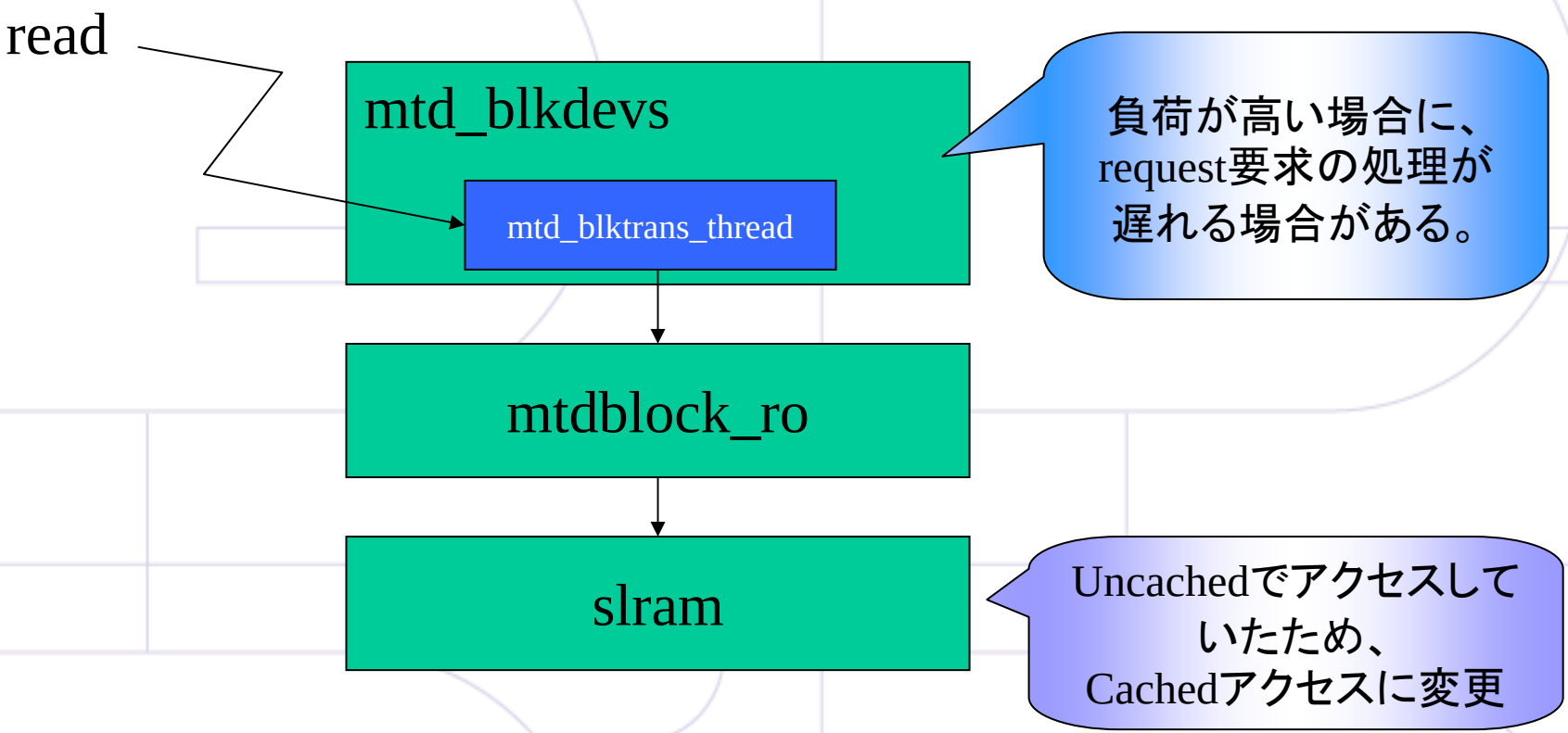
なんらかの改善策が必要となった。

RAM Disk使用時

- Initrdイメージをreadし、/dev/ramにwriteする。(init/do_mounts_rd.c)

```
for (i = 0, disk = 1; i < nblocks; i++) {
  if (i && (i % devblocks == 0)) {
            ...
      printk("Loading disk #%d... ", disk);
  }
  sys_read(in_fd, buf, BLOCK_SIZE);
  sys_write(out_fd, buf, BLOCK_SIZE);
}
```

MTD Uncached system RAM使用時



Linear アクセス対応版 SquashFS

- Block Deviceを経由せずに、SquashFS内で直接RAM(Cached)にアクセスする。

  →CramfsのLinearアクセス対応と同等の機能をSquashFSに実装。

評価システムの起動時間において、
RAM Diskを使用した場合と比較して
約500msec改善出来た。

SquashFSの圧縮単位(ブロックサイズ)

- 高圧縮率を得るためにブロックサイズを大きくする(最大64KB)。

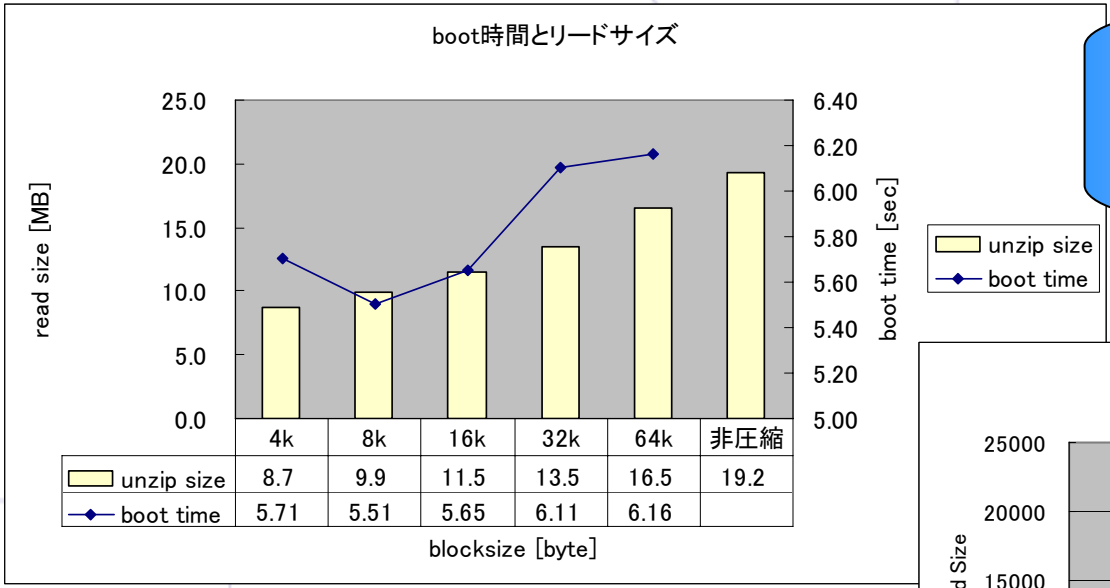
しかし・・・

- その時点では不要な部分も解凍される可能性がある。
- SquashFSのOverheadにより、アプリケーションのパフォーマンスに影響することがある。

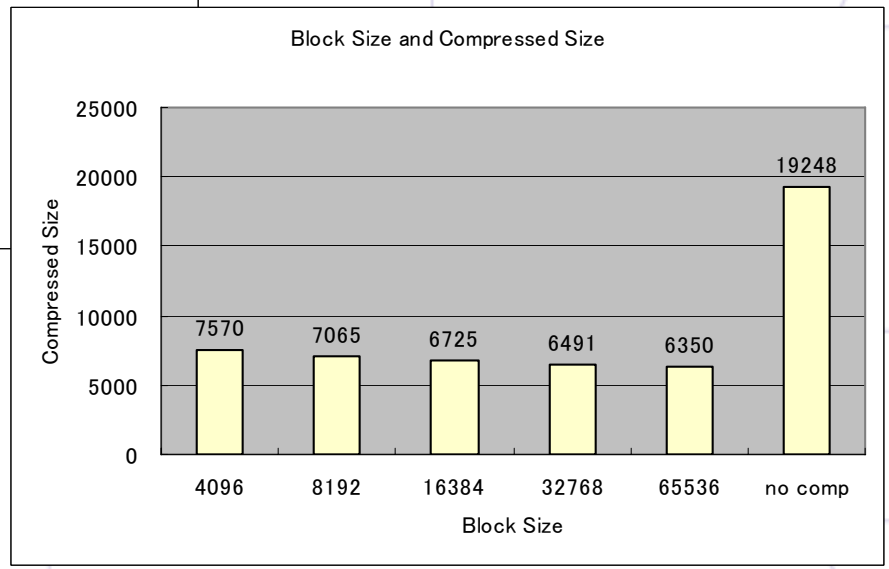
そこで・・・

- 圧縮率とアプリケーションのパフォーマンスを総合的に考慮し、圧縮単位(ブロックサイズ)を決定する必要がある。

SquashFSの圧縮単位(ブロックサイズ)



評価システムにおける
圧縮後のバイナリサイズ



評価システムの起動時における
起動時間・解凍サイズと
ブロックサイズ

まとめ

- Cramfs XIPを使用していた評価システムに対して、SquashFSを適用し、NAND Flashの使用量を増加させることなく、RAM使用量を削減した。
- SquashFSを適用することによるデメリットを、拡張機能を実装することで最小限に抑えることが出来た。
- SquashFSの圧縮単位（ブロックサイズ）は、アプリケーションのパフォーマンス等、目的にあわせて最適なものを選択するのが良い。

End