Integrated Intra- and Inter-task Cache Analysis for Preemptive Multi-tasking Real-Time Systems

Yudong Tan[1] and Vincent J. Mooney III[2]

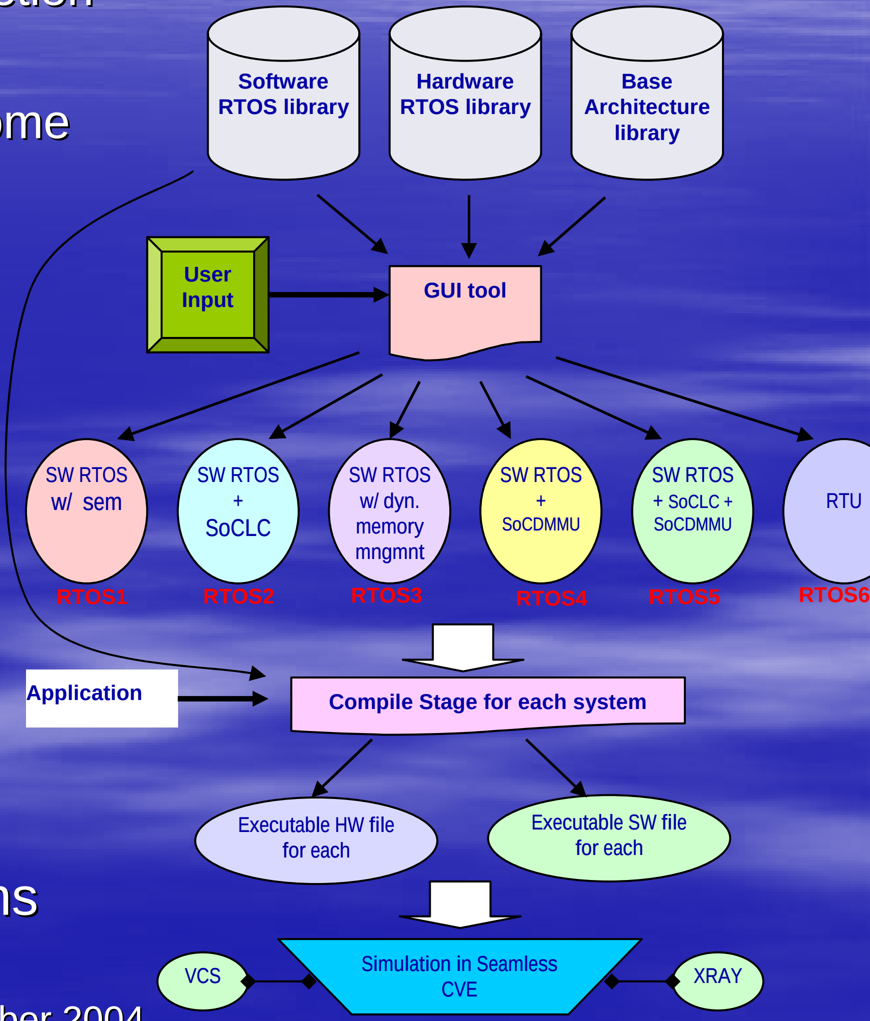
[2]Associate Professor, [1]School of Electrical and Computer Engineering
[2]Adjunct Associate Professor, College of Computing
[1,2]Center for Research on Embedded Systems and Technology (CREST)
Georgia Institute of Technology
Atlanta, Georgia USA
www.crest.gatech.edu

OUTLINE

Problem Statement

- Motivation
  - Task deadlines in real-time systems
    - ✓ Worst Case Response Time (WCRT): time from arrival of a task to its completion
  - HW vs. SW
    - ✓ Timing of HW is more predictable – some SW functions transferred to HW
    - ✓ SW timing analysis is unavoidable
  - Caches complicate timing analysis
- Assumptions
  - Multi-tasking, uniprocessor
  - Fixed Priority Scheduling (e.g., RMS)
  - Preemptive
  - L1 Cache (inc. set associative)
- Objective
  - WCRT estimate
  - Cache reload costs due to preemptions
  - Schedulability analysis

The δ Hardware/Software RTOS Generation Framework


Software RTOS library
Hardware RTOS library
Base Architecture library
User Input
GUI tool
SW RTOS w/ sem
RTOS1
SW RTOS + SoCLC
RTOS2
SW RTOS w/ dyn. memory mngmnt
RTOS3
SW RTOS + SoCDMMU
RTOS4
SW RTOS + SoCLC + SoCDMMU
RTOS5
RTU
RTOS6
Application
Compile Stage for each system
Executable HW file for each
Executable SW file for each
VCS
Simulation in Seamless CVE
XRAY

Previous Work

- Methods for timing analysis
  - Limit cache usage
  - Static timing analysis
  - Monitor

Previous Work: Limit Cache Usage

- Limit cache usage
  - Hardware approaches
    - ✓ SMART (Strategic Memory Allocation for Real-Time Systems) Cache [Kirk]
      - ✓ Assign cache lines to tasks according to their CPU utilization
    - ✓ Column Cache [Suh and Rudolph]
      - ✓ Cache is partitioned at the granularity of cache columns
      - ✓ No cache evictions among tasks
    - ✓ Data-Replace Controlled Cache [Maki et al.]
      - ✓ Specific instructions are used to lock cache lines in order to prevent cache eviction
  - Software Approaches
    - ✓ OS-Controlled Cache Predictability [Liedtke]
    - ✓ Compiler Support for Software-based Cache Partitioning [Mueller]
  - Need customized hardware or specific OS/compliers

Previous Work: Static Timing Analysis

- Static Analysis of Single Task Worst Case Execution Time (WCET)
  - Integer Linear Programming (ILP), Cinderella, [Li and Malik]
    - ✓ Implicit Path Enumeration with ILP
    - ✓ WCET analysis at the granularity of basic blocks
  - SYMTA (SYMbolic Timing Analysis), [Wolf and Ernst]
    - ✓ Extend basic blocks to program segments
    - ✓ Reduce over-estimate of WCET on boundaries of basic blocks
  - Cluster calculation of WCET [Ermerahl]
    - ✓ Reduce over-estimate of WCET on boundaries of basic blocks
  - Symbolic Analysis Methods with abstract interpretation, [Wilhelm],[Stenstrom]
    - ✓ Analyze WCET without knowing exact input data
  - Only consider single task systems
    - ✓ Preemption in multi-tasking systems
    - ✓ Context switch cost

Previous Work: Static Timing Analysis

- Worst Case Response Time (WCRT) Analysis for Multi-tasking Systems
  - WCRT [Tindell]
    - ✓ Cache not considered
  - Busquests-Mataix's Method [Busquests-Mataix et al.]
    - ✓ Preemption-related cache reload cost overestimated: all cache lines used by the preempting task have to be reloaded.
  - Lee's Approach [Lee et al.]
    - ✓ Useful memory blocks: used before the preemption and requested after the preemption by the preempted task
    - ✓ ILP
    - ✓ Exponential w.r.t. the number of tasks
    - ✓ May include infeasible preemptions
    - ✓ No inter-task cache eviction analysis method proposed

Previous Work: Monitor

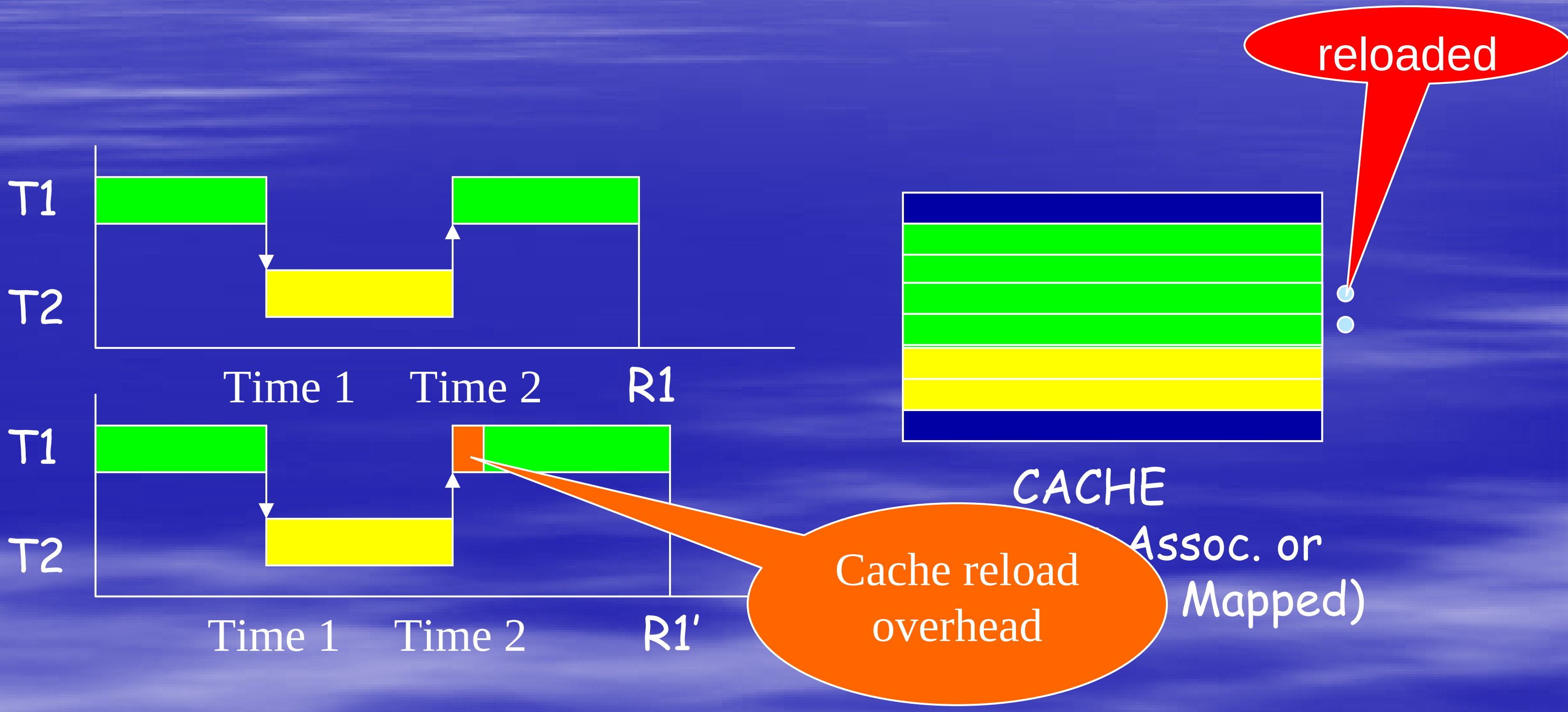
- Monitor
  - MAMon (Multi-processor Application Monitor) [Shobaki]
  - Performance Monitor (PM) in IBM PowerPC 604
  - Disadvantages:
    - ✓Need additional hardware
    - ✓Possibly need to insert instrumental instructions in software
    - ✓Dependent on the execution path, no guarantee on obtaining the worst case execution/response time

Preemption-related Cache Reload Cost

- Why do cache lines need to be reloaded?
  - Inter-task cache eviction
  - Intra-task cache dependency
- Assumptions
  - Only two levels of memory hierarchy

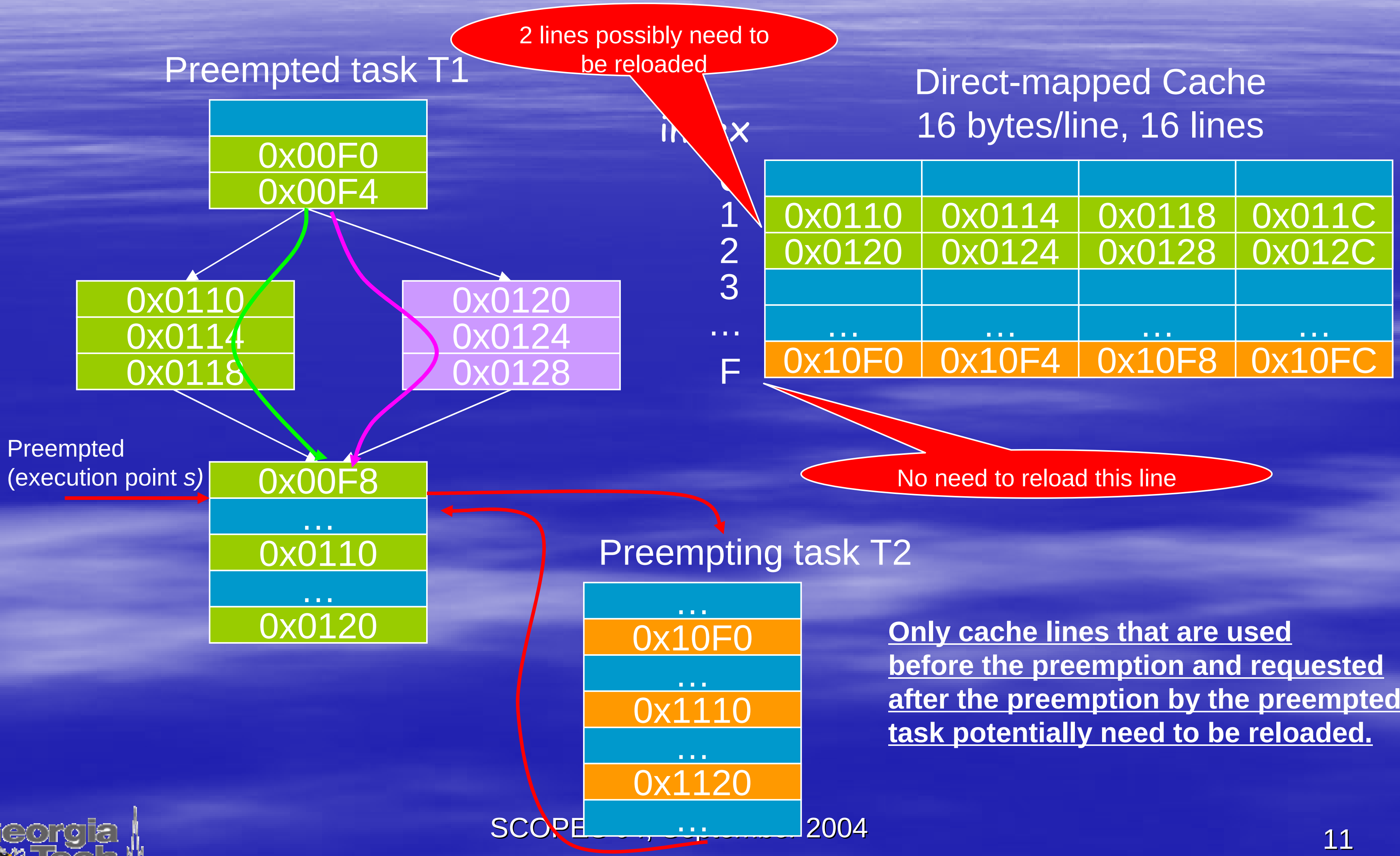
Inter-task Cache Eviction

- Two Tasks: T1 and T2
- T2 has a higher priority than T1



- **<u>Only cache lines used by both the preempting and the preempted task possibly need to be reloaded.</u>**

Intra-task Cache Dependency



2 lines possibly need to be reloaded
Preempted task T1
0x00F0
0x00F4
0x0110
0x0114
0x0118
0x0120
0x0124
0x0128
Preempted (execution point *s*)
0x00F8
…
0x0110
…
0x0120
Direct-mapped Cache 16 bytes/line, 16 lines
index
1
2
3
…
F
0x0110 0x0114 0x0118 0x011C
0x0120 0x0124 0x0128 0x012C
… … … …
0x10F0 0x10F4 0x10F8 0x10FC
No need to reload this line
Preempting task T2
…
0x10F0
…
0x1110
…
0x1120
…

Only cache lines that are used before the preemption and requested after the preemption by the preempted task potentially need to be reloaded.

Two Conditions for Cache Reload After a Preemption

- Used by both the preempting and the preempted task
  - Only cache lines in the intersection set of cache lines used by the preempting task and the preempted task
- Loaded to the cache before the preemption and requested after the preemption by the preempted task
  - Only cache lines mapped from “useful memory blocks”
    - ✓ Lee’s approach [Lee et al.]

Inter-task Cache Eviction Analysis

- **Memory Trace (No dynamic memory allocation)**
- **Cache Index Induced Partition (CIIP)**
  - Partition a memory block set according to the index of each memory block
  - Memory blocks in the same partition have the same index
  - Cache eviction can only happen among memory blocks in the same partition

Given an *L*-way set associative cache with *N* sets and a memory trace with *K+1* addresses

$$M = \{m_0, m_1, \ldots, m_K\}$$

CIIP of M: $\hat{M} = \{\hat{m}_0, \hat{m}_1, \ldots, \hat{m}_{N-1}\}$

Where $\hat{m}_i = \{m_j \in M \mid idx(m_j) = i\}$

Inter-task Cache Eviction Analysis (Cont.)

- Use CIIP to estimate the upper bound of inter-task cache eviction cost

$$M_1 = \{m_{10}, m_{11}, \ldots, m_{1K_1}\} \quad \hat{M}_1 = \{\hat{m}_{10}, \hat{m}_{11}, \ldots, \hat{m}_{1,N-1}\}$$

$$M_2 = \{m_{20}, m_{21}, \ldots, m_{2K_2}\} \quad \hat{M}_2 = \{\hat{m}_{20}, \hat{m}_{21}, \ldots, \hat{m}_{2,N-1}\}$$

Upper bound of the number of memory blocks that possibly conflict in the cache:

$$S(M_1, M_2) = \sum_{r=0}^{N-1} \min(L, |\hat{m}_{1r}|, |\hat{m}_{2r}|)$$

L is the number of ways in the cache

When the cache miss penalty is fixed, the inter-task cache eviction cost (i.e., the *cost* of *preemption*):

$$C_{pre}(T_1, T_2) = S(M_1, M_2) \times C_{miss}$$

- Path analysis can be applied to tighten the estimate of inter-task cache eviction cost

Y. Tan and V. Mooney, "Timing Analysis for Preemptive Multi-tasking Real-time Systems with Caches," ***Proceedings of Design, Automation and Test in Europe (DATE'04)*****, pp. 1034-1039, February 2004.**

Inter-task Cache Eviction Analysis (Cont.)

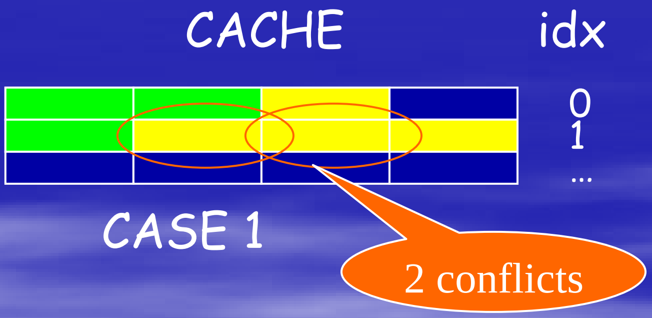
- An example

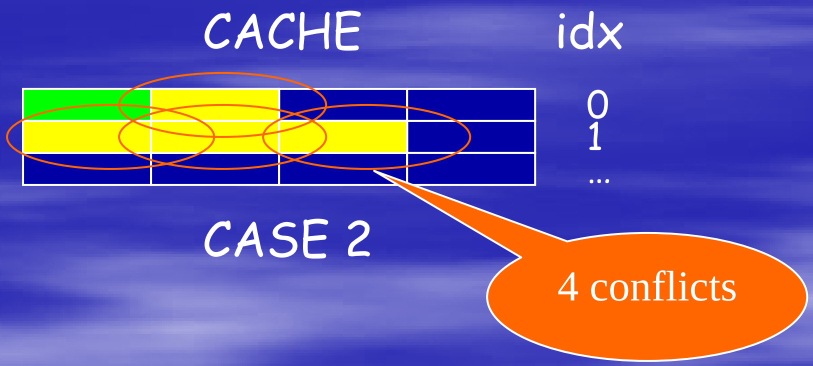
A 4-way SA cache with 16 sets, each line has 16 bytes

Two Sets of Memory Blocks:

$M_1 = \{0x700; 0x800; 0x710; 0x810; 0x910\}$ $\hat{M}_1 = \{\hat{m}_{10}, \hat{m}_{11}\} = \{\{0x700; 0x800\}, \{0x710; 0x810; 0x910\}\}$

$M_2 = \{0x200; 0x310; 0x410; 0x510\}$ $\hat{M}_2 = \{\hat{m}_{20}, \hat{m}_{21}\} = \{\{0x200\}, \{0x310; 0x410; 0x510\}\}$





$S(M_1, M_2) = 1 + 3 = 4$ -- gives an upper bound

Intra-task Cache Dependency Analysis

- Reaching Memory Blocks (RMB)
  - all possible memory blocks that may reside in the cache when the task reaches an execution point *s*
- Living Memory Blocks (LMB)
  - all possible memory blocks that may be one of the first *L* distinct references* to the cache set *after* execution point *s*, where *L* is the number of ways in the cache
- Useful Memory Blocks (UMB) at an execution point *s*
  - intersection of RMB and LMB at the execution point *s*
- Maximum Useful Memory Block Set (MUMBS)
  - the maximum intersection set of LMB and RMB over all the execution points of a task

* By "distinct reference" we mean a memory block that ia mapped to the same cache set but with a different address. Here we assume that LRU is used in the set associative cache.

Integrate Inter- and Intra-task Cache Timing Analysis

- Only useful memory blocks are potentially required to be reloaded
- *MUMBS* of the preempted task is used in the CIIP calculation

$$M_1 = \{m_{10}, m_{11}, \ldots, m_{1K_1}\} \qquad \hat{M}_1 = \{\hat{m}_{10}, \hat{m}_{11}, \ldots, \hat{m}_{1,N-1}\}$$

$$M_2 = \{m_{20}, m_{21}, \ldots, m_{2K_2}\} \qquad \hat{M}_2 = \{\hat{m}_{20}, \hat{m}_{21}, \ldots, \hat{m}_{2,N-1}\}$$

MUMBS $\tilde{M}_2 = \{\tilde{m}_{20}, \tilde{m}_{21}, \ldots, \tilde{m}_{2K_2}\} \qquad \hat{\tilde{M}}_2 = \{\hat{\tilde{m}}_{20}, \hat{\tilde{m}}_{21}, \ldots, \hat{\tilde{m}}_{2,N-1}\}$

Without considering UMBs $S(M_1, M_2) = \sum_{r=0}^{N-1} \min(L, |\hat{m}_{1r}|, |\hat{m}_{2r}|)$

With considering UMBs $S(M_1, \tilde{M}_2) = \sum_{r=0}^{N-1} \min(L, |\hat{m}_{1r}|, |\hat{\tilde{m}}_{2r}|)$

Note that $\left|\hat{\tilde{M}}_2\right| \le \left|\hat{M}_2\right| \Rightarrow \left|S(M_1, \tilde{M}_2)\right| \le \left|S(M_1, M_2)\right|$

WCRT Analysis

- WCRT Analysis without considering cache

T_i All tasks in the system sorted in the descending order of their priorities

C_i WCET of T_i P_i Period of T_i , also defines the deadline

$hp(i)$ The set of tasks with higher priorities than T_i

Response time

$$R_i^k = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i^{k-1}}{P_j} \right\rceil \times C_j$$

WCRT Analysis (Cont.)

- WCRT with Cache
  - Iterative calculation

$$R_i^0 = C_i;$$

$$R_i^1 = C_i + \sum_{j \in hp(i)} \lceil \frac{R_i^0}{P_j} \rceil \times (C_j + C_{pre}(T_i, T_j) + 2C_{cs})$$

...

$$R_i^k = C_i + \sum_{j \in hp(i)} \lceil \frac{R_i^{k-1}}{P_j} \rceil \times (C_j + C_{pre}(T_i, T_j) + 2C_{cs})$$

Twice Context Switch: one for preemption and one for resuming

RMS is used for scheduling.

Schedulability

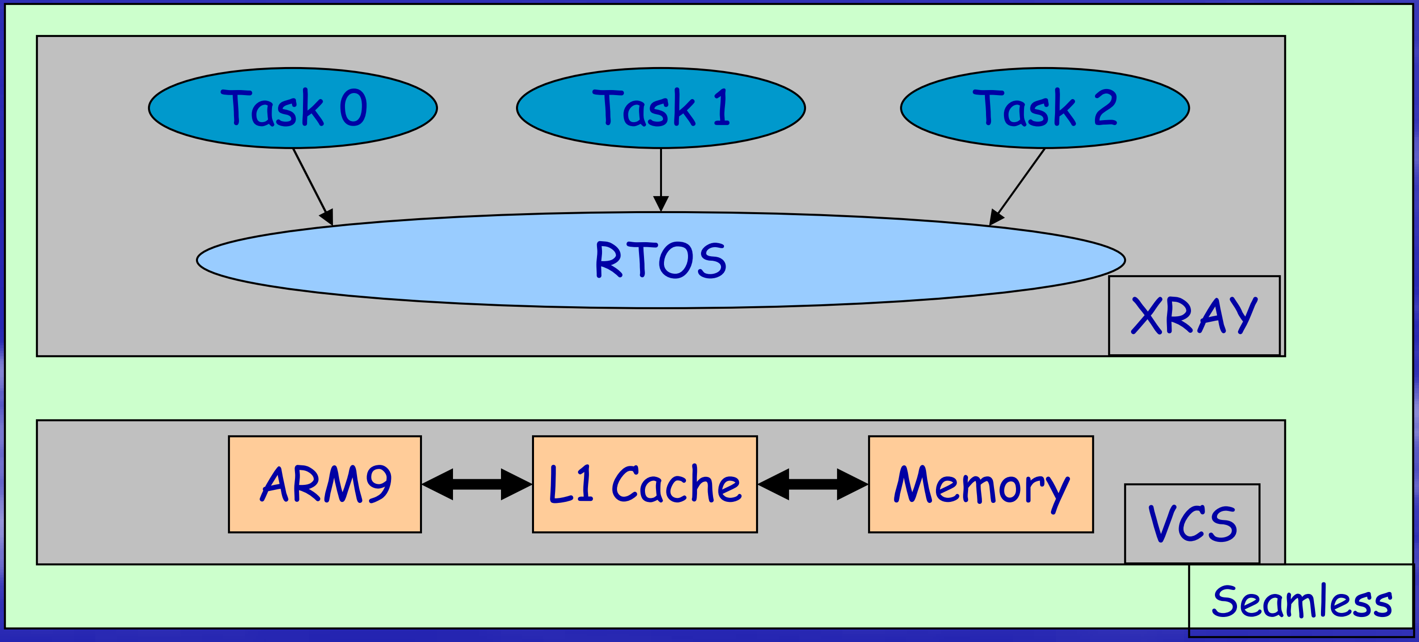
- The tasks are schedulable if both
  - the iteration above converges and
  - the WCRT of all tasks are less than their periods
- Otherwise, our method does not find a feasible schedule for the tasks
  - no guarantee that no feasible schedule exists; only know that we did not find a feasible schedule

WCRT Analysis (Cont.)

- WCRT estimated for each task in the descending order of priorities of tasks
- Computational Complexity
  - The number of iterations for each task is bounded by P_i / P_0
  - The computational complexity in each iteration is proportional to the number of tasks
  - All tasks except the task with the highest priority need to be estimated
  - The total computation complexity is $O(n^2)$, where *n* is the number of tasks

Experiment

- Simulation Architecture
  - ARM9TDMI
  - 32KB 4-way set associative cache (256 lines in each way)
  - Atalanta RTOS developed at Georgia Tech
  - Seamless CVE for simulation

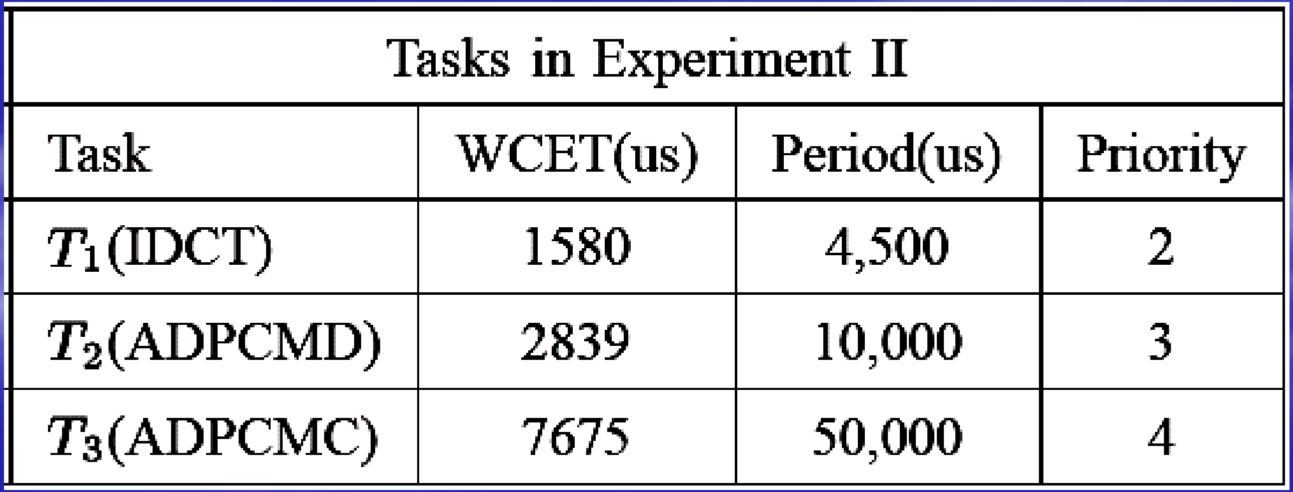


Experiment

- Five approaches
  - App1 (Busquests-Mataix's method): All cache lines used by preempting task are reloaded for a preemption.
  - App2: Only lines in the intersection set of lines used by the preempting task and the preempted task are reloaded after a preemption. Inter-task cache eviction method proposed in this paper is used here.
  - App3: All useful memory blocks are reloaded to the cache. (Lee's Approach)
  - App4: Integrated Inter- and Intra-task cache eviction analysis. No path analysis is applied here.
  - App5: Intra-task cache eviction analysis, inter-task cache eviction analysis plus path analysis.

Experiment I

- DSP application
  - Adaptive Differential Pulse Coding Modulation Coder (ADPCMC)
  - ADPCM Decoder (ADPCMD)
  - Inverse Discrete Cosine Transform (IDCT)



Tasks in Experiment II

Task	WCET(us)	Period(us)	Priority
T_1(IDCT)	1580	4,500	2
T_2(ADPCMD)	2839	10,000	3
T_3(ADPCMC)	7675	50,000	4

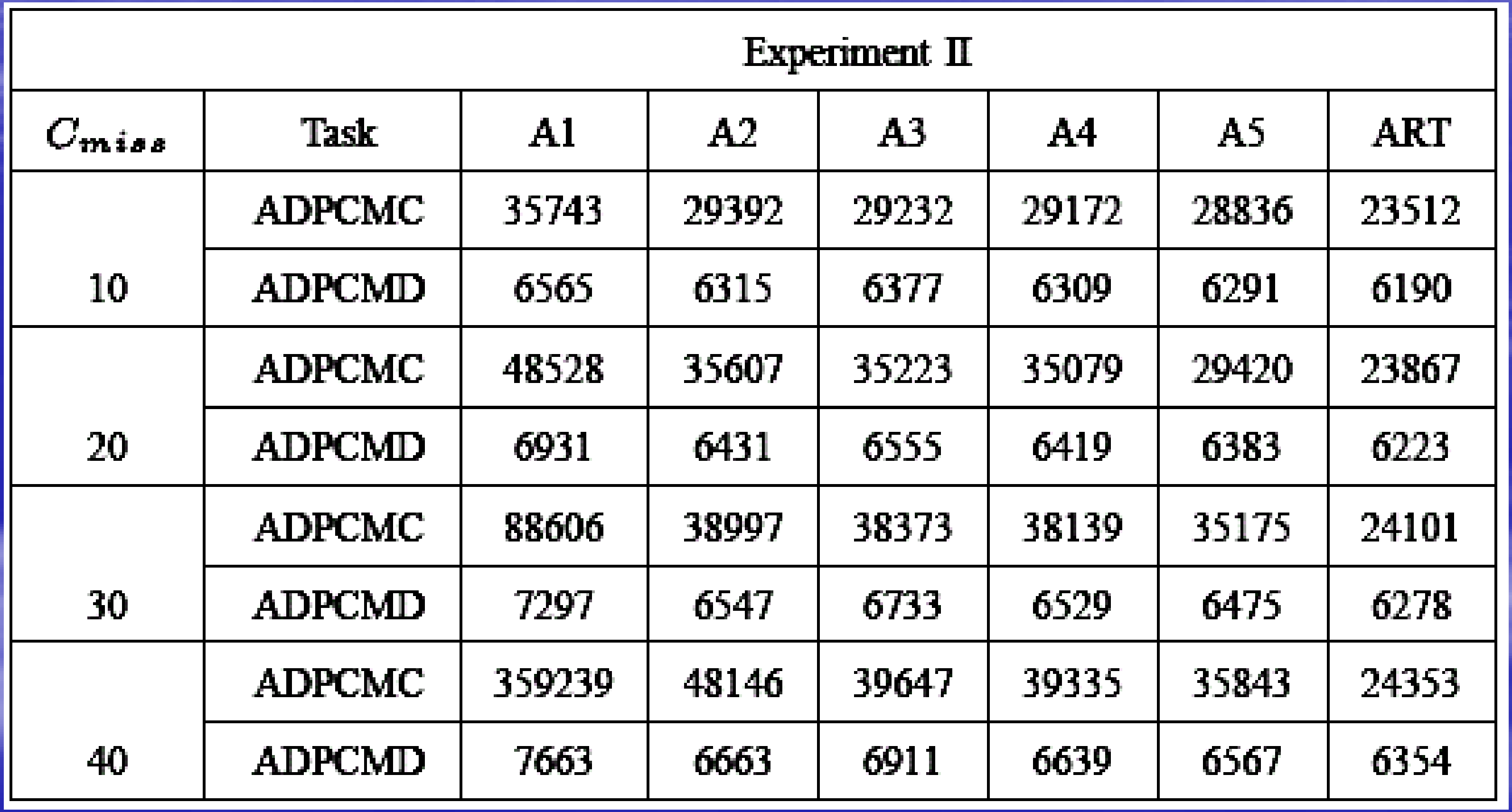
Results of Experiment I

- Three types of preemption
  - ADPCMD preempted by IDCT
  - ADPCMC preempted IDCT
  - ADPCMC preempted by ADPCMD
- Estimate of the number of cache lines to be reloaded

preemptions	App.1	App.2	App.3	App.4	App.5
ADPCMD by IDCT	249	68	98	64	56
ADPCMC by IDCT	220	114	98	92	64
ADPCMC by ADPCMD	183	58	89	55	46

Results of Experiment I

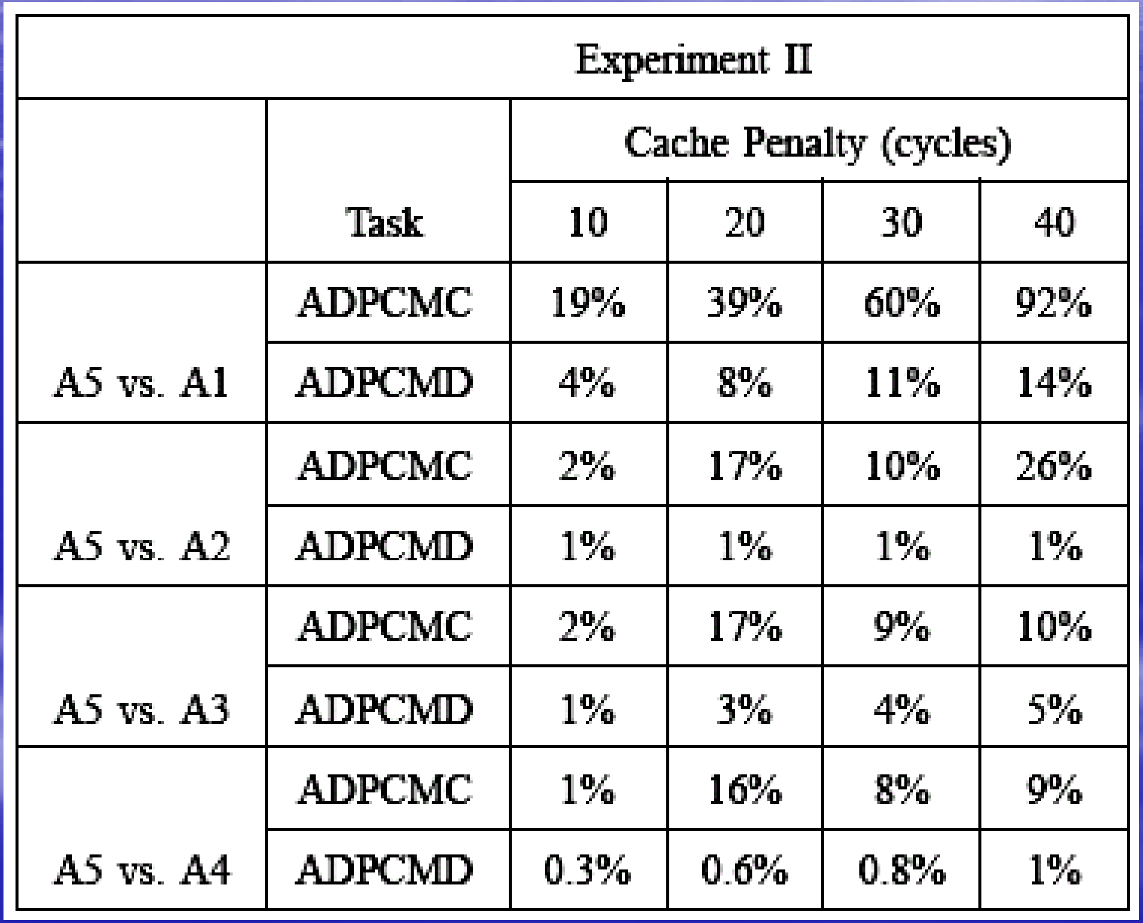
- WCRT estimates



Experiment II							
C_{miss}	Task	A1	A2	A3	A4	A5	ART
10	ADPCMC	35743	29392	29232	29172	28836	23512
	ADPCMD	6565	6315	6377	6309	6291	6190
20	ADPCMC	48528	35607	35223	35079	29420	23867
	ADPCMD	6931	6431	6555	6419	6383	6223
30	ADPCMC	88606	38997	38373	38139	35175	24101
	ADPCMD	7297	6547	6733	6529	6475	6278
40	ADPCMC	359239	48146	39647	39335	35843	24353
	ADPCMD	7663	6663	6911	6639	6567	6354

Results of Experiment I

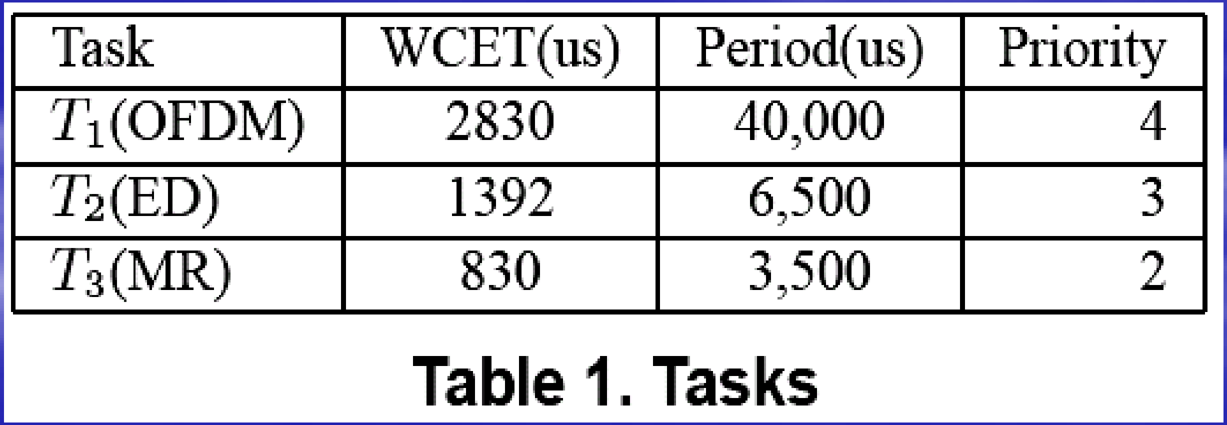
- Comparison of Approach 5 with other approaches



Experiment II					
		Cache Penalty (cycles)			
	Task	10	20	30	40
A5 vs. A1	ADPCMC	19%	39%	60%	92%
	ADPCMD	4%	8%	11%	14%
A5 vs. A2	ADPCMC	2%	17%	10%	26%
	ADPCMD	1%	1%	1%	1%
A5 vs. A3	ADPCMC	2%	17%	9%	10%
	ADPCMD	1%	3%	4%	5%
A5 vs. A4	ADPCMC	1%	16%	8%	9%
	ADPCMD	0.3%	0.6%	0.8%	1%

Experiment II

- A mobile robot application with three tasks
  - Edge Detection (ED)
  - Mobile Robot control (MR)
  - OFDM for communication



Task	WCET(us)	Period(us)	Priority
T_1(OFDM)	2830	40,000	4
T_2(ED)	1392	6,500	3
T_3(MR)	830	3,500	2

Table 1. Tasks

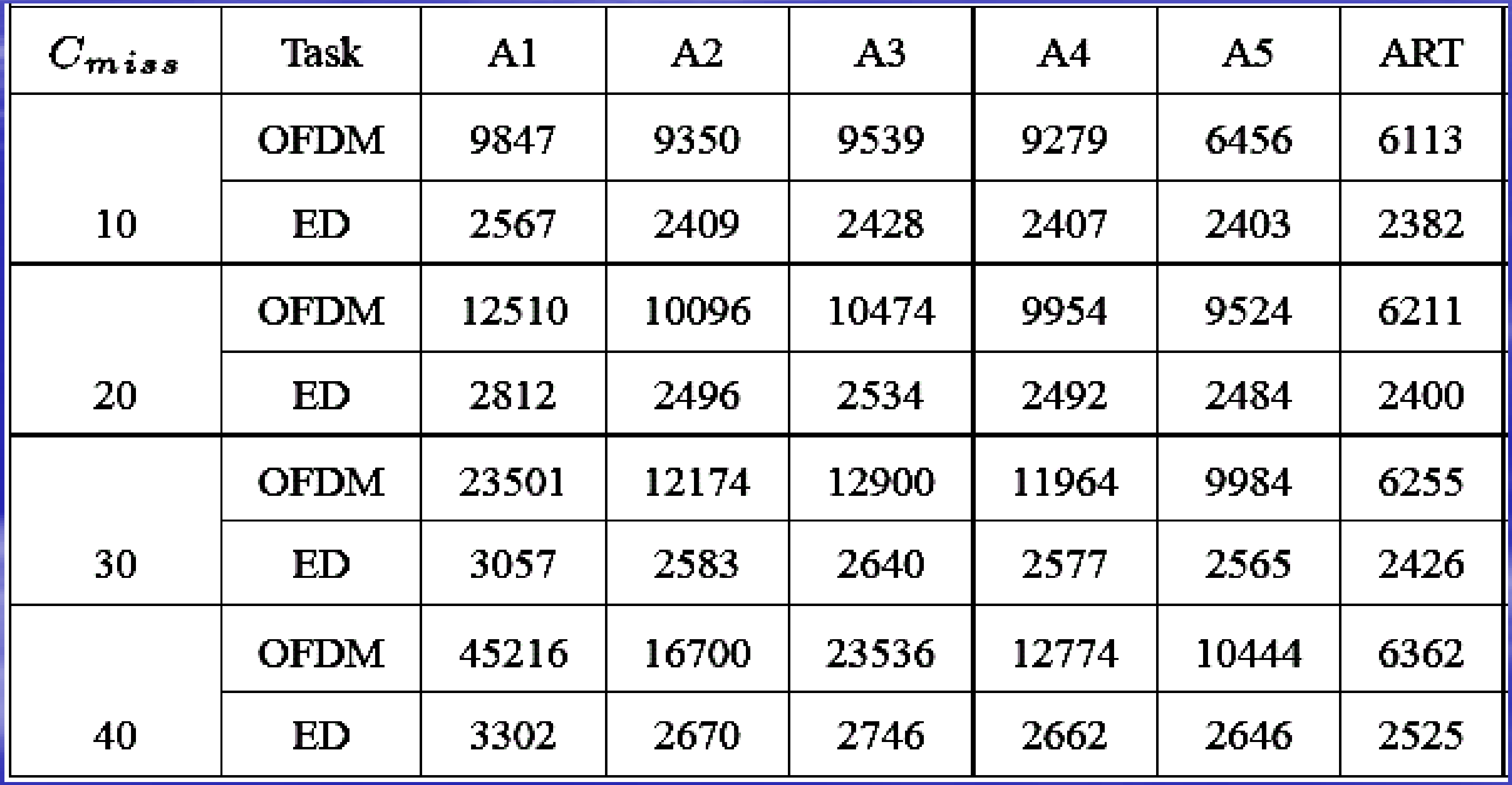
Results of Experiment II

- Three types of preemption
  - ED preempted by MR
  - OFDM preempted by MR
  - OFDM preempted by ED
- Estimate of the number of cache lines to be reloaded

preemptions	App.1	App.2	App.3	App.4	App.5
ED by MR	245	134	187	118	88
OFDM by MR	254	172	187	135	98
OFDM by ED	245	87	106	85	81

Results of Experiment II

- WCRT estimates

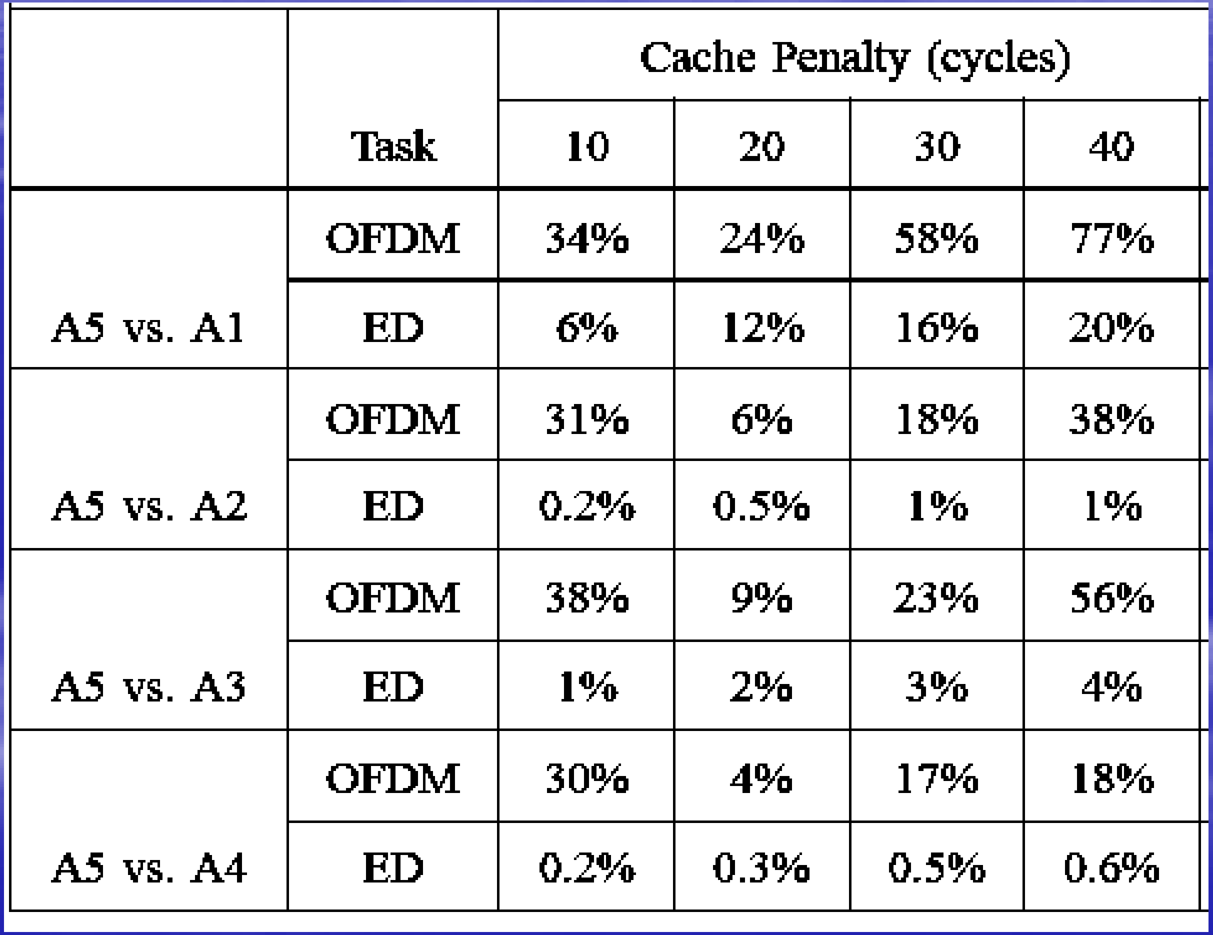


C_{miss}	Task	A1	A2	A3	A4	A5	ART
10	OFDM	9847	9350	9539	9279	6456	6113
	ED	2567	2409	2428	2407	2403	2382
20	OFDM	12510	10096	10474	9954	9524	6211
	ED	2812	2496	2534	2492	2484	2400
30	OFDM	23501	12174	12900	11964	9984	6255
	ED	3057	2583	2640	2577	2565	2426
40	OFDM	45216	16700	23536	12774	10444	6362
	ED	3302	2670	2746	2662	2646	2525

(ART: Actual Response Time)

Results of Experiment II

Improvement of Approach 5 over other approaches



		Cache Penalty (cycles)			
	Task	10	20	30	40
A5 vs. A1	OFDM	34%	24%	58%	77%
	ED	6%	12%	16%	20%
A5 vs. A2	OFDM	31%	6%	18%	38%
	ED	0.2%	0.5%	1%	1%
A5 vs. A3	OFDM	38%	9%	23%	56%
	ED	1%	2%	3%	4%
A5 vs. A4	OFDM	30%	4%	17%	18%
	ED	0.2%	0.3%	0.5%	0.6%

Experiment III

Six Tasks

Tasks	MR	IDCT	ED	ADPCMD	OFDM	ADPCMC
Period (cycles)	7000	9000	13000	20000	40000	50000
WCET (cycles)	830	1580	1392	2839	2830	7675
Priorities	2	3	4	5	6	7

WCRT estimates of OFDM and ADPCMC

	WCRT estimates of ADPCMC					WCRT estimates of OFDM				
C_{miss}	A1	A2	A3	A4	A5	A1	A2	A3	A4	A5
10	51572	34837	38091	34336	33781	16901	16217	16399	15948	15643
20	75585	58646	59576	51990	38235	25904	17531	17895	16993	16383
30	258814	75673	97381	69025	57496	50831	25756	32408	24697	17123
40	6837328	152023	233839	76729	68599	116464	33690	50843	31834	17863

Comparison Approach 5 with Approach 4:

	C_{miss}			
Task	10	20	30	40
ADPCMC	2%	27%	18%	11%
OFDM	2%	4%	31%	44%

Experiment IV

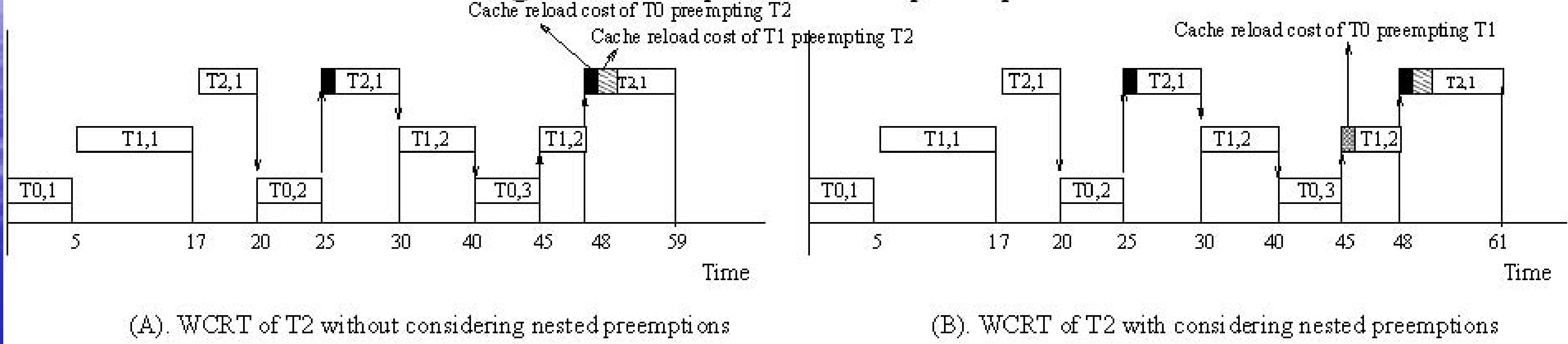
- Show the effects of infeasible preemptions in Lee's approach
  - Use the same tasks specified in Lee's experiments
  - Compute the WCRT with our WCRT estimate formula

Task	Period	WCET
FFT	$320,000$	$60,234 + 280 \times C_{miss}$
LUD	$1,120,000$	$255,998 + 364 \times C_{miss}$
LMS	$1,920,000$	$365,893 + 474 \times C_{miss}$
FIR	$25,600,000$	$557,589 + 405 \times C_{miss}$

Cache miss penalty = 100 cycles (used in Lee's experiment)
WCRT of FIR with Lee's Approach = 5,323,620 cycles
WCRT of FIR with our approach (Approach 5) = 3,297,383 cycles
Reduction in WCRT estimate = 38%

Inaccuracy With Nested Preemptions

Fig. 5. Example of nested preemptions



Cache reload cost of T0 preempting T2
Cache reload cost of T1 preempting T2
Cache reload cost of T0 preempting T1
T0,1
T1,1
T2,1
T0,2
T1,2
T0,3
5
17
20
25
30
40
45
48
59
61
Time

(A). WCRT of T2 without considering nested preemptions

(B). WCRT of T2 with considering nested preemptions

- T1 preempts T2
- T0 preempts T1
- => T0 affects cache reload cost for T2

Solution for Nested Preemptions

- The tasks whose priorities are lower than the preempting task and higher than the preempted tasks can possibly be preempted indirectly.
- The MUMBS of all these tasks, together with the MUMBS of the preempted task should be considered.
- The cache reload cost is calculated with the following formula.

$$C(T_a,T_b)=C_{miss}\times S\left(\bigcup_{l=b+1}^{a}\tilde{M}_l, M_b^k\right)=\sum_{r=0}^{N-1}\min\left\{\left|\bigcup_{l=b+1}^{a}\hat{\tilde{M}}_{l,r}^{k}\right|,\left|\hat{m}_{b,r}\right|,L\right\}$$

Note: By applying this new formula, the results of previous experiments are different from the results presented in this paper. For example, as reported in this paper, Approach 5 can achieve a reduction of up to 44% in WCRT estimate, if compared with Approach 4. However, Approach 5 can only achieve a reduction of up to 32%, as compared with Approach 4, by using the new formula. Please refer to the following technical report for the details of the new formula and the experimental results.

Y. Tan and V. Mooney, “Cache-related Timing Analysis for Multi-tasking Real-Time Systems with Nested Preemptions,” Technical Report, GIT-CC-04-11, Georgia Institute of Technology, October, 2004, http://www.cc.gatech.edu/tech_reports/index.04.html

Future Work

- Fix estimate accuracy for nested preemptions
- Investigate the factors that cause overestimate in WCRT analysis
- Investigate how cache parameters affect WCRT analysis
- Investigate combination with specialized cache hardware and/or specialized compilers

Conclusion

- Preemption-related cache reload cost is determined by both inter- and intra-task cache dependency
- The WCRT estimate is also affected by the estimate of the number of preemptions
- Integrating inter- and intra-task cache dependency analysis with path analysis can tighten the WCRT analysis significantly

Thank you!