Parallel-Task Scheduling on Multiple Resources

Mike Holenderski, Reinder J. Bril, Johan Lukkien

Eindhoven University of Technology
Surveillance camera
Surveillance camera

- Sensor
- Main memory
- Local memory
- DSP
- DMA controller
- CPU
- Local memory
- Network
Surveillance camera

Sensor → Main memory → DSP → DMA controller → CPU → Local memory → Network
Surveillance camera

Sensor → Main memory → Local memory → DMA controller → CPU

CPU: P, $, Local memory → Network

DSP: P → Local memory
Surveillance camera

- Sensor
- Main memory
- DSP
- Local memory
- DMA controller
- CPU
- Local memory
- Network

bus

4
Surveillance camera

- Sensor
- Main memory
- DSP
  - Local memory
- DMA controller
  - Local memory
- CPU
  - P
  - P
  - $4$
- Network

Bus
Surveillance camera
Problem

• Existing synchronization protocols for multiprocessors assume tasks execute on one processor at a time

• Existing parallel-task real-time scheduling algorithms assume independent tasks

• Simple approach of treating the entire platform as a single resource is inefficient
Goal

• Scheduling algorithm for parallel tasks with real-time constraints

• Exploit parallelism on a platform comprised of multiple heterogeneous resources.
Multiple heterogeneous resources

- Each resource consists of **multiple units**
  - Each unit is a serially accessible entity
  - Each resource has a capacity $\geq 1$

- Each resource is either **preemptive** or **non-preemptive**
  - Preemption does not corrupt a preemptive resource
Application

• Each task $\tau_i$ has a
  - $\pi_i$: fixed priority
  - $T_i$: period
  - $D_i$: deadline ($D_i \leq T_i$)
  - $S_i$: sequence of segments

• Each segment $\tau_{ij} \in S_i$ has a
  - $E_{ij}$: worst-case execution time
  - $R_{ij}$: set of resource requirements

• Each resource requirement $(r_k, n_k) \in R_{ij}$ has a
  - $r_k$: required resource
  - $n_k$: number of required units
Resource requirements graph

Resources are accessed simultaneously
Local vs. global resources

Preemptive resources

Segments

Non-preemptive resources

$p_1$ $p_2$ $p_3$

$\tau_{1,1}$ $\tau_{2,1}$ $\tau_{3,1}$ $\tau_{3,2}$ $\tau_{4,1}$ $\tau_{4,2}$

$n_1$ $n_2$
Local vs. global resources

Preemptive resources

Segments

Non-preemptive resources

Local vs. global resources
Local vs. global resources

Preemptive resources

Segments

$\tau_{1,1}$  $\tau_{2,1}$  $\tau_{3,1}$  $\tau_{3,2}$  $\tau_{4,1}$  $\tau_{4,2}$

Non-preemptive resources

$\text{Local resources}$  $\text{Global resources}$
PSRP

- Equip each ...
  - local preemptive resource with a priority queue
  - global resource with a FIFO queue
  - local non-preemptive resource with a ceiling (according to SRP)
  - local preemptive resource with a system ceiling $\pi_p$ (according to SRP)
  - global resource with ceiling = highest task priority

- Upon arrival of $\tau_{i,j}$, it is added atomically to all queues in $R_{i,j}$
- Upon completion of $\tau_{i,j}$, it is removed from all queues in $R_{i,j}$
- $\tau_{i,j}$ can start if ...
  - $\tau_{i,j}$ is at the head of all queues in $R_{i,j}$, and
  - $\Pi_i > \pi_p$ for all $p \in R_{i,j}$

- Schedule segments starting from the head of queues, as long as:
  - enough resource units are available, and
  - all other resources required by the segment are available
    - otherwise busy wait (on global resources)
PSRP example
Analysis

• Compute the worst-case response time of each segment

• Worst-case response time of a task = Worst-case response time of its last segment

• Analysis is exponential in the number of tasks (for tasks which contain more than one segment)

SEE PAPER
Conclusions & future work

✓ First scheduling algorithm for
  - partitioned parallel tasks
  - with real-time constraints
  - requiring multiple heterogenous resources

✓ Improved parallelism vs. treating the entire platform as a single resource

• Preemptive resources have capacity = 1

• Potentially large delays for high priority tasks
  - Global resource queues are sorted according to FIFO
  - Global resources are scheduled non-preemptively