A Comparative Study of the Realization of Rate-Based Computing Services in General Purpose Operating Systems

Kevin Jeffay
Department of Computer Science
University of North Carolina
at Chapel Hill
jeffay@cs.unc.edu

Gerardo Lamastra
ReTiS Lab
Scuola Superiore di Studi
Universitari e Perfezionamento
S. Anna, Pisa, Italy

http://www.cs.unc.edu/Research/Dirt/

Rate-Based Resource Allocation
A technology for real-time computing on the desktop

• The Problem:
  – How to provide integrated real-time computation and communication services in a time-shared operating system?

• The Solution:
  – Rate-based resource allocation
    » Proportional share
    » Server algorithms
    » “x out of y” algorithms

• Which solution works best?

Rate-Based Resource Allocation
Integrated real-time resource allocation example

• Data arrives for a video conference over the network
• It is processed by the operating system and delivered to the application
• The application further processes and sends to the window system
• The window system paints the screen

Device Driver Layer
Protocol input queue
Protocol Layer
Protocol input queue
Socket receive queues
Socket Layer
MPEG Play
X Server
User Space
Network interface card
Rate-Based Resource Allocation
Integrated real-time resource allocation example

- Technical challenges:
  - Device scheduling and protocol processing
  - Application and system call scheduling

- Candidate technologies
  - Proportional share scheduling (EEVDF)
  - Constant Bandwidth Servers (CBS)
  - Rate-Based extensions to Liu and Layland (RBE)

---

Rate-based resource allocation schemes
Proportional share resource allocation

- Processes are allocated a share of the processor’s capacity
  - Process $i$ is assigned a weight $w_i$
  - Process $i$’s share of the CPU at time $t$ is
    \[ f_i(t) = \frac{w_i}{\sum_{j \neq i} w_j} \]

- If processes’ weights remain constant in $[t_1, t_2]$ then process $i$ receives
  \[ S_i(t_1, t_2) = \int_{t_1}^{t_2} f_i(t) \, dt = \frac{w_i}{\sum_{j} w_j} (t_2 - t_1) \]

units of execution time in $[t_1, t_2]$

---

Rate-Based Resource Allocation
Integrated real-time resource allocation example

- Our study:
  - Compare the performance of applications of rate-based scheduling technology at various levels in the kernel
  - For various characterizations of real-time processing workloads
    - Well-behaved periodic job/task arrivals
    - Bursty job/task arrivals
    - “Misbehaved” job/task arrivals

---

Rate-based resource allocation schemes
Constant bandwidth server

- A server process executes tasks every $T_S$ time units
- When a request arrives it is serviced with a deadline of
  \[ d_k = \text{MAX}(t_k, d_{k-1}) + c_k / U_S \]

where
  - $t_k$ is the arrival time of the $k^{th}$ task request
  - $c_k$ is the execution cost of the $k^{th}$ task request
  - $U_S = C_S / T_S$ is the capacity (utilization) of the server
  - $d_{k-1}$ is the deadline of the previous task request
Rate-based resource allocation schemes

Rate-based Liu & Layland scheduling

- Processes make progress at the rate of processing \( x \) events every \( y \) time units and each event is processed within \( d \) time units

- For task \( i \) with rate specification \((x_i, y_i, d_i)\), the \( j^{th} \) event for task \( i \), arriving at time \( t_{ij} \), will be processed by time

\[
D(i,j) = \begin{cases} 
  t_{ij} + d_i & \text{if } 1 \leq j \leq x_i \\
  \max(t_{ij} + d_i, D(i,j-x_i)+y_i) & \text{if } j > x_i
\end{cases}
\]

- Deadlines occur at least \( d \) time units after a job is released
- Deadlines separated by at least \( y \) time units

---

Empirical comparisons

Experimental setup

- Modify FreeBSD UNIX to support rate-based scheduling in the “top” and “bottom” halves of the kernel
- Consider the performance of each rate-based scheme in isolation and in combinations
- Consider the performance across a variety of multimedia workloads

---

Experimental Setup

Workload generation

- Audio receiver (5% CPU utilization)
- M-JPEG receiver (45% CPU utilization)
- tftp receiver (20% CPU utilization)
- Dhrystone (100 - x% utilization)

---

Empirical Comparisons

Performance metrics setup

- Packets dropped at the IP layer
- Packets dropped at the socket layer
- Packets delivered to the application
- Dhrystone performance
- NIC to application response time
- Deadline miss percentage

---
**Empirical Comparisons**

Experimental plan

- First consider using only
  - Proportional share,
  - CBS, and
  - RBE
- scheduling for all resource allocation problems

- Then attempt to match algorithms to the specific allocation problems where they are best suited

---

**Experimental Results Summary**

Well-behaved, periodic packet arrivals

<table>
<thead>
<tr>
<th></th>
<th>Prop Share</th>
<th>CBS</th>
<th>RBE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phone</td>
<td>0 0 2,993</td>
<td>0 0</td>
<td>0 0 3,027</td>
</tr>
<tr>
<td>ftp</td>
<td>0 0 11,961</td>
<td>2 0 11,914</td>
<td>0 0 11,944</td>
</tr>
<tr>
<td>M-JPEG</td>
<td>0 0 5,346</td>
<td>0 0 5,388</td>
<td>0 0 5,443</td>
</tr>
</tbody>
</table>

In isolation, all rate-based schemes give “perfect” (or very good) performance
- No packets are dropped

- Liu & Layland rate-based scheduling (RBE) provides the best response times
  - (Not surprising)

---

Experimental Results Summary

Bursty (pareto) packet arrivals

<table>
<thead>
<tr>
<th></th>
<th>Prop Share</th>
<th>CBS</th>
<th>RBE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phone</td>
<td>1,585 0 1,312</td>
<td>0 0 2,938</td>
<td>0 0 3,027</td>
</tr>
<tr>
<td>ftp</td>
<td>5,315 0 5,408</td>
<td>5 0 10,760</td>
<td>0 0 10,778</td>
</tr>
<tr>
<td>M-JPEG</td>
<td>2,705 0 2,498</td>
<td>0 0 3,192</td>
<td>0 0 5,287</td>
</tr>
</tbody>
</table>

- Proportional share scheduling degrades the performance of all applications uniformly
  - A (bad) artifact of quantum-based allocation

- CBS and RBE smooth the arrival process
  - Event driven scheduling works well here
  - Pure event-driven scheduling (RBE) gives lowest response times

---

Experimental Results Summary

“Misbehaved” ftp packet arrivals

<table>
<thead>
<tr>
<th></th>
<th>Prop Share</th>
<th>CBS</th>
<th>RBE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phone</td>
<td>5 0 2,997</td>
<td>0 0 2,978</td>
<td>0 0 2,998</td>
</tr>
<tr>
<td>ftp</td>
<td>17,999 0 11,902</td>
<td>17,880 0 12,120</td>
<td>0 9,052 20,794</td>
</tr>
<tr>
<td>M-JPEG</td>
<td>56 0 5,390</td>
<td>0 0 5,391</td>
<td>0 0 5,444</td>
</tr>
</tbody>
</table>

- Proportional share and CBS provide excellent protection/isolation for well-behaved tasks
  - ftp packets dropped at the IP layer

- RBE scheduling drops ftp packets at the socket layer
  - Pure event-driven scheduling provides no isolation
  - Dhrystone performance suffers drastically
First-Round Experiments Summary

So what?

- When workload is well-behaved all schemes perform well
- Pure-event driven scheduling and quantum allocation don’t work well for “bottom-half” kernel processing
- Server-based allocation doesn’t work well for application-level processing

Combine the scheduling schemes to better match the processing requirements at each level in the system

Rate-Based Resource Allocation

Conclusions

- “One size does not fit all” (unless the external environment is (perfectly) well-behaved)
  - Quantum allocation within the kernel leads to coarse-grained control
  - Server-based allocation impractical for applications
  - Pure event scheduling doesn’t provide isolation
- Different scheduling algorithms work best at different levels of the kernel
  - Event scheduling best at the device layer
  - Server/quantum scheduling best at the application/system call layer

Combining Allocation Policies

Getting the best of all worlds

- CBS+Proportional Share scheduling

<table>
<thead>
<tr>
<th></th>
<th>Constant Rate</th>
<th>Bursty</th>
<th>Misbehaved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phone</td>
<td>0 0 2,869</td>
<td>0 0 2,998</td>
<td>0 0 2,797</td>
</tr>
<tr>
<td>ftp</td>
<td>0 0 11,722</td>
<td>0 0 10,340</td>
<td>17,898 0 11,545</td>
</tr>
<tr>
<td>M-JPEG</td>
<td>0 0 5,343</td>
<td>0 0 4,951</td>
<td>0 0 5,398</td>
</tr>
</tbody>
</table>

- RBE+Proportional Share scheduling

<table>
<thead>
<tr>
<th></th>
<th>Constant Rate</th>
<th>Bursty</th>
<th>Misbehaved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phone</td>
<td>0 0 2,873</td>
<td>0 0 2,954</td>
<td>0 0 2,789</td>
</tr>
<tr>
<td>ftp</td>
<td>0 0 11,802</td>
<td>0 0 10,437</td>
<td>17,872 0 11,647</td>
</tr>
<tr>
<td>M-JPEG</td>
<td>0 0 5,324</td>
<td>0 0 4,956</td>
<td>0 0 5,393</td>
</tr>
</tbody>
</table>

IP Drops  | Socket Drops  | Packets Delivered