Internet of Tiny Linux (IoTL): Episode IV

Nicolas Pitre
<nico@linaro.org>
2017-09-25

Nicolas Pitre

This is a discussion on various methods, from low hanging fruits to more daring approaches, being put forward to reduce the size of Linux kernel and user space binaries to make them suitable for very small IoT applications. Latest results in terms of effectiveness and upstream acceptance will be discussed.

Internet of Tiny Linux (IoTL)

"Internet of Things" (IoT)

Goals

- Ubiquitous and Low Cost
- Reliable and Easy to Use
- Secure and Field-Upgradable

Pick two!

Internet of Tiny Linux (IoTL)

"Internet of Things" (IoT)

Solutions

- Avoid custom base software
- Leverage the Open Source community
- Gather critical mass around common infrastructure
- Share the cost of non-differentiating development

Internet of Tiny Linux (IoTL)

"Internet of Things" (IoT)

***Linux* is a logical choice**

- Large community of developers
- Best looked-after network stack
- Extensive storage options
- Already widely used in embedded setups
- Etc.

Internet of Tiny Linux (IoTL)

"Internet of Things" (IoT)

The Linux kernel is a logical choice... BUT

- it is featureful → Bloat
- its default tuning is for high-end systems
- the emphasis is on scaling up more than scaling down
- is the largest component in most Linux-based embedded systems

Linux Kernel Size Reduction is part of the solution

Internet of Tiny Linux (IoTL)

Reducing the Linux Kernel Size

REducing the code size

Automatic size reduction techniques:

- Linker Section Garbage Collection (-gc-sections
- Link Time Optimization (LTO)

Manual size reduction techniques:

- Extra kernel configuration options
- Alternative code implementations

Internet of Tiny Linux (IoTL)

Reducing the Linux Kernel Size

Subsystem specific trimming

Already Done (few examples):

- Compile out block device support.
- Compile out NTP support.
- Compile out support for capabilities (only root is granted permission).
- Compile out support for non-root users and groups.
- Compile out printk() and related strings.
- Etc.

Internet of Tiny Linux (IoTL)

Reducing the Linux Kernel Size

Let's remove POSIX timers

CONFIG_POSIX_TIMERS=y

```
   text    data     bss     dec     hex filename
  36516    2956    3912   43384    a978 ./kernel/time/built-in.o
```

CONFIG_POSIX_TIMERS=n

```
   text    data     bss     dec     hex filename
  27329    2884    1200   31413    7ab5 ./kernel/time/built-in.o
```

28% size reduction on the whole time subsystem.

Available in Linux v4.10.

Internet of Tiny Linux (IoTL)

Reducing the Linux Kernel Size

Let's remove PI futexes

CONFIG_FUTEX_PI=y

```
$ size kernel/futex.o
   text    data     bss     dec     hex filename
   7174      19       0    7193    1c19 kernel/futex.o
```

CONFIG_FUTEX_PI=n

```
$ size kernel/futex.o
   text    data     bss     dec     hex filename
   3484      17       0    3501     dad kernel/futex.o
```

51% size reduction.

Merged in Linux v4.14-rc1.

Internet of Tiny Linux (IoTL)

Reducing the Linux Kernel Size

The SLOB of schedulers: nanosched

```
$ make tinyconfig && make kernel/sched/
$ size kernel/sched/built-in.o
   text    data     bss     dec     hex filename
  20325    1308     120   21753    54f9 kernel/sched/built-in.o
```

```
$ echo CONFIG_SCHED_NANO=y >> .config
$ make kernel/sched/
$ size kernel/sched/built-in.o
   text    data     bss     dec     hex filename
   9853     324     208   10385    2891 kernel/sched/built-in.o
```

What are the chances for inclusion into the mainline kernel tree?

Internet of Tiny Linux (IoTL)

Reducing the Linux Kernel Size

Mmake the deadline scheduler class configurable

A much smaller patch with the preliminary work already merged upstream.

Before:

```
$ size -t kernel/sched/built-in.o
   text    data     bss     dec     hex filename
[...]
  22661    3372     116   26149    6625 (TOTALS)
```

With CONFIG_SCHED_DL=n:

```
$ size -t kernel/sched/built-in.o
   text    data     bss     dec     hex filename
[...]
  17601    3308     100   21009    5211 (TOTALS)
```

Internet of Tiny Linux (IoTL)

Reducing the Linux Kernel Size

Mmake the realtime scheduler class configurable

Before:

```
   text    data     bss     dec     hex filename
  22661    3372     116   26149    6625 (TOTALS)
```

With CONFIG_SCHED_DL=n:

```
   text    data     bss     dec     hex filename
  17601    3308     100   21009    5211 (TOTALS)
```

With CONFIG_SCHED_DL=n && CONFIG_SCHED_RT=n:

```
   text    data     bss     dec     hex filename
  15149    3276      28   18453    4815 (TOTALS)
```

Patches available here:

http://git.linaro.org/people/nicolas.pitre/linux.git optional_sched_classes

Internet of Tiny Linux (IoTL)

Reducing the Linux Kernel Size

Upstream maintainer acceptance

> 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.
>
> *June 11th 2017*
> — Ingo Molnar

Internet of Tiny Linux (IoTL)

Reducing the Linux Kernel Size

Let's pick a real hardware target

- STM32F469 Discovery kit
  - STM32F469NI MCU (ARM Cortex-M4)
  - 16MB SDRAM
  - 16MB Quad-SPI NOR Flash

Already accommodates embedded Linux comfortably.

Internet of Tiny Linux (IoTL)

Reducing the Linux Kernel Size

The Ultimate Goal

- STM32F469NI MCU
  - BGA216 package
  - ARM Cortex-M4 core
  - 2 Mbytes of Flash
  - 324 Kbytes of RAM

Ought to run Linux on its own.

Internet of Tiny Linux (IoTL)

Reducing RAM usage

Change of strategy

- Shrink the kernel "size" by eXecuting it In Place from Flash (XIP)
- Do XIP of user space as well
- Concentrate efforts on **data** size reduction rather than code reduction

Internet of Tiny Linux (IoTL)

Reducing RAM usage

Kernel XIP

- Already available !
- Improved it by storing kernel .data compressed into Flash

Patches available here:
http://git.linaro.org/people/nicolas.pitre/linux.git xip_zdata

Internet of Tiny Linux (IoTL)

Reducing RAM usage

User Space XIP requirements

- Executable binary format allowing independent code and data access

BFLT Binary Flat Format:

- very lightweight
- poorly documented
- requires ad hoc tools
- shared library support is a hack

FDPIC ELF:

- designed for split segments from the start
- can be inspected with standard ELF tools
- supports dynamic libraries

Patches available here:
http://git.linaro.org/people/nicolas.pitre/linux.git fdpic

Internet of Tiny Linux (IoTL)

Reducing RAM usage

User Space XIP requirements

- File system driver allowing direct Flash mappings

I chose cramfs because:

- it is lightweight
- easy to add direct ROM access
- may dispense with the block layer
- very low memory usage

For XIP support I added:

- the possibility to not compress some data blocks
- the ability to locate data blocks arbitrarily for alignment purposes
- automatic mapping from ROM whenever possible when mmap() is used

The mkcramfs tool knows how to locate ELF read-only segments in a file and mark them for XIP while still compressing writable data segments to be loaded into RAM.

Patches available here:

http://git.linaro.org/people/nicolas.pitre/linux.git xipcramfs

Internet of Tiny Linux (IoTL)

Reducing RAM usage

still large RAM kernel footprint

```
$ size vmlinux
   text    data     bss     dec     hex filename
 957220   79744   53940 1090904  10a558 vmlinux
```

The text segment is executed from ROM but data and bss are loaded into RAM.

Internet of Tiny Linux (IoTL)

Reducing RAM usage

My symsize script

```
#!/bin/sh
{
read addr1 type1 sym1
while read addr2 type2 sym2; do
        size=$((0x$addr2 - 0x$addr1))
        case $type1 in
        b|B|d|D)
                echo -e "$type1 $size\t$sym1"
                ;;
        esac
        type1=$type2
        addr1=$addr2
        sym1=$sym2
done
} < System.map | sort -n -r -k 2
```

Internet of Tiny Linux (IoTL)

Reducing RAM usage

symsize output

```
b 16384 __log_buf
D 8192  init_thread_union
d 4288  timer_bases
b 4108  in_lookup_hashtable
b 4096  ucounts_hashtable
d 3760  cpuhp_ap_states
D 2048  tasklist_lock
d 2048  page_wait_table
d 1296  init_sighand
D 1072  runqueues
[...]
```

Internet of Tiny Linux (IoTL)

Reducing RAM usage

Static memory usage reduction

arch/arm/include/asm/thread_info.h

```
-#define THREAD_SIZE_ORDER      1
+#define THREAD_SIZE_ORDER      0
```

fs/dcache.c

```
-#define IN_LOOKUP_SHIFT 10
+#define IN_LOOKUP_SHIFT 5
```

kernel/time/timer.c

```
-#define LVL_BITS       6
+#define LVL_BITS       4
```

kernel/ucount.c

```
-#define UCOUNTS_HASHTABLE_BITS 10
+#define UCOUNTS_HASHTABLE_BITS 5
```

i. and so on.

Internet of Tiny Linux (IoTL)

Reducing RAM usage

Static memory usage reduction

Result:

```
   text    data     bss     dec     hex filename
 947884   64424   31964 1044272   fef30 vmlinux
```

data reduction: 15320 bytes

bss reduction: 21976 bytes

Total reduction: 37296 bytes (28%)

Internet of Tiny Linux (IoTL)

Reducing RAM usage

Runtime memory usage reduction

Time to boot the board:

```
Booting Linux on physical CPU 0x0
Linux version 4.13.0-00061-ga90flee20e-dirty (nico@xanadu.home) (gcc version 6.3.1 20170404 (Linaro GCC 6.3-2017.05)) #468 Mon Sep 18 23:30:43 EDT 2017
CPU: ARMv7-M [410fc241] revision 1 (ARMv7M), cr=00000000
OF: fdt: Machine model: STMicroelectronics STM32F469i-DISCO board
[...]
On node 0 totalpages: 4096
  Normal zone: 32 pages used for memmap
  Normal zone: 0 pages reserved
  Normal zone: 4096 pages, LIFO batch:0
pcpu-alloc: s0 r0 d32768 u32768 alloc=1*32768
[...]
Memory: 16064K/16384K available (663K kernel code, 51K rwdata, 232K rodata, 44K init, 29K bss, 320K reserved, 0K cma-reserved)
```

- 80 unreleased memblock pages
  - 32 pages for memmap! (time to shrink memory)
  - 8 pages for pcpu-alloc
  - 13 pages for rwdata
  - 11 pages of initdata

Where are the missing 16 pages?

Answer: At least 13 are taken by unflatten_and_copy_device_tree().

Internet of Tiny Linux (IoTL)

Reducing RAM usage

Runtime memory usage reduction

- Tweak workqueue allocation to shrink its per-cpu memory usage
- Reduce the per-cpu memory pool from 8 pages to only one
- Reduce available RAM from 16MB to 800KB
  - Shrinks mem_map from 32 to 2 pages

What to do with the unflatten device tree?

Internet of Tiny Linux (IoTL)

Reducing RAM usage

Runtime memory usage reduction

What about initcalls?

init/main.c

```
+       pages = global_node_page_state(NR_SLAB_UNRECLAIMABLE);
        if (initcall_debug)
                ret = do_one_initcall_debug(fn);
        else
                ret = fn();
+       pages = global_node_page_state(NR_SLAB_UNRECLAIMABLE) - pages;
+       if (pages)
+               printk("memory: %d pages allocated by initcall %pf\n", pages, fn);
```

Internet of Tiny Linux (IoTL)

Reducing RAM usage

Runtime memory usage reduction

What about initcalls?

Result:

```
  2 pages allocated by spawn_ksoftirqd
  2 pages allocated by regulator_init
  1 pages allocated by stm32f469_pinctrl_init
 11 pages allocated by of_platform_default_populate_init
  1 pages allocated by topology_init
  1 pages allocated by init_pipe_fs
  1 pages allocated by proc_loadavg_init
  1 pages allocated by chr_dev_init
  1 pages allocated by kswapd_init
  1 pages allocated by stm32_timers_driver_init
  1 pages allocated by stm32_rtc_driver_init
  1 pages allocated by i2c_dev_init
  1 pages allocated by pm_qos_power_init
```

25 total pages allocated from initcalls.

But... 11 pages for of_platform_default_populate_init() !?

Internet of Tiny Linux (IoTL)

Reducing RAM usage

Runtime memory usage reduction

A view from user space:

```
/ # dmesg | grep Memory
Memory: 628K/800K available [...]

/ # free
             total       used       free     shared    buffers     cached
Mem:           640        496        144          0          0         24
-/+ buffers/cache:        472        168

/ # cat /proc/meminfo
MemTotal:             640 kB
MemFree:              136 kB
MemAvailable:         136 kB
Cached:                24 kB
MmapCopy:              84 kB
KernelStack:           60 kB
```

Internet of Tiny Linux (IoTL)

Reducing RAM usage

Runtime memory usage reduction

Memory distribution:

- 140 KB used by Device Tree data
  - 96 KB from unflatten_and_copy_device_tree()
  - 44 KB from of_platform_default_populate_init()
- 84 KB used by user space tasks
  - 2 tasks
  - 16 KB stack (2 x 8 KB)
  - 68 KB data
- 64 KB kernel data
  - 52 KB .data
  - 8 KB .bss
- 56 KB initcalls
  - besides of_platform_default_populate_init()
- 60 KB kernel stacks
  - 13 kernel threads
  - 2 user tasks
- A few KB's from memblock_alloc
- 168 KB available memory

Internet of Tiny Linux (IoTL)

Reducing RAM usage

Conclusion:

- Shrinking data usage is easier than reducing code size
- Current solution is still incomplete
  - Lots of memory is still unaccounted for and must be tracked
  - Leveraging the kmemleak infrastructure should be investigated
- Device Tree data is a big memory hog
- Still not ready for the STM32F469NI's 324 KB of internal RAM but getting closer.