Full virtualization of real-time systems by temporal partitioning

Authors
Timo Kerstan (morpheus@upb.de)
Daniel Baldin (dbaldin@upb.de)
Stefan Grösbrink (morenga@upb.de)

Heinz Nixdorf Institute
University of Paderborn, Germany
- CRC614 performs research of self optimizing mechatronical systems

- OCM is the core module of the CRC 614
  - Cognitive operator
  - Reflective operator
  - Controller

- Properties:
  - Hard & Soft realtime
  - High dynamics in
    - resource requirements
    - active services
    - operation modes
  - Fail safe behaviour
- Problems
  - Complexity
    - Distribution
  - Dependability
    - Redundancy
  - Different requirements on
    - Functionality
      - High Level API
      - Low level API
    - Timing
      - Non real time
      - Soft real time
      - Hard real time
    - OS platform
      - Linux
      - RTOS
  - Integration?
Goal

- **Assumptions:**
  - Given real-time systems RS\(_1\),...,RS\(_n\) executing their periodic tasksets \(\Gamma_1,...,\Gamma_n\)
  - RTOS using its own scheduler
    - EDF or
    - RM
  - Executed on dedicated CPU

- **Goal:** Execution of RS\(_1\),...,RS\(_n\) as virtual machines VM\(_1\),...,VM\(_n\) on a single CPU

  - Question 1: What CPU to use for the virtual real-time system?
  - Question 2: How to schedule the virtual machines while preserving full virtualization?
Question 1: What CPU to use for the virtual real-time system?
- Idea: Normalization on slowest given RS.
- Assumptions
  - Comparable CPUs!
  - Infinitesimal time slicing

Normalized system executable?
- Examination of its utilization
- Decision depends on applied scheduling algorithm
- Speedup based on Utilization and Scheduling Bound of applied scheduling algorithm
**Transformation from** $\Gamma_i$ **to** $\Gamma_i'$ (**EDF**)  

- **Time slices based on proportional weight $W$ of the normalized tasksets $\Gamma_i$.**
  - **Example:**  
    - $U(\Gamma_{1s}) = 0.5$, $U(\Gamma_{2s}) = 0.75$, $U(\Gamma_{3s}) = 0.75$
    - $W(\Gamma_{1s}) = 0.25$, $W(\Gamma_{2s}) = 0.375$, $W(\Gamma_{3s}) = 0.375$

- **Assumption: Infinitesimal time slicing: $P \rightarrow 0$**

  **Example:**  
  - Speedup: $U(\Gamma_{1s}) = 0.5$, $U(\Gamma_{2s}) = 0.75$, $U(\Gamma_{3s}) = 0.75 \Rightarrow S = 2$
  - $\Rightarrow U(\Gamma_{1'}) = 0.5$, $U(\Gamma_{2'}) = 0.375$, $U(\Gamma_{3'}) = 0.375$
  - Speedup $S$ is a design hint for choosing the minimal needed CPU speed.
Summary

- Goal: Execution of RS$_1$, ..., RS$_n$ as virtual machines VM$_1$, ..., VM$_n$ on a single CPU

  - Question 1: What CPU to use for the virtual real-time system?
    - Normalization on slowest given RS.
    - Speedup based on EDF/RM
    - Transformation of $\Gamma_1$, ..., $\Gamma_n$ into $\Gamma'_1$, ..., $\Gamma'_n$
    - Up to now not realizable
      - Assumption: Infinitesimal time slicing!

  - Question 2: How to schedule the virtual machines while preserving full virtualization?
    - Idea: Usage of single time slot periodic partitions
      - Period length not infinitesimal!
      - But how to choose the period?
Resource Partitions

A resource partition $\Pi$ is a tuple $(\gamma, P)$

$\gamma = \{(S_1, E_1), \ldots, (S_N, E_N)\}$ with $0 \leq S_1 < E_1 < \ldots < S_N < E_N$ for some $N \geq 1$

$P$ is the partition period.

Physical resource is available only during intervals $(S_i + j \cdot P, E_i + j \cdot P)$, $1 \leq i \leq N$, $j \geq 0$.

- A resource partition with multiple time pairs ($N>1$) is called Multiple Time Slot Periodic Partition
- A resource partition with only one time pair ($N=1$) is called Single Time Slot Periodic Partition
- Examples:
Virtual real-time system consists of:
$$\Gamma_i = \{\tau_k(T_k, C_k) | k = 1, \ldots, m\}, \quad i = 1, \ldots, n$$
$$\Pi_i = \{(S_i, E_i)\}, \quad P \mid S_1 = 0, E_i = S_i + \alpha_i \cdot U(\Gamma_i) \cdot P, \quad S_i = E_{i-1}\}, \quad \text{with } P \text{ being equal for all } \Pi_i.$$

- Activation length of a virtual machine depends on the utilization of $\Gamma_i$ being $U(\Gamma_i)$ and on its used scheduling algorithm

- $\alpha_i$ is a scaling factor depending on the utilization bound of the applied scheduling algorithm
  - $\alpha_i = 1$ for EDF
  - $\alpha_i = 1/U_{\text{ub}}$ for RM
The period $p$ cannot be chosen arbitrarily!

- Two EDF virtual machines
  - $T_1=\{(2,8),(2.5,10)\}$
  - $T_2=\{(2,8),(2.5,10)\}$
  - $U_1=U_2=0.5$
  - $\text{STSPP}_1=\{(0,4),8\}$
  - $\text{STSPP}_2=\{(4,8),8\}$

![Diagram of virtual machine scheduling](image)
The period $p$ cannot be chosen arbitrarily!

- Two EDF virtual machines
  - $VM_1 = \{(2,8),(2.5,10)\}$
  - $VM_2 = \{(2,8),(2.5,10)\}$
  - $U_1 = U_2 = 0.5$
  - $STSPP_1 = \{\{(0,4)\}, 8\}$
  - $STSPP_2 = \{\{(4,8)\}, 8\}$
The period $p$ cannot be chosen arbitrarily!

- Two EDF virtual machines
  - $VM_1 = \{(2,8),(2.5,10)\}$
  - $VM_2 = \{(2,8),(2.5,10)\}$
  - $U_1 = U_2 = 0.5$
  - $STSPP_1 = \{((0,4)), 8\}$
  - $STSPP_2 = \{((4,8)), 8\}$
The period p cannot be chosen arbitrarily!

- Two EDF virtual machines
  - VM$_1$=\{(2,8),(2.5,10)\}
  - VM$_2$=\{(2,8),(2.5,10)\}
  - U$_1$=U$_2$=0.5
  - STSPP$_1$=\{\{(0,4)\},8\}
  - STSPP$_2$=\{\{(4,8)\},8\}
The period $p$ cannot be chosen arbitrarily!

- Two EDF virtual machines
  - $VM_1 = \{(2,8),(2.5,10)\}$
  - $VM_2 = \{(2,8),(2.5,10)\}$
  - $U_1 = U_2 = 0.5$
  - $STSPP_1 = \{(0,4), 8\}$
  - $STSPP_2 = \{(4,8), 8\}$
The period $p$ cannot be chosen arbitrarily!

- Two EDF virtual machines
  - $VM_1=\{(2,8),(2.5,10)\}$
  - $VM_2=\{(2,8),(2.5,10)\}$
  - $U_1=U_2=0.5$
  - $STSPP_1=\{(0,4),8\}$
  - $STSPP_2=\{(4,8),8\}$

Virtual Machine 1

Virtual Machine 2

Hypervisor

The period $p$ cannot be chosen arbitrarily!
The period $p$ cannot be chosen arbitrarily!

- Two EDF virtual machines
  - $VM_1 = \{(2, 8), (2.5, 10)\}$
  - $VM_2 = \{(2, 8), (2.5, 10)\}$
  - $U_1 = U_2 = 0.5$
  - $STSPP_1 = \{(0, 4)\}, 8$
  - $STSPP_2 = \{(4, 8)\}, 8$
The period $p$ cannot be chosen arbitrarily!

- Two EDF virtual machines
  - $\text{VM}_1 = \{(2,8),(2.5,10)\}$
  - $\text{VM}_2 = \{(2,8),(2.5,10)\}$
  - $U_1 = U_2 = 0.5$
  - $\text{STSPP}_1 = \{(0,4)\}, 8$
  - $\text{STSPP}_2 = \{(4,8)\}, 8$

![Diagram showing EDF Schedulers and VMs](image-url)
Supply Function of Example Schedule

VM₁ = {(8;2), (10;2,5)}  STSPP₁ = {(0;4), 8}
VM₂ = {(8;2), (10;2,5)}  STSPP₂ = {(4;8), 8}

Idealized supply with infinite time slicing
Supply Function of Example Schedule

\[ P = \gcd\left(\left\{ T_k \mid \tau_k \in \bigcup_{i=1}^{n} \Gamma_i \right\}\right) \]
## Summary

- **Goal:** Execution of $RS_1, \ldots, RS_n$ as virtual machines $VM_1, \ldots, VM_n$ on a single CPU

  - Question 1: What CPU to use for the virtual real-time system?
    - Normalization on slowest given RS.
    - Speedup based on EDF/RM
    - Transformation of $\Gamma_1, \ldots, \Gamma_n$ into $\Gamma'_1, \ldots, \Gamma'_n$
    - Theorem 1 ensures the needed allocation!
    - Assumption: Infinitesimal time slicing!

  - Question 2: How to schedule the virtual machines while preserving full virtualization?
    - Usage of single time slot periodic partitions

    $$P = \gcd\{\tau_k | \tau_k \in \bigcup_{i=1}^n \Gamma_i \}'\}$$

    - $P$ may get very small
    - Tradeoff between interrupt latency and switching overhead

---

### Diagram

![Diagram](image-url)
Reducing the switching overhead

- Small values of P lead to a larger weight of the switching overhead

- Is it possible to increase the value of P without missing deadlines?

- Idea:
  - Choose a P larger than the GCD of all deadlines
  - Calculate for every VM at its tasks deadlines within the hyperperiod
    - the needed computation time N(t)
    - the assigned computation time Z(t)
  - The fraction N(t)/Z(t) is the necessary speedup to fulfill the requested computation time
  - The maximum fraction within the hyperperiod is the speedup to apply the chosen STSPP length P
Reducing the switching overhead

\[ P = \text{GCD}\{40, 80, 100, 200\} = 20 \]

\[ VM_1 = \{(40, 10), (100, 25)\} \]

\[ VM_2 = \{(80, 16), (200, 60)\} \]

\[ STSPP_1 = \{(0, 10), 20\} \]

\[ STSPP_2 = \{(10, 20), 20\} \]
Reducing the switching overhead

\[ \begin{align*}
VM_1 &= \{(40,10),(100,25)\} \\
VM_2 &= \{(80,16),(200,60)\}
\end{align*} \]

\[ P = GCD(\{40,80,100,200\}) = 20 \]

\[ STSPP_1 = \{(0,10),20\} \quad STSPP_2 = \{(10,20),20\} \]
Conclusion

- **Goal:**
  - Transform given real-time systems into a virtualized system
    - Performance of Host CPU?
    - Full virtualization
    - VM Scheduling using FTS

- **We developed a simple methodology to derive**
  - CPU speedup
  - STSPPs
  - FTS Schedule

- **Switching overhead can be severe with a small P**
  - Reduction by additional speedup
  - Analysis to determine the required speedup for a given P
Conclusion

Goal:
- Transform given real-time systems into a virtualized system
  - Performance of Host CPU?
  - Full virtualization
  - VM Scheduling using FTS

We developed a simple methodology to derive
- CPU speedup
- STSPPs
- FTS Schedule

Switching overhead can be severe with a small P
- Reduction by additional speedup
- Analysis to determine the required speedup for a given P

Thank you for your attention.