Linux-tiny
And Directions For Small Systems

Matt Mackall
mpm@digeo.com
July, 2004
The Problem: Kernel Bloat
The Problem: Kernel Bloat

- 1994:
  - 0.99 kernel
  - 16MHz 386SX
  - 4MB of RAM
The Problem: Kernel Bloat

- 1994:
  0.99 kernel
  16MHz 386SX
  4MB of RAM

- 2004:
  2.6 kernel
  1024 node IA64
  > 1TB of RAM
The Problem: Kernel Bloat

- **1994:**
  0.99 kernel
  16MHz 386SX
  4MB of RAM

- **2004:**
  2.6 kernel
  1024 node IA64
  > 1TB of RAM

- not happy with 4MB of RAM any more
What Happened?
What Happened?

- massive boom in personal computing and internet use
What Happened?

- massive boom in personal computing and internet use
- constant decrease of hardware cost
What Happened?

- massive boom in personal computing and internet use
- constant decrease of hardware cost
- Linux became a serious commercial endeavor
What Happened?

- massive boom in personal computing and internet use
- constant decrease of hardware cost
- Linux became a serious commercial endeavor
- kernel hackers got jobs (and big machines on their desks)
But there are still small machines!
But there are still small machines!

- *embedded devices are getting increasingly sophisticated*
But there are still small machines!

- **embedded devices are getting increasingly sophisticated**
- **hand-held computers are far more powerful than the original Linux desktops**
But there are still small machines!

- embedded devices are getting increasingly sophisticated
- hand-held computers are far more powerful than the original Linux desktops
- and people around the world still rely on trailing edge “legacy” machines
Where is the growth?
Where is the growth?

- accumulates change by change
Where is the growth?

- accumulates change by change
- features: wholly new code
Where is the growth?

- accumulates change by change
- features: wholly new code
- performance: trading size for speed
Where is the growth?

- accumulates change by change
- features: wholly new code
- performance: trading size for speed
- duplication: trading size for laziness
Where is the growth?

- accumulates change by change
- features: wholly new code
- performance: trading size for speed
- duplication: trading size for laziness
- compatibility: covering more bases
Where is the growth?

- accumulates change by change
- features: wholly new code
- performance: trading size for speed
- duplication: trading size for laziness
- compatibility: covering more bases
- correctness: fixing the corner cases
Where is the growth?

- accumulates change by change
- features: wholly new code
- performance: trading size for speed
- duplication: trading size for laziness
- compatibility: covering more bases
- correctness: fixing the corner cases
- migration: supporting ever more interfaces
Linux-tiny for small systems
Linux-tiny for small systems

- a collection of patches against 2.6
Linux-tiny for small systems

- a collection of patches against 2.6
- independently configurable
Linux-tiny for small systems

- a collection of patches against 2.6
- independently configurable
- minimally invasive and self-contained
Linux-tiny for small systems

- a collection of patches against 2.6
- independently configurable
- minimally invasive and self-contained
- mergeability is a priority
Linux-tiny for small systems

- a collection of patches against 2.6
- independently configurable
- minimally invasive and self-contained
- mergeability is a priority
- emphasis on debugging, auditing, and other support for embedded developers
Linux-tiny for small systems

- a collection of patches against 2.6
- independently configurable
- minimally invasive and self-contained
- mergeability is a priority
- emphasis on debugging, auditing, and other support for embedded developers
- initial target: minimal x86 web server
## Finding bloat: using size(1) and nm

```
2.6.5$ size vmlinux */built-in.o
    text     data     bss     dec     hex filename
  3366220   673296   166824  4206340    402f04 vmlinux
  1181276   250808   48000  1480084    169594 drivers/built-in.o
  735152    32593   30628   798373    c2ea5 fs/built-in.o
   18151    1120    1316    20587     506b init/built-in.o
   21841     172     204    22217     56c9 ipc/built-in.o
  159632   16115  42402  218149    35425 kernel/built-in.o

[...]
```

```
2.6.5$ nm --size -r vmlinux | head -20
00008000 b  __log_buf
00007000 D  irq_desc
00004e78 d  pci_vendor_list
00003c00 B  ide_hwifs
0000213a T  vt_ioctl
00002000 D  init_thread_union
00001880 D  contig_page_data
0000163b T  journal_commit_transaction
00001500 b  irq_2_pin
000012f5 T  tcp_sendmsg
00001162 t  n_tty_receive_buf
00001080 d  per_cpu_tvec_bases
00001000 t  translation_table
[...]
```
Finding bloat: measuring inlining

1560  get_current (1294 in *.c)
calls:
callers: <other>(336) capable(122) unlock_kernel(44)
lock_kernel(33) flush_tlb_page(11) flush_tlb_mm(10)
find_process_by_pid(6)
flush_tlb_range(4) current_is_kswapd(4) current_is_pdflush(3)
rwsem_down_failed_common(2) on_sig_stack(2) do_mmap2(2)
__exit_mm(2) walk_init_root(1) scm_check_creds(1)
save_i387_fsave(1)
sas_ss_flags(1) restore_i387_fsave(1) read_zero_pagealigned(1)
handle_group_stop(1) get_close_on_exec(1) fork_traceflag(1)
ext2_init_acl(1) exec_permission_lite(1) dup_mmap(1)
do_tty_write(1) de_thread(1) copy_signal(1) copy_sighand(1)
copy_fs(1) check_sticky(1) cap_set_all(1) cap_emulate_setxuid
(1) arch_get_unmapped_area(1)

546  current_thread_info (286 in *.c)
calls:
callers: <other>(207) copy_to_user(95) copy_from_user(86)
tcp_set_state(22) test_thread_flag(20) verify_area(13)
tcp_enter_memory_pressure(6) sock_orphan(3) icmp_xmit_lock(2)
csum_and_copy_to_user(2) tcp_v4_lookup(1) sock_graft(1)
set_thread_flag(1) neigh_update_hhs(1) ip_finish_output2(1)
gfp_any(1) fn_flush_list(1) do_getname(1) clear_thread_flag(1)
alloc_buf(1) activate_task(1)
Finding bloat: Vlasenko's inline hunter

<table>
<thead>
<tr>
<th>Size</th>
<th>Uses</th>
<th>Wasted</th>
<th>Name and definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>56</td>
<td>461</td>
<td>16560</td>
<td>copy_from_user</td>
</tr>
<tr>
<td>122</td>
<td>119</td>
<td>12036</td>
<td>skb_dequeue</td>
</tr>
<tr>
<td>164</td>
<td>78</td>
<td>11088</td>
<td>skb_queue_purge</td>
</tr>
<tr>
<td>97</td>
<td>141</td>
<td>10780</td>
<td>netif_wake_queue</td>
</tr>
<tr>
<td>43</td>
<td>468</td>
<td>10741</td>
<td>copy_to_user</td>
</tr>
<tr>
<td>43</td>
<td>461</td>
<td>10580</td>
<td>copy_from_user</td>
</tr>
<tr>
<td>145</td>
<td>77</td>
<td>9500</td>
<td>put_page</td>
</tr>
<tr>
<td>49</td>
<td>313</td>
<td>9048</td>
<td>skb_put</td>
</tr>
<tr>
<td>109</td>
<td>101</td>
<td>8900</td>
<td>skb_queue_tail</td>
</tr>
<tr>
<td>381</td>
<td>21</td>
<td>7220</td>
<td>sock_queue_rcv_skb</td>
</tr>
<tr>
<td>55</td>
<td>191</td>
<td>6650</td>
<td>init_MUTEX</td>
</tr>
<tr>
<td>61</td>
<td>163</td>
<td>6642</td>
<td>unlock_kernel</td>
</tr>
<tr>
<td>59</td>
<td>165</td>
<td>6396</td>
<td>lock_kernel</td>
</tr>
<tr>
<td>127</td>
<td>59</td>
<td>6206</td>
<td>dev_kfree_skb_any</td>
</tr>
<tr>
<td>41</td>
<td>289</td>
<td>6048</td>
<td>list_del</td>
</tr>
<tr>
<td>73</td>
<td>83</td>
<td>4346</td>
<td>dev_kfree_skb_irq</td>
</tr>
<tr>
<td>131</td>
<td>39</td>
<td>4218</td>
<td>netif_device_attach</td>
</tr>
</tbody>
</table>
Finding bloat: tracking allocations

```plaintext
# cat /proc/kmalloc

total bytes allocated:   266848
slack bytes allocated:    37774
net bytes allocated:     145568
number of allocs:           732
number of frees:            282
number of callers:           71
lost callers:                 0
lost allocs:                  0
unknown frees:                0

<table>
<thead>
<tr>
<th>total</th>
<th>slack</th>
<th>net</th>
<th>alloc/free</th>
<th>caller</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>203</td>
<td>256</td>
<td>8/0</td>
<td>alloc_vfsmnt+0x73</td>
</tr>
<tr>
<td>8192</td>
<td>3648</td>
<td>4096</td>
<td>2/1</td>
<td>atkbd_connect+0x1b</td>
</tr>
<tr>
<td>192</td>
<td>48</td>
<td>64</td>
<td>3/2</td>
<td>seq_open+0x10</td>
</tr>
<tr>
<td>12288</td>
<td>0</td>
<td>4096</td>
<td>3/2</td>
<td>seq_read+0x53</td>
</tr>
<tr>
<td>8192</td>
<td>0</td>
<td>0</td>
<td>2/2</td>
<td>alloc_skb+0x3b</td>
</tr>
<tr>
<td>960</td>
<td>0</td>
<td>0</td>
<td>10/10</td>
<td>load_elf_interp+0xa1</td>
</tr>
<tr>
<td>1920</td>
<td>288</td>
<td>0</td>
<td>10/10</td>
<td>load_elf_binary+0x100</td>
</tr>
<tr>
<td>320</td>
<td>130</td>
<td>0</td>
<td>10/10</td>
<td>load_elf_binary+0x1d8</td>
</tr>
<tr>
<td>192</td>
<td>48</td>
<td>96</td>
<td>6/3</td>
<td>request_irq+0x22</td>
</tr>
<tr>
<td>7200</td>
<td>1254</td>
<td>7200</td>
<td>75/0</td>
<td>proc_create+0x74</td>
</tr>
<tr>
<td>64</td>
<td>43</td>
<td>64</td>
<td>2/0</td>
<td>proc_symlink+0x40</td>
</tr>
<tr>
<td>4096</td>
<td>984</td>
<td>0</td>
<td>1/1</td>
<td>check_partition+0x1b</td>
</tr>
<tr>
<td>69632</td>
<td>0</td>
<td>45056</td>
<td>17/6</td>
<td>dup_task_struct+0x38</td>
</tr>
</tbody>
</table>
```
Opportunities for trimming
Opportunities for trimming

- debugging interfaces: printk(), bug(), warn(), panic()
Opportunities for trimming

- debugging interfaces: printk(), bug(), warn(), panic()
- optional APIs: sysfs, ptrace, aio, posix-timers, uid16, vm86, ethtool, tcpdiag, igmp, rtnetlink, etc.
Opportunities for trimming

- debugging interfaces: printk(), bug(), warn(), panic()
- optional APIs: sysfs, ptrace, aio, posix-timers, uid16, vm86, ethtool, tcpdiag, igmp, rtnetlink, etc.
- memory structures: lookup tables, kernel stacks, etc.
Opportunities for trimming

- debugging interfaces: printk(), bug(), warn(), panic()
- optional APIs: sysfs, ptrace, aio, posix-timers, uid16, vm86, ethtool, tcpdiag, igmp, rtinetlink, etc.
- memory structures: lookup tables, kernel stacks, etc.
- replacing SLAB with SLOB
Opportunities for trimming

- debugging interfaces: printk(), bug(), warn(), panic()
- optional APIs: sysfs, ptrace, aio, posix-timers, uid16, vm86, ethtool, tcpdiag, igmp, rtnetlink, etc.
- memory structures: lookup tables, kernel stacks, etc.
- replacing SLAB with SLOB
- TinyVT: a minimal console
Results
Results

- historic 0.99pl15 kernel: Slackware 1.1.2, 1994 301K
Results

• historic 0.99pl15 kernel: Slackware 1.1.2, 1994 301K

• 2.6.5-tiny1 test config: IDE, ext2, TCP, NIC 363K boots with mem=2M (1664K w/ BIOS)
Results

- historic 0.99pl15 kernel:
  Slackware 1.1.2, 1994
  301K
- 2.6.5-tiny1 test config:
  IDE, ext2, TCP, NIC
  363K
  boots with mem=2M (1664K w/ BIOS)
- Highly-modularized vendor kernels:
  Fedora Core 2
  1.2M
  SuSE 9.1
  1.5M
Results

- historic 0.99pl15 kernel: Slackware 1.1.2, 1994  301K
- 2.6.5-tiny1 test config: IDE, ext2, TCP, NIC  363K
  boots with mem=2M (1664K w/ BIOS)
- Highly-modularized vendor kernels: Fedora Core 2  1.2M
  SuSE 9.1  1.5M
- Stock 2.6.5 default config  1.9M
A sample boot
A sample boot

Uncompressing Linux...
A sample boot

Uncompressing Linux... Ok, booting the kernel.
A sample boot

Uncompressing Linux... Ok, booting the kernel.

#
A sample boot

Uncompressing Linux... Ok, booting the kernel.
# mount /proc
A sample boot

Uncompressing Linux... Ok, booting the kernel.
# mount /proc
#
A sample boot

Uncompressing Linux... Ok, booting the kernel.
# mount /proc
# cat /proc/meminfo
Uncompressing Linux... Ok, booting the kernel.
# mount /proc
# cat /proc/meminfo
MemTotal:          980 kB
MemFree:           312 kB
Buffers:            32 kB
Cached:            296 kB
SwapCached:          0 kB
Active:            400 kB
Inactive:           48 kB
HighTotal:           0 kB
HighFree:            0 kB
LowTotal:          980 kB
LowFree:           312 kB
SwapTotal:           0 kB
SwapFree:            0 kB
Dirty:               0 kB
Writeback:           0 kB
Mapped:            380 kB
Slab:                0 kB
Committed_AS:      132 kB
PageTables:         24 kB
VmallocTotal:  1032172 kB
VmallocUsed:         0 kB
VmallocChunk:  1032172 kB
#
Getting involved

• Much more to do
• Suggestions welcome!
• Project web page: http://selenic.com/tiny-about
• Mailing list: linux-tiny@selenic.com http://selenic.com/mailman/listinfo/linux-tiny