Scheduler Support for Video-oriented Multimedia on Client-side Virtualization

Hwanju Kim¹, Jinkyu Jeong¹, Jaeho Hwang¹, Joonwon Lee², and Seungryoul Maeng¹

Korea Advanced Institute of Science and Technology (KAIST)¹
Sungkyunkwan University²

ACM Multimedia Systems (MMSys)
February, 22-24, Chapel Hill, North Carolina, USA
Virtualization Everywhere

• **A new layer** between OS and HW

= hypervisor

VM

OS

VM

OS

Hypervisor

Multiple OSes
High resource utilization
Strong Isolation
Easy management
Live maintenance
Fast provisioning
...

Server-side virtualization

Cloud computing
Virtual desktop infrastructure

Client-side virtualization
Client-side Virtualization

- Multiple OS instances on a local device
- Primary use cases
  - Different OSes for application compatibility
  - Consolidating business and personal computing environments on a single device
- BYOD: Bring Your Own Device
Multimedia on Virtualized Clients

- Multimedia is ubiquitous on any VM

1. Multimedia workloads are dominant on virtualized clients
2. Interactive systems can have concurrently mixed workloads
Issues on Multi-layer Scheduling

- A multimedia-agnostic hypervisor invalidates OS policies for multimedia

Larger CPU proportion & Timely dispatching

I’m unaware of any multimedia-specific OS policies in a VM, since I see each VM as a black box.
Multimedia-agnostic Hypervisor

- Multimedia QoS degradation
  - Two VMs with equal CPU shares
  - Multimedia VM + Competing VM

Video playback (720p) on VLC media player

Quake III Arena (demo1)
Possible Solutions to Semantic Gap

• Explicit vs. Implicit

**Explicit**
- OS cooperation

**Explicit User involvement**
- Simple

**Implicit Hypervisor-only**
- Transparency

- Infeasible w/o multimedia-friendly OS schedulers
- Inconvenient
- Unsuitable for dynamic workloads
- Difficult to identify workload demands at the hypervisor
Proposed Approach

- Multimedia-aware hypervisor scheduler
  - Transparent scheduler support for multimedia
  - No modifications to upper layer SW (OS & apps)
  - “Feedback-driven VM scheduling”

Challenges

1. How to estimate multimedia QoS based on a small set of HW events?
2. How to control CPU scheduler based on the estimated information?
Multimedia QoS Estimation

- What is estimated as multimedia QoS?
  - "Display rate" (i.e., frame rate)
    - Used by HuC scheduler [TOMCCAP’06]

- How is a display rate captured at the hypervisor?
  - Two types of display
    1. Memory-mapped display (e.g., video playback)
    2. GPU-accelerated display (e.g., 3D game)
Memory-mapped Display (1/2)

- How to estimate a display update rate on the memory-mapped framebuffer
- Write-protection for virtual address space mapped to framebuffer

The hypervisor can inspect any attempt to map memory

Sampling to reduce trap overheads (1/128 pages, by default)
Memory-mapped Display (2/2)

• Accurate estimation
  • Maintaining display rate per task
    • An aggregated display rate does not represent multimedia QoS
  • Tracking guest OS task at the hypervisor
    • Inspecting address space switches (Antfarm [USENIX’06])

• Monitoring audio access (RSIO [SIGMETRIC’10])
  • Inspecting audio buffer access with write-protection
  • A task with a high display rate and audio access ➔ a multimedia task
GPU-accelerated Display (1/2)

- Naïve method
  - Inspecting GPU command buffer with write-protection or polling
    - Too heavy due to huge amount of GPU commands

- Lightweight method
  - Little overhead, but less accuracy
    - 3D games are less sensitive to frame rate degradation than video playback
  - **GPU interrupt-based estimation**
    - An interrupt is typically used for an application to manage buffer memory

- **Hypothesis**
  - “A GPU interrupt rate is in proportion to a display rate”
A GPU interrupt rate can be used to estimate a display rate without additional overheads

- Linear relationship between display rates and GPU interrupt rates

- Exponential weighted moving average (EWMA) is used to reduce fluctuation
  - \( \text{EWMA}_t = (1-w) \times \text{EWMA}_{t-1} + w \times \text{current value} \)
Multimedia Manager

• A feedback-driven CPU allocator
  • Base assumption
    • “Additional CPU share (or higher priority) improves a display rate”
  • Desired frame rate (DFR)
    • A currently achievable display rate
      • Multiplied by tolerable ratio (0.8)

IF current FPS < previous FPS AND current FPS < DFR THEN
  Increase CPU share

/* Exceptional cases:
* 1) No relationship between CPU and FPS
* 2) FPS is saturated below DFR
* 3) Local CPU contention in a VM
*/
If no FPS improvement by CPU share increase (3 times) Then
  Increase CPU share
Decrease CPU share by half

If in initial phase Then
  Exponential increase
Else
  Linear increase
Priority Boosting

• Responsive dispatching
  • Problem
    • The hypervisor does not distinguish the types of events for priority boosting
      • A VM that will handle a multimedia event cannot preempt a currently running VM handling a normal event.
  • Higher priority for multimedia-related events
    • e.g., video, audio, one-shot timer

<table>
<thead>
<tr>
<th>Priority</th>
<th>MMBOOST</th>
<th>IOBOOST</th>
<th>Normal priority</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Multimedia events</td>
<td>Other events</td>
<td>Based on remaining CPU shares</td>
</tr>
</tbody>
</table>
Evaluation

• Experimental environment
  • Intel MacBook with Intel GMA 950
  • Xen 3.4.0 with Ubuntu 8.04
    • Implementation based on Xen Credit scheduler

• Two-VM scenario
  • One with direct I/O + one with indirect (hosted) I/O
  • Presenting the case of direct I/O in this talk
    • See the paper for the details of the indirect I/O case
Estimation Accuracy

- Estimation accuracy
  - Error rates: 0.55%~3.05%

Video playback (720p) (w/ CPU-bound VM)  

Quake 3 (w/ CPU-bound VM)
Estimation Overhead

- CPU overhead caused by page faults
  - Video playback
    - 0.3~1% with sampling
    - Less than 5% with tracking all pages

<table>
<thead>
<tr>
<th>Overhead</th>
<th>All pages</th>
<th>Sampling</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1/8 pages</td>
<td>1/32 pages</td>
</tr>
<tr>
<td>Low resolution</td>
<td>4.95%</td>
<td>1.10%</td>
</tr>
<tr>
<td>(640x354)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>High resolution</td>
<td>3.91%</td>
<td>1.04%</td>
</tr>
<tr>
<td>(1280x720)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Low resolution (640x354):
- 4.95% overall
- 1.10% sampling
- 0.54% 1/8 pages
- 0.58% 1/32 pages
- 0.58% 1/128 pages

High resolution (1280x720):
- 3.91% overall
- 1.04% sampling
- 0.69% 1/8 pages
- 0.33% 1/32 pages
- 0.33% 1/128 pages
Multimedia Manager

- Video playback (720p) + CPU-bound VM
Performance Improvement

- Performance improvement
- Closed to maximum achievable frame rates

Video playback (720p) on VLC media player
- Credit scheduler
- Credit scheduler w/ multimedia support

Competing workloads in another VM
- Video playback or 3D game
- Competing workloads

Quake III Arena (demo1)
- Credit scheduler
- Credit scheduler w/ multimedia support

Competing workloads in another VM
Limitations & Discussion

- Network-streamed multimedia
  - Additional preemption support required for multimedia-related network packets
- Multiple multimedia workloads in a VM
  - Multimedia manager algorithm should be refined to satisfy QoS of mixed multimedia workloads in the same VM
- Adaptive management for SMP VMs
  - Adaptive vCPU allocation based on hosted multimedia workloads
Conclusions

• Demands for multimedia-aware hypervisor
  • Multimedia are increasingly dominant in virtualized systems
• “Multimedia-friendly hypervisor scheduler”
  • Transparent and lightweight multimedia support on client-side virtualization

• Future directions
  • Multimedia for server-side VDI
  • Multicore extension for SMP VMs
  • Considerations for network-streamed multimedia
Thank You!

• Questions and comments

• Contact
  • hjukim@calab.kaist.ac.kr
EXTRA SLIDES
### Workload Details

- Descriptions and CPU usages for used workloads

<table>
<thead>
<tr>
<th>Workloads</th>
<th>Description</th>
<th>CPU usage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Direct I/O</td>
</tr>
<tr>
<td>Multimedia</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VLC-LowRes</td>
<td>640x354 video playback <em>(vlc)</em></td>
<td>17</td>
</tr>
<tr>
<td>VLC-HighRes</td>
<td>1280x720 video playback <em>(vlc)</em></td>
<td>62</td>
</tr>
<tr>
<td>Quake3</td>
<td>Quake III arena demo play</td>
<td>72</td>
</tr>
<tr>
<td>Competitor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Idle</td>
<td>No workload</td>
<td>0</td>
</tr>
<tr>
<td>CPU-bound</td>
<td>CPU-hungry loop</td>
<td>100</td>
</tr>
<tr>
<td>Net-bound</td>
<td>TCP streaming benchmark <em>(iperf)</em></td>
<td>11</td>
</tr>
<tr>
<td>NetCPU</td>
<td>Net-bound with CPU-bound</td>
<td>100</td>
</tr>
<tr>
<td>Kbuild</td>
<td>Build Linux 2.6.34 kernel <em>(make)</em></td>
<td>94</td>
</tr>
<tr>
<td>Encoding</td>
<td>Encode a video <em>(mencoder)</em></td>
<td>100</td>
</tr>
<tr>
<td>Download</td>
<td>Download a file from Web <em>(wget)</em></td>
<td>41</td>
</tr>
</tbody>
</table>
Task-based Display Rate Management

- Different types of frame rate accounting

For Unix-like OSes, a display rate is accounted to a previously scheduled task when a framebuffer write is observed (this heuristic works well because a display interface is interactive and is highly likely to be scheduled promptly on requests)

Video playback (640x354) + Directory listing on the screen (ls –alR /) (w/ CPU-bound VM)

Error rate = 1.78%
Local CPU Contention in a VM

- Increasing the number of CPU-bound tasks with a video playback application

Our scheme provides sufficient CPU to a VM (IDD) that hosts a multimedia workload even though local CPU contention increases

- No starvation of another CPU-bound VM (guest)