Boot-Time BoF

Tim Bird
Principal Engineer, Sony

Outline

- Boot-Time SIG
- Instrumentation
- Tools
- Tests
- Patches for upstream that are pending
- Existing projects (round-table discussion)
- Ideas for future projects

Boot-Time Special Interest Group (SIG)

- Ad hoc collection of boot-time interested parties
  - No organization, no membership, no dues
- Monthly meeting to discuss ideas and projects
- Use linux-embedded@vger.kernel.org for discussions
- See https://elinux.org/Boot-Time_SIG

Instrumentation to work on

- Unified Boot Log
  - recording start_kernel() sub-function duration before time-init
- printk levels
- better probe descriptions (function or module name?)
- printk using cycle-counter (for pre-time printks)
- bootstage stash driver

printk using read_cycle_counter

- Turns out that 3 printks gives you some useful data:
  - Enough data to calibrate the cycle_counter
  - Measure the time before the kernel starts
  - Measure the duration of the kernel printk timestamp blind spot

```
[    0.000000] read_cycle_counter() at kernel start=1985792291
...
[    0.000182] read_cycle_counter() after time_init=2052657313
...
[    1.973535] read_cycle_counter() before running init process =2447327841
```

Tools

- show_delta
- analyze-initcall-debug
- boottime-tools (cntvct)
- others in progress or not upstream?

Tests

- boot-time-regression-test.py
  - By Tim Bird at Sony
  - initcall and printk region test (Described earlier today)
  - No upstream patchset yet
- test_boot_time.py
  - by Laura Nao at Collabora
  - Uses embedded bootconfig with ftrace probes
  - See https://lore.kernel.org/all/20240725110622.96301-1-laura.nao@collabora.com/T/#ma568382acdc81af65c30fe3823a7be3e49f98108
  - See https://lpc.events/event/18/contributions/1700/attachments/1497/3164/LPC2024-Boot_Time_Testing_with_ftrace-Laura_Nao-Collabora.pdf

Status of patches being considered for upstream

- print initlevel
- bootstage stash driver
- deferred initcalls
- better probe details (function name, to map to source line)
- deferred memory init?
- show_delta 2.0?
- analyze-initcall-debug.py?

Round-table of on-going work

- See https://elinux.org/Boot-Time_Activities

Optimizations already upstream

- Base memory init improvements (RedHat)
  - To about 70ms (from about 700 ms)
  - May be described in this presentation: https://elinux.org/Boot_Time_Presentations#Accelerating_Linux_Boot_Time:_Techniques_and_Strategies_for_Optimal_Performance_.5BDevConf.US_2024.5
- Deferred memory init
  - Init part or memory, then use hotplug to add memory after boot
  - Described in this presentation: https://elinux.org/Boot_Time_Presentations#Deferred_Memblocks_Init_for_Boot_Time_Reduction_.5BELC_2024.5D

Optimizations already upstream (cont.)

- udev integration with systemd (Federico)
  - udev efficiency improvements
  - Reduction of user space time by about 1.2 to 1.5 seconds due
  - Patch is upstream, documented in wiki at https://elinux.org/Systemd_Udev_tuning
  - See https://github.com/systemd/systemd/pull/35155
  - Can modify the initial probe statement to fire off all events at once

Stuff already upstream (cont.)

- Parallelizing boot operations
  - Async probing – does it work
  - Deferred probe ordering?
    - device-tree dependencies?
  - Systemd unit ordering?

Documentation or tuning guide

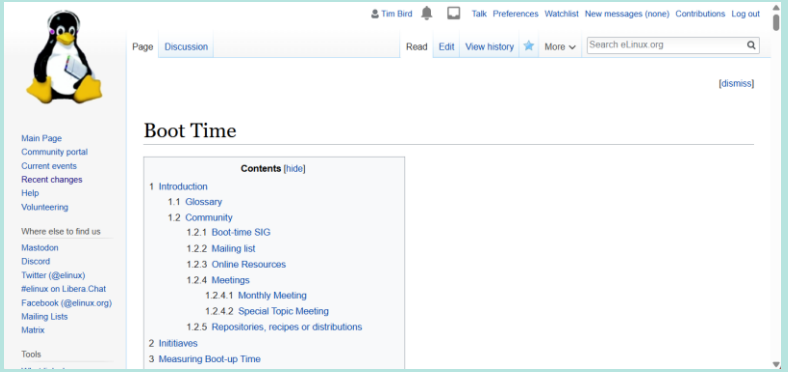
- Need lots of data for this
  - Would like to populate with data like: on rpi4, doing X reduced boot time by N ms
  - Would be nice to automatically gather data:
    - Autotest config variations
    - Autotest command-line variations
  - Would be great to track boot-times as a company manually applied fixes
    - Run grab-boot-data.sh at each step, to record effect of each change
  - Need more instrumentation before doing this:
    - have grab-boot-data.sh grab bootstage cache, and "systemd-analyze blame"
    - fix kernel blind split
    - use ftrace on top-level functions in start_kernel()
- How to keep it up-to-date?

Ideas to follow up on:

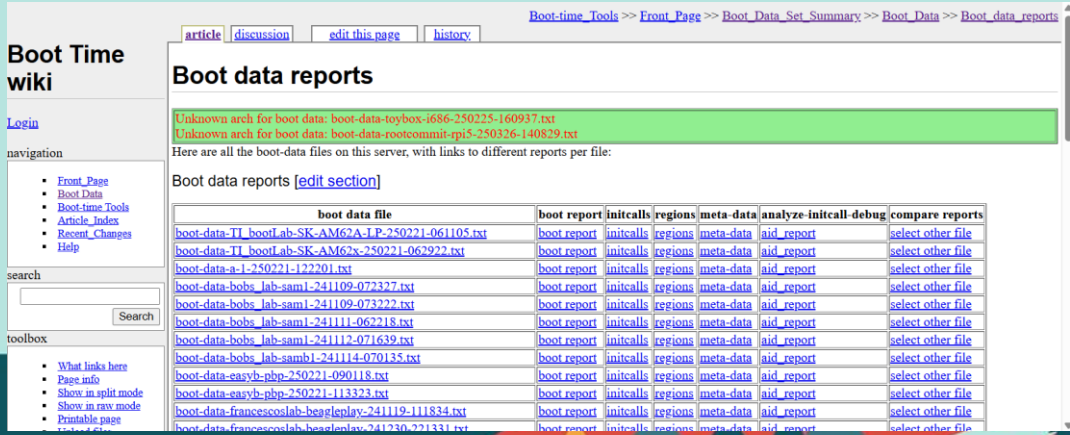
- Populating the elinux wiki Boot-Time section (Tim)
  - See https://elinux.org/Boot_Time_Project_Ideas
- Populating the boot-time data wiki
- Google Android team working on parallelized module loading (Saravana)
  - Systemd does NOT load modules in parallel, Modules.Load.D
  - modprobe - modified to support parallel module loading
  - presented at LPC 2024 (in the refereed track)
    - Doesn't need async probing, but you get bugs and issues same as with async probing
    - See https://lpc.events/event/18/contributions/1734/ - see page 13

Resources

- Boot-time section of the elinux wiki:
  - https://elinux.org/Boot_Time



- Boot time (data) wiki
  - https://birdcloud.org/boot-time



Conference and meetups:

- July SIG meeting
  - July 22, 2025
- Open Source Summit Europe?
  - August 25-27, Amsterdam, The Netherlands
  - Can do a BOF or meetup, in Amsterdam?

Anything Else?