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

Hyoseung Kim and Ragunathan (Raj) Rajkumar
Carnegie Mellon University
hyoseung@cmu.edu, raj@ece.cmu.edu

I. Virt/RK Framework

In this demo, we present Virt/RK, a real-time virtualization framework specifically developed for multi-core platforms. Virt/RK combines (i) a theoretical framework to analyze the timing requirements of virtualized workloads, (ii) a real-time hypervisor to host multiple real-time guest OSs, and (iii) a suite of tools to allocate resources to real-time applications hosted in a virtualized environment. Virt/RK uses the resource kernel (RK) approach [6] to explicitly account for and enforce the resource usage of individual guest virtual machines. In addition, Virk/RK includes virtualization-aware multi-core synchronization [4] and interrupt handling [5] schemes to satisfy the timing requirements of real-world virtualization scenarios. Virt/RK is currently implemented in KVM/QEMU running on x86 and ARM and is being ported to L4/Fiasco.

A. VCPU Resource Reservation

Virtualization technologies typically provide a two-level hierarchical scheduling structure. Each VM has one or more virtual CPUs (VCPUs) that are scheduled by the hypervisor on physical CPUs (PCPUs). The tasks of a VM are scheduled by a guest OS on the VCPUs of that VM. In this hierarchical scheduling structure, the timeliness of tasks running in a VM is affected by the timing parameters of VCPUs, such as the amount of resource assigned to a VCPU and the resource supply policy for VCPUs.

The VCPU resource reservation of Virt/RK is based on our experience in resource-centric real-time kernel design and analysis [6]. With this approach, VCPUs can specify their resource demands and the hypervisor takes the responsibility of satisfying the specified resource and timing demands using its own resource management policies. The hypervisor also guarantees that the demands of the VCPUs will be satisfied by performing admission control tests and preventing the VCPUs from overrunning their pre-specified resource demands. VCPU resource reservation can be configured to cooperate with virtualization-aware synchronization and interrupt handling schemes [4, 5].

B. VM Memory Reservation

Modern multi-core platforms have multiple levels of shared memory resources, such as caches, buses, and DRAM banks. Tasks running on different PCPUs may contend with each other to access those shared memory resources, resulting in significant increases in execution time. As memory-intensive applications become more prevalent in real-time systems, such contention on shared memory resources should be considered in multi-core virtualization environments. Therefore, the VM memory reservation of Virt/RK combines the traditional memory reservation approach [3] with cache and bank partitioning techniques [1, 2]. First, a guest VM can specify its physical memory demands, and once the memory is reserved, the reserved memory pages are exclusively used by the VM. This ensures that the potential abuse of memory by any VM does not lead to memory pressure on other VMs. Secondly, a guest VM can reserve a specific portion of the cache and DRAM banks of the host machine. Assigning a dedicated set of cache and DRAM bank resources to a VM significantly reduces the negative impact of cache and memory interference in multi-core platforms. Virt/RK also enables the cache allocation of individual tasks running in a VM.

II. Demonstration

We demonstrate the benefits of Virt/RK with two scenarios. The first scenario is to virtualize the driving context of an autonomous vehicle [7, 8]. The software simulates a self-driving car traveling around a test loop in Pittsburgh, PA, USA. Without Virt/RK, the autonomous vehicle software running in a VM experiences temporal interference from the executions of other VMs. With Virt/RK, it is nearly unaffected by the other VMs. The second scenario is to execute multiple cache-sensitive tasks in the same VM. Without Virt/RK, tasks suffer from cache interference penalties and their execution times are significantly increased. However, with Virt/RK, each task can be assigned dedicated cache partitions and does not experience any cache interference.

References

[1] H. Kim et al. A coordinated approach for practical OS-level cache management in multi-core real-time systems. In *ECRTS*, 2013.
[2] H. Kim et al. Bounding memory interference delay in cots-based multi-core systems. In *RTAS*, 2014.
[3] H. Kim and R. R. Rajkumar. Memory reservation and shared page management for real-time systems. *Journal of Systems Architecture*, 60(2):165–178, 2014.
[4] H. Kim, S. Wang, and R. Rajkumar. vMPCP: A synchronization framework for multi-core virtual machines. In *RTSS*, 2014.
[5] H. Kim, S. Wang, and R. Rajkumar. Responsive and enforced interrupt handling for real-time system virtualization. In *RTCSA*, 2015.
[6] R. Rajkumar et al. Resource kernels: A resource-centric approach to real-time and multimedia systems. In *MMCN*, 1998.
[7] C. Urmson et al. Autonomous driving in urban environments: Boss and the urban challenge. *Field and Robotics*, 2008.
[8] J. Wei et al. Towards a viable autonomous driving research platform. In *IEEE Intelligent Vehicles Symposium (IV)*, 2013.