Reinitialization of devices after a soft-reboot

2007/2/12

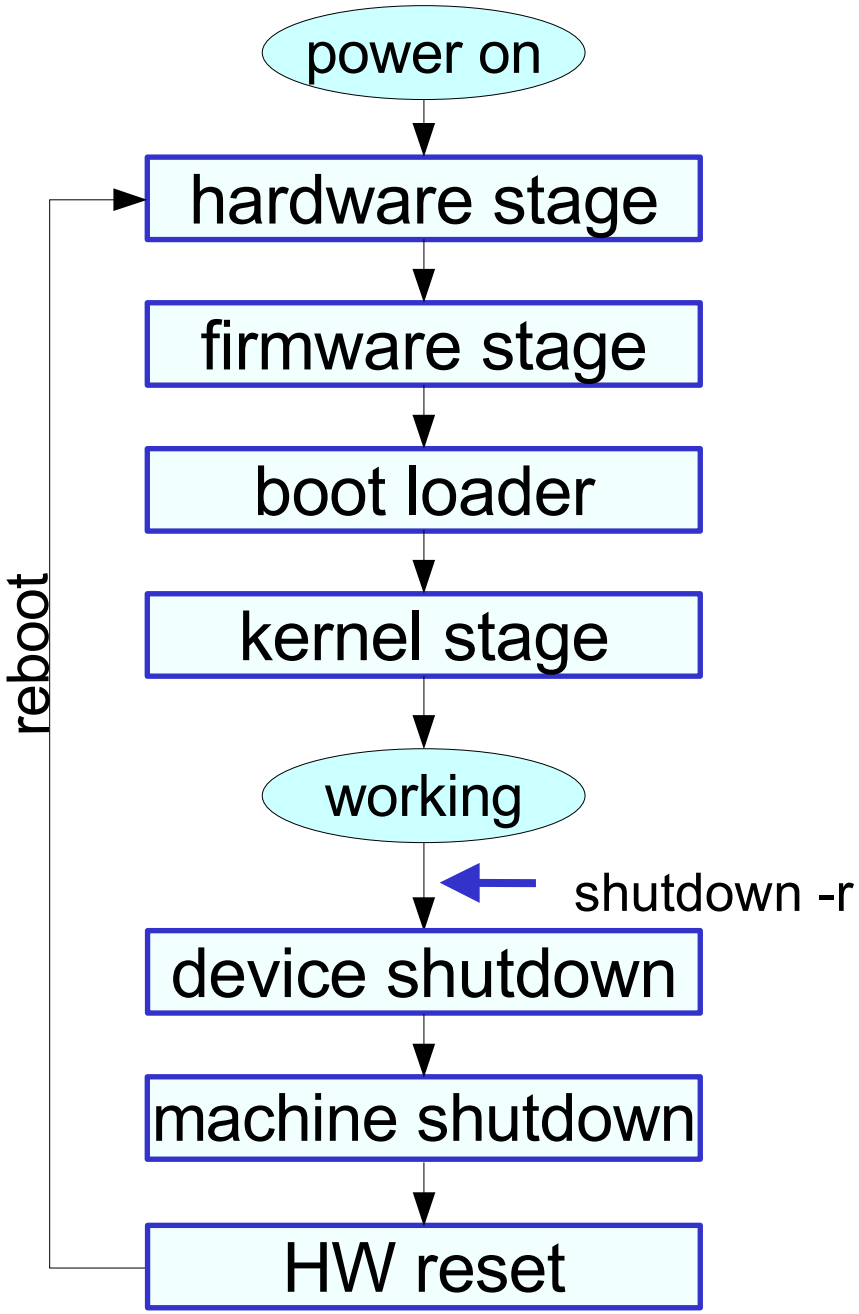
NTT Open Source Software Center
Fernando Luis Vázquez Cao

Agenda

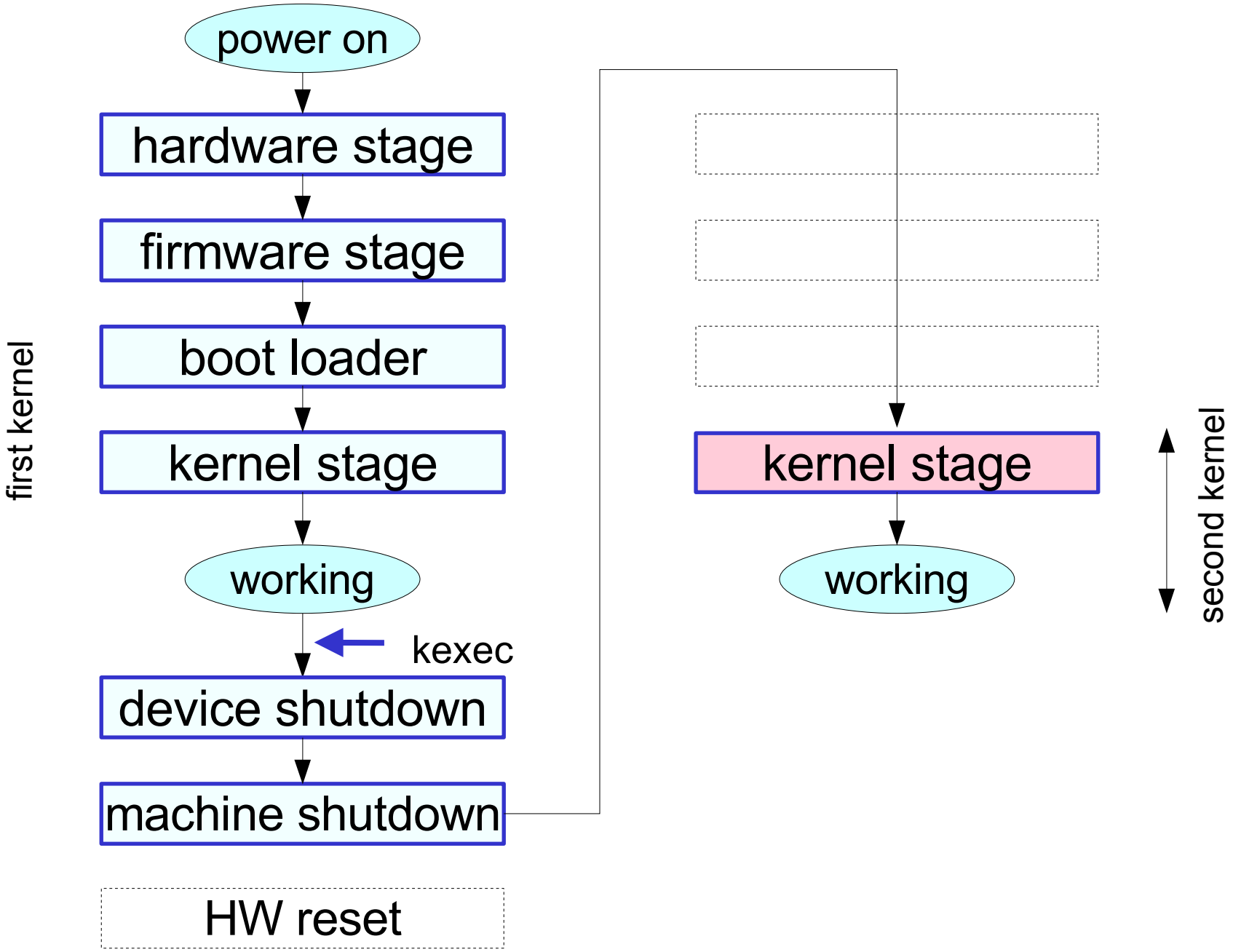
1. Kexec/kdump reboot
2. Device reinitialization
3. Tackling device reinitialization
4. Device configuration restore

1 kexec/kdump reboot

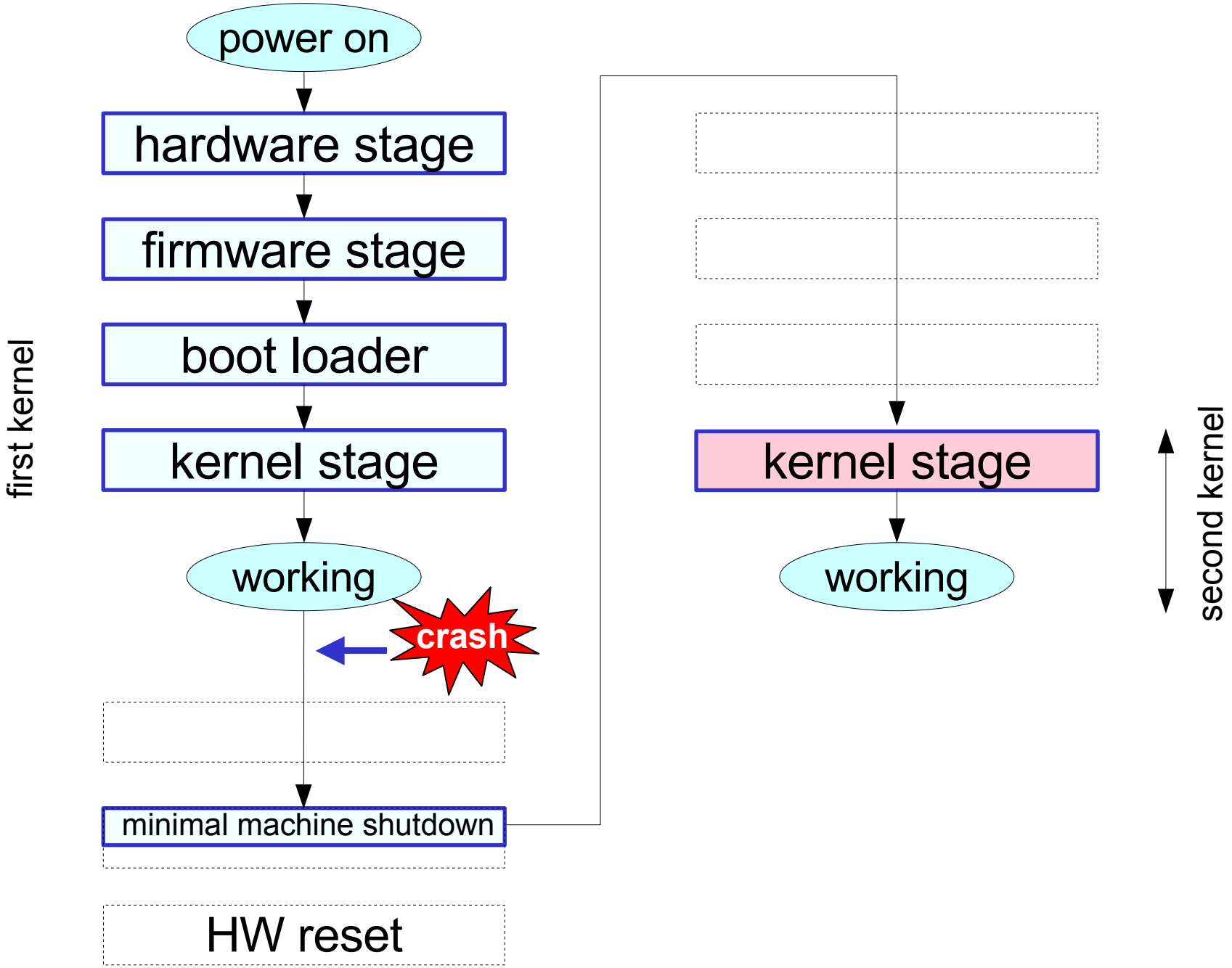
1.1. Standard boot process



1.2. Kexec boot process



1.3. Kdump boot process



2 device reinitialization

2.1. Device reinitialization issue

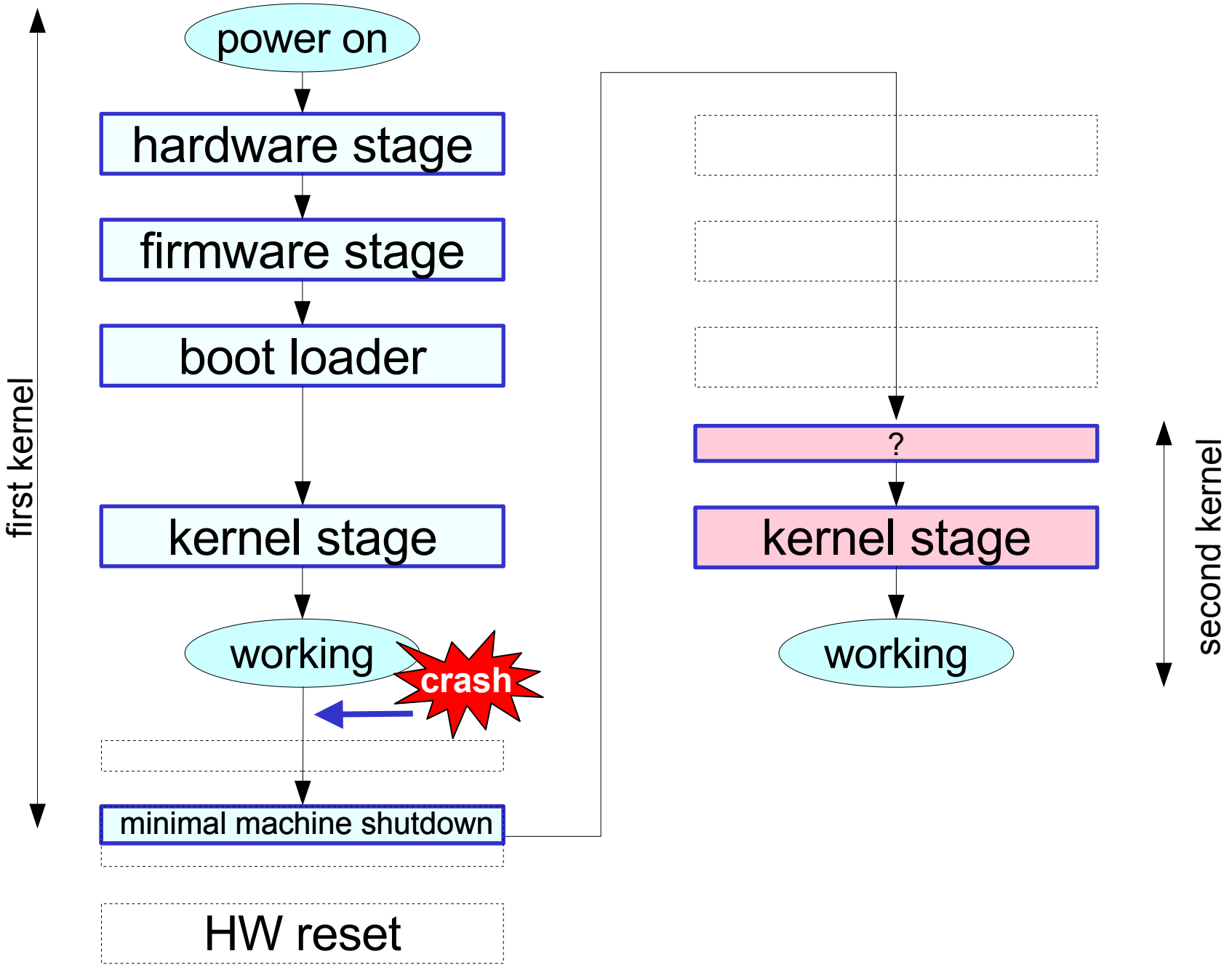
- State of devices after a kdump boot is unknown
  - The first kernel and what it knows is unreliable
    - ✗ No device shutdown in the crashing kernel
  - Firmware stage of the boot process is skipped
    - ✗ Devices are not reset
- Consequences
  - Devices may be operational or in an unstable state
- Kexec is also vulnerable when the first kernel's shutdown functions do not do their job properly

2.2. Invalid assumptions

- Drivers (implicitly) assume that the devices have been reset and/or that some pre-initialization has been performed during the firmware stage
  - Drivers find devices in an unexpected state or receive a message generated from the context of the previous kernel
    - ✗ This is an anomalous situation so the kernel panics or raises an oops

3 tackling device reinitialization

3.1. Tackling device reinitialization



3.2. Possible solutions

- Create a black list of drivers that are known to have problems (use a white list instead?)
- Device/bus reset
- Driver hardening to be able to initialize in potentially unreliable environments
  - ➢ Device configuration restore

3.3. Requirements

- Notify the second kernel that it is booting in a potentially unstable environment (use kernel parameter `reset_devices`)
- If needed, use the mechanisms offered by kexec to pass information between the first and the second kernel
- Implement the necessary solutions keeping the linux device model in mind

3.4. Device reset

- Two possibilities
  - Bus level reset (PCI, etc): need new `bus_type` method?
  - Per-device soft reset: call a device driver specific reset function from the device driver `probe`?
- Problems
  - Individual device soft-reset
    - ✗ Not all devices have this capability
    - ✗ It is a time-consuming operation in some devices
  - Bus level reset
    - ✗ Reset functionality not supported by all buses

3.5. Driver hardening

- Things that can be done to initialize a device in an unreliable environment
  - Add hacks to the initialization code
  - Relax driver's consistency checks
  - Put devices into a good known state before proceeding with the standard initialization process (device configuration restore)

4 device configuration restore

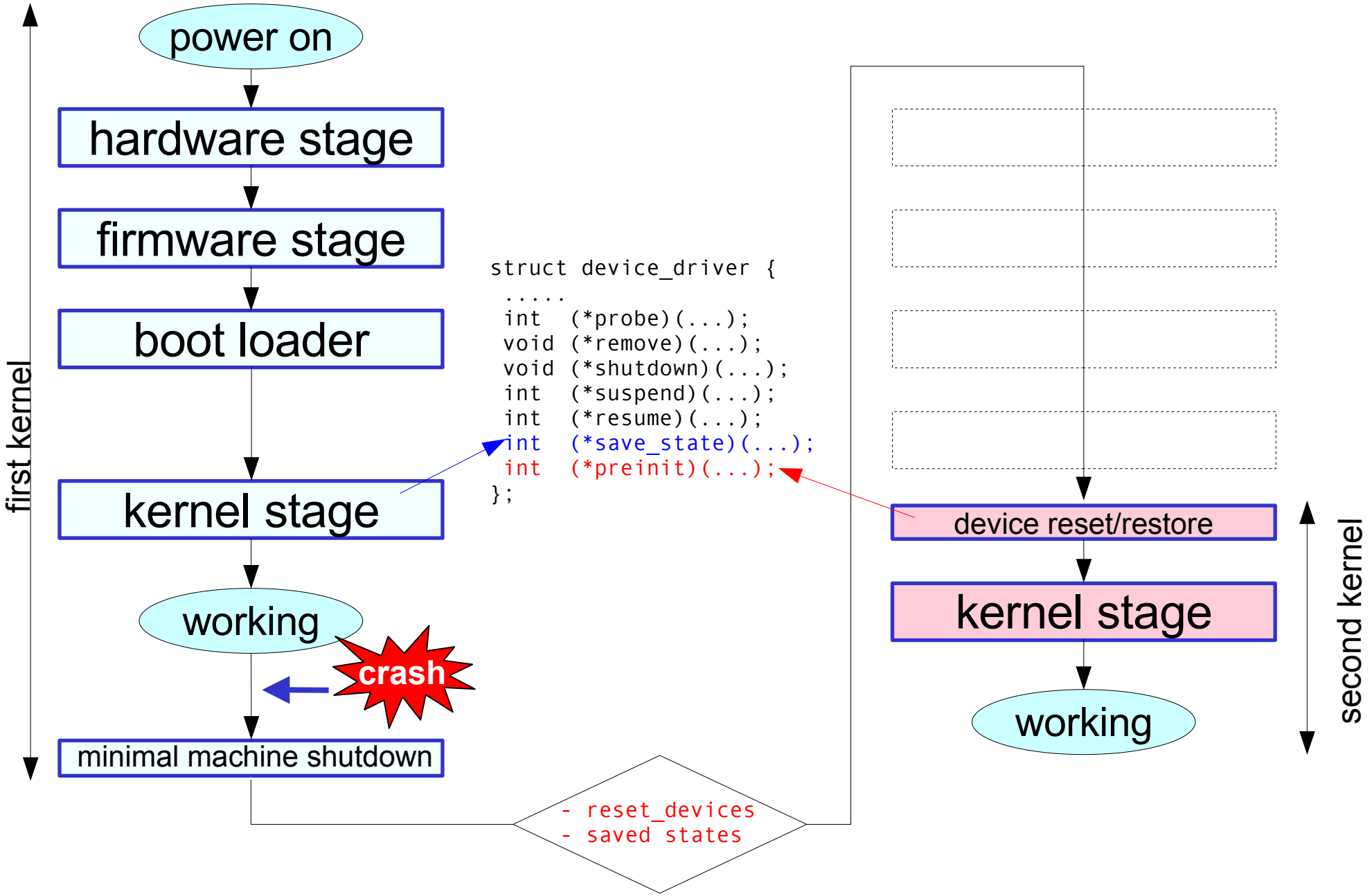
4.1. Device configuration restore

- How do we know what the right configuration is?
  - Documentation available: follow the instructions
  - No documentation available: need to find out a good configuration
- During a normal boot the firmware performs part of the configuration and the driver does the rest
  - Need an infrastructure in the second kernel doing the job the firmware usually does for us during a regular boot

4.2. Device configuration restoration

- Save/restore device configuration
  - Save the configuration as performed by the firmware in the first kernel: add new `save_early_state` method to `bus_type`, `device_driver` and `class` structures?
  - In the event of a crash notify and pass this information to second kernel (basic infrastructure exists in kexec)
  - Use this information to pre-configure devices
    - This simulates the work done by the firmware
    - Can we reuse the PM resume method? Use a new one instead (`preinit` for example)?
  - Proceed with the standard initialization

4.3. Tackling device reinitialization

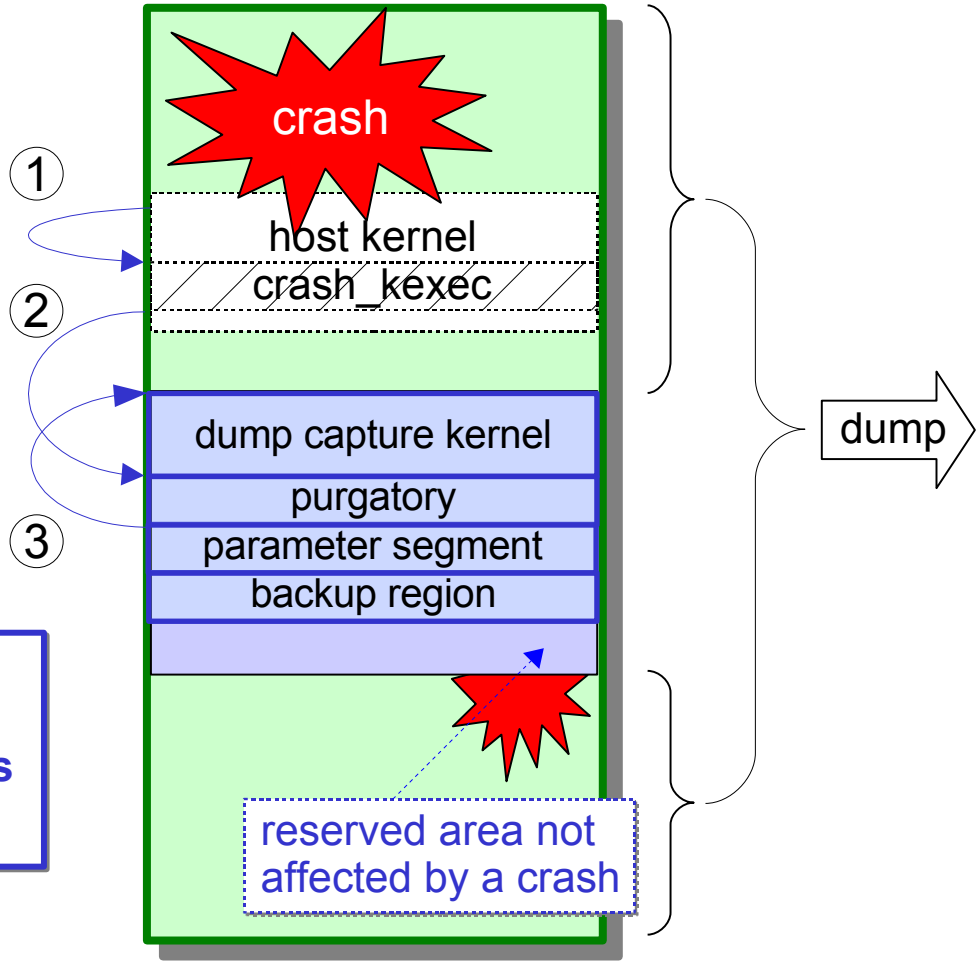


4.4. Kdump internals

1. crash detection: kdump takes control of the system

2. minimal machine shutdown: stop CPUs, APICs, etc

3. crash dump capture: performed by the dump capture kernel, which runs from a reserved area



Host kernel text and data

Reserved memory area

In-kernel machine shutdown

Thanks for your attention

Contact: fernando@oss.ntt.co.jp