We are developing a system, Virtuoso, for distributed computing using virtual machines (VMs). Virtuoso must be able to mix batch and interactive VMs on the same physical hardware, satisfying constraints on responsiveness and compute rates for both kinds of workloads. VSched is the component of Virtuoso that provides this capability. VSched is a user-level tool that interacts with the stock Linux kernel running below any type-II virtual machine monitor to schedule all VMs (indeed, any process) using a periodic real-time scheduling model. This abstraction allows compute rate and responsiveness constraints to be straightforwardly described using a period and a slice within the period, and it allows for fast and simple admission control. This paper makes the case for periodic real-time scheduling for VM-based computing environments, and then describes and evaluates VSched. VSched can be downloaded from http://virtuoso.cs.northwestern.edu.

### 1 Evaluation

Our evaluation focuses on the resolution limits of VSched running on various different platforms: what combinations of period and slice lead to low deadline miss rates and what happens when the limits are exceeded. In the next section, we discuss how well the achievable combinations support interactive applications.

We run our evaluation for three different environments, as shown in Figure 1. The key differences between these environments are the processor speed (1 GHz P3 versus 2 GHz P4) and the timers available (2.4 kernel, 2.4 with KURT, and 2.6 kernel).

### 1.1 Methodology

For our detailed evaluation, we use the following methodology. Our primary metric is the miss rate, the number of times we miss the deadlines of a task divided by the total number of deadlines. For tasks that miss their deadlines, we also collect the miss time, the time by which the deadline was overrun. We want to understand how the miss rate varies with period and slice (or, equivalently, period and utilization), and what by how much we typically miss a deadline when this happens.

We run VSched with 1 test task. For each period, we have two deadlines to meet. The first one is at the point when the task exhausts its slice and should be suspended by VSched. The second is the point at which the current period ends and VSched should resume running the task for the next period. Between these two points in time, the task is suspended and VSched is asleep. Note that if we meet the first deadline, we also meet the second, so we conservatively measure whether we meet the first deadline. Conceptually, if we missed the first deadline and met the second, it would mean that VSched has allowed the VM to run for longer than was strictly necessary given its (period, slice) specification.

### 1.2 Miss rate versus \((\text{period, slice})\)

By studying this relationship, we seek to answer the following questions:

- What is the maximum utilization that VSched can schedule?
- What is the maximum resolution that VSched can achieve?
- What is the safe region of operation for VSched on different platforms?

Figure 2 shows the different configurations on which we evaluated VSched.
Machine 1: Pentium 4, 2.00GHz, 512MB Mem, Linux version 2.4.20-31.9 (Red Hat Linux 9.0)

Machine 2: Dual CPUs (Pentium III Coppermine, 1.0 GHZ), 1G Mem, non-SMP Linux kernel 2.4.18 patched with KURT 2.4.18-2

Machine 3: Pentium 4, 2.00GHz, 512MB Mem, Linux version 2.6.8.1 (Red Hat Linux 9.0)

Figure 1. Testbed Machines

<table>
<thead>
<tr>
<th>Kernel version</th>
<th>Machine (from table 1)</th>
<th>Utilization</th>
<th>Period</th>
<th>Slice</th>
<th>Pass (no. of periods)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux kernel 2.4.20-31.9</td>
<td>1</td>
<td>10% - 99% (increased by 10%)</td>
<td>1016 ms - 16 ms</td>
<td>105.8 ms - 1.6 ms</td>
<td>1000</td>
</tr>
<tr>
<td>KURT 2.4.18-2</td>
<td>2</td>
<td>1% - 99% (increased by 1%)</td>
<td>10.1 ms - 1.1 ms</td>
<td>9.999 ms - 0.011 ms</td>
<td>2000</td>
</tr>
<tr>
<td>Linux kernel 2.6.8.1</td>
<td>3</td>
<td>1% - 99% (increased by 1%)</td>
<td>101 ms - 1 ms</td>
<td>99.99 ms - 0.01 ms</td>
<td>2000</td>
</tr>
</tbody>
</table>

Figure 2. Testing setup

Figure 3 shows the miss rate as a function of period and slice for Machine 1, a 1 GHz P3 running a stock 2.4 kernel. The top graph is a 3D representation of this function, while the bottom graph is a countour map of the function. Considering the bottom graph, all combinations in the upper left, above the contour lines, reflect the 100% miss rate that is a necessary outcome when utilization exceeds 100%. The combinations in the lower right reflect nearly 0% miss rates. Clearly, utilizations to within a few percent of 100% are possible with nearly 0% miss rate.

Figure 4 shows a similar presentation of data for Machine 2, the 1 GHz P3 running a 2.4 kernel augmented with the KURT timers. Here, we focus on the highest resolutions, down to about 100 µs. An explanation of the two furrows that can be seen on the contour plot is needed. For a given period, as utilization increases, an abrupt wall is reached when the slice comes to within about 2 ms (less for larger periods) of the period. The miss rate reaches 100% at this point. As the slice continues to be increased (but remains smaller than the period), we see a ditch in which the miss rate drops back down to about 0%. Then, as the slice approaches to within 200 µs of the period, we see the miss rate shoot back up to 100%. This “double dip” effect is quite mysterious and shows up only with the KURT timer.

Figure 5 shows the data for Machine 3, a 2 GHz P4 running a stock 2.6 kernel. Here, the behavior is similar to that of the 2.4 kernel without KURT, but with the ability to successfully schedule down to a 1 ms range and to achieve a bit higher utilization for any particular period.

1.3 Characterizing misses

We would hope that when deadlines are missed, that the time by which they are missed is small, or at least commensurate with the period and slice. This is what we
see. Misses tend to occur in one of two situations:

- Utilization misses: The utilization needed is too high (but less than 1).
- Resolution misses: The period or slice is too small for the available timer and VSched overhead to support.

Figure 6 illustrates utilization misses on Machine 1. Here, we are requesting a period of 16 ms (feasible) and a slice of 15.8 ms (feasible). This utilization of 98.75% is too high, however, for to be able to schedule it, VSched requires slightly less than 1.25% of the CPU. The figure shows a histogram of the miss times. Notice that the vast majority of misses miss by less than 260 μs, less than 2% of the period.

Figure 7 shows a similar histogram illustrating resolution misses on Machine 2. Here, we are requesting a period of 1.1 ms and a slice of 0.011 ms. The slice is slightly below what is feasible for this system; the combination of the KURT timers and VSched cannot react within 11 μs. We can see that the majority of the miss times are less than 0.06 milliseconds, or less than 5% of the period.

Figure 8 illustrates further resolution misses on Machine 2. Here, the period is 1 ms and the slice is 0.308 ms. Slices in the range of 0.3 ms are feasible, but the precision (0.001 ms) being requested here exceeds what is possible. Again, we see that the majority of the miss times are a tiny fraction of the period.
Figure 6. Distribution of miss times when utilization is exceeded for Machine 1 (2 GHz P4, 2.4 kernel).

Figure 7. Distribution of miss times when resolution is exceeded for Machine 2 (1 GHz P3, 2.4 kernel, KURT timers).

Figure 8. Distribution of miss times when resolution is exceeded for Machine 2 (1 GHz P3, 2.4 kernel, KURT timers).

Figure 9. Distribution of miss times when resolution is exceeded for Machine 3 (2 GHz P4, 2.6 kernel).

Figure 10. Summary of 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>

Figure 9 illustrates utilization misses on Machine 3. Here, the period is 11 ms (feasible) and the slice is 10.89 ms (feasible). The utilization (99%) exceeds what is possible. Again, we can see that the miss times are generally a small fraction of the period.

Figure 10 summarizes the utilization and resolution limits of VSched running on our different configurations. Beyond these limits, miss rates are close to 100%, while before these limits, miss rates are close to 0%.