Boot-Time Optimization for the Real World

Embedded Linux Conference Europe 2020

Michael Olbrich – m.olbrich@pengutronix.de
Motivations for this Talk

ELC-E 2019

- “We Need to Talk About Systemd: Boot Time Optimization for the New init daemon”
  - Basic introduction to boot time optimization
- “Timing Boot Time Reduction Techniques”
  - Many good techniques, impressive results
  - Unacceptable compromises for any of my projects
- That’s it?
Motivations for Boot-Time Optimization

- Hard requirements
  - Required interactions with the outside world within a certain deadline after power-on.

- Soft requirements
  - User experience
Choose Your Optimization Targets

Examples:

- First CAN message on the bus
- First content on the display
- Limited user interaction possible
- Full user interaction possible
Priorities of Conflicting Requirements

- Debugging devices in the field
- Robustness
- Security
- Development & testing
- Maintenance
Techniques

- **Disable**
  - Handled in previous presentations

- **Delay**
  - Do thing after the optimization target is reached

- **Improve**
  - Optimize initialization code

- **Cheat**
  - Find new ways to satisfy the requirements
Serial Console

- Kernel output on a serial console is very slow
- Userspace is better but still unnecessary overhead

- \texttt{loglevel=5}
  - Only show warnings or worse (should be none)
- \texttt{systemd.log_level=warning}
- \texttt{systemd.show_status=auto}
  - Only show output after an error occurs
udev Coldplug

- Enumerate existing hardware while booting
- Ensures that devices are available before accessed
- Takes a long time

→ Avoid dependencies in the hot path
udev Coldplug - Data Partitions

Use automounts

- No direct dependency on the device of the partition
- udev coldplug happens while the application is starting
- The application waits for the filesystem on the first access
Trick systemd to skip device dependencies

- The device must exist when userspace starts
- Manual fsck handling required
- What=UUID=...
  - Only works as explicit mount unit, not via fstab
- What=/symlink/outside/dev
  - Works as explicit mount unit and via fstab
Simple Qt QML Application

1. Create QGuiApplication & QQuickView
2. Read dummy file from the data partition
3. Load QML
4. Show Window
5. sd_notify()
udev Coldplug - Data Partitions - Example

- **Hardware:** STM32MP1 (Dual Cortex-A7 800MHz), eMMC
- **Start:** ~8.0s
- **Automount:** ~7.4s
- **Fake device:** ~6.7s
- **Automount + Fake device:** ~6.7s
Avoiding coldplug dependencies is not always possible

→ Two coldplug stages:
  - `udevadm trigger --type=devices --subsystem-match=drm` ...
  - `udevadm trigger --type=devices --subsystem-nomatch=drm` ...
Early Splash Screen

- Run as pid 1
- Show splash screen
- Release DRM master (drmDropMaster())
- Fork
  - Exec systemd in pid 1
  - Just wait to be killed in the child
Early Application

- fork() + exec() systemd
- Cannot take advantage of the systemd features
  - Resource control, watchdog / monitoring, security
- Possible solutions:
  - Import into service
    - Write pid to /sys/fs/cgroup/system.slice/myapp.service/cgroup.procs
    - Pass the sd_notify fd for watchdog handling
    - Still no security features
  - Restart application is a service
    - State must be transferred
Debug Features vs. Boot-Time

- Kernel tracing infrastructure
- Kernel startup until rootfs is mounted:
  - Tracing enabled: ~1.4s
  - Tracing disabled: ~0.5s
- Most of the time is spent in trace_eval_init()
  - Maybe this could be done later / on demand?
Security - Challenges and Opportunities

- Security enforces software architecture design
  - multiple processes for privilege separation
  - defined resource requirements for access permissions
- Reuse software architecture for boot-time optimization
  - Not everything needs to start immediately
  - Process ordering and startup priorities
  - Avoid dependencies in the hot path
  - ...

Designing Hardware to Boot Fast

- Fast mass storage
- No USB in the hot path
- Avoid FPGA setup in the bootloader
Questions?

Michael Olbrich – m.olbrich@pengutronix.de