Feasible Preemption Point Analysis for Data Cache Reference Patterns

Harini Ramaprasad

Department of Computer Science & Center for Embedded Systems Research
North Carolina State University
Advisor: Dr. Frank Mueller

Motivation

- Scheduling in Real-Time systems requires
  - Worst-case Execution time (WCET) → Static Timing Analysis
- Data Caches in Real-Time systems
  - Creates unpredictability in timing
- Preemptive Systems
  - Further complicates timing analysis
  - Need to add data cache delay due to preemption (D-CRPD)
- Our contribution
  - Significantly tighter bounds for D-CRPD and hence response times

Calculating D-CRPD

- Critical instance when preemption delay is considered
  - Does not necessarily occur when all tasks released simultaneously
  - Per-job analysis performed
- Steps involved
  - Preemption delay
  - Calculate delay given preempted/preempting task info
  - Number of preemptions
  - Calculate max # of preemptions possible for task
  - Worst-case scenario
  - Identify placement of preemption pts. in iteration space of task

Preemption Delay Calculation

- In every interval between consecutive preemption points
  - Find shortest possible exec time for current task (min_t)
  - Use WCETs of hp tasks
  - Find longest possible exec time for current task (max_t)
  - Use BCETs of hp tasks
  - For both min_t and max_t, in iteration space of current task,
  - Find earliest iteration point that is guaranteed to be reached
  - Find latest iteration point that can potentially be reached
  - Calculate max # of preemptions possible for task
  - For both min_t and max_t, in iteration space of current task,
  - Place BCETs of hp tasks in interval
  - If any time remains, current task may be scheduled
  - Provide improvements of an order of magnitude over previous methods

Feasible preemption points

- Potential preemption points
  - Release points of higher priority (hp) tasks
  - Feasibility check – consider interval between two consecutive points
  - Check whether task can be scheduled in interval
  - Place BCETs of hp tasks in interval
  - If any time remains, current task may be scheduled
  - Check if portion of task remains beyond interval
  - Use WCETs of hp tasks + WCET of current task

Findings

- Calculation of max. # of preemptions
  - Comparison methods
    - Higher Priority Job bound (HJ Bound)
    - Old method
    - Method proposed by Staschulat et al.
    - New method
  - For all methods
    - Our new method provides the tightest estimates
    - Provide improvements of an order of magnitude over
      - HJ bound
      - Our old method
      - Provide significant improvements over method by Staschulat et al.

Experimental Results

- Utilization = 50%
- Utilization = 80%

Conclusions

- Contributions
  - Determination of new critical instance
  - Preemption delay affects response time
  - Max. # of preemptions & Response time
    - Tighter bound calculated
    - Max. # preemptions
      - 12x - 20x improvement over HJ bound and our old method
      - 7x improvement over method by Staschulat et al.
  - Response Time
    - 1.5x - 2.5x improvement over HJ bound and our old method
    - 1.5x improvement over method by Staschulat et al.
  - Worst-case scenario
    - More realistic
  - First work considering staggered releases in D-CRPD calculation