Virt/RK: A Real-Time Virtualization Framework for Multi-Core Platforms

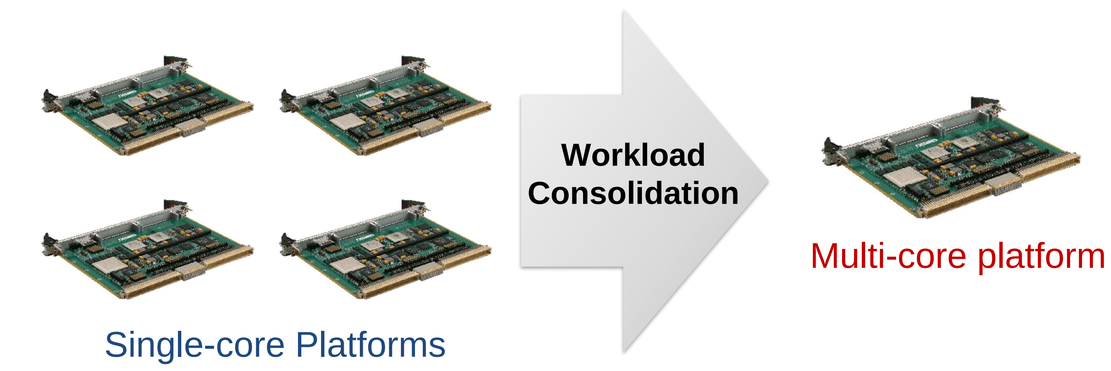
Hyoseung Kim and Ragunathan (Raj) Rajkumar

Electrical and Computer Engineering
Carnegie Mellon University

Electrical & Computer ENGINEERING

Benefits of Multi-Core Platforms

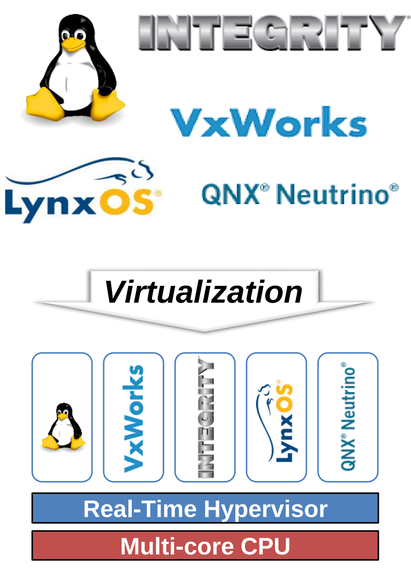
- Workload consolidation onto a multi-core CPU



- Reduces the number of CPUs and wiring harnesses among them
- Leads to a significant reduction in size, weight, power and cost requirements

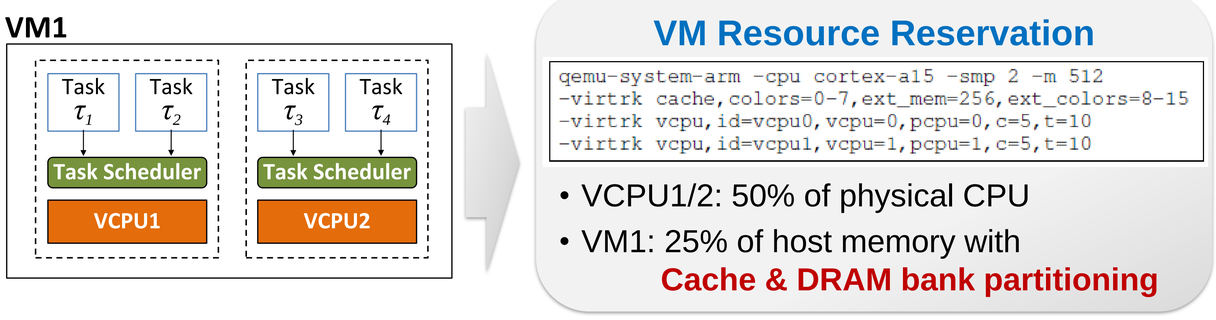
Real-Time System Virtualization

- **Barriers** to consolidation:
  - Applications are typically developed independently by different vendors
    - Bare-metal, proprietary OS
    - Linux, Android
  - IP and licensing considerations
- **Consolidation** via **virtualization**
  - Each application can maintain its own implementation
  - Minimizes re-certification process
  - IP protection and license segregation
  - Fault isolation



Virt/RK

- Real-time virtualization with resource kernel approach
  - CPU reservation for VCPUs + Memory reservation for VMs



VM Resource Reservation

```
qemu-system-arm -cpu cortex-a15 -smp 2 -m 512
-virtrk cache,colors=0-7,ext_mem=256,ext_colors=8-15
-virtrk vcpu,id=vcpu0,vcpu=0,pcpu=0,c=5,t=10
-virtrk vcpu,id=vcpu1,vcpu=1,pcpu=1,c=5,t=10
```

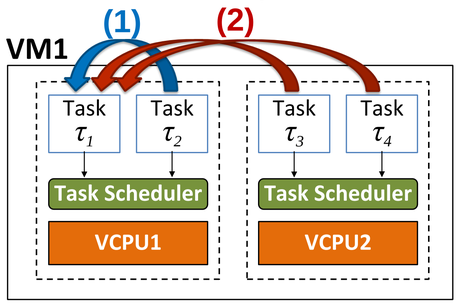
- VCPU1/2: 50% of physical CPU
- VM1: 25% of host memory with **Cache & DRAM bank partitioning**

Prevents inter-VM interference

- Current implementation: Virt/RK::KVM-x86, Virt/RK::KVM-ARM
  - Virt/RK::L4 under development

Cache Interference within a VM

- **Problem**: cache interference among tasks within the same VM
  - Each VM can be assigned private cache partitions
  - But, those cache partitions are *shared* among tasks running in the VM



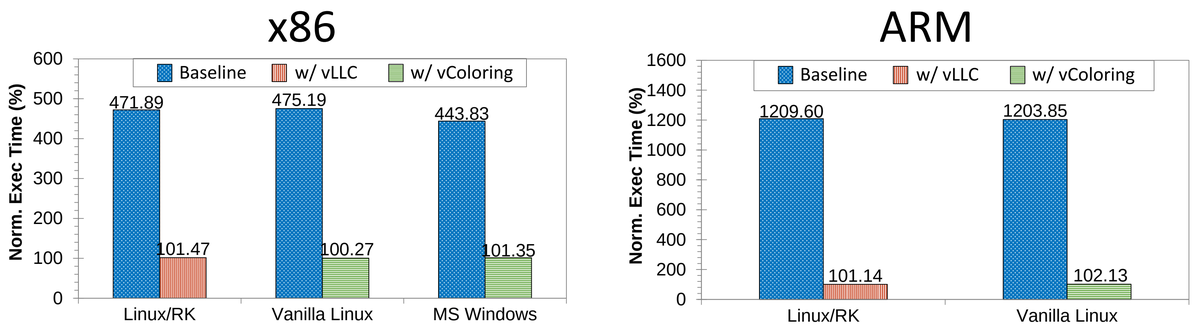
(1) Intra-VCPU cache interference: tasks running on the same VCPU

(2) Inter-VCPU cache interference: tasks running on different VCPUs

- **Solutions**: vLLC & vColoring, hypervisor-level methods to manage the cache allocation of individual tasks running in a VM
  - Supports proprietary, closed-source guest OSs

Experimental Results

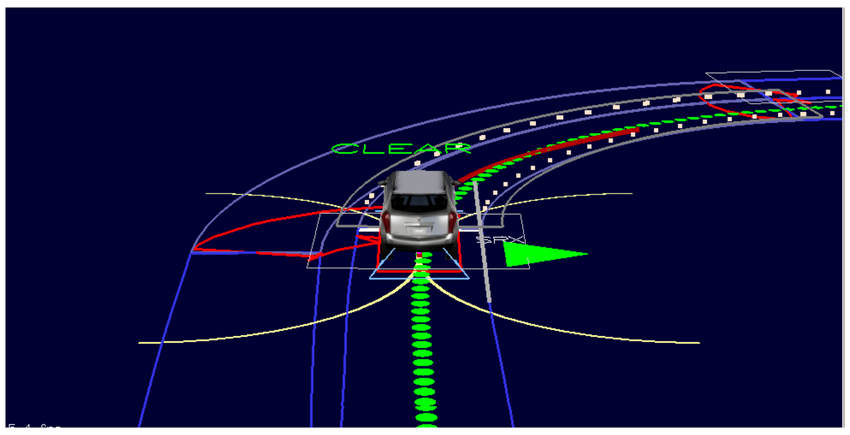
- Experimental Setup
  - x86: Intel i7-2600 four cores, ARM: Exynos 5422 (four ARM Cortex-A15 cores)
  - Guest OSs: Linux/RK, Vanilla Linux, MS Windows Embedded (x86 only)
- Inter-VCPU interference among cache-sensitive tasks within a VM



Virt/RK with vLLC & vColoring effectively prevents cache interference inside a VM

Demonstration

- Virtualization of the driving context of an autonomous vehicle*†



* C. Urmson et al., "Autonomous Driving in Urban Environments: Boss and the Urban Challenge", *Field and Robotics*, 2008.
† J. Wei et al., "Towards a Viable Autonomous Driving Research Platform", In *IEEE Intelligent Vehicles Symposium* (IV), 2013.