Tightening contention delays while scheduling parallel applications on multi-core architectures

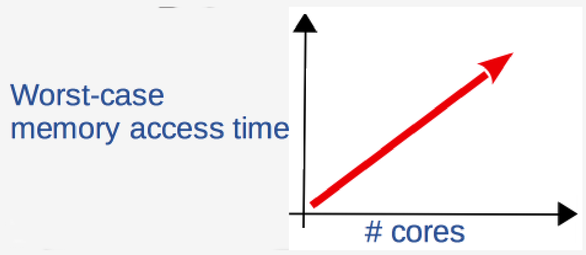
Benjamin Rouxel, Steven Derrien and Isabelle Puaut
PACAP/Cairn team, University of Rennes 1 and CNRS, France

Motivation

Critical Embedded Systems

- Reliability and safety
- Demand for high-performance
- Complex architectures

Memory access latency is always the bottleneck in average case and in the worst case.



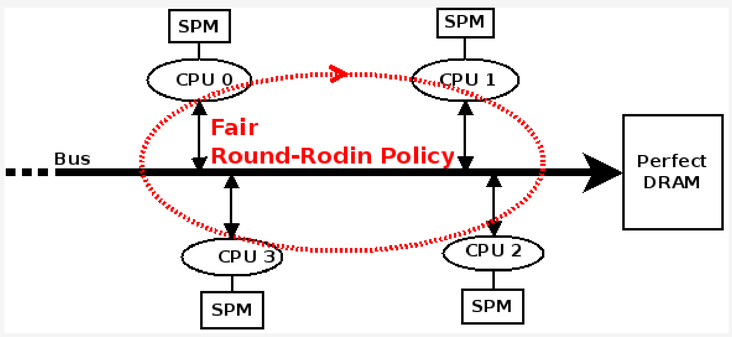
Contribution

- Method to derive precise bounds on **worst-case contention** using application structure knowledge at schedule time.
- ILP formulation and heuristic: a time-triggered partitioned schedule
- Experimental data to evaluate the benefit of precise estimation of contentions and a discussion on the interest of allowing concurrency (opposed to contention-free)

What's next?

- to enable asynchronous communication (ongoing, submitted to RTAS)
- to further refine the contention model, thus increasing accuracy
- to handle NoC-based architectures

Hardware model



- $NbCores$: number of cores
- T_{slot} : bus max. access duration per core
- D_{slot} : max. amount of data transmitted in T_{slot}
- $data$: amount of data in the request

$\#chunks = \lfloor \frac{data}{D_{slot}} \rfloor$
$remainingTime = (data \% D_{slot}) * (\frac{T_{slot}}{D_{slot}})$
$\#waitingSlots = \lceil \frac{data}{D_{slot}} \rceil$
$delay =$

$$\underbrace{T_{slot} * \#waitingSlots * \textbf{\#interference}}_{\text{Total waiting time}} + \underbrace{T_{slot} * \#chunks + remainingTime}_{\text{Total access time}}$$

Communication cost computation: #interference

Worst-case $\#interference$

$\#interference = NbCores - 1$

Contention-aware $\#interference$

#interference = #phases effectively interfering
= #phases overlapping in the schedule on different cores

- **Non-overlapping phases i and j**

$$\neg(end_j^Y \leq start_i^X \vee end_i^X \leq start_j^Y)$$

where X and Y can either represent a *read* or a *write*

- **Non-concurrent tasks can not overlap**

Transitive closure of the DAG with Warshall's algorithm

$$\#interference_i^X = \sum_{j \in T | are\_concurrent(i,j)} are\_overlapping(\tau_i^X, \tau_j^r) + \sum_{j \in T | are\_concurrent(i,j)} are\_overlapping(\tau_i^X, \tau_j^w)$$

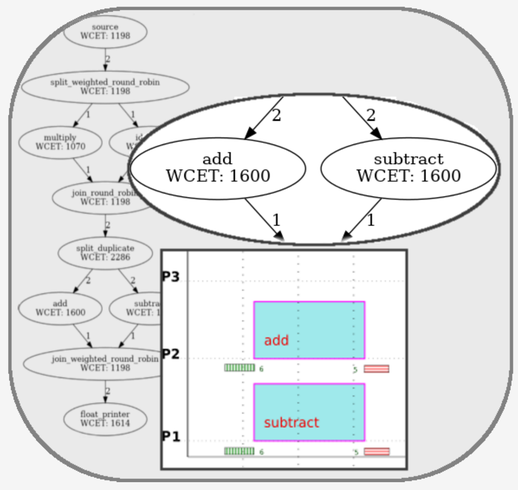
where X can either represent a *read* or a *write*

Software model

- Directed Acyclic Graph (DAG)
- Predictable Execution Model (PREM)
  - Isolate communications and computations: *read-exec-write*

Example

$NbCores = 8 \quad T_{slot} = 3 \; ; D_{slot} = 3; \quad data = 2$

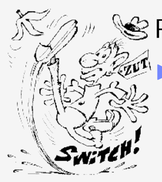


$\#chunks = \lfloor 2/3 \rfloor = 0$
$remainingTime = 2 \% 3 = 2$
$\#waitingSlots = \lceil 2/3 \rceil = 1$

Worst-case	Contention-aware
$\#interference = 7$	$\#interference = 1$
$delay = 3 * 1 * 7 + 2 = 23$	$delay = 3 * 1 * 1 + 2 = 5$

Integer Linear Programming (ILP)

- Objective function: minimize schedule length
- Mapping constraints
  - a task is mapped on only one core
  - a task must be mapped on one core
- Scheduling constraints
  - non-overlapping phases on the same core
  - precedence between phases and tasks
  - communication cost computations



Problem hidden here:
- Quadratic and non-convex constraints for communication costs
  Total waiting time has been linearized

Forward list scheduling heuristic

- Order tasks
- On each core, try to map the task the earliest possible
  - two tasks do not overlap on the same core
  - ensure task's precedence (data dependency)
  - compute communication costs
- Keep the mapping/scheduling with the minI makespan
- Start again with an other tasks

Problem hidden here:
- Finding contention-free schedule version
- Computing interference at schedule modifies past decisions

Experiments

validating the heuristic against the ILP

- Task Graph For Free (TGFF)
- Synthetic benchmarks, wide range of graph topologies, communication configuration, ...
- Parameters:

#Task-graphs	#Tasks <min,max,avg>	Max width <min,max,avg>	WCET	Amount of data exchanged
200	3, 34, 14	1, 11, 3	$[1;70]$	$[0;11]$

  - Number of cores: $[2;15]$
  - T_{slot} : $[1;10]$

Table 1: Degradation of the heuristic compared with the ILP

% of exact results (ILP only)	degradation <min,max,avg> %
98%	-8%, 71%, 7%

Gain of contention-aware w.r.t worst-contention

- StreamIT benchmark suite
- Real-case benchmarks
- Fork-join graph
- Fixed data-rate, amount of data, WCET estimates known at compile time