The Quest-V Separation Kernel for Mixed Criticality Systems

Richard West richwest@cs.bu.edu

Ye Li, Eric Missimer {liye, missimer}@cs.bu.edu





Background

- Multi- / many-core processors increasingly popular in embedded systems
- Many now feature hardware virtualization

 ARM Cortex A15, Intel VT-x, AMD-V
- H/W Virtualization provides opportunity to partition resources amongst guest VMs

H/W Virtualization + Resource Partitioning = Platform for Mixed Criticality Systems

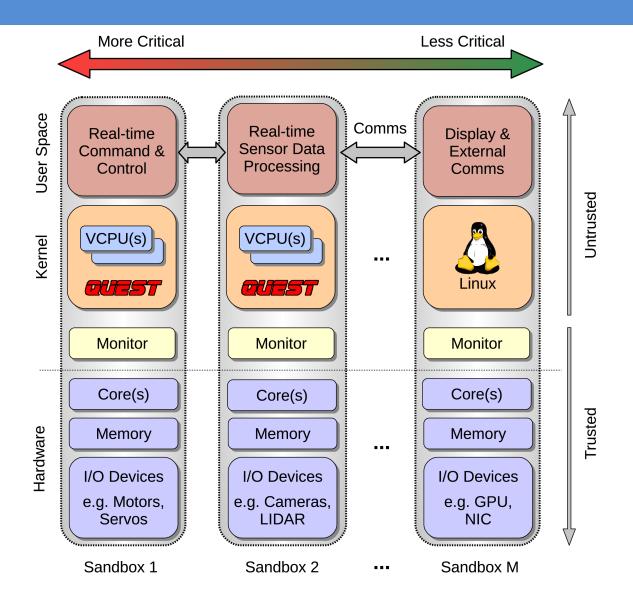
Problem

- Traditional Virtual Machine approaches too
 expensive
 - Require traps to VMM (a.k.a. hypervisor) to mux & manage machine resources for multiple guests
 - e.g., ~1500 clock cycles VM-Enter/Exit on Xeon E5506

Contributions

- Quest-V Separation Kernel
 - Uses H/W virtualization to partition resources amongst services of different criticalities
 - Each partition, or sandbox, manages its own CPU cores, memory area, and I/O devices w/o hypervisor intervention
 - Hypervisor typically only needed for bootstrapping system + managing comms channels b/w sandboxes

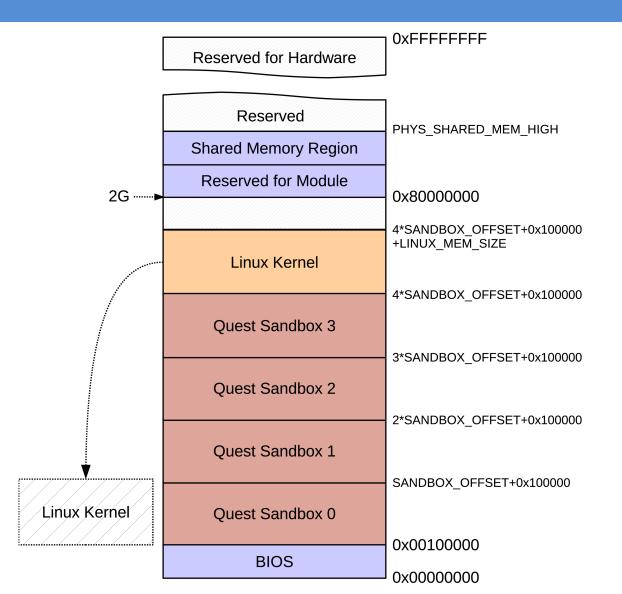
Architecture Overview



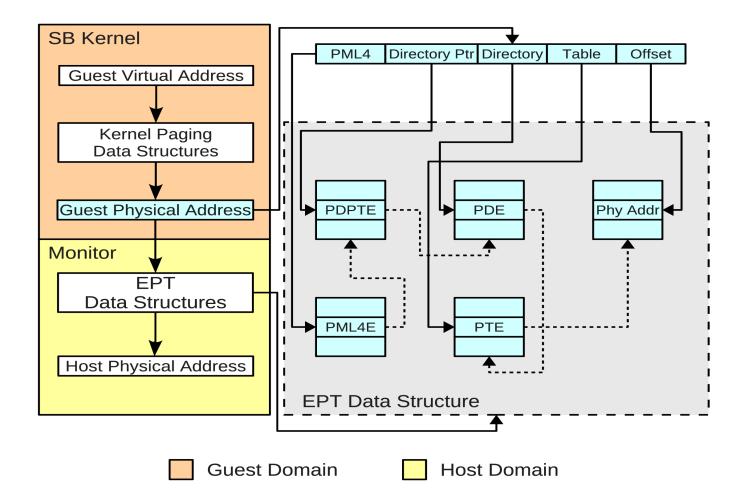
Memory Partitioning

- Guest kernel page tables for GVA-to-GPA translation
- EPTs (a.k.a. shadow page tables) for GPA-to-HPA translation
 - EPTs modifiable only by monitors
 - Intel VT-x: 1GB address spaces require 12KB EPTs w/ 2MB superpaging

Quest-V Linux Memory Layout



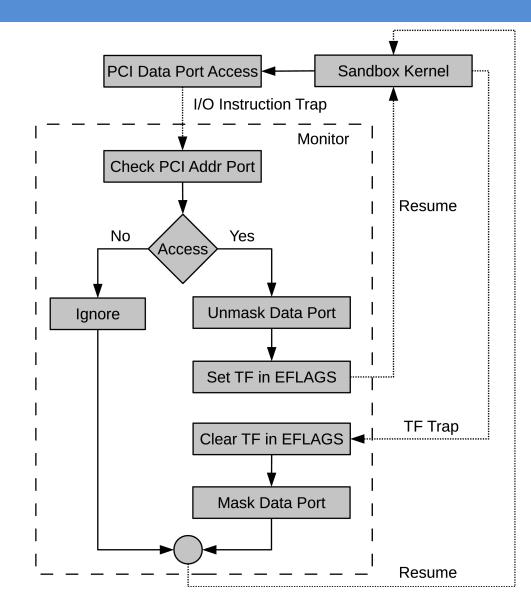
Quest-V Memory Partitioning



I/O Partitioning

- Device interrupts directed to each sandbox
 - Use I/O APIC redirection tables
 - Eliminates monitor from control path
- EPTs prevent unauthorized updates to I/O APIC memory area by guest kernels
- Port-addressed devices use in/out instructions
- VMCS configured to cause monitor trap for specific port addresses
- Monitor maintains device "blacklist" for each sandbox
 - DeviceID + VendorID of restricted PCI devices

Quest-V I/O Partitioning



10

CPU Partitioning

- Scheduling local to each sandbox
 - partitioned rather than global
 - avoids monitor intervention
- Uses VCPU approach for Quest native kernels (real-time)

Cache Partitioning

- Shared caches controlled using color-aware memory allocator
- Cache occupancy prediction based on h/w performance counters

$$- E' = (1-E/C) * m_{\mu} - E/C * m_{o}$$

Linux Front End

- For low criticality legacy services
- Based on Puppy Linux 3.8.0
- Runs entirely out of RAM including root filesystem
- Low-cost paravirtualization
 - less than 100 lines
 - Restrict observable memory
 - Adjust DMA offsets
- Grant access to VGA framebuffer + GPU
- Quest native SBs tunnel terminal I/O to Linux via shared memory using special drivers

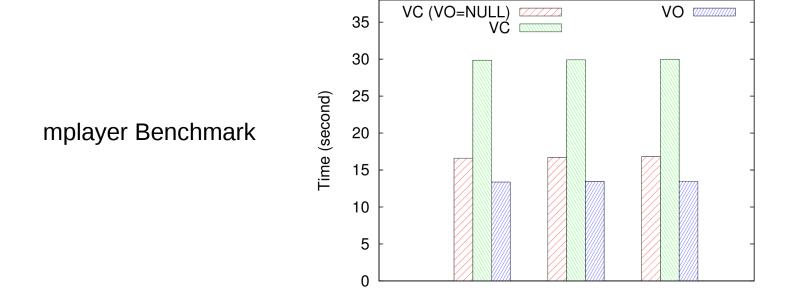
Quest-V Linux Screenshot

* Copyri	st-V kernel version: 0.1a *** //- ght Boston University, 2013 * urrent Output: 0x0001 ******* _		
	to Quest-V Sandbox 1!		qis
Image: second secon	■ rxvt * **** Quest-V kernel version: C * Copyright Boston University, ****** Current Output: 0x0000 CPUID says: Intel(R) Co Multi-processing detected. Nu VMX support detected Welcome to Quest-V Sandbox 0! Waiting for VM-Forks All sandboxes forked	, 2013 * \\\ ******* \\ \ \\/\\\ ore(TM) i5-2500K CPU @ 3.30GHz	_ = ×
connect ▲# uname -a Linux puppypc2033 3.8.0 # free total	ROOT: RAMDISK free kernel pages=323 >	_ □ × 3 i686 GNU/Linux buffers	
Mem: 508444 -/+ buffers: Swap: 0 # cat /proc/cpuinfo processor : 0 vendor_id : Genuin cpu family : 6 model : 42		46272	
model name : Intel(stepping :7 microcode <td:0x25< td=""> cpu MHz <td:3400.0< td=""> cache size <td:6144 k<="" td=""> fdiv_bug <td:no< td=""> hlt_bug <td:no< td=""> f00f bug <td:no< td=""></td:no<></td:no<></td:no<></td:6144></td:3400.0<></td:0x25<>		2	
a cmov pat pse36 clflush plm constant_tsc arch_p	e de psetscmsrpaemcecx8apic dtsacpimmxfxsrssesse2ssht erfmonpebsbtsnonstop_tscaperfm tordscplesttm2sse3cx16xtpr	tm pbe nx rdtsc nperfeagerfpu p	
_1 sse4_2 popcnt tsc_dea aveopt pln pts dtherm bogomips : 6584.9 clflush size : 64 cache_alignment : 64	dline_timer aes xsave avx lahf_lm		
f # 📕	nvt	prvt	03:57

14

Quest-V Performance Overhead

- Measured time to play back 1080P MPEG2 video from the x264 HD video benchmark
- Mini-ITX Intel Core i5-2500K 4-core, HD3000 graphics, 4GB RAM



Linux

Quest Linux Quest Linux 4SB

Conclusions

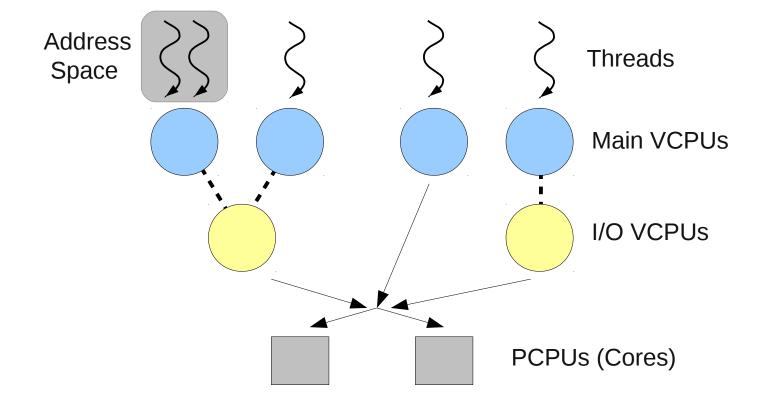
- Quest-V separation kernel built from scratch
 - Distributed system on a chip
 - Uses (optional) h/w virtualization to partition resources into sandboxes
 - Protected comms channels b/w sandboxes
- Sandboxes can have different criticalities
 - Linux front-end for less critical legacy services
- Sandboxes responsible for local resource management
 - avoids monitor involvement

See: www.questos.org for more details

Future Work

- Online fault detection and recovery
- Technologies for secure monitors
 - e.g., Intel TXT

VCPUs in Quest(-V)



Predictability

- VCPUs for budgeted real-time execution of threads and system events (e.g., interrupts)
 - Threads mapped to VCPUs
 - VCPUs mapped to physical cores
- Sandbox kernels perform local scheduling on assigned cores
 - Avoid VM-Exits to Monitor eliminate cache/TLB flushes

Memory Virtualization Costs

- Example Data TLB overheads
- Xeon E5506 4-core @ 2.13GHz, 4GB RAM

