
Raoul Rivas, Ahsan Arefin, Klara Nahrstedt

UPCRC, University of Illinois at Urbana-Champaign
Motivation

• Video Sharing, Internet TV and Teleimmersive Systems have made Content Delivery Networks ubiquitous

• Virtualization has emerged as a popular architecture for server consolidation

• Current VMMs are not suitable for these multimedia systems
Motivation

• In the teleimmersive system, gateways form a CDN for correlated multistreams.

• The TEEVE system uses virtualization to enable multiple simultaneous sessions

• **Janus resides at each gateway**
Current Efforts

• Two groups of Virtual Machine Monitors (VMMs)
  – Hard-Realtime → Critical Infrastructure (e.g. avionics)
    • Xtratum, RTS
  – Best-Effort → throughput / general-purpose (e.g. data centers, desktop virtualization)
    • Xen, VMWare Server ESX, L4Ka, Hyper-V.
Hard Real-Time Hypervisors

- Use GRMS and EDF schedulers
- They provide complete hardware separation
  - Low Hardware Utilization
  - Limited Virtualization Possibilities
  - Hardware Duplication

Examples: RTS Real-Time Hypervisor, Xtratum
Best-Effort Hypervisors

• Proportional Schedulers and Fairness
  – Credit Scheduler, BVT [Duda et al.], Round Robin and SEDF [Leslie et al.]

• Unsuitable for multimedia systems
  – No QoS guarantees (e.g. Bandwidth, Delay)
Problem Description

- Guest OS level CPU contention
  - Different Processes compete for the Virtual CPU (VCPU) time
- Hypervisor level CPU contention
  - Different VMs compete for the CPU time
- Compatible Schedules between Guest OS and VMM
  - Hierarchical to Flat Allocation Problem
Resource Allocation

**TASKS** (OS level)

**VCPUs** (VMM level)

**CPUs** (Hardware Level)
Hierarchical to Flat Allocation Problem

Our solution must also meet the QoS constraints of the application!
Assumptions

• Our Real-time applications use the Liu and Layland Model
  – Each application has a periodic job with a deadline.
  – Deadline is equal to the period

• The number of CPU cores is limited
  – We have more applications than cores
Our Contribution

• An architecture that solves the Hierarchical to Flat Resource Allocation Problem for CPUs through cross-layering and cooperation

• Algorithms and a protocol design that provide soft real-time capabilities suitable for multimedia applications used in CDN

• An experimental validation of Janus based on Xen
Solution Overview

• Three phase application lifecycle
  – Registration Phase
    • QoS parameters: (VM_id, Task_id, Period, Slice)
  – Running Phase
  – Unregistration Phase
• Distributed Kernel-level Coordinator
• Real-Time Scheduler: partitioned EDF-based multicore scheduler
• Janus Executive: Service layer, DKLC ↔ RTS
Janus Architecture
Xen-Based Janus

Feedback Daemon

RT App

PROC

Kernel-level Coordinator

CPU Affinity Manager

Linux CPU Affinity Manager

Linux Scheduler

VCPU

VCPU

VCPU

VCPU

Janus Executive

Admission Control

VCPU Free Bitmap

VCPU Mapper

VCPUP Classifier

Real-Time Scheduler

BE Credit Scheduler

RT Taskset Partition Manager

CPU

CPU

Linux Kernel (Guest OS)
Janus Protocols

Control Plane
• Admission Control
• Mapping of Tasks into VCPU
• Mapping of VCPU into CPU

Execution Plane
• Scheduling
• Policy Enforcement
1. App sends the QoS parameters to KLC (C,P,PID)
Control Plane

- 1. App sends the QoS parameters to KLC (C,P,PID)
- 2. KLC forwards them to Admission control and includes VM_id
Control Plane

1. App sends the QoS parameters to KLC (C,P,PID)
2. KLC forwards them to Admission control and includes VM_id
3. Check Admission Control condition and assign a physical CPU (next slide)
4. Request a VCPU from the VCPU Mapper
Control Plane

- 5. VCPU Mapper assigns a free VCPU to the RT task
5. VCPU Mapper assigns a free VCPU to the RT task

6. VCPU Mapper replies to the admission control with the number

7. Admission control returns the number

8. KLC notifies App on the Admission request

9. Admission Control notifies the Physical CPU to the RT scheduler
Admission Control

- Partitioned EDF requires to partition the tasks in bins (each bin is a CPU)
- We use the worst fit algorithm for partitioning
- Utilization based admission control

\[ U_{\text{Task}} = \frac{C_{\text{Task}}}{P_{\text{Task}}} \]

\[ U_{\text{cpu}[i]} = \sum_{\text{Task} \in \text{cpu}[i]} U_{\text{Task}} \]

\[ U_{\text{cpu}[i]} + U_{\text{NewTask}} \leq 1 \]
Admission Control

- \( U(T1) = 0.2, \; U(T2) = 0.5 \)

\[
\begin{align*}
U(CPU1) + U(T1) &= 0.8 \leq 1 \\
U(CPU2) + U(T1) &= 0.6 \leq 1 \\
U(CPU3) + U(T1) &= 0.5 \leq 1
\end{align*}
\]

- \( U(CPU1) = 0.6 \)
- \( U(CPU1) = 0.4 \)
- \( U(CPU1) = 0.3 \)
Admission Control

- $U(T1) = 0.2$, $U(T2) = 0.5$

$U(CPU1) + U(T2) = 1.1 \geq 1$

$U(CPU2) + U(T2) = 0.9 \leq 1$

$U(CPU3) + U(T2) = 1.0 \leq 1$

$U(CPU1) = 0.6$  $U(CPU2) = 0.4$  $U(CPU3) = 0.5$
**Execution Plane**

- BE VCPU follow the BE VM Scheduler
- BE VCPU contain multiple BE and follow the Linux Scheduler Policy
- RT VCPU contain one RT task
- RT VCPU can also contain kernel threads managed by the Linux Scheduler
- CPU Affinity Manager is responsible of enforcing RT Applications follow their QoS
CPU Affinity Manager

- A RT CPU can be scheduled by the BE Scheduler and the RT Scheduler
- RT Scheduler → RT App
- BE Credit Scheduler → Linux Scheduler handle the interrupt or the kernel thread

RT App  Kernel Threads  BE Apps  CPU Affinity Manager

VCPU  VCPU  Linux Scheduler

VCPU Classifier

Real-Time Scheduler  BE Credit Scheduler

CPU  CPU
Experimental Evaluation

• Realtime Workload
  – FFMPEG decoding 480p MJPEG videos

• Best-Effort Workload
  – 64-bit integer factorial

• Xen 3.4.1 with paravirtualized Linux 2.6.18 (64-bit)

• Thinkpad T61: Core 2 Duo 2.6 Ghz and 4Gb RAM
Jitter Evaluation

• Trace of 100 jobs for RT workload (33 and 40 ms).
Experiment 2

- One RT task in one VM (period 66ms). One BE task in each VM (2 and 3 VM)

![Graph showing missed deadlines and average jitter for different configurations.

- Missed Deadlines (%)
- Avg Jitter (ms)

2VM Credit 2VM Janus 3VM Credit 3VM Janus

0 5 10 15 20 25 30 35 40 45 50
Conclusion

• Janus ensures RT guarantees to application based on QoS parameters
• Transparent approach to the Guest OS
• Minimal kernel and Xen modifications
• We leverage the hot plugging interface in the OS
• We leverage the hard CPU affinity of Linux
• Each VCPU can be scheduled by both the Proportional Share Scheduler or the RT Scheduler (Cooperative Scheduling)
  – Allows interrupts, Watchdog timers, Migration Threads
• A soft real-time work conserving architecture
  – Better Power Management
Future Work

• Adaptation of QoS parameters.
  – Slice and Period Changes
• Semi- heterogeneous architectures
  – Different ISA, speed, pipeline
  – Pentium II and Core 2 Duo (MMX and SSE2)
• Intel Turbo Boost and Dynamic Voltage Scaling
  – Overloading
  – Power Management
• Other Architectures: Windows/Hyper-V