

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

Yudong Tan and Vincent J Mooney III

Center for Research on Embedded Systems and Technology
(CREST)

School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia USA

OUTLINE

Problem Statement

Previous Work

Preemption-related Cache Reload Cost
Analysis

WCRT Estimation

Experimental Results

Conclusion

Future Work

Problem Statement

Motivation

Tasks in real-time systems need to meet deadlines.

Worst Case Response Time (WCRT) analysis: The 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.

Using cache complicates the timing analysis problem.

Assumption

Multi-tasking, uniprocessor

Fixed Priority Scheduling (e.g., RMS)

Preemptive

L1 Cache (Set Associative / Direct Mapped)

Objective

WCRT estimate

Including cache reload cost due to preemptions

Schedulability analysis

SCOPES'04, September

Previous Work

Methods for timing analysis

- Limit cache usage

- Analyze the timing properties of tasks statically

- 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

Lock Cache [Maki]

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 [Muller]

Need customized hardware or specific

OS/compilers

SCOPES 04, September

2004

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, [Wilhem],[Stenstrom]

- Analyze WCET without knowing exact input data

Only consider single task systems

- Preemption in multi-tasking systems

Previous Work: Static Timing

Analysis

Worst Case Response Time (WCRT) Analysis for Multi-tasking Systems

WCRT [Tindell]

Cache not considered

Busquests-Mataix's Method

Preemption-related cache reload cost overestimated: all cache lines used by the preempting task have to be reloaded.

Lee's Approach

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?

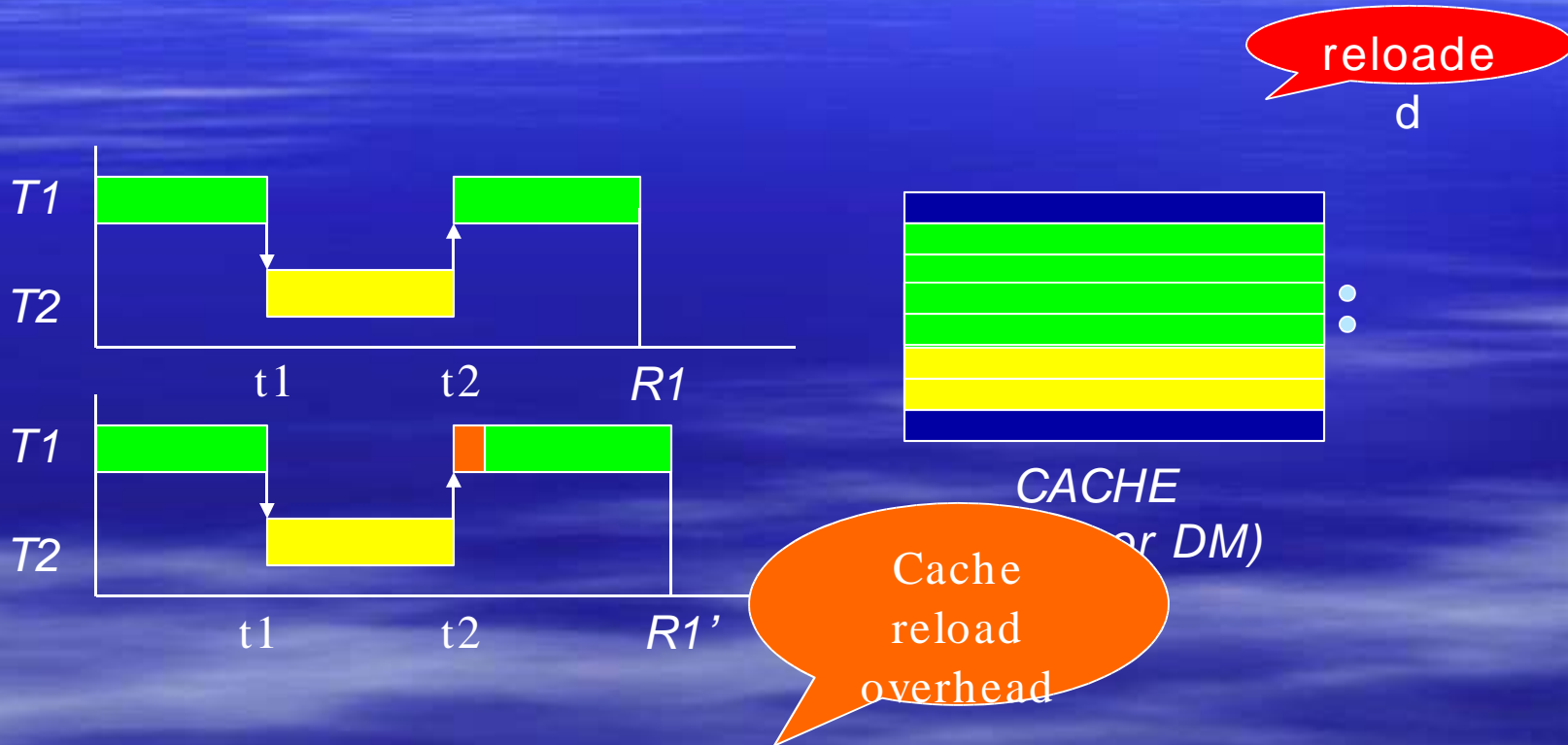
Inter-task cache eviction

Intra-task cache dependency

Inter-task cache eviction

Two Tasks: T1 and T2

T2 has a higher priority than T1

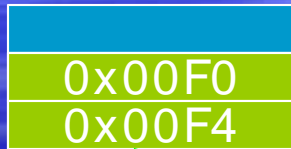


Only cache lines used by both the preempting and the preempted task possibly need to be reloaded.

SCOPES'04, September
2004

Intra-task Cache dependency

Preempted task T1

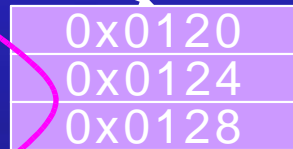


2 lines possibly
need to be
reloaded

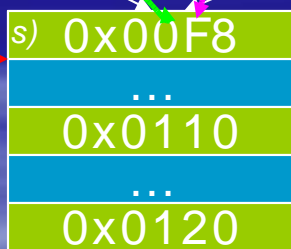
index

Direct-mapped Cache
16 bytes/line, 16 lines

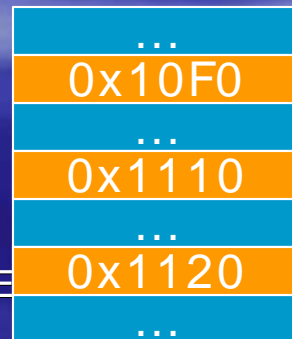
0				
1	0x0110	0x0114	0x0118	0x011C
2	0x0120	0x0124	0x0128	0x012C
3				
...
F	0x10F0	0x10F4	0x10F8	0x10FC



Preempted
(execution point)



Preempting task T2



No need to reload this
line

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

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”.

Inter-task Cache Eviction Analysis

Memory Trace (No dynamic memory allocation)

Cache Index Induced Partition (CIIP)

Partition a memory block set according to their index
Memory blocks in the same partition have the same index.

Cache eviction can only happen among memory blocks in the same partition.
An associative cache with N sets.

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

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

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

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

SCOPES'04, September
2004

Inter-task Cache Eviction Analysis (Cont.)

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

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

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

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

$$S(M_1, M_2) = \sum_{r=0}^{N-1} \min(L, |\hat{m}_{1r}|, |\hat{m}_{2r}|) \quad L - \text{the number of ways in the cache}$$

When the cache miss penalty is fixed, the inter-task cache eviction cost:

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

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" *Proceedings of Design, Automation and Test in Europe (DATE'04)*, pp. 1034-1039, February 2004

Inter-task Cache Eviction Analysis (Cont.)

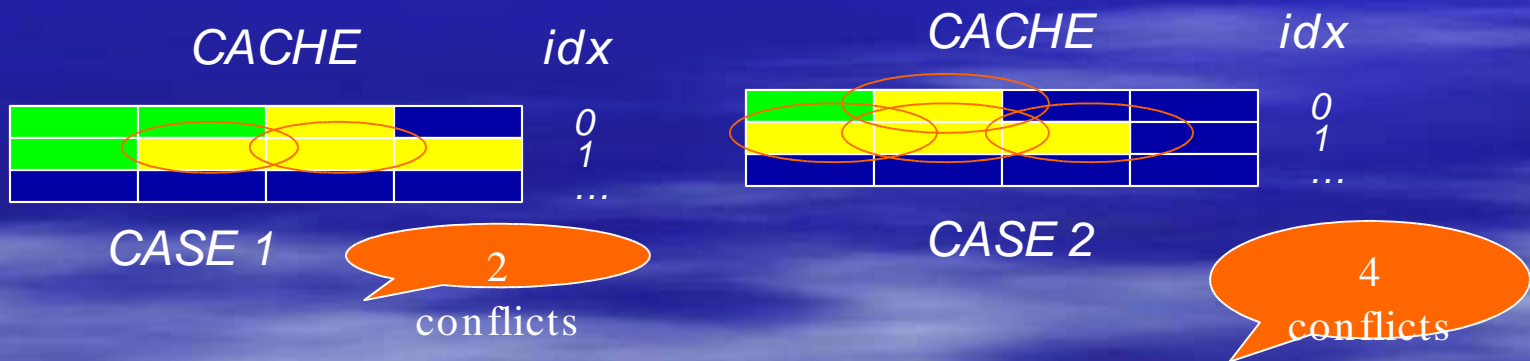
An example

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

Two Sets of Memory Blocks:

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

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



$$S(M_1, M_2) = 1 + 3 = 4 \quad \text{-- gives an upper bound}$$

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 *after* execution point s , where L is the number of ways in the cache.

Useful Memory Blocks (UMB) at the execution point s

Intersection of RMB and LMB at the execution point s

Maximum Useful Memory Block Set (MUMBS)

* By "distinct references" we mean the memory blocks that have the same index but different addresses. So these memory blocks are mapped to the same set in the cache. We also assume that LRU is used in the cache.

The maximum intersection set of LMB and RMB over all the execution points of a task

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

$$\begin{array}{ll}
 M_1 = \{m_{10}, m_{11}, \dots, m_{1K_1}\} & \hat{M}_1 = \{\hat{m}_{10}, \hat{m}_{11}, \dots, \hat{m}_{1,N-1}\} \\
 M_2 = \{m_{20}, m_{21}, \dots, m_{2K_2}\} & \hat{M}_2 = \{\hat{m}_{20}, \hat{m}_{21}, \dots, \hat{m}_{2,N-1}\} \\
 \text{MUMBS } \tilde{M}_2 = \{\tilde{m}_{20}, \tilde{m}_{21}, \dots, \tilde{m}_{2K_2}\} & \hat{\tilde{M}}_2 = \{\hat{\tilde{m}}_{20}, \hat{\tilde{m}}_{21}, \dots, \hat{\tilde{m}}_{2,N-1}\}
 \end{array}$$

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

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

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:

The iteration above converges.

The WCRT of all tasks are less than their periods.

Otherwise, we cannot 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_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. $O(n^2)$

The total computation complexity is $O(n^2)$, where n is the number of tasks.

Experiment

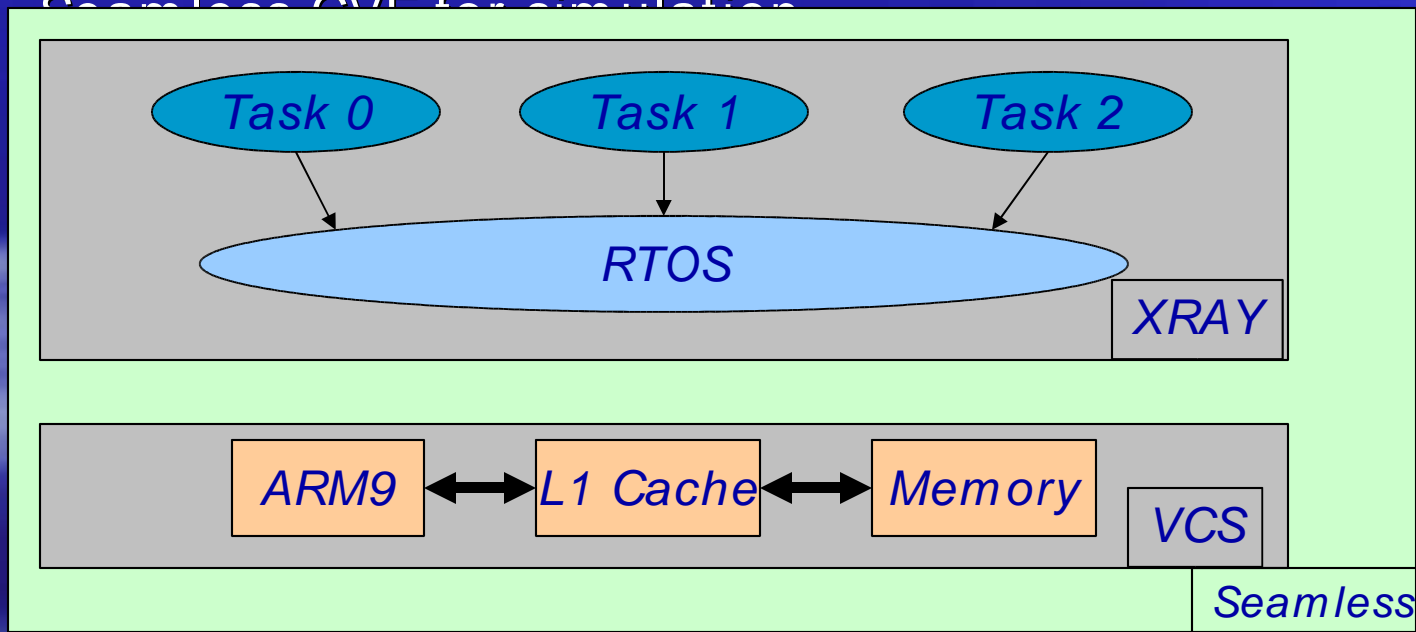
Simulation Architecture

ARM9TDMI

32KB 4-way set associative cache (256 lines in each way)

Atalanta RTOS developed at Georgia Tech

Complete QV5 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

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 I

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 I

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)

SCOPES'04, September
2004

Results of Experiment I

Improvement of Approach 5 over other approaches

	Task	Cache Penalty (cycles)			
		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%

SCOPES'04, September
2004

Experiment II

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

SCOPES'04, September
2004

Results of Experiment II

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 II

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 II

Comparison of Approach 5 with other approaches

Experiment II					
	Task	Cache Penalty (cycles)			
		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 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:

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

Experiment IV

Show the affects 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%

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.

SCOPES'04, September

2004

Future Work

Fix estimate accuracy for nested preemptions

Investigate the factors that cause overestimate in WCRT analysis

Investigate how cache parameters affect WCRT analysis

Thank you!

SCOPES'04, September
2004

