

Feasible Preemption Point Analysis for Data Cache Reference Patterns

Harini Ramaprasad

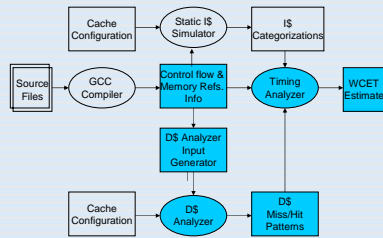


NC STATE UNIVERSITY

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

Motivation

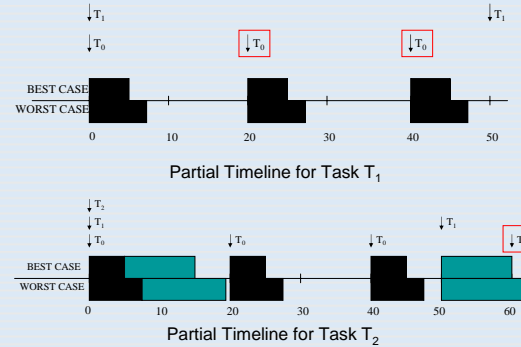
- Scheduling in Real-Time systems requires
 - Worst-case Execution time (WCET) → Static Timing Analysis
- Data Caches in Real-Time systems
 - Creates unpredictability in timing
- Preemptive Systems
 - Further complicates timing analysis
 - Need to add data cache delay due to preemption (D-CRPD)
- Our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



Static Timing Analyzer Framework

An Example

Task	Period	WCET	BCET
T_0	20	7	5
T_1	50	12	10
T_2	200	30	25



Theoretical Results

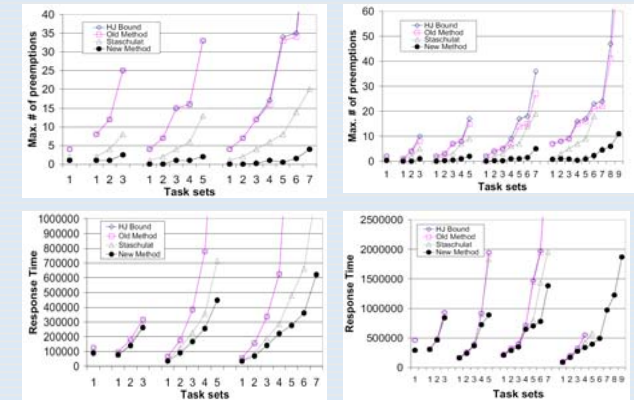
Task ID	Period	WCET	# Preemption pts New Method (Min/Max/Avg)					# Pts HJ Bound	# Pts Old method	# Pts Staschulat
			W/B=1	W/B=1.5	W/B=2	W/B=2.5	W/B=3			
1	80000	16000	1/1/1	1/1/1	1/1/1	1/1/1	1/1/1	8	8	2
2	100000	5000	0/1/0.25	0/1/0.25	0/2/0.5	0/2/0.5	0/2/0.5	12	12	4
3	200000	30000	3/3/3	3/4/3.5	3/5/4	3/5/4	3/5/4	25	25	8

Max # of preemptions for varying WCET/BCET – U = 50%

Task ID	Period	WCET	# Preemption pts New Method (Min/Max/Avg)					# Pts HJ Bound	# Pts Old method	# Pts Staschulat
			W/B=1	W/B=1.5	W/B=2	W/B=2.5	W/B=3			
1	80000	20000	2/2/2	2/2/2	2/2/2	2/2/2	2/2/2	8	8	3
2	100000	12000	1/2/1.5	1/3/1.75	1/3/1.75	1/4/2	1/4/2	12	12	6
3	200000	50000	6/6/6	7/10/8.5	8/12/10	9/14/11.5	9/14/11.5	25	25	19

Max # of preemptions for varying WCET/BCET – U = 80%

Experimental Results



Utilization = 50%

Utilization = 80%

Calculating D-CRPD

- Critical instance when preemption delay is considered
 - Does not necessarily occur when all tasks released simultaneously
 - Need to consider entire hyperperiod
 - 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
 - Pick point with max. delay in the range obtained above

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.*

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