LinuxTinyの内容分析

2005年7月15日

NEC Linux推進センター

池田 宗広

m-ikeda（あっとまーく）ds.jp.nec.com

0. セッションの概要

LinuxTiny は、カーネルのサイズ縮小・使用メモリ削減のためのパッチ集です。

このセッションでは、NEC Linux推進センターがでこれまでに行った LinuxTiny の内容分析についてご紹介します。

- カーネルサイズ／使用メモリ削減の必要性
- LinuxTinyの概要
- パッチの目的分類
- パッチの規模
- カーネルサイズ縮小／使用メモリサイズ削減効果検証
  - 効果の高い設定／パッチ
  - その他の設定／パッチ
- 今後の予定

1. カーネルサイズ／使用メモリ削減の必要性

- 一般的なPCに搭載されるメモリおよびストレージの容量は、近年飛躍的に増大してきた。
- 「軽い」と言われていたLinuxも、機能追加・強化に伴ってカーネルサイズ・使用メモリともに増加してきている。
- しかしこんな要望は根強い。
  - うちの古くなったマシン、ハードディスクもメモリも小さいけどLinuxでまだまだ使いたい！
  - せっかくLinuxを使うんだから、カーネルはやっぱり最新の2.6！
- そのためにはカーネルサイズを縮小し、使用メモリを削減することが必要となる。

→ LinuxTinyによるサイズ／メモリ削減を検討

「小さく・軽く」という方向性は、組み込み機器への要求とも合致する。

2. LinuxTinyの概要

- Developed by Matt Mackall
  http://www.selenic.com/tiny-about/
- カーネルサイズ・使用メモリを極限まで縮小削減するためのパッチ集
  - パッチ数120、修正ファイル数369ファイル
- 機能概要
  - 組み込みシステム向け（サイズ縮小・メモリ削減）
    - printk()、BUG、panic() を削除し、カーネルサイズを縮小する設定が可能。
    - あまり使われない機能を削除し、カーネルサイズを縮小する設定が可能。(ptrace、direct I/O、POSIXタイマー、IGMP、rtnetlink etc.)
    - SLABアロケータを、よりメモリ効率に優れたSLOBアロケータへと置換。
    - インライン関数の呼び出し箇所の割り出し・ビルド後のサイズ計算用ツールを含む。
  - Kexec
    - クラッシュダンプ用途にkexec機能の追加が可能。
  - Kgdb
    - kgdbの追加が可能。イーサネットを用いたリモートデバッグもサポート。

※ 今回は2.6.10用について分析・検討。

3. パッチの目的分類

- Kconfig に追加される設定項目で目的を分類
  - 組み込み向け
    - General setup ---> Configure standard kernel features (for small systems) ---> 以下に数十項目の設定が追加される。
    - カーネルサイズの縮小・使用メモリ削減が目的。
  - kexec 機能追加
    - Processor type and features ---> kexec system call
  - kgdb 機能追加
    - Kernel hacking ---> Include kgdb kernel debugger

4. パッチの規模

今回の分析対象

目的分類	パッチ数	修正ファイル数	修正行数(+)	修正行数(-)
組み込み向け	91	249	8350	2769
kexec	25	71	2179	171
kgdb	3	48	6060	50
その他	1	1	2	2
合計	120	369	16591	2992

※ 修正ファイル数、行数は延べの値。

- パッチ数、修正数ともに、組み込み向けのものが最も多い。
- 「その他」は Makefile EXTRAVERSION を -tiny に変更するもの。

5. カーネルサイズ縮小／使用メモリ削減効果検証

3種のカーネルを作成して比較を行う。

1) vanilla-full：参考
   - 一般的なLinuxカーネル。
   - 設定はほとんど触らず vanilla カーネル(2.6.10)を make。
     - ローダブルモジュールサポート = n, SMP・プリエンプティブカーネル = n
       パワーマネージメント = n, フレームバッファ = n

2) vanilla-size：tinyの比較対象
   - vanilla でどこまで小さくできるか？
   - サイズが最小となるように設定。
     - for small systems = y, シンボルロード =n , Mathエミュレーション = n,
       Optimize for size = y,
       ドライバは最小限, fs は ext2, ext3, reiser, ISO9660, proc, sysfs, tmpfs のみ。

3) tiny：今回の検証対象
   - LinuxTiny パッチ適用でどのくらい小さくなるか？
   - vanilla-size の設定＋LinuxTinyパッチ適用で追加された設定項目は全てサイズ縮小／使用メモリが減少すると予想される設定を行う。

5．1．カーネルサイズの縮小効果

- カーネルサイズ測定結果
  - Kernel 2.6.10
  - x86(PC compatible)用カーネル
  - gcc 3.3.5

	vmlinux[KB]	bzImage[KB]	
(vanilla-full)	(7709)	(3071)	(参考)
vanilla-size	2833	1061	
tiny	2360	869	

→LinuxTiny の適用によって、2.6カーネルでも1MB以下のカーネルイメージが作成可能。

5．2．使用メモリの削減効果

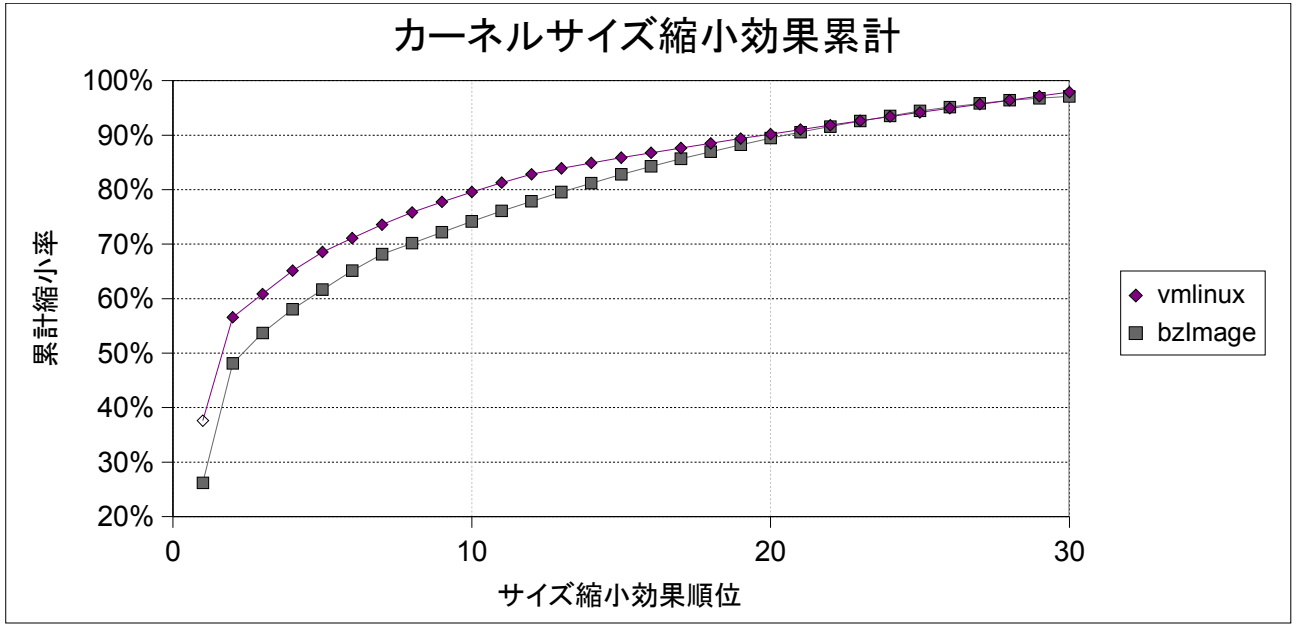
- 使用メモリ測定結果
  - Kernel 2.6.10
  - x86(PC compatible)用カーネル
  - gcc 3.3.5
  - Machine：96MB RAM / Celeron 400MHz
  - 各種デーモンは停止（起動しない）
  - 起動直後に free で測定

	total[KB]	used[KB]	free[KB]	buffers[KB]	cached[KB]	
(vanilla-full)	(90200)	(11572)	(78628)	(1324)	(4052)	(参考)
vanilla-size	94656	9668	84988	1316	4028	
tiny	95036	8988	86048	1392	4004	

→LinuxTiny の適用によって、1MB以上空きメモリが増加。

5. 3. サイズ縮小効果の高い設定／パッチ

- vanilla-size と tiny のサイズ差を100%とした場合、
  サイズ縮小効果の約80%は上位10設定（パッチ）によるもの。
  - vmlinux　：　上位10設定で 80% の効果
  - bzImage　：　上位10設定で 74% の効果



5. 3. サイズ縮小効果の高い設定／パッチ

1) TINY_CFLAGS = n,TINY_CFLAGS_VAL(tiny-cflags.patch):26.2%

- CFLAGS にアーキテクチャ最適化のためのオプションを追加する。(default : ″-march=i386″))

2) PRINTK = n (kill-printk.patch) : 21.9%

- printk(), do_syslog(), sys_syslog(), call_console_drivers() を空関数に置換し、文字列データを削除。
- arch/i386/kernel/head.S の無効割り込みハンドラ部分(ignore_int ラベル)の printk() 呼び出しを削除。

✗ dmesg, syslog が無効になるため、トレーサビリティが低下する。

3) TINYVT = n (tinyvt.patch) : 5.6%

- ヴァーチャルターミナルのコードを標準(vt_ioctl.c, vc_screen.c etc.)からコンパクトなコード tinyvt.c に変更。

✗ 不安定でしばしばブートしなくなった。
2.6.11用パッチからは削除されている。

※ パーセントはvanilla-size と tiny の bzImageサイズ差に対する縮小効果割合。

5. 3. サイズ縮小効果の高い設定／パッチ

4) RTNETLINK = n (rtnetlink.patch) : 4.4%

- rtnetlink関連の関数群(rtnl(), rtattr_strcmp(), rtnetlink_send() etc.)を空マクロ、または0を返すだけのインライン関数 _rtnull() に置換。

5) DIRECTIO = n (direct-io-core.patch) : 3.6%

- fs/direct-io.c をコンパイル対象から除外。
- direct-io.c に含まれる blockdev_direct_IO(), blockdev_direct_IO_no_locking() , generic_file_direct_IO()は、-EINVAL を返すだけのインライン関数またはマクロに置換。

6) FILE_LOCKING = n (fslock.patch) : 3.5%

- POSIXファイルロック関数群(locks_mandatory_locked(), locks_verify_truncate() etc.)を空関数に置換。

※ パーセントはvanilla-size と tiny の bzImageサイズ差に対する縮小効果割合。

5. 3. サイズ縮小効果の高い設定／パッチ

7) BUG = n (nobug.patch) : 3.0%
- BUG(), BUG_ON(), WARN_ON(), PAGE_BUG() を空マクロに置換。

8) IGMP = n (igmp.patch) : 2.0%
- net/ipv4/igmp.c をコンパイル対象から除外。
- igmp関連の関数(igmp_rcv(), ip_mc_join_group() etc.)を空マクロに置換。

9) POSIX_TIMERS = n (posix-timers.patch) : 2.0%
- kernel/posix-timers.c はほとんど削除され、exit_itimers(), clock_was_set() は空関数、do_posix_clock_monotonic_gettime() はjiffiesからナノ秒を単純計算して返すように置換。

※ パーセントはvanilla-size と tiny の bzImageサイズ差に対する縮小効果割合。

5. 3. サイズ縮小効果の高い設定／パッチ

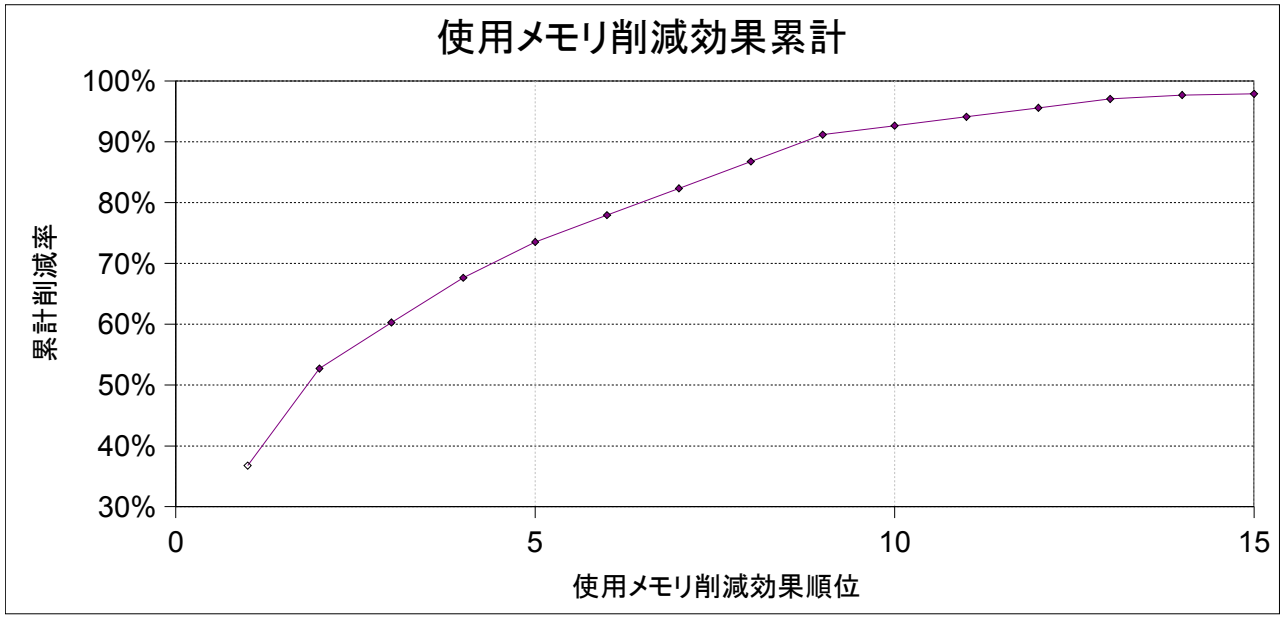
10)FULL_BUG = n (tiny-bug.patch) : 2.0%

- BUG(), WARN_ON() マクロ定義から _FILE_, _LINE_ を削除。

※ パーセントはvanilla-size と tiny の bzImageサイズ差に対する縮小効果割合。

5. 4. 使用メモリ削減効果の高いパッチ／設定

- 結果vanilla-size と tiny の使用メモリ差を100%とした場合に、使用メモリ削減効果の約75%は上位5設定（パッチ）によるもの。
  - 上位5設定で 74% の効果
  - 上位10設定で 93% の効果



5. 4. 使用メモリ削減効果の高い設定／パッチ

1) MEMPOOL = n (mempool-shrink.patch) : 36.8%

- mempool_create() ではエレメントを1つだけ確保し、mempool_t::cache にポインタを代入しておく。
  - ✔ vanilla では引数 で指定された数のエレメントを確保し、ポインタを mempool::elements が指すポインタ配列に代入する。

- mempool_alloc() 内でエレメントの確保試行に失敗した場合、1つだけ確保されているエレメントを返し、mempool_t::cacheにはNULLを入れておく。
  次回呼び出しで再度確保に失敗した場合、NULLが返される。
  - ✔ vanilla ではエレメントの確保試行に失敗した場合、メモリプールに十分(プール容量の1/2以上)エレメントがあればそこから、なければbdflushカーネルスレッドを起床して確保できるまで繰り返し試行する。

- ✗ 負荷の高い状況ではデッドロックの可能性がある。

※ パーセントはvanilla-size と tiny の使用メモリ差に対する削減効果割合。

5. 4. 使用メモリ削減効果の高い設定／パッチ

2) SLOB = y, SLAB = n (slob.patch) : 16.0%

- SLABアロケータをSLOB(Simple List Of Blocks)アロケータに置換。
- SLOBアロケータはSLABオブジェクトを持たない。
  kmalloc()/kmem_cache_alloc() によるメモリ要求に対しては、その都度空き領域を探してポインタを返す。
- 空き領域は slob_t 型構造体の片方向リンクで管理される。slob_t 構造体は領域のサイズと次の空き領域へのポインタをメンバに持つ。
- slob_t型構造体は領域の先頭に置かれる。
  kmalloc()/kmem_cache_alloc() は slob_alloc() を呼び出してslob_t構造体のポインタpを取得し、(void*)(p+1)を返す。

✗ ページサイズ以下の複数サイズを確保する場合のメモリ利用効率は高いが、確保・解放を繰り返すと断片化が発生する。

✗ 確保・解放のたびにリンク検索を行うためパフォーマンス(動作速度)は落ちる可能性あり。

✗ 2.6.10用のSLOBアロケータは設定によってはブートしないことあり。
2.6.11用は安定。(原因は掴みきれていない)

※ パーセントはvanilla-size と tiny の使用メモリ差に対する削減効果割合。

5. 4. 使用メモリ削減効果の高い設定／パッチ

3) AIO = n (no-aio.patch) : 7.6%

- fs/aio.c の非同期I/O(Asynchronous I/O)関連の関数群(aio_compete(), exit_aio(), kick_iocb() etc.)が空定義される。

4) DIRECTIO = n (direct-io-core.patch) : 7.4%

- （サイズ縮小効果 5) を参照）

5) IGMP = n (igmp.patch) : 5.9%

- （サイズ縮小効果 8) を参照）

※ パーセントはvanilla-size と tiny の使用メモリ差に対する削減効果割合。

5. 5. その他「どうかな？」と思う設定／パッチ

- カーネルの機能低下／パフォーマンスダウンが予想されるが、サイズ縮小／使用メモリ削減にはほとんど寄与しないものもある。
  - インライン関数の非インライン化
    - fs/inode.c ... ifind()
      (inode-inlines.patch) : 0.2%/-
    - fs/namei.c ... __vfs_follow_link()
      (namei-inlines.patch) : 0.0%/0.2%
    - include/arch/i386/kernel/semaphore.h ... up(), down()
      (semaphore-inline.patch) : 0.0%/-

    etc...

    ✗ セマフォ関連の非インライン化を適用するとブートしない。
    (2.6.10, 2.6.11用とも)
  - APIの削除
    - ptrace (ptrace.patch) : 0.9%/
    - vm86 (remove-vm86.patch) : -/-

※ パーセントはvanilla-size と tiny の
bzImageサイズ差に対する縮小効果割合／使用メモリ差に対する削減効果割合。

6. 今後の予定

- **組み込みプラットフォームでの評価**
  - ARMプラットフォームへの移植
    - arch 以下へのパッチ数：40パッチ
  - サイズ・使用メモリ測定
  - ベンチマーク実施
    - mempoolあり／なし、SLAB／SLOBアロケータのパフォーマンスを検証
    - SLAB/SLOBアロケータのメモリ利用効率を比較

- **より効果の高い手法検討・開発**
  - より小さなサイズ
  - より少ない使用メモリ
  - より高いパフォーマンス
  - トレーサビリティの確保

- **CELF SystemSize W/G とのリンク**

NEC & Linux