Embedded Linux Size BoF

Michael Opdenacker
Free Electrons
michael.opdenacker@free-electrons.com

Michael Opdenacker

- Michael Opdenacker
- Founder and Embedded Linux engineer at ***free electrons***
  - 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
```

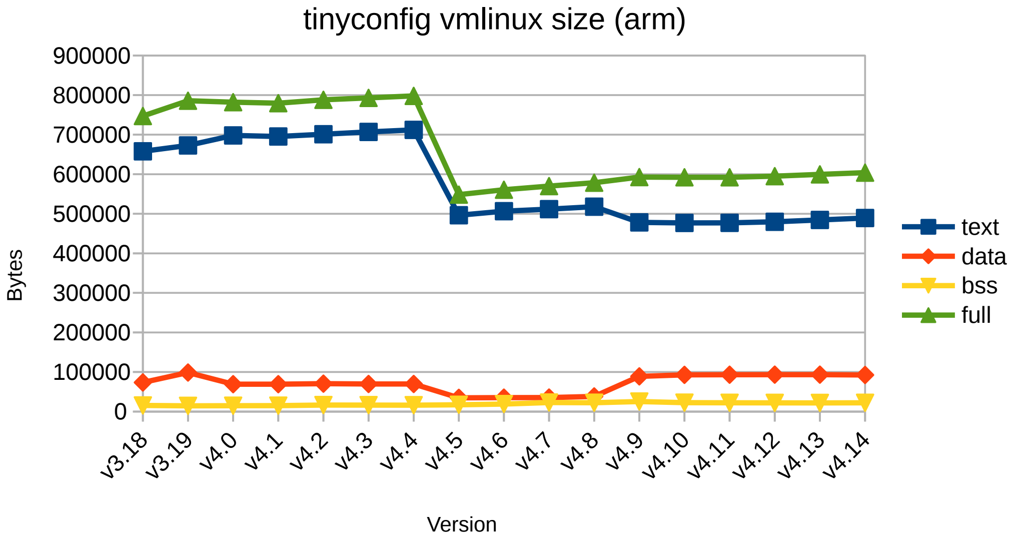
kernel/configs/tiny.config

```
# 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_OPTIMIZE_INLINING=y
# CONFIG_SLAB is not set
# CONFIG_SLUB is not set
CONFIG_SLOB=y
```

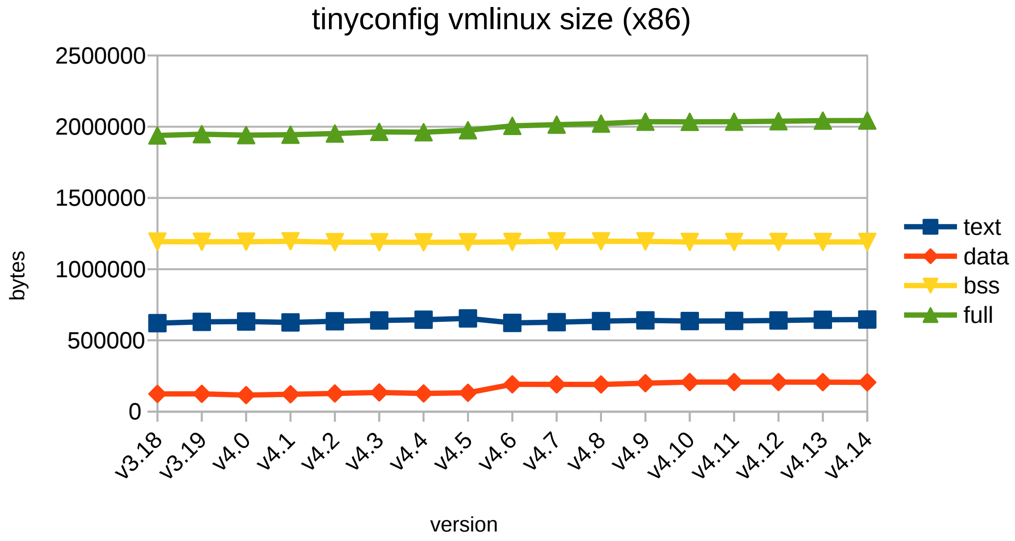
arch/x86/configs/tiny.config

```
CONFIG_NOHIGHMEM=y
# CONFIG_HIGHMEM4G is not set
# CONFIG_HIGHMEM64G is not set
```

tinyconfig Linux kernel size (arm)



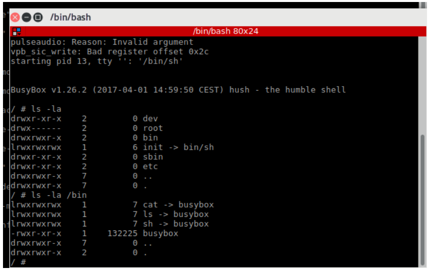
tinyconfig Linux kernel size (x86)



Demo

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



/bin/bash

/bin/bash 80x24

```
pulseaudio: Reason: Invalid argument
vpb_sic_write: Bad register offset 0x2c
starting pid 13, tty '': '/bin/sh'


BusyBox v1.26.2 (2017-04-01 14:59:50 CEST) hush - the humble shell

/ # ls -la
drwxr-xr-x    2         0 dev
drwx------    2         0 root
drwxrwxr-x    2         0 bin
lrwxrwxrwx    1         6 init -> bin/sh
drwxr-xr-x    2         0 sbin
drwxr-xr-x    2         0 etc
drwxrwxr-x    7         0 ..
drwxrwxr-x    7         0 .
/ # ls -la /bin
lrwxrwxrwx    1         7 cat -> busybox
lrwxrwxrwx    1         7 ls -> busybox
lrwxrwxrwx    1         7 sh -> busybox
-rwxr-xr-x    1    132225 busybox
drwxrwxr-x    7         0 ..
drwxrwxr-x    2         0 .
/ #
```

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
```

http://free-electrons.com/pub/conferences/2017/elce/opdenacker-size-bof/demo/4.14-rc5/

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:
`http://free-electrons.com/pub/conferences/2017/elce/opdenacker-size-bof/demo/4.14-rc5-minitty/`

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`

Reaction from upstream maintainers

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.

Supporting real hardware

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.

How to help with kernel tinification (2)

- Look for tinification opportunities, looking for the biggest symbols:

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

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

```
> ./scripts/bloat-o-meter vmlinux-4.9 vmlinux-4.10
add/remove: 101/135 grow/shrink: 155/109 up/down: 19517/-19324 (193)
function                                     old     new   delta
page_wait_table                                -    2048   +2048
sys_call_table                                 -    1600   +1600
cpuhp_bp_states                              980    1800    +820
...
```

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.htm

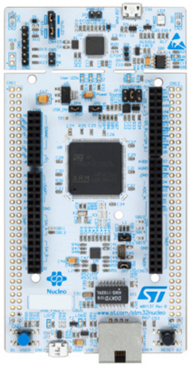
Other ideas

- 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!



STM32 Nucleo F767ZI

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) `http://free-electrons.com/pub/conferences/2017/jdll/opdenacker-embedded-linux-in-less-than-4mb-of-ram/`
- 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?

Michael Opdenacker

michael.opdenacker@free-electrons.com