UNIVERSITÄT
DES
SAARLANDES

Experimental Evaluation of Cache-Related Preemption Delay Aware Timing Analysis

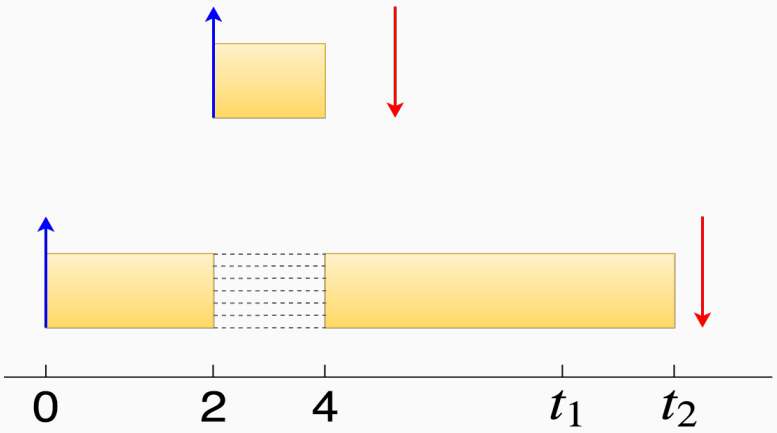
WCET, 2018

Darshit Shah, Sebastian Hahn, Jan Reineke

July 3, 2018

Universität des Saarlandes

Experimental Evaluation of Cache-Related Preemption Delay Aware Timing Analysis

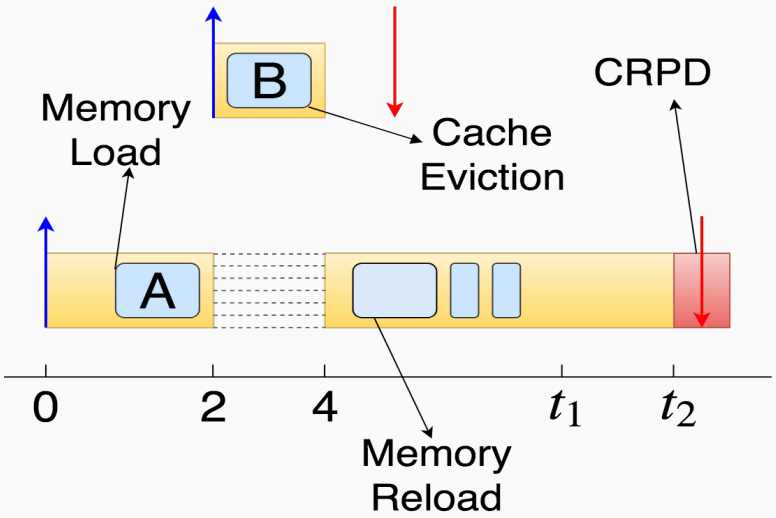


- **WCET:** Maximum time to completion *in isolation*
- **Response Time:** Maximum time to completion

$$R_1 = C_1 + C_2$$

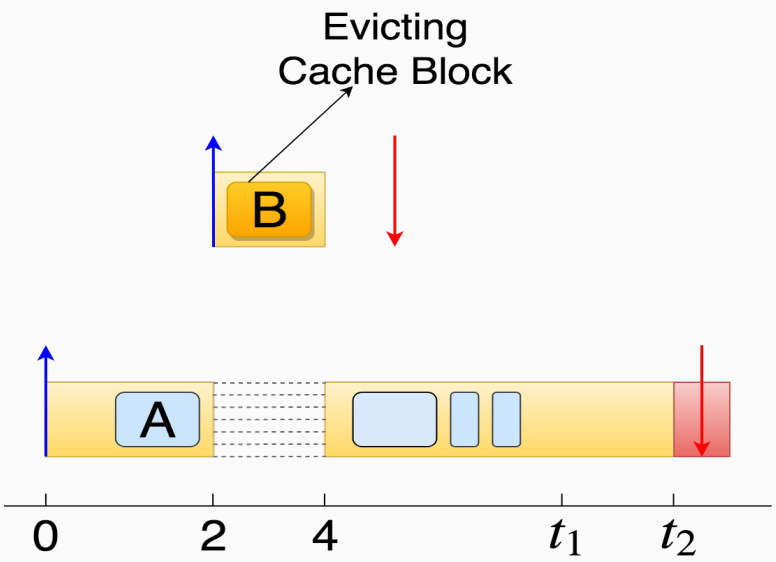
- Preemptions are free?

Experimental Evaluation of Cache-Related Preemption Delay Aware Timing Analysis



- Preempting task may evict cache blocks
- Causes extra memory reloads
- Additional time required is called *Cache-Related Preemption Delay*

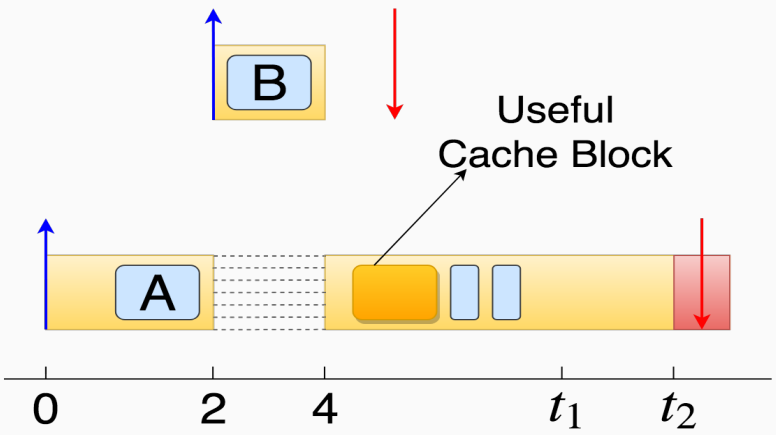
Experimental Evaluation of
Cache-Related Preemption Delay Aware Timing Analysis



- By considering *effect of preempting task*

  e.g.: ECB-Only, UCB-Union, UCB-Union Multiset

Experimental Evaluation of
Cache-Related Preemption Delay Aware Timing Analysis



- By considering *effect of preempting task*

  e.g.: ECB-Only, UCB-Union, UCB-Union Multiset

- By considering *effect on the preempted task*

  e.g.: UCB-Only, ECB-Union, ECB-Union Multiset

Experimental Evaluation of Cache-Related Preemption Delay Aware Timing Analysis

For a comprehensive evaluation we would like to have:

- Real program code
  - To compute WCET
  - Cache and Memory access patterns
- Complete tasksets
- Deadlines
- Relative priorities of tasks
- Task periods

However, not enough real world data available for a comprehensive evaluation

Synthetic Generation of Task Characteristics[1]

Standard Task Parameters:

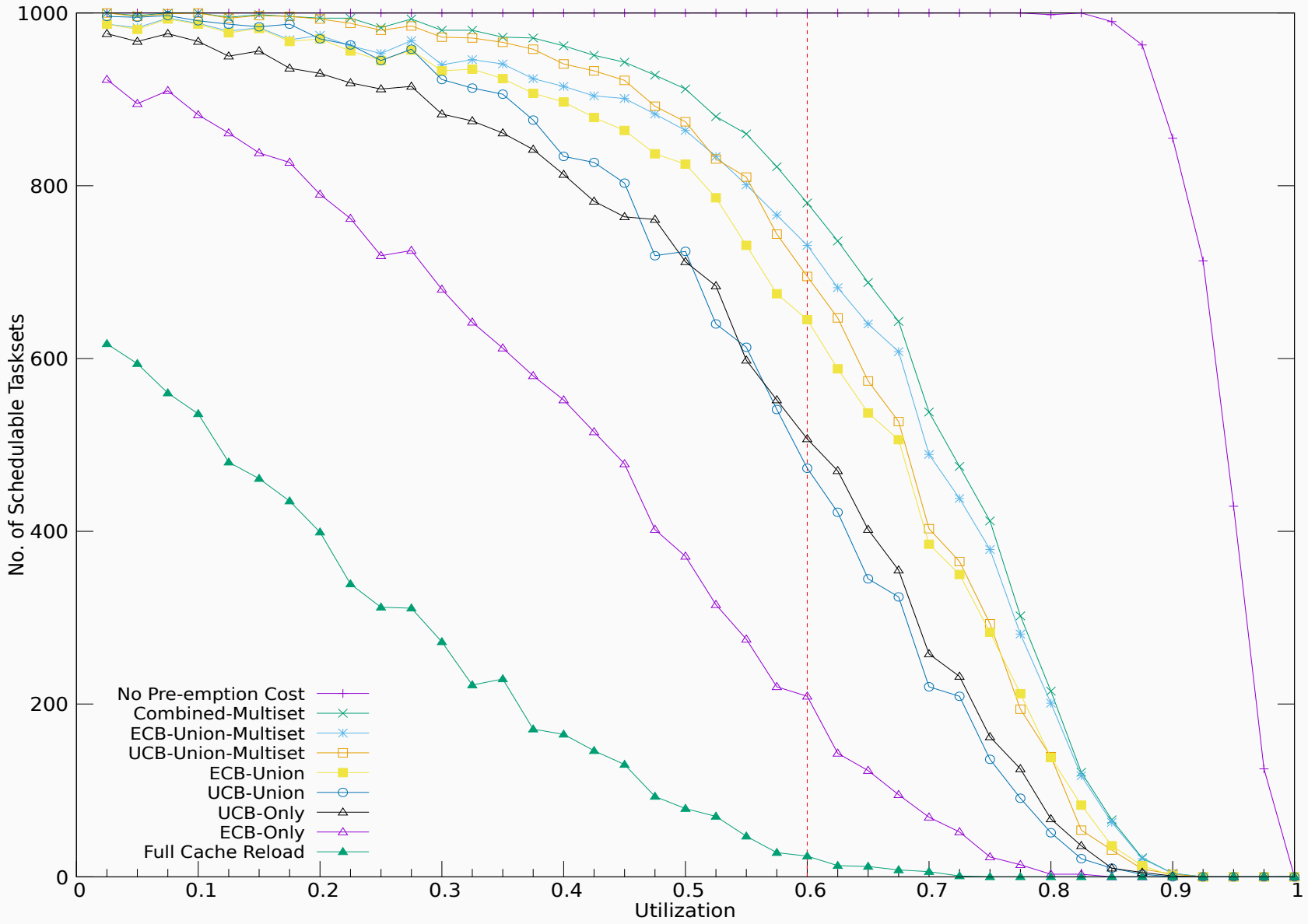
- Periods (T_i): [5ms, 500ms] from a Log-Uniform Distribution
- Utilization (U_i): UUnifast
- WCET (C_i): $T_i \times U_i$

CRPD Parameters:

- ECB: UUnifast
- UCB: Upto 30% of ECBs

[1] *Altmeyer et al.* Improved cache related pre-emption delay aware response time analysis for fixed priority pre-emptive systems. Real-Time Syst., 2012.

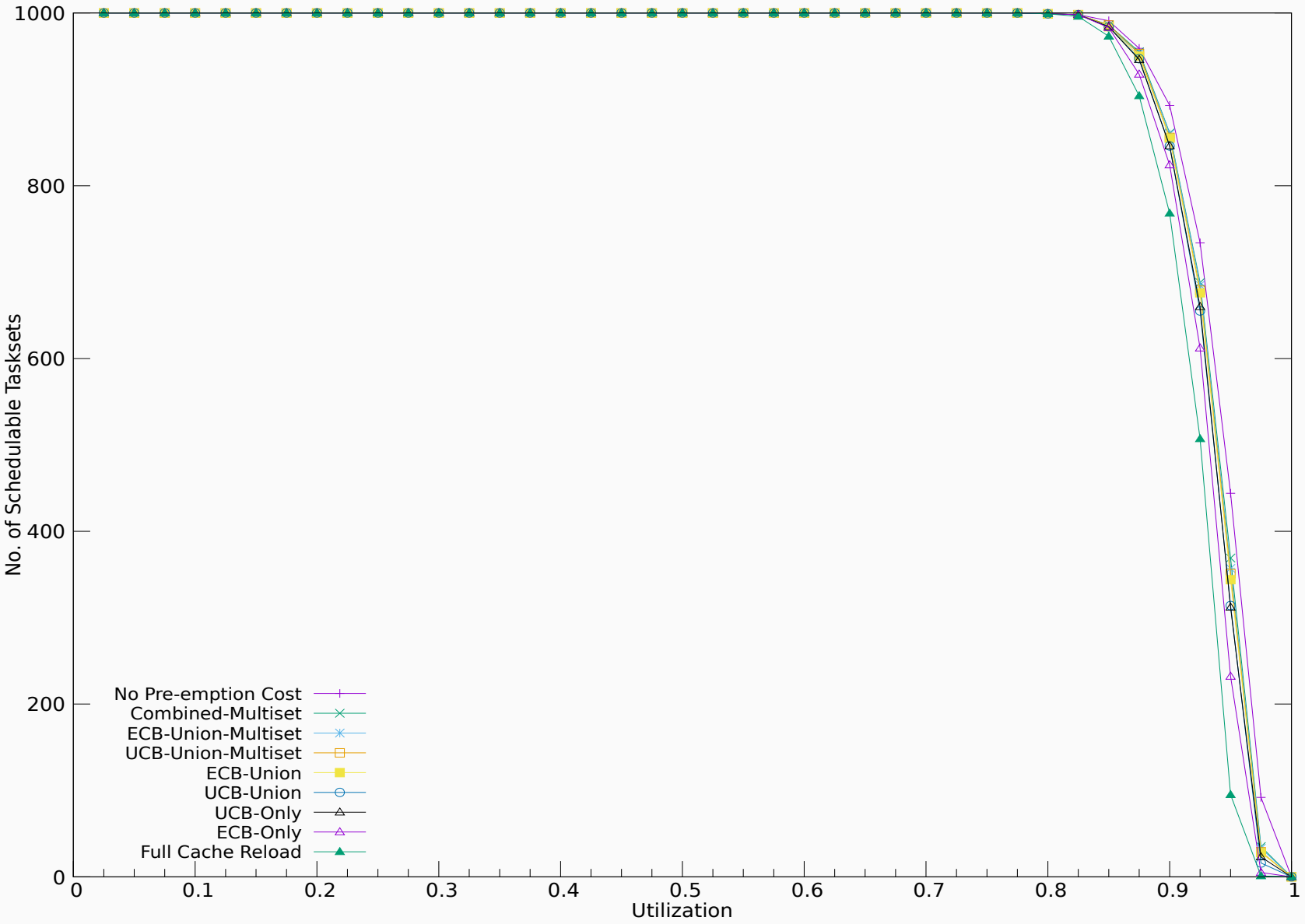
CRPD-Aware Schedulability



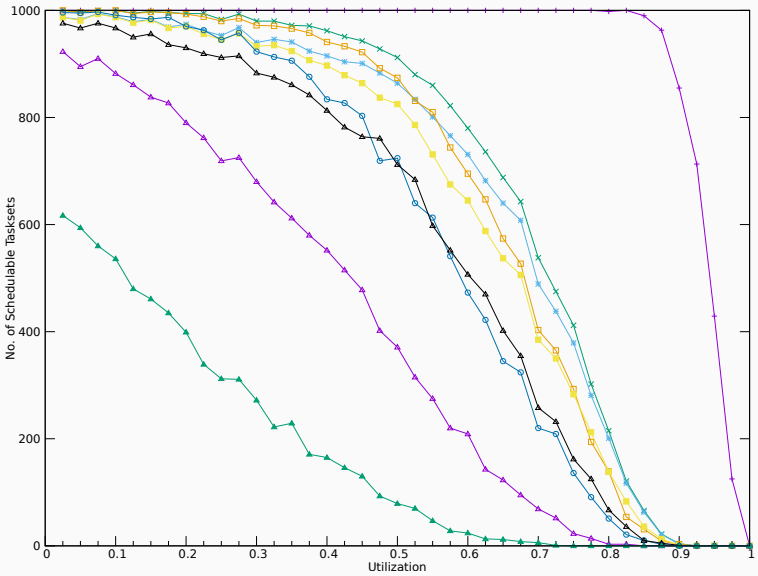
Block Reload Times

- Timing penalty incurred due to a cache miss
- Usually considered equal to the *Memory Latency*
- Original experiment considers BRT = 8μs
- Real DRAMs have lower BRT (approx. 0.2μs)

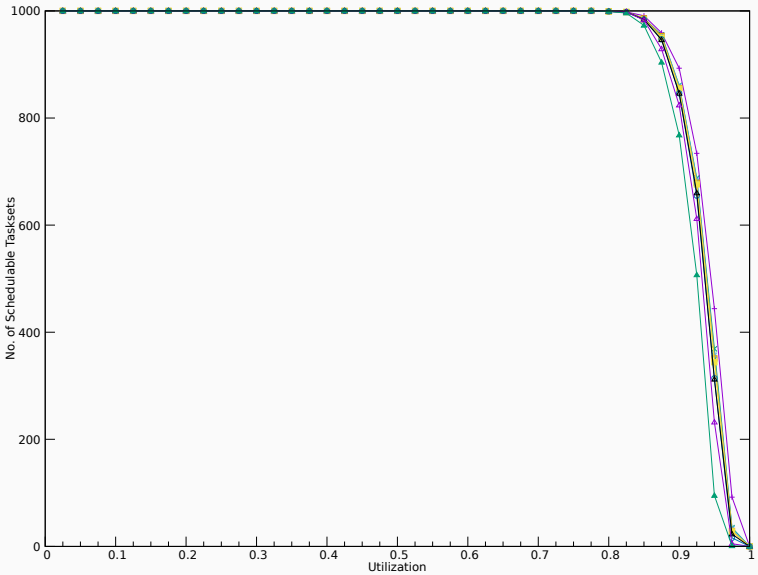
CRPD-Aware Schedulability



CRPD-Aware Schedulability



At BRT=8μs, effect of CRPD seems to be very *significant*



At BRT=0.2μs, effect of CRPD seems to be *negligible*

Synthetic Generation of Task Characteristics

Standard Task Parameters:

- Periods (T_i): [5ms, 500ms] from a Log-Uniform Distribution
- Utilization (U_i): UUnifast
- WCET (C_i): $T_i \times U_i$

CRPD Parameters:

- ECB: UUnifast
- UCB: Upto 30% of ECBs

Correlations between parameters not captured

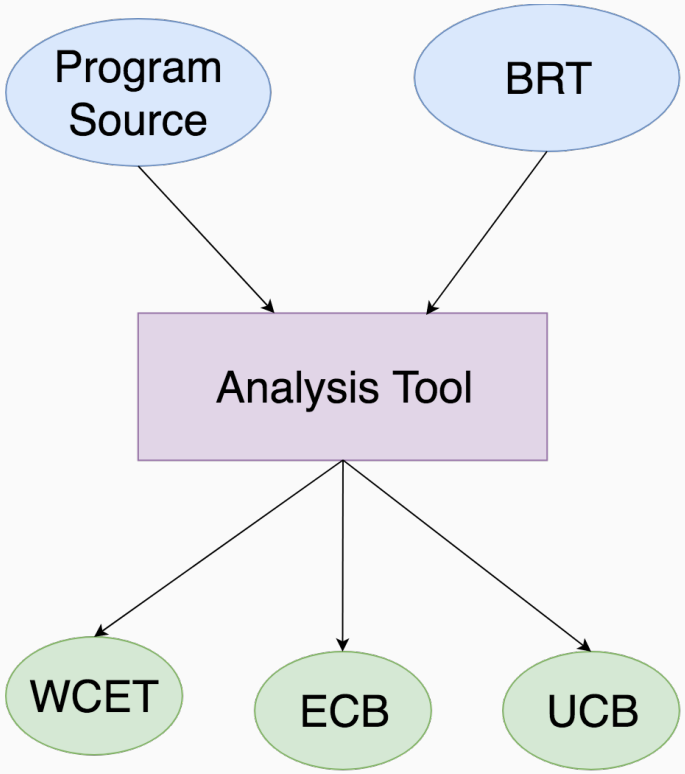
Analysis Based Task Characteristic Generation

Standard Task Parameters:

- Benchmark: Mälardalen Suite
- WCET: **Analysis Tool**
- Utilization: UUnifast
- Periods: $T_i = \frac{C_i}{U_i}$

CRPD Parameters:

- ECB: **Analysis Tool**
- UCB: **Analysis Tool**



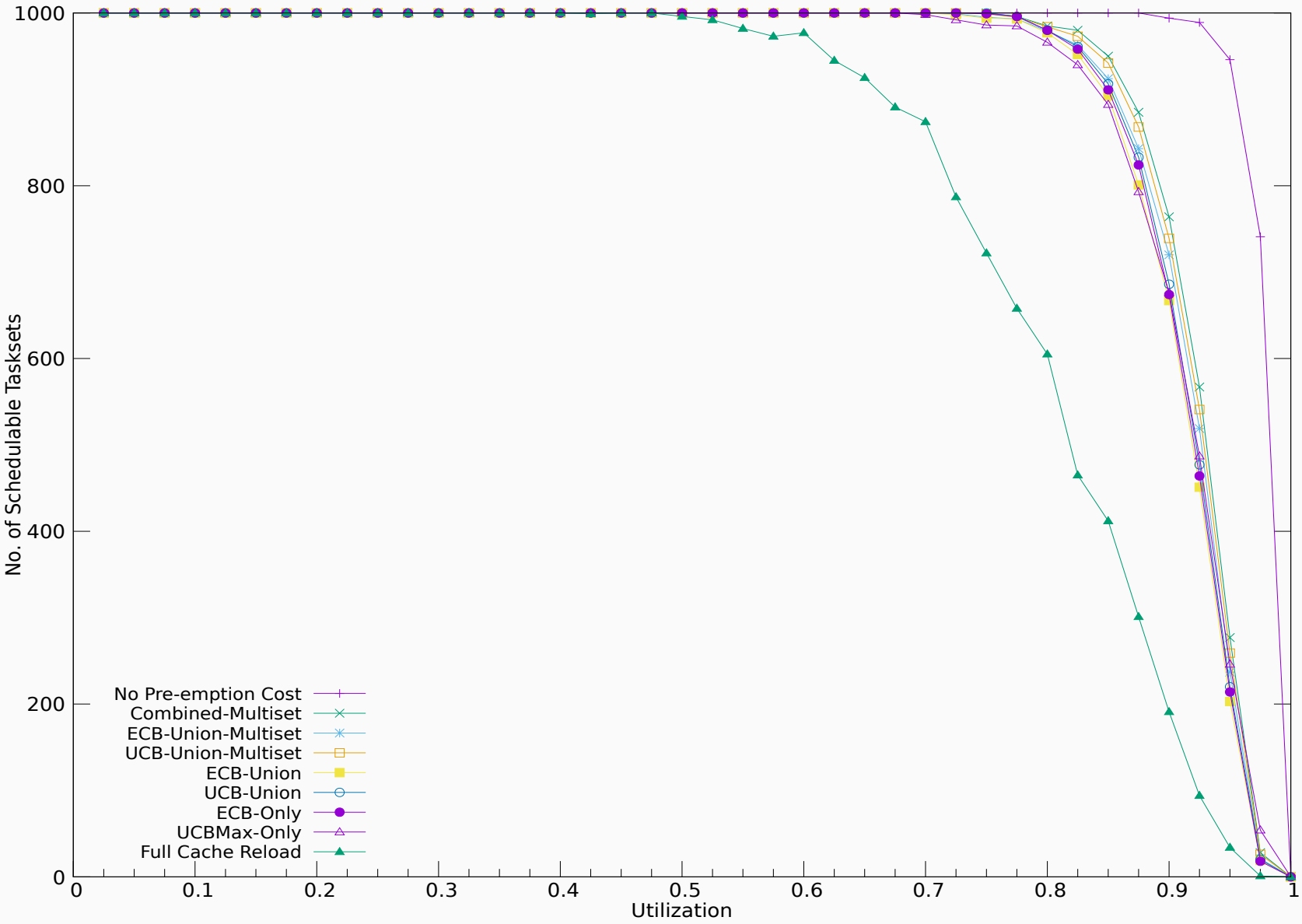
Single analysis correctly captures various correlations

LLVM-Timing Analysis

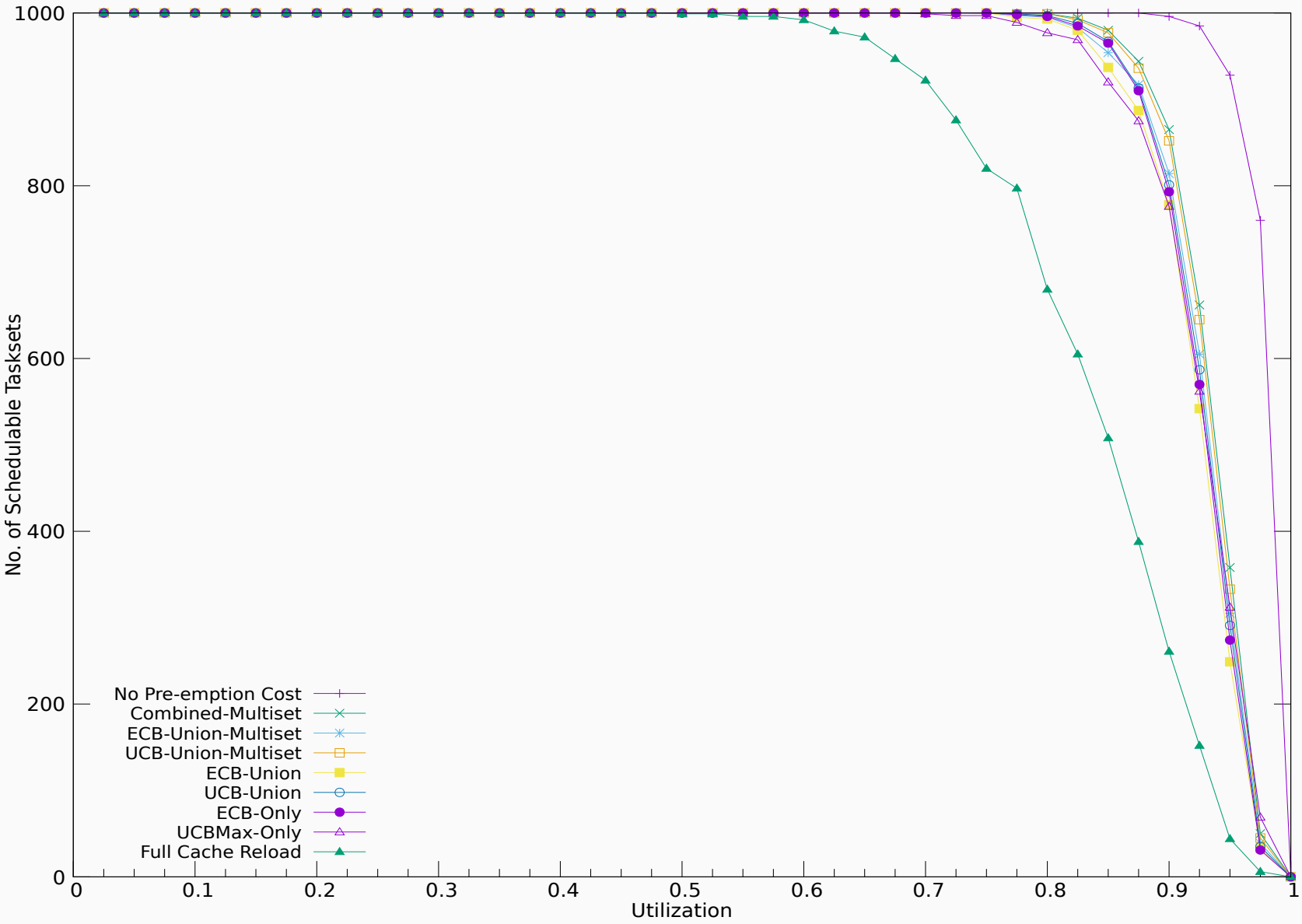
LLVMTA is a low-level timing analysis tool written on LLVM

- Performs micro-architectural analysis of target system
  - 100 Mhz ARM Core
  - 5-stage In-order pipeline
  - Branch prediction
  - Native Floating-Point support
  - No Data Cache, scratchpad memory available for data
  - 2 KiB Instruction Cache (Direct Mapped, 256 Sets)
- Virtual Loop-Peeling for context sensitive analysis
- Must and May analyzes to predict cache behavior

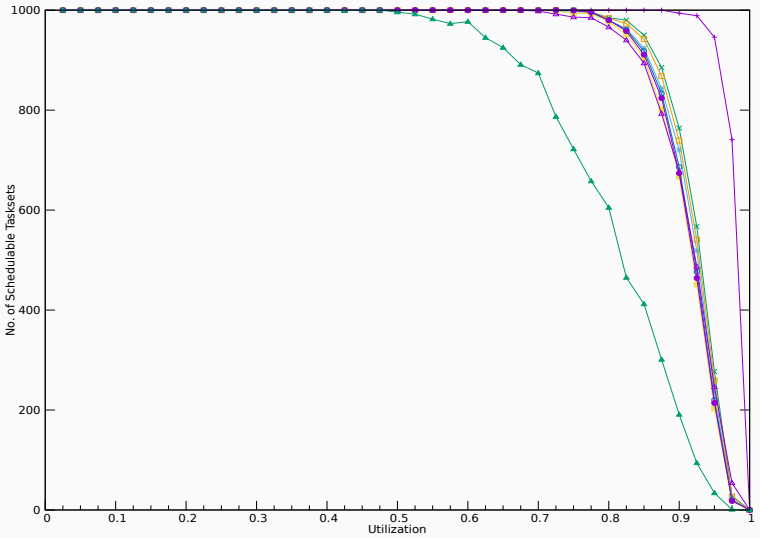
CRPD-Aware Schedulability (BRT = $8\mu s$)

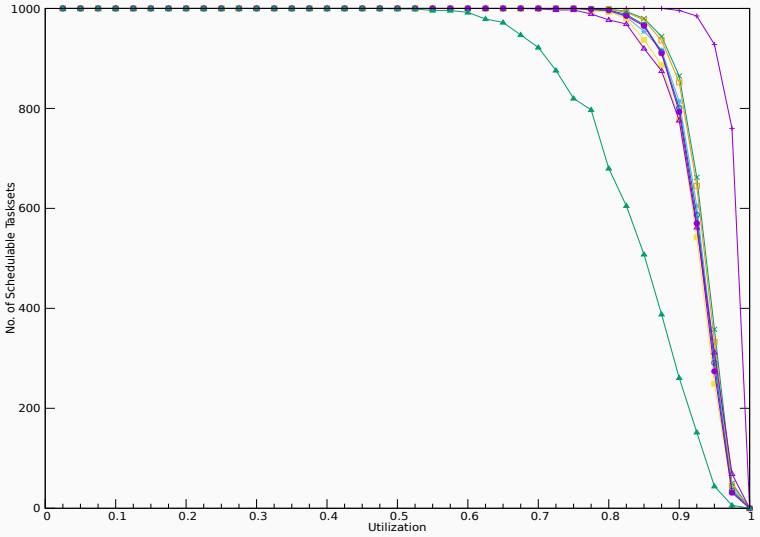


CRPD-Aware Schedulability (BRT = $0.2\mu s$)



CRPD-Aware Schedulability





- CRPD is non-trivial share of the response time
- All bounding approaches have similar performance
- Both cases perform almost similarly

Conclusions

Preemptions are not free and CRPD must be well bounded

Problems facing experimental evaluation:

- Current models not representative of actual executions
- No way to generate such models

Possible solutions:

- Generate tasksets from analytical data
- Generate synthetic task characteristics that capture correlations

Thank You!

Taskset Generation

- Utilizations in range [0.025, 1] in steps of 0.025
- **1000** tasksets per utilization
- **10** tasks per taskset