VSched: Mixing Batch And Interactive Virtual Machines Using Periodic Real-time Scheduling

Bin Lin
Peter A. Dinda
Prescience Lab
Department of Electrical Engineering and Computer Science
Northwestern University
http://www.presciencelab.org
Overview

- Periodic real-time model for scheduling diverse workloads onto hosts
  - Virtual machines in our case
- Periodic real-time scheduler for Linux
  - VSched – publicly available
  - Works with any process
  - We use it with type-II VMs
- Promising evaluation for many workloads
  - Interactive, batch, batch parallel
Outline

• Scheduling virtual machines on a host
  • Virtuoso system
  • Challenges

• Periodic real-time scheduling

• VSched, our scheduler

• Evaluating our scheduler
  • Performance limits
  • Suitability for different workloads

• Conclusions and future work
  • Putting the user in direct control of scheduling
Virtuoso: VM-based Distributed Computing

User
Orders a raw machine

<table>
<thead>
<tr>
<th>Storage Price /month</th>
<th>37.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running Price /hour</td>
<td>0.3096</td>
</tr>
</tbody>
</table>
User’s View in Virtuoso Model

A VM is a replacement for a physical computer

Multiple VMs may run simultaneously on the same host
Challenges in Scheduling Multiple VMs Simultaneously on a Host

• VM execution *priced* according to interactivity and compute rate constraints
  – How to *express*?
  – How to *coordinate*?
  – How to *enforce*?

• Workload-diversity
  – Scheduling must be general
Our Driving Workloads

• Interactive workloads
  – substitute a remote VM for a desktop computer.
  – desktop applications, web applications and games

• Batch workloads
  – scientific simulations, analysis codes

• Batch parallel workloads
  – scientific simulations, analysis codes that can be scaled by adding more VMs

• Goals
  – interactivity does not suffer
  – batch machines meet both their advance reservation deadlines and gang scheduling constraints.
Scheduling Interactive VMs is Hard

- Constraints are highly user dependent
- Constraints are highly application dependent
- Users are very sensitive to jitter

- Conclusions based on extensive user studies
  - User comfort with resource borrowing [HPDC 2004]
  - User-driven scheduling [Grid 2004, in submission papers]
Batch Workloads

- Notion of compute rate
- Application progress proportional to compute rate
- Ability to know when job will be done
Batch Parallel Workloads

- Notion of compute rate
- Application progress proportional to compute rate
- Ability to know when job will be done
- Coordination among multiple hosts
  - Effect of gang scheduling
Outline

• Scheduling virtual machines on a host
  • Virtuoso system
  • Challenges

• Periodic real-time scheduling

• VSched, our scheduler

• Evaluating our scheduler
  • Performance limits
  • Suitability for different workloads

• Conclusions and future work
  • Putting the user in direct control of scheduling
Periodic Real-time Scheduling Model

- Task runs for **slice** seconds every **period** seconds
  - “1 hour every 10 hours”, “1 ms every 10 ms”
    - Does NOT imply “1 hour chunk” (but does not preclude it)
  - **Compute rate**: $\frac{\text{slice}}{\text{period}}$
    - 10 % for both examples, but radically different interactivity!
  - **Completion time**: $\frac{\text{size}}{\text{rate}}$
    - 24 hour job completes after 240 hours

- Unifying abstraction for diverse workloads
  - We schedule a VM as a single task
  - VM’s (slice, period) enforced
EDF Online Scheduling

- Dynamic priority preemptive scheduler
- Always runs task with highest priority
- Tasks prioritized in reverse order of impending deadlines
  - Deadline is end of current period

EDF="Earliest Deadline First"
EDF Admission Control

• If we schedule by EDF, will all the (slice, period) constraints of all the VMs always be met?

• EDF Schedulability test is simple
  – Linear in number of VMs

\[ U(n) = \sum_{k=1}^{n} \frac{\text{slice}_k}{\text{period}_k} \leq 1 \implies \text{Schedulable} \]
A detailed VSc hed schedule for three VMs

(VM1(50, 20) VM2(100, 10) VM3(1000, 300))
Outline

• Scheduling virtual machines on a host
  • Virtuoso system
  • Challenges

• Periodic real-time scheduling

• VSched, our scheduler

• Evaluating our scheduler
  • Performance limits
  • Suitability for different workloads

• Conclusions and future work
  • Putting the user in direct control of scheduling
Our implementation - VSched

- Provides soft real-time (limited by Linux)
- Runs at user-level (no kernel changes)
- Schedules any set of processes
  - We use it to schedule type-II VMMs
- Supports very fast changes in constraints
  - We know immediately whether performance improvement is possible or if VM needs to migrate
Our implementation – VSched

• Supports (slice, period) ranging into days
  – Fine millisecond and sub-millisecond ranges for interactive VMs
  – Coarser constraints for batch VMs
• Client/Server: remote control scheduling
  – Coordination with Virtuoso front-end
  – Coordination with other VScheds
• Publicly released
  [http://virtuoso.cs.northwestern.edu](http://virtuoso.cs.northwestern.edu)
Exploiting SCHED_FIFO

• Linux feature for simple preemptive scheduling without time slicing

• FIFO queue of processes for each priority level
• Runs first runnable process in highest priority queue
• VSched uses the three highest priority levels
VSched structure

• Client
  – Securely manipulate Server over TCP/SSL
  – Remote control

• Server module
  – EDF admission control
  – Remote control

• Scheduling Core
  – Online EDF scheduler manipulates SCHED_FIFO priorities

• Kernel
  – Implements SCHED_FIFO scheduling
Outline

• Scheduling virtual machines on a host
  • Virtuoso system
  • Challenges

• Periodic real-time scheduling

• VSched, our scheduler

• Evaluating our scheduler
  • Performance limits
  • Suitability for different workloads

• Conclusions and future work
  • Putting the user in direct control of scheduling
Basic Metrics

• **miss rate**
  – Missed deadlines / total deadlines

• **miss time**
  – Time by which deadline is missed when it is missed
  – We care about its distribution

• How do these depend on (period, slice) and number of VMs?
Reasons For Missing Deadlines

• Resolution misses: The period or slice is too small for the available timer and VScheduler overhead to support.

• Utilization misses: The utilization needed is too high (but less than 1).
Performance Limits

• Resolution
  – How small can period and slice be before miss rate is excessive?

• Utilization limit
  – How close can we come to 100% utilization of CPU?
Deterministic study

- Deterministic sweep over period and slice for a single VM
- Determines maximum possible utilization and resolution
  - Safe region of operation for VSched

- We look at lowest resolution scenario here
Near-optimal Utilization

**Contour of (Period, Slice, Miss Rate)**

Impossible Region:
utilization exceeds 100%

Extremely narrow range where feasible, near 100% utilizations cannot be achieved

≈0% Miss rate Possible and Achieved

2 GHz P4 running a 2.4 kernel (10 ms timer)
Performance Limits on Three Platforms

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Maximum Utilization</th>
<th>Minimum Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine 1</td>
<td>0.90</td>
<td>10 ms</td>
</tr>
<tr>
<td>Machine 2</td>
<td>0.75</td>
<td>0.2 ms</td>
</tr>
<tr>
<td>Machine 3</td>
<td>0.98</td>
<td>1 ms</td>
</tr>
</tbody>
</table>

- **Machine 1**: P4, 2GHz, Linux 2.4.20 (RH Linux 9) (**10 ms timer**).
- **Machine 2**: PIII, 1GHZ, Linux 2.4.18 patched with KURT 2.4.18-2 (**~10 us timer**).
- **Machine 3**: P4, 2GHz, Linux 2.6.8 (RH Linux 9) (**1 ms timer**).

- **Beyond these limits, miss rates are close to 100%**
- **Within these limits, miss rates are close to 0%**
Miss Times Small When Limits Exceeded

(Period:16, Slice:15.8), Unit: millisecond

Request 98.75% utilization; too high!

< 2.5 % of slice
Randomized Study

• Testcase consists of
  – A random number of VMs
  – Each with a feasible, different, randomly chosen \((period, slice)\) constraint

• We plot each testcase as a point in the following
Average Miss Rates Very Low and Largely Independent of Utilization and Number of VMs

Example: random testcases with 3 VMs

~1% Miss Rate For All Utilizations
Miss Rates Grow At Very High Utilization

Example: random testcases with 3 VMs
Miss Time is Very Small When Misses Do Occur
Independence from number of VMs

- Miss rates are largely independent of the number of VMs after two VMs
  - more frequent context switches from one to two VMs
- Miss time is very small and independent of the number of VMs
User Study of Mixing Batch and Interactive VMs

• Each user ran an interactive VM simultaneously with a batch VM
  – P4 2GHz, 512MB Mem, Linux 2.6.3, VMWare GSX 3.1
  – Interactive VM: WinXP Pro VM
  – Batch VM: RH 7.3 VM with cycle soaker
Activities in Interactive VM

- Listening to MP3 (Microsoft Media Player)
- Watching MPEG (Microsoft Media Player)
- Playing 3D First Person Shooter Game (QUAKE II)
- Browsing web (Internet Explorer)
  - using multiple windows, Flash Player content, saving pages, and performing fine-grain view scrolling.
Setup

- Batch VM: (1 minute, 10 minutes) (10%)
- Varied period and slice of interactive VM
- For each activity, user qualitatively assessed effect of different combinations of (period, slice) to find minimum acceptable combination
### Impressive Worst Case Results

<table>
<thead>
<tr>
<th>(period, slice)(ms)</th>
<th>Quake (with sound)</th>
<th>MP3 playback</th>
<th>MPEG (with sound) playback</th>
<th>Web Browsing</th>
</tr>
</thead>
<tbody>
<tr>
<td>5, 1</td>
<td>good</td>
<td>good</td>
<td>tiny audio jitter</td>
<td>good</td>
</tr>
<tr>
<td>6, 1</td>
<td>good</td>
<td>good</td>
<td>tiny audio jitter</td>
<td>good</td>
</tr>
<tr>
<td>7, 1</td>
<td>tiny jitter</td>
<td>good</td>
<td>tiny audio jitter</td>
<td>good</td>
</tr>
<tr>
<td>8, 1 (10-15% Utilization)</td>
<td>small jitter</td>
<td>tiny jitter</td>
<td>tiny jitter</td>
<td>good</td>
</tr>
<tr>
<td>9, 1</td>
<td>jitter</td>
<td>noisy</td>
<td>tiny jitter</td>
<td>jitter</td>
</tr>
<tr>
<td>10, 1</td>
<td>jitter</td>
<td>noisy</td>
<td>jitter</td>
<td>jitter</td>
</tr>
</tbody>
</table>

- Most sensitive user can still tolerate applications at very low utilization
- Can clearly run a mix of interactive and batch VMs on the same machine, keeping users of both happy
- Considerable headroom for interactive VMs
Scheduling Batch Parallel Applications

• Can we linearly control the execution rate of a parallel application running on VMs mapped to different hosts in proportion to the cycles we give it? **YES**

• Can we protect such an application from external load? **YES**

• *BSP benchmark; all-to-all communication; 4 cluster nodes; compute/communicate ratio = 0.5; MFLOP/s as our metric*
Existence of \((period, slice)\) constraint that achieves desired utilization while resulting in only a corresponding decrease in execution rate

MFLOP/s varies in direct proportion to utilization given the right \((period, slice)\) constraints

Our target line

Inappropriate \((period, slice)\) combinations
VSched Makes Parallel Application Performance Impervious to External Load Imbalance

Contestion: average number of competing processes that are runnable
Conclusions

- Proposed periodic real-time model for VM-based distributed computing
- Designed, implemented and evaluated a user-level scheduler (VSched)
- Mixed batch computations with interactive applications with no reduction in usability
- Applied VSched to schedule parallel applications
Future work

• Automating choosing schedules straightforwardly for all kinds of VMs
• Automating coordination of schedules across multiple machines for parallel applications
• Incorporate direct human input into the scheduling process
  – Forthcoming papers
Letting the Naïve User Choose Period and Slice

• Goal: Non-intrusive interface
  – Used only when user is unhappy with performance
  – Instantly manipulated to change the schedule

• Preview of further results
  – GUI (showing cost)
  – Non-centering joystick
• For More Information
  – Prescience Lab (Northwestern University)
    • http://www.presciencelab.org
    • http://virtuoso.cs.northwestern.edu
• VSched is publicly available from
  • http://virtuoso.cs.northwestern.edu