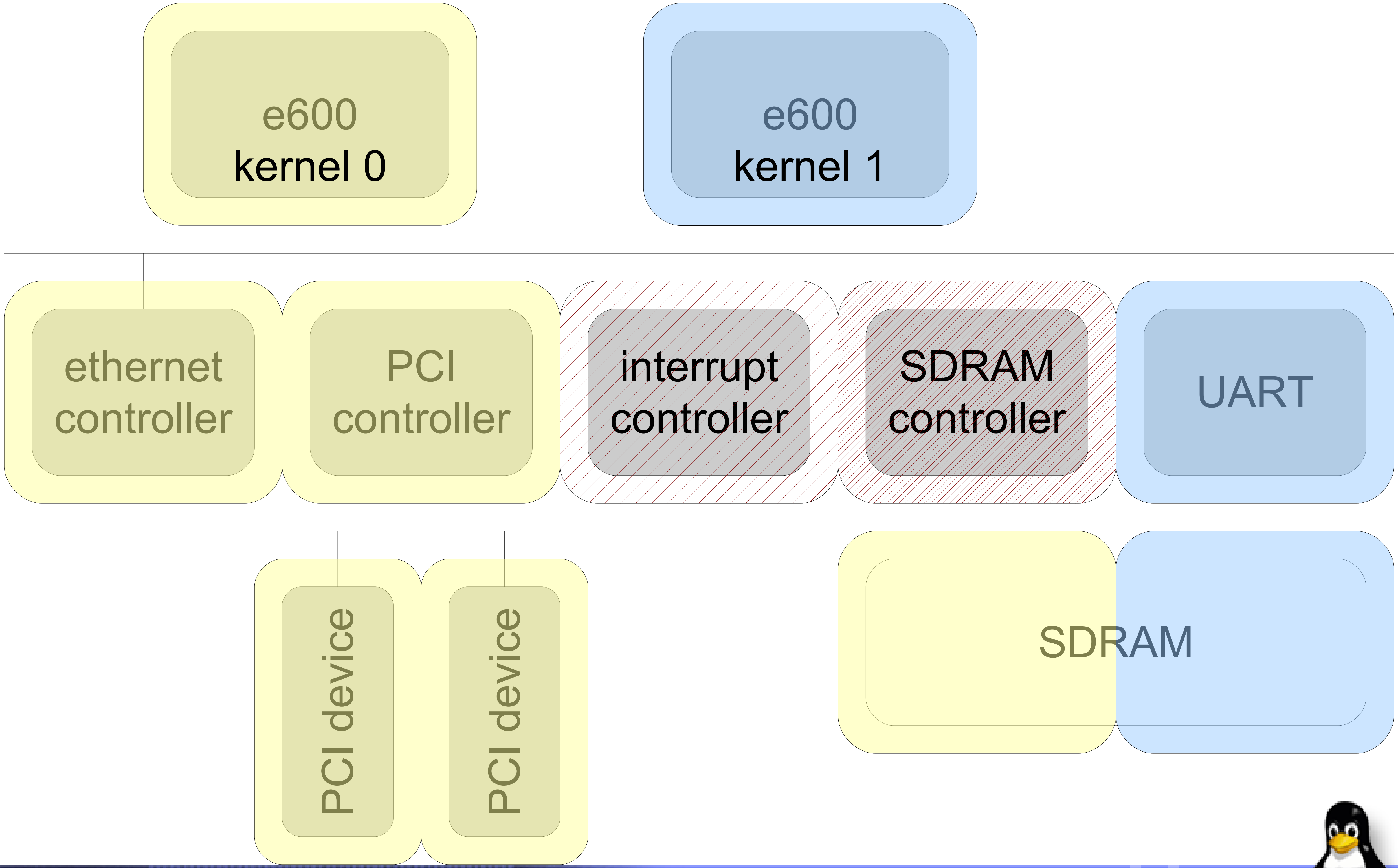
Asymmetric Multiprocessing
Linux Plumbers Conference
24 Sep 2009

Hollis Blanchard
IBM Linux Technology Center

Freescale 8641D SoC



Why do people want it?

- Multicore processors entering embedded space
- Linux + embedded OS
  - Enhance embedded OS capabilities with Linux UI, application availability, etc
  - Enhance Linux capabilities with dedicated-purpose software
    - Networking use case: control plane + data plane
    - One networking manufacturer claimed two orders of magnitude better throughput with data plane “custom executive”

What do you need?

- A way to load both kernels
  - U-boot – load both
  - CONFIG_OFFLINE – “unplug” resources from running Linux kernel
- Tell them what hardware (cores, memory, IO) they own
  - Device tree
- Modified kernels
  - E.g partial initialization of shared hardware
  - Non-zero RAM base
- Separate interrupt vectors for each core
- A way to share required hardware
- Inter-kernel communication

Shared Devices

- Devices initialized at boot
  - E.g. SDRAM controller
  - One kernel initializes, the other leaves it alone
  - Initializing kernel must have system-wide visibility to initialize correctly
    - Device tree
- Devices shared at runtime
  - Interrupt controller
    - But can't share interrupts (which core should be interrupted?)
  - Hardware must support concurrent accesses
    - e.g. write-to-clear registers for interrupt acknowledge

Inter-core Communication

- Could just use IO devices (e.g. 2 NICs + switch)
- Ideal: shared memory + interrupt
  - Starting to sound like virtio, right?
- Patches from Ira Snyder for virtio-over-pci
  - CompactPCI backplane

Related but Different: Embedded Virtualization

- Partitioning *with* isolation
  - Good for reliability and debugging
  - Even good for bringup: boot, crash, fix, repeat
- Issues
  - Many embedded processors still lack hardware virtualization features
  - Many embedded workloads are dominated by IO, which tends to suffer the greatest performance impact from virtualization

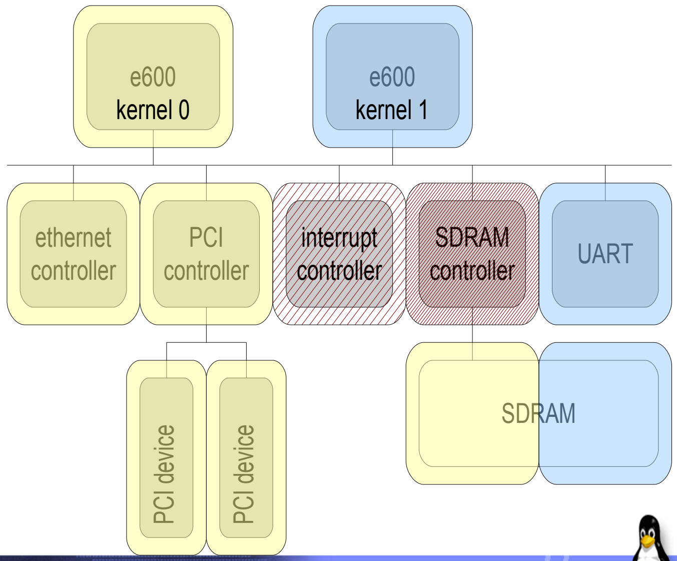
Summary

- Pros
  - Utilize multicore processors for power/space/BoM savings
  - Use specialized kernels to solve different problems
  - Doesn't require hardware virtualization support
- Cons
  - No isolation could cause very difficult debugging problems
  - Possibly invasive kernel modifications required
  - May need to duplicate IO devices
  - May need a software proxying protocol to share IO devices

Asymmetric Multiprocessing
Linux Plumbers Conference
24 Sep 2009

Hollis Blanchard
IBM Linux Technology Center

Freescale 8641D SoC



Why do people want it?

- Multicore processors entering embedded space
- Linux + embedded OS
  - Enhance embedded OS capabilities with Linux UI, application availability, etc
  - Enhance Linux capabilities with dedicated-purpose software
    - Networking use case: control plane + data plane
    - One networking manufacturer claimed two orders of magnitude better throughput with data plane “custom executive”

What do you need?

- A way to load both kernels
  - U-boot – load both
  - CONFIG_OFFLINE – “unplug” resources from running Linux kernel
- Tell them what hardware (cores, memory, IO) they own
  - Device tree
- Modified kernels
  - E.g partial initialization of shared hardware
  - Non-zero RAM base
- Separate interrupt vectors for each core
- A way to share required hardware
- Inter-kernel communication

Shared Devices

- Devices initialized at boot
  - E.g. SDRAM controller
  - One kernel initializes, the other leaves it alone
  - Initializing kernel must have system-wide visibility to initialize correctly
    - Device tree
- Devices shared at runtime
  - Interrupt controller
    - But can't share interrupts (which core should be interrupted?)
  - Hardware must support concurrent accesses
    - e.g. write-to-clear registers for interrupt acknowledge

Inter-core Communication

- Could just use IO devices (e.g. 2 NICs + switch)
- Ideal: shared memory + interrupt
  - Starting to sound like virtio, right?
- Patches from Ira Snyder for virtio-over-pci
  - CompactPCI backplane

Related but Different: Embedded Virtualization

- Partitioning *with* isolation
  - Good for reliability and debugging
  - Even good for bringup: boot, crash, fix, repeat
- Issues
  - Many embedded processors still lack hardware virtualization features
  - Many embedded workloads are dominated by IO, which tends to suffer the greatest performance impact from virtualization

Summary

- Pros
  - Utilize multicore processors for power/space/BoM savings
  - Use specialized kernels to solve different problems
  - Doesn't require hardware virtualization support
- Cons
  - No isolation could cause very difficult debugging problems
  - Possibly invasive kernel modifications required
  - May need to duplicate IO devices
  - May need a software proxying protocol to share IO devices