A Workload-oriented Programming Model for Temporal Isolation with VBS

Ana Sokolova

Department of Computer Sciences
University of Salzburg

joint work with
Silviu Craciunas and Christoph Kirsch
It is about

- Scheduling processes in **temporal isolation**
  - the time it takes to execute a piece of code is bounded, independently of concurrently running code

- Using variable bandwidth servers for **predictability**

- Server design for **performance**
Process model

- action is a piece of code
- process is a sequence of actions
Problem

schedule the processes so that each of their actions maintains its response time
Problem

Solvable with variable bandwidth servers (VBS)

Results:
- a constant-time scheduling algorithm
- a constant-time admission test

schedule the processes so that each of their actions maintains its response time
VBS is determined by a bandwidth cap \( u \)

VBS processes dynamically adjust speed (resource)

\[
\frac{\lambda_1}{\pi_1} \leq u \quad \text{and} \quad \frac{\lambda_2}{\pi_2} \leq u
\]

Generalization of constant bandwidth servers (CBS)

[Abeni and Buttazzo 2004]
One process on a VBS

- Process running on a VBS
  - Load
  - Action $\alpha_1$
  - Response time
  - Load
  - Action $\alpha_2$
  - Response time

- Response time under VBS
  - $\pi_1$
  - $\lambda_1$
  - $\pi_2$
  - $\lambda_2$
VBS

process running on a VBS

arrival

response time under VBS

time

termination
multiple processes are EDF-scheduled
Scheduling result and bounds

Processes $P_1, P_2, \ldots, P_n$ on VBSs $u_1, u_2, \ldots, u_n$, are schedulable if $\sum u_i \leq 1$

For any action $\alpha$ on a resource $(\lambda, \pi)$ we have

- Upper response time bound: $\lceil \text{load} / \lambda \rceil \pi + \pi - 1$
- Lower response time bound: $\lfloor \text{load} / \lambda \rfloor \pi$
- Jitter: $\pi - 1$
Implementation

- constant-time scheduling algorithm
- different queue management plugins

<table>
<thead>
<tr>
<th></th>
<th>list</th>
<th>array</th>
<th>matrix/tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>$O(n^2)$</td>
<td>$O(\log(t) + n\log(t))$</td>
<td>$\Theta(t)$</td>
</tr>
<tr>
<td>space</td>
<td>$\Theta(n)$</td>
<td>$\Theta(t + n)$</td>
<td>$O(t^2 + n)$</td>
</tr>
</tbody>
</table>

$n$ - number of processes  
$t$ - number of time instants
## Implementation

- constant-time scheduling algorithm
- different queue management plugins

---

### Trade-off time and space

<table>
<thead>
<tr>
<th></th>
<th>list</th>
<th>array</th>
<th>matrix/tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>$O(n^2)$</td>
<td>$O(\log(t) + n \log(t))$</td>
<td>$\Theta(t)$</td>
</tr>
<tr>
<td>space</td>
<td>$\Theta(n)$</td>
<td>$\Theta(t+n)$</td>
<td>$O(t^2+n)$</td>
</tr>
</tbody>
</table>

$n$ - number of processes  
$t$ - number of time instants
Programming model

loop {
    int number_of_frames = determine_rate();

    allocate_memory(number_of_frames);
    read_from_network(number_of_frames);

    compress_data(number_of_frames);
    write_to_disk(number_of_frames);
}

} until (done);

loop period

different throughput and latency requirements
for different portions of code
How do we get VBS parameters for an action?

“server design problem”
Response-time function

Time (ms)

Memory allocation requests in number of frames

Bad

Good
Response-time function

\[ f_{\text{R}} : \mathbb{R}_D \rightarrow \mathbb{Q}_+ \]

Time (ms)

Memory allocation requests in number of frames

Allocation rate of 240 fps
Execution-time function

Time (ms)

Bad

load

$f_E : E_D \rightarrow Q_+$

$f_E$ is a parametric version of WCET

Memory allocation requests in number of frames

$f_R$
Utilization

\[ f_U(w) = \frac{f_E(w) - d_E}{f_R(w) - d_R} \]

\[ f_U(w) \leq c_U \]
Timing of the allocate_memory action

\[ f_R(w) = 4w + 4 \]

\[ f_E(w) = 0.4w + 0.2 \]

\[ c_U = 10\% \]
\[ d_R = 4 \text{ ms} \]
\[ d_E = 200 \mu \text{s} \]
Response-time sampling

Scheduled response time (under VBS)

\[ f_R(w) = 4w + 4 \]
\[ f_E(w) = 0.4w + 0.2 \]
Server Design Problem

Finding the right $\lambda, \pi$ is difficult.

For $f_S(w) \leq f_R(w)$ one can choose $\pi$ as follows:

- $0 < \pi < d_R - d_E / c_U$
- $\pi$ divides $d_R$ evenly
- $\pi$ divides $f_R(w) - d_R$ evenly or
  $\lambda$ divides $f_E(w) - d_E$ evenly

$\lambda = \pi \times c_U$
Server Design Problem

Smallest $\pi$ possible:
- $f_S$ approximates $f_R$ best 😞
- less response-time jitter 😊
- increased scheduler overhead 😞

Scheduler overhead accounting:
- utilization accounting 😊
- response-time accounting 😞
- combined accounting

Higher-level scheduler:
- small period for the first part of an action
- large period for the remaining part
Conclusion

For scheduling processes in temporal isolation:

- Programming model as a link to VBS
- VBS provide predictability
- Server design for better performance

http://tiptoe.cs.uni-salzburg.at/