Programmable Temporal Isolation for Real-Time Systems

Silviu Craciunas
Department of Computer Sciences
University of Salzburg

joint work with Christoph Kirsch and Ana Sokolova
Structure
VBS process model

varying real-time constraints while maintaining temporal isolation
Structure

VBS process model
  varying real-time constraints while maintaining temporal isolation

VBS scheduling result
  EDF-based scheduler, response time bounds, queue mechanism
Structure

VBS process model
  varying real-time constraints while maintaining temporal isolation

VBS scheduling result
  EDF-based scheduler, response time bounds, queue mechanism

Overhead accounting
  include scheduler overhead in the schedulability analysis
VBS process model
- varying real-time constraints while maintaining temporal isolation

VBS scheduling result
- EDF-based scheduler, response time bounds, queue mechanism

Overhead accounting
- include scheduler overhead in the schedulability analysis

Power-aware VBS
- reduce CPU power consumption
Motivation
Motivation

http://javiator.cs.uni-salzburg.at
When things go wrong...
Flying is difficult but fun
Real-time control loop

```python
loop {
    read_sensors();
    compute_actuators();
    write_actuators();

    update_state();
    log();
}
```
Real-time control loop

```c
loop {
    read_sensors();
    compute_actuators();
    write_actuators();
    update_state();
    log();
}
```

deadline

Wednesday, December 1, 2010
Real-time control loop

```
loop {
  read_sensors();
  compute_actuators();
  write_actuators();
  update_state();
  log();
}
```
Real-time control loop

```c
loop {
    read_sensors();
    compute_actuators();
    write_actuators();
    update_state();
    log();
}
```

- different portions of code have different timing requirements
- temporal behavior should not be affected by other processes (temporal isolation)

Wednesday, December 1, 2010
Process model
Process model

process  arrival  time
Process model

process

arrival

termination

time
Process model

- action is a piece of code
- action is a piece of code
Process model

- action $\alpha_1$

- process

- arrival

- load

- termination

- response time

- action is a piece of code
• action is a piece of code
• process is a sequence of actions
Process model

- action is a piece of code
- process is a sequence of actions
- throughput vs latency of process execution
Scheduling problem

process 1

process 2

... uniprocessor ...

process n

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

process 1

process 2

Solvable with variable-bandwidth servers (VBS)

process n

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

Schedule the processes so that each of their actions maintains its response time.

Solvable with variable-bandwidth servers (VBS)

Results:
- constant-time scheduling algorithm
- constant time admission test
### Virtual Periodic Resources

<table>
<thead>
<tr>
<th>Period $\pi$</th>
<th>Limit $\lambda$</th>
<th>Utilization $\frac{\lambda}{\pi}$</th>
</tr>
</thead>
</table>
Resources and VBS

virtual periodic resources

period $\pi$  limit $\lambda$  utilization $\frac{\lambda}{\pi}$

Wednesday, December 1, 2010
• VBS is determined by a bandwidth cap ($u$)
• VBS processes dynamically adjust speed (change resources)

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

• generalization of constant bandwidth servers (CBS)

[Abeni and Buttazzo 2004]
One process on a VBS

action $\alpha_1$

load

response time

process running on a VBS
One process on a VBS

Process running on a VBS

Time

Load

Action $\alpha_1$

Response time

$\pi_1$
One process on a VBS

process

load

action $\alpha_1$

response time

process running on a VBS

$\pi_1$
One process on a VBS

process running on a VBS

action $\alpha_1$

load

response time

$\lambda_1$

$\pi_1$
One process on a VBS
One process on a VBS

action $\alpha_1$

load

response time

load

response time

process running on a VBS

$\lambda_1$

$\pi_1$

$\lambda_2$

$\pi_2$
VBS

process running on a VBS

time
VBS

arrival

process running on a VBS

time

Wednesday, December 1, 2010
process running on a VBS
process running on a VBS

arrival

limit

release

VBS
VBS

process running on a VBS

arrival

limit

release

release

time
VBS

process running on a VBS

arrival

limit

preemption

release

release
VBS

process running on a VBS

arrival

limit

preemption

limit

release

release

Wednesday, December 1, 2010
VBS

process running on a VBS

arrival

preemption

release

release

release

time

Wednesday, December 1, 2010
VBS

process running on a VBS

arrival

limit

preemption

limit

limit

completion

time

release

release

release

termination
VBS

A process running on a VBS has events such as arrival, limit, preemption, and completion. The response time under VBS is shown through these events over time.
VBS

multiple processes are EDF-scheduled
multiple processes are EDF-scheduled
multiple processes are EDF-scheduled
multiple processes are EDF-scheduled
multiple processes are EDF-scheduled
multiple processes are EDF-scheduled
Processes $P_1, P_2, \ldots, P_n$ on VBSs $u_1, u_2, \ldots, u_n$ are schedulable if $\sum_{i=1}^{n} u_i \leq 1$
Processes $P_1, P_2, \ldots, P_n$ on VBSs $u_1, u_2, \ldots, u_n$ are schedulable if $\sum_{i=1}^{n} u_i \leq 1$

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

- upper response-time bound $\lceil \frac{\text{load}}{\lambda} \rceil \pi + \pi - 1$
- lower response-time bound $\left\lfloor \frac{\text{load}}{\lambda} \right\rfloor \pi$
- jitter $\pi - 1$
Scheduling result and bounds

Processes $P_1, P_2, \ldots, P_n$ on VBSs $u_1, u_2, \ldots, u_n$ are schedulable if $\sum_{i=1}^{n} u_i \leq 1$

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

- upper response-time bound $\left\lceil \frac{load}{\lambda} \right\rceil \pi + \pi - 1$
- lower response-time bound $\left\lceil \frac{load}{\lambda} \right\rceil \pi$
- jitter $\pi - 1$

temporal isolation
Programmable temporal isolation

the “speed“ of an action is programmable
  (influencing response time and jitter)
smaller $\pi \Rightarrow$

  + smaller jitter
  + VBS response time closer to „ideal“ response time
- higher administrative overhead
  (more scheduler invocations)

Finding the right $\lambda, \pi$ is difficult (server design problem).
Scheduler overhead
Bare-metal experiment

![Graph showing response time and CPU utilization vs. number of processes.](image-url)
Bare-metal experiment

![Graph showing response-time jitter and CPU utilization with respect to the number of processes.]

- **Y-axis:** response-time jitter (ms)
- **X-axis:** number of processes
- **Legend:**
  - Actual response-time jitter
  - Theoretical bound on response-time jitter
  - CPU utilization (dotted line)

---

*Wednesday, December 1, 2010*
Overhead accounting

process 1
(1,4)

process 2
(1,6)
Overhead accounting

process 1
(1,4)

process 2
(1,6)

\[ N = \left\lceil \frac{\pi}{\gcd(\text{all periods})} \right\rceil + 1 \]
Overhead accounting

utilization

response time
Overhead accounting

utilization

normal VBS action execution

response time
Overhead accounting

utilization

normal VBS action execution

response accounting

response time
Overhead accounting

utilization

response time

utilization accounting

response accounting

normal VBS action execution
Overhead accounting

utilization

utilization accounting

combined response time and utilization accounting

response accounting

response time
Overhead accounting

- Utilization accounting
- Combined response time and utilization accounting
- Response accounting

Sufficient schedulability tests and response time bounds in all cases
Without overhead

response time

utilization

normal VBS action execution
Without overhead

utilization

normal VBS
action execution

response time

test: \[ \sum_{i} u_i \leq 1 \]

bounds:
\[ \left\lfloor \frac{\text{load}}{\lambda} \right\rfloor \pi \leq \text{RT} \leq \left\lceil \frac{\text{load}}{\lambda} \right\rceil \pi + \pi - 1 \]
Response accounting

utilization

normal VBS action execution

response accounting

response time
Response accounting

utilization

response time

normal VBS action execution response accounting

test: $\sum_{i} u_i \leq 1$

bounds:

$\left\lfloor \frac{\text{load}^*}{\lambda} \right\rfloor \pi \leq \text{RT} \leq \left\lceil \frac{\text{load}^*}{\lambda} \right\rceil \pi + \pi - 1$

$\text{load} + \left\lceil \frac{\text{load}}{\lambda - \delta} \right\rceil \delta$
Utilization accounting
Utilization accounting

Utilization accounting

Utilization accounting

normal VBS

action execution

response accounting

response time

test: \[ \sum_i \max_j \frac{\lambda_{i,j} + \delta_{i,j}}{\pi_{i,j}} \leq 1 \]

bounds:

\[
\left\lfloor \frac{\text{load}}{\lambda} \right\rfloor \pi \leq RT \leq \left\lfloor \frac{\text{load}}{\lambda} \right\rfloor \pi + \pi - 1
\]
Utilization accounting

- Utilization accounting
- Combined response time and utilization accounting
- Response accounting

utilization

response time

Wednesday, December 1, 2010
Utilization accounting

Utilization

Combined response time and utilization accounting

Response time

Test:
\[ \sum_{i} \max_{j} \frac{\lambda_{i,j} + \delta_{i,j}^{u}}{\pi_{i,j}} \leq 1 \]

Bounds:
\[ \left\lfloor \frac{\text{load}^{*}}{\lambda} \right\rfloor \pi \leq RT \leq \left\lceil \frac{\text{load}^{*}}{\lambda} \right\rfloor \pi + \pi - 1 \]

\[ \text{load}' + \left\lfloor \frac{\text{load}'}{\lambda} \right\rfloor \delta^{u} \]
\[ \text{load} + \left\lfloor \frac{\text{load}}{\lambda - \delta^{b}} \right\rfloor \delta^{b} \]

\[ \delta = \delta^{u} + \delta^{b} \]
Overhead accounting
Optimization of the estimate

Scheduling invocations due to release can be considered as a separate process

process 1
(1,4)

process 2
(1,6)

scheduler process

Wednesday, December 1, 2010
Experiments
Power-Aware VBS

Dynamic Voltage and Frequency Scaling
Power-Aware VBS

Dynamic Voltage and Frequency Scaling

Maintain VBS properties (temporal isolation, bounds)
Power-Aware VBS

Dynamic Voltage and Frequency Scaling

Maintain VBS properties (temporal isolation, bounds)

process 1
(1,4)

process 2
(1,4)
Power-Aware VBS

Dynamic Voltage and Frequency Scaling

Maintain VBS properties (temporal isolation, bounds)

- **Process 1**: (2,4)
  - Time intervals: 0-4, 4-8, 8-12

- **Process 2**: (2,4)
  - Time intervals: 0-4, 4-8, 8-12
Power-Aware VBS

Dynamic Voltage and Frequency Scaling

Maintain VBS properties (temporal isolation, bounds)

Possible whenever there is slack in the system
Power-Aware VBS

EDF frequency scaling result:

An EDF-schedulable set of tasks is still schedulable if the processor frequency in between any two release times is set to at least

\[ U_c \cdot f_{\max} \]

current total utilization of all released tasks in the considered interval of time between two releases

process 1
(2,4)

0 4 8 12

0.5 \( f_{\max} \)
Frequency-scaling VBS

Slack
Frequency-scaling VBS

Slack

Static Slack

Dynamic Slack

Action Slack

Termination Slack
Frequency-scaling VBS

Frequency is scaled to the sum of the bandwidth caps and not changed at runtime.
Frequency-scaling VBS

- **Static Slack**: Frequency is scaled to the sum of the bandwidth caps and not changed at runtime.
- **Action Slack**: Frequency is scaled at release time to the sum of the utilizations of the released actions.
- **Dynamic Slack**: Frequency is scaled at release time to the sum of the utilizations of the released actions.
- **Termination Slack**: Frequency is scaled to the sum of the bandwidth caps and not changed at runtime.
Frequency-scaling VBS

**Static Slack**

Frequency is scaled to the sum of the bandwidth caps and not changed at runtime.

**Dynamic Slack**

Frequency is scaled at release time to the sum of the utilizations of the released actions.

**Action Slack**

New limits are computed for each action such that the upper response-time bound is maintained.

**Termination Slack**

Frequency is scaled to the sum of the bandwidth caps and not changed at runtime.
Frequency-scaling VBS
Frequency-scaling VBS

Static slack

\[ f = \sum_{i=1}^{n} u_i \cdot f_{\text{max}} \]
Frequency-scaling VBS

Static slack

\[ f = \sum_{i=1}^{n} u_i \cdot f_{max} \]

Action slack

\[ f = \sum_{i=1}^{n} \frac{\lambda_{i,j}}{\pi_{i,j}} \cdot f_{max} \]
Frequency-scaling VBS

Static slack

\[ f = \sum_{i=1}^{n} u_i \cdot f_{max} \]

Action slack

\[ f = \sum_{i=1}^{n} \frac{\lambda_{i,j}}{\pi_{i,j}} \cdot f_{max} \]

Termination slack

\[ f = \sum_{i=1}^{n} \frac{\lambda_{i,j}^*}{\pi_{i,j}} \cdot f_{max} \]

\[ \lambda_{i,j}^* = \left[ \frac{l_{i,j}}{n_{i,j}} \right] \]

\[ n_{i,j} = \left[ \frac{l_{i,j}}{\lambda_{i,j}} \right] \]
Frequency-scaling VBS

Static slack

\[ f = \sum_{i=1}^{n} u_i \cdot f_{max} \]

Action slack

\[ f = \sum_{i=1}^{n} \frac{\lambda_{i,j}}{\pi_{i,j}} \cdot f_{max} \]

Termination slack

\[ f = \sum_{i=1}^{n} \frac{\lambda_{i,j}^*}{\pi_{i,j}} \cdot f_{max} \]

\[ \lambda_{i,j}^* = \left\lceil \frac{l_{i,j}}{n_{i,j}} \right\rceil \]

\[ n_{i,j} = \left\lceil \frac{l_{i,j}}{\lambda_{i,j}} \right\rceil \]

Termination and action slack can be used separately or together.
Power-Aware VBS

Assuming a simple power model \( (P \propto V^2) \)
Look-ahead FS-VBS
Look-ahead FS-VBS

With knowledge of future events:
redistribute computation time between periods
Look-ahead FS-VBS

With knowledge of future events:
- redistribute computation time between periods
- optimal offline method
Look-ahead FS-VBS

With knowledge of future events:
- redistribute computation time between periods
- optimal offline method
- feasible online method
Look-ahead FS-VBS

With knowledge of future events:
- redistribute computation time between periods
- optimal offline method
- feasible online method

May help to handle:
- more complex power models
Look-ahead FS-VBS

With knowledge of future events:
- redistribute computation time between periods
- optimal offline method
- feasible online method

May help to handle:
- more complex power models
- frequency switching cost (time and power)
With knowledge of future events:
  redistribute computation time between periods
  optimal offline method
  feasible online method

May help to handle:
more complex power models
frequency switching cost (time and power)
  time overhead included using overhead accounting
Look-ahead FS-VBS

- Process 1: 30%
- Other utilization: 70% (30%)
- Total utilization: 100% (60%)
Look-ahead FS-VBS

process 1 modified

other utilization

total utilization
Look-ahead FS-VBS

- Process 1 modified
- Other utilization
- Total utilization

Actual improvement depends on the power model.
Look-ahead FS-VBS

Assuming a simple power model ($P \propto V^2$)
Look-ahead online FS-VBS
Look-ahead online FS-VBS

Assume a simple power model ($P \propto V^2$)
Look-ahead online FS-VBS

Assume a simple power model \((P \propto V^2)\)
Look-ahead online FS-VBS

Assume a simple power model \((P \propto V^2)\)

Modify the limits in each period (whenever possible)

s.t. the utilization approximates the average utilization

knowledge of future events
Look-ahead online FS-VBS

Assume a simple power model ($P \propto V^2$)

Modify the limits in each period (whenever possible) s.t. the utilization approximates the average utilization
Conclusions

- Reservation-based scheduling for temporal isolation
Conclusions

- Reservation-based scheduling for temporal isolation
- VBS enables variable execution speed
Conclusions

- Reservation-based scheduling for temporal isolation

- VBS enables variable execution speed

- Overhead accounting
Conclusions

- Reservation-based scheduling for temporal isolation

- VBS enables variable execution speed

- Overhead accounting

- Power-aware VBS

Wednesday, December 1, 2010
Thank you

