Parallelizing Linux Boot on CE Devices

Vitaly Wool
Embedded Alley Solutions Inc.
Linux boot-up

• kernel initialization: sequential
  – low-level init
  – sequential subsystems init
  – sequential drivers init
  – no good means to set dependencies

• userspace initialization: sequential
  – SysV initscripts or derivative
Sequential Driver Init

• suboptimal
  – waiting for a device to init might be a long story
  – nothing happens at that time

• incomprehensive
  – some drivers depend on others to be init’ed first
  – no means in Linux kernel to explicitly specify dependencies
  – using init levels for drivers init prioritization is bogus
Boot time profiling example

- System configuration
  - ARM926 CPU, 300 Mhz
  - kernel boots and RAM disk loads from NAND
  - AC97 (WM9712) hardware for touchscreen and audio

- Boot time
  - best achievement is about 3 seconds to boot
  - 50+% spent on NAND and AC97
Boot time profiling example

- NAND init, ms: 1340 ms
- NAND disk copy, ms: 940 ms
- TS init, ms: 220 ms
- sound init, ms: 150 ms
- rest, ms: 280 ms
Asynchronous Driver Init

- start of driver init routines
  - sequential
  - asynchronous
  - dependency-based
  - “weight” for an init routine may be specified
    - counts if some routines are “equal” otherwise
  - callbacks to let the system know the routine has completed

- should go well with the device tree model
Asynchronous Driver Init

Asynchronous Driver Start w/ Dependencies

Row 1
Row 2
Row 3
Row 4

0 50 100 150 200 250 300 350 400 450 500 550 600 650 700

- USB
- I2C
- NAND Init
- AC'97 init
- wait time
Async Driver Init

• Benefits
  – Performance
    • Faster kernel startup
  – Robustness
    • manageable sequence of initialization
    • system doesn’t hang if someone’s init hangs

• Problems
  – sometimes boot time may even increase
    • more overhead
  – more concurrency, so potentially more races
Async Init: you can’t always win 😊
Async Init: only kernelspace?

- asynchronous driver init
  - helps to improve boot time
  - helps to solve driver dependency problems
- why only kernel?
  - an idea to apply this approach to initscripts
- and it’s not only kernel
  - such approaches exist already
    - upstart
    - initng
Userspace Init: legacy way

- SysV initscripts
- init daemon
  - jobs separated by run levels
  - runs a job when a particular run level is entered
    - e.g. /etc/init.d/rc 2
- Assumptions on sequence
  - e.g. a storage device must have been before mount from /etc/fstab.
SysV init: why legacy?

- Drives can be plugged in and removed at any point
- Storage buses allow more than a fixed number of drives
  - they must be scanned for new ones
- Network devices can be plugged in/removed at any point.
- Firmware may need to be loaded after the device detection, but before it is usable by the system.
upstart: an event-based daemon

• events
  – can be generated by the daemon or sent by processes
  – cause jobs to be started/stopped

• typical events
  – the system has started,
  – the root filesystem is now writable,
  – a block device has been added to the system,
  – a filesystem has been mounted
Upstart state machine

- rest states are in red
  - the job remains in this states until an event comes in
- transition states are in blue
  - allow a job to run shell script to prepare to be run/stopped/respawned
• dependency-based
  – pretty similar to async driver init solution
• Jobs declare dependencies on other jobs
  – Starting the job causes its dependencies to be started first
    • and their dependencies, and so on….
  – When jobs are stopped, if running jobs have no dependencies, they themselves can be stopped
initng VS upstart

• initng problems
  – dependency on Apache would need the daemon to be running where a dependency on “checkroot” would need the script to have finished running
  – you might not know whether something is a dependency or not without reading other configuration
    • mount NFS may be a dependency of everything under /usr or may just be a dependency of anything allowing the user to log in

• upstart doesn’t have such
Conclusions

- plain sequential init is obsolete both for kernel and userspace
- different approaches should be employed for kernelspace and userspace init
  - kernelspace: dependency-based
    - “async init”
  - userspace: event-based
    - upstart
Questions?

mailto:vital@embeddedalley.com