Virtual-CPU Scheduling in the Quest Operating System

Matt Danish, Ye Li and Richard West
Contact: richwest@cs.bu.edu
Goals

• Develop system with improved predictability
• Integrated management of tasks & I/O events
• Enforce *temporal isolation* between threads
Approach

• Introduce “virtual CPUs” for scheduling
  – Resource containers for CPU usage
  – Have budgets (reservations) and replenishment periods

• Scheduling hierarchy
  – Threads mapped to VCPUs
  – VCPUs mapped to PCPUs
VCPUs in Quest

• Two classes
  – **Main** → for conventional tasks
  – **IO** → for IO event threads (e.g., ISRs)

• Scheduling policies
  – **Main** → sporadic server (SS)
  – **IO** → priority inheritance bandwidth-preserving server (PIBS)
SS Scheduling

• Model periodic tasks
  – Each SS has a pair \((C,T)\) s.t. A server is guaranteed no more than \(C\) CPU cycles every period of \(T\) cycles
    • Guarantee applied at *foreground* priority
    • Can exceed this utilization at *background* priority
  – Rate-Monotonic Scheduling theory applies
PIBS Scheduling

• IO VCPUs have utilization factor, $V_U$

• IO VCPUs inherit priorities of tasks (or Main VCPUs) associated with IO events
  – Currently, priorities are $f(T)$ for corresponding Main VCPU
  – IO VCPU budget is limited to:
    • $V_{T,\text{main}} \times V_U$ for period $V_{T,\text{main}}$
• IO VCPUs have *eligibility* times, when they can execute

• $V_e = V_e + \frac{C_{\text{actual}}}{V_U}$
• About 11,000 lines of kernel code
• About 175,000 lines including lwIP, drivers, regression tests
• SMP, IA32, paging, VCPU scheduling, USB, PCI, networking, etc
Experiments

- Intel Core2 Extreme QX6700 @ 2.66GHz
- 4GB RAM
- Gigabit Ethernet (Intel 8254x “e1000”)
- UHCI USB Host Controller
  - 1GB USB memory stick
- Parallel ATA CDROM in PIO mode

- Measurements over 5sec windows using bandwidth-preserving logging thread
Experiments

- CPU-bound threads: increment a counter
- CD ROM/USB threads: read 64KB data from filesystem on corresponding device
## I/O Effects on VCPUs

<table>
<thead>
<tr>
<th>VCPU</th>
<th>CPU-bound threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCPU0</td>
<td>2, 5, CPU-bound</td>
</tr>
<tr>
<td>VCPU1</td>
<td>2, 8, Reading CD, CPU-bound</td>
</tr>
<tr>
<td>VCPU2</td>
<td>1, 4, CPU-bound</td>
</tr>
<tr>
<td>VCPU3</td>
<td>1, 10, Logging, CPU-bound</td>
</tr>
<tr>
<td>IOVCPU</td>
<td>10%, ATA</td>
</tr>
</tbody>
</table>
I/O Effects on VCPUs

The diagram illustrates the CPU usage comparison between two scenarios:

- **Without CD-ROM I/O**
- **With CD-ROM I/O**

The CPU usage is divided into different sections for each VCPU:
- VCPU0
- VCPU1
- VCPU2
- VCPU3
- IOVCPU

The graph shows a significant increase in CPU usage when CD-ROM I/O is enabled.
<table>
<thead>
<tr>
<th>VCPU</th>
<th>$V_C$</th>
<th>$V_T$</th>
<th>threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCPU0</td>
<td>1</td>
<td>20</td>
<td>CPU-bound</td>
</tr>
<tr>
<td>VCPU1</td>
<td>1</td>
<td>30</td>
<td>CPU-bound</td>
</tr>
<tr>
<td>VCPU2</td>
<td>10</td>
<td>100</td>
<td>Network, CPU-bound</td>
</tr>
<tr>
<td>VCPU3</td>
<td>20</td>
<td>100</td>
<td>Logging, CPU-bound</td>
</tr>
<tr>
<td>IOVCPU</td>
<td>1%</td>
<td></td>
<td>Network</td>
</tr>
</tbody>
</table>
t=50 start ICMP ping flood. Here, we see comparison overheads of two scheduling policies
PIBS vs SS IO VCPU Scheduling

Network bandwidth of two scheduling policies
### IO VCPU Sharing

<table>
<thead>
<tr>
<th>VCPU</th>
<th>$V_c$</th>
<th>$V_t$</th>
<th>threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCPU0</td>
<td>30</td>
<td>100</td>
<td>USB, CPU-bound</td>
</tr>
<tr>
<td>VCPU1</td>
<td>10</td>
<td>110</td>
<td>CPU-bound</td>
</tr>
<tr>
<td>VCPU2</td>
<td>10</td>
<td>90</td>
<td>Network, CPU-bound</td>
</tr>
<tr>
<td>VCPU3</td>
<td>100</td>
<td>200</td>
<td>Logging, CPU-bound</td>
</tr>
<tr>
<td>IO VCPU</td>
<td>1%</td>
<td></td>
<td>USB, Network</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VCPU</th>
<th>$V_c$</th>
<th>$V_t$</th>
<th>threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCPU0</td>
<td>30</td>
<td>100</td>
<td>USB, CPU-bound</td>
</tr>
<tr>
<td>VCPU1</td>
<td>10</td>
<td>110</td>
<td>CPU-bound</td>
</tr>
<tr>
<td>VCPU2</td>
<td>10</td>
<td>90</td>
<td>Network, CPU-bound</td>
</tr>
<tr>
<td>VCPU3</td>
<td>100</td>
<td>200</td>
<td>Logging, CPU-bound</td>
</tr>
<tr>
<td>IO VCPU1</td>
<td>1%</td>
<td></td>
<td>USB</td>
</tr>
<tr>
<td>IO VCPU2</td>
<td>1%</td>
<td></td>
<td>Network</td>
</tr>
</tbody>
</table>
IO VCPU Sharing

![Graph showing comparison of USB, Network, and USB (pingflood) between Shared and Separate scenarios. The x-axis represents the two scenarios (Shared and Separate), and the y-axis represents the data rate in kilobytes per second (kB/s). The graph indicates a significant difference in data rates between the two scenarios, with the Separate scenario showing much higher data rates for USB and Network.]
Conclusions

• Temporal isolation on IO events and tasks
• PIBS + SS Main & IO VCPUs can guarantee utilization bounds
• Future investigation of higher-level policies
• Future investigation of h/w performance counters for VCPU-to-PCPU scheduling