Embedded Linux Size BoF

Michael Opdenacker
Bootlin
michael.opdenacker@bootlin.com
Michael Opdenacker

- Founder and Embedded Linux engineer at Bootlin
  - Embedded Linux expertise
  - Development, consulting and training

- Long time interest in embedded Linux boot time, and one of its prerequisites: small system size.

- From Orange, France.
Why reduce size?

There are multiple reasons for having a small kernel and system

▶ Run on very small systems (IoT)
▶ Run on old machines
▶ Run Linux as a bootloader
▶ Boot faster (for example on FPGAs)
▶ Reduce power consumption. Even conceivable to run the whole system in CPU internal RAM or cache (DRAM is power hungry and needs refreshing)
▶ Security: reduce the attack surface
▶ Cloud workloads: optimize instances for size and boot time.
▶ Spare as much RAM as possible for applications and maximizing performance.
▶ Other reasons?

See https://tiny.wiki.kernel.org/use_cases
Compiler optimizations

- Using a recent compiler
  Compiling the kernel with gcc 6.3 vs 4.7: only 0.8% smaller size!

- Compiling with gcc LTO
  Compiling `oggenc.c` with `-Os -flto` instead of `-Os`:
  only -2.6% (arm) and -2.8% (x86_x64)

- Using Clang `-Oz` instead of gcc `-Os`
  Compiling `oggenc.c`: -5.7%

- ARM: compiling with `-mthumb` instead of `-marm`:
  -6.8% with `oggenc`

- Any further technique you’d like to share?
Reduce user-space size

- Replace BusyBox by Toybox (less configurable, mature and featureful). Can save a few tens of KB.
- Replace glibc or uClibc by musl
  musl vs glibc: 76% size reduction in static BusyBox
- For small static executables, musl also wins vs glibc and uclibc
  7300 bytes (musl) vs 492792 (glibc) in static hello world.
- sstrip can be used to shave off an extra KB.
- Any further technique you’d like to share?
How to get a small kernel?

- Run `make tinyconfig` (since version 3.18)
- `make tinyconfig` is `make allnoconfig` plus configuration settings to reduce kernel size
- You will also need to add configuration settings to support your hardware and the system features you need.

```
tinyconfig:
  $(Q)$(MAKE) -f $(srctree)/Makefile allnoconfig tiny.config
```
# CONFIG_CC_OPTIMIZE_FOR_PERFORMANCE is not set
CONFIG_CC_OPTIMIZE_FOR_SIZE=y
# CONFIG_KERNEL_GZIP is not set
# CONFIG_KERNEL_BZIP2 is not set
# CONFIG_KERNEL_LZMA is not set
CONFIG_KERNEL_XZ=y
# CONFIG_KERNEL_LZO is not set
# CONFIG_KERNEL_LZ4 is not set
CONFIG_OPTIMIZEINLINEING=y
# CONFIG_SLAB is not set
# CONFIG_SLUB is not set
CONFIG_SLOB=y
CONFIG_NOHIGHMEM=y

# CONFIG_HIGHMEM4G is not set
# CONFIG_HIGHMEM64G is not set
tinyconfig Linux kernel size (arm)

tinyconfig vmlinux size (arm)

<table>
<thead>
<tr>
<th>Version</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>v3.18</td>
<td>100000</td>
</tr>
<tr>
<td>v3.19</td>
<td>200000</td>
</tr>
<tr>
<td>v4.0</td>
<td>300000</td>
</tr>
<tr>
<td>v4.1</td>
<td>400000</td>
</tr>
<tr>
<td>v4.2</td>
<td>500000</td>
</tr>
<tr>
<td>v4.3</td>
<td>600000</td>
</tr>
<tr>
<td>v4.4</td>
<td>700000</td>
</tr>
<tr>
<td>v4.5</td>
<td>800000</td>
</tr>
<tr>
<td>v4.6</td>
<td>900000</td>
</tr>
<tr>
<td>v4.7</td>
<td>1000000</td>
</tr>
<tr>
<td>v4.8</td>
<td>1100000</td>
</tr>
<tr>
<td>v4.9</td>
<td>1200000</td>
</tr>
<tr>
<td>v4.10</td>
<td>1300000</td>
</tr>
<tr>
<td>v4.11</td>
<td>1400000</td>
</tr>
<tr>
<td>v4.12</td>
<td>1500000</td>
</tr>
<tr>
<td>v4.13</td>
<td>1600000</td>
</tr>
<tr>
<td>v4.14</td>
<td>1700000</td>
</tr>
</tbody>
</table>

Not worse than 10 years back!

Kernel, drivers and embedded Linux - Development, consulting, training and support - https://bootlin.com
Mainline Linux 4.14-rc5 booting on QEMU ARM VersatilePB, with 3121 KB of RAM

- **zImage**: 409240 bytes
- **text**: 986792 (program code)
- **data**: 986792 (initialized data)
- **bss**: 25772 (initialized data)
- **total**: 1129872

Test it by yourself (all in a single line):
```
qemu-system-arm -M versatilepb -nographic -kernel zImage
-initrd initarmfs.cpio.gz -dtb versatile-pb.dtb -m 3121k
```

Demo with the Minitty patches

Mainline Linux 4.14-rc5 booting on QEMU ARM VersatilePB, with 2993 KB of RAM

- zImage: 393928 bytes (-15 KB!)
- 128 KB of RAM saved!

Try it by yourself:
Ongoing mainlining efforts

Most (if not all) of the work is currently done by Nicolas Pitre (Linaro). Here are the main ideas:

- **Minitty**: a minimalistic tty layer for very small systems.
  See https://lwn.net/Articles/721074/.

- **Nanosched**: a lightweight scheduler (little chance to get accepted).

- Instead, proposed to make some scheduling classes optional: sched/deadline and sched/rt.

See his latest presentation:
http://connect.linaro.org/resource/sfo17/sfo17-100/. Code available on
http://git.linaro.org/people/nicolas.pitre/linux.git
Ingo Molnar, June 11th 2017

*But you can prove me wrong: show me a Linux kernel for a real device that fits into 32KB of RAM (or even 256 KB) and then I’ll consider the cost/benefit equation. Until that happens I consider most forms of additional complexity on the non-hardware dependent side of the kernel a net negative.*
To convince upstream maintainers, Nicolas’ ultimate goal is to run Linux on the STM32F469NI MCU:

- BGA216 package
- ARM Cortex-M4 core
- 2 Mbytes of Flash
- 324 Kbytes of RAM

Nicolas started to work on the STM32F469 Discovery kit (with 16 MB of SDRAM, already well supported by Linux).
Reducing RAM usage

- Running the kernel and user-space in place (XIP)
- Reducing kernel defines to reduce the size of kernel structures
  
  fs/dcache.c
  
  - #define IN_LOOKUP_SHIFT 10
  + #define IN_LOOKUP_SHIFT 5

- Investigating the big memory consumption from device tree loading. Reducing RAM usage is easier than reducing code size!
How to help with kernel tinification (1)

- Look for `obj-y` in kernel Makefiles:
  ```
  obj-y = fork.o exec_domain.o panic.o \
  cpu.o exit.o softirq.o resource.o \
  sysctl.o sysctl_binary.o capability.o ptrace.o user.o \
  signal.o sys.o kmod.o workqueue.o pid.o task_work.o \
  extable.o params.o \
  kthread.o sys_ni.o nsproxy.o \
  notifier.o ksysfs.o cred.o reboot.o \
  async.o range.o smpboot.o ucount.o
  ```

- What about allowing to compile Linux without ptrace support (14K on arm) or without reboot (9K)?

- Another way is to look at the compile logs and check whether/why everything is needed.
Look for tinification opportunities, looking for the biggest symbols:

```bash
nm --size-sort vmlinux
```

Look for size regressions with the *Bloat-O-Meter*:

```bash
> ./scripts/bloat-o-meter vmlinux-4.9 vmlinux-4.10
```

<table>
<thead>
<tr>
<th>function</th>
<th>old</th>
<th>new</th>
<th>delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>page_wait_table</td>
<td></td>
<td>2048</td>
<td>+2048</td>
</tr>
<tr>
<td>sys_call_table</td>
<td></td>
<td>1600</td>
<td>+1600</td>
</tr>
<tr>
<td>cpuhp_bp_states</td>
<td>980</td>
<td>1800</td>
<td>+820</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Projects to follow

- Compiling Linux with LLVM/Clang
  Google (Greg Hackmann and Nick Desaulniers) managed to compile the 4.4 and 4.9 stable kernels, opening the door to size and performance optimizations:
  https://lwn.net/Articles/734071/

- Compiling Linux with gcc LTO
  Some efforts (Andy Kleen, Nicolas Pitre) but not in mainline yet Details on
  https://linuxplumbersconf.org/2015/ocw/system/presentations/3369/original/slides.html
Resurrect patches from Josh Triplett which didn’t get in: https://git.kernel.org/cgit/linux/kernel/git/josh/linux.git/

Simplify the filesystem layer: you don’t want things like readahead or page writeback support when you don’t have storage. However, that’s very difficult to remove!

Remove kernel features such as ptrace, reboot support, etc.

Revive single-user (CONFIG_NON_ROOT) support: https://lwn.net/Articles/631853/

Modify the kernel binary to remove symbols that were not used during runtime tests? At least, can be done without hurting the mainline code! How to do that?

Other ideas?
Another hardware platform worth supporting

Try to support a board with no SDRAM:

- 512K of on-chip RAM
- 2M of flash
- ARM Cortex M7 CPU
- Cost: 23 EUR

Hoping to have a system with a very good battery life!
Useful resources

▶ Internet of Tiny Linux (IoTL): Episode IV (Nicolas Pitre, Sep 2017)
http://connect.linaro.org/resource/sfo17/sfo17-100/

▶ My detailed presentation about reducing Linux size (with benchmark details)

▶ Home of the Linux tinification project https://tiny.wiki.kernel.org/

▶ Ideas ideas and projects which would be worth reviving
http://elinux.org/Kernel_Size_Reduction_Work
Questions? Suggestions? Comments?

Michael Opdenacker
michael.opdenacker@bootlin.com

Slides under CC-BY-SA 3.0
http://bootlin.com/pub/conferences/2017/elce/