Cache-related Timing Analysis for Preemptive Multi-tasking Real-Time Systems

Yudong Tan, Vincent J. Mooney III

Center for Research in Embedded Systems and Technology
School of Electrical and Computer Engineering
Georgia Institute of Technology
DATE'04 Feb. 2004

OUTLINE

- Previous Work
- Problem Statement
- Inter-task cache eviction analysis
- Path Analysis
- WCRT estimation
- Experiment
- Conclusion

Previous Work

- Cache-related WCET analysis
  - Customized Hardware/Software
    - SMART [Krik]
    - Column Cache
  - Static Analysis
    - ILP, Cinderrela, [Li and Malik]
    - SYMTA, [Wolf and Ernst]
    - Symbolic Analysis Methods, [Wilhem],[Stenstrom]
- WCRT Analysis for Multi-tasking Systems
  - WCRT, [Tindell]
  - Busquests-Mataix's Method
    - Cache Eviction Cost: all cache lines used by the preempting task.
  - Lee's Approach
    - Useful memory blocks

Problem Statement

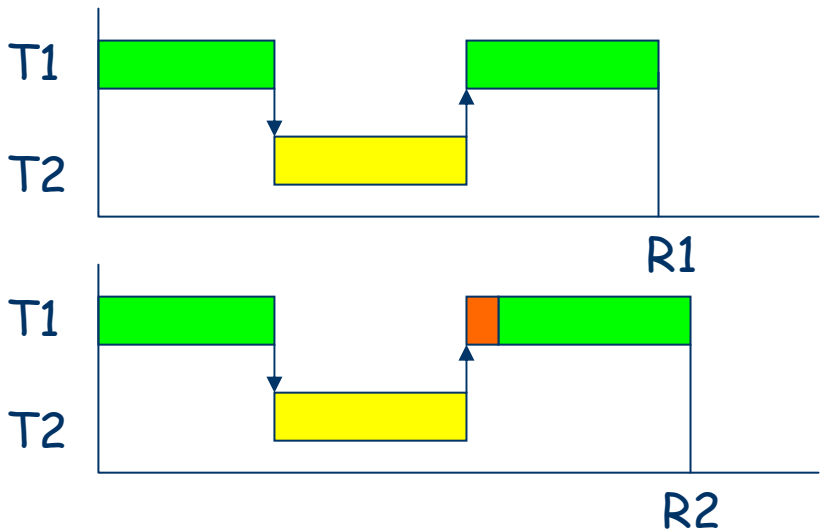
- Assumption
  - Multi-tasking
  - Fixed Priority Scheduling (e.g., RMS)
  - Preemptive
  - L1 Cache (Set Associative)
- Objective
  - WCRT estimate
  - Cache eviction caused by preemptions included
  - Schedulability

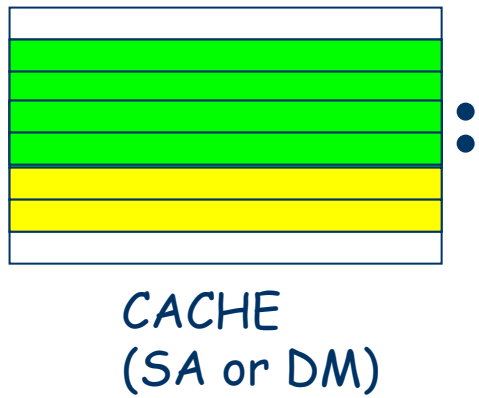
Cache eviction due to preemptions

- Cache lines used by the preempted task are evicted by the preempting task during the preemption. (Inter task cache eviction)
- The evicted cache lines are requested by the preempted again after the preemption (i.e., after the preempted task resumes)

Inter-task Cache Eviction

- Two Tasks: T1 and T2
- T2 has a higher priority than T1
- Cache reloading cost
  - Two cache lines need to be reloaded by T1 after preemption.
  - The response time of T1 is extended.
  - **Only cache lines used by both the preempting and the preempted task need to be considered.**



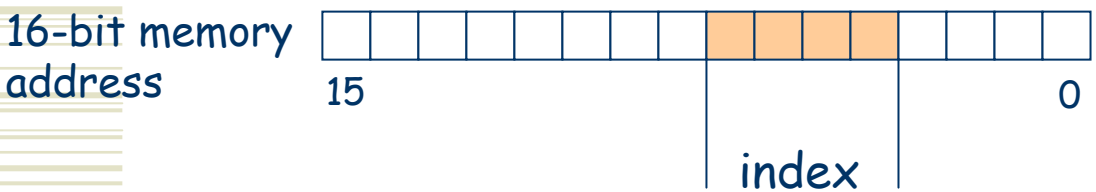


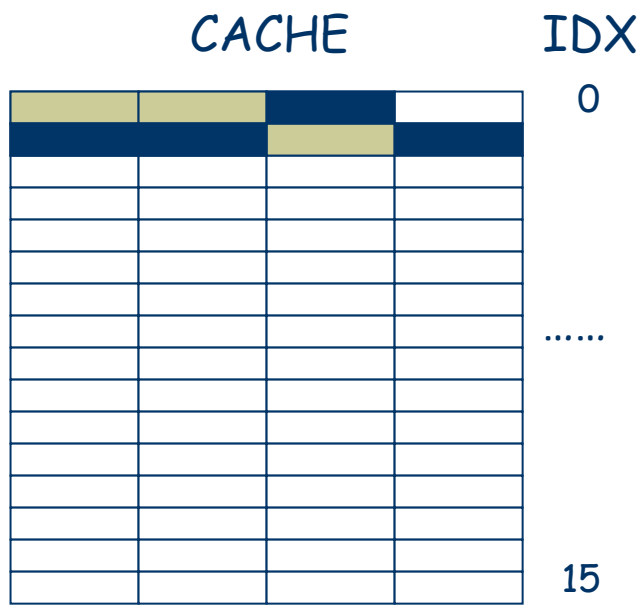
Inter-task Cache Eviction

- Memory trace (No dynamic data allocation)
- Memory vs. Cache
  - Index of a memory block vs. cache line
  - Only memory blocks with the same index can possibly conflict in the cache.
  - An example

Two Sets of Memory Blocks:
M1={0x700; 0x800; 0x710; 0x810; 0x910}
M2={0x200; 0x310; 0x410; 0x510}

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





Inter-task Cache Eviction (Cont.)

- 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 *L*-way set associative cache with *N* lines in each set.

CIIP of M:

$$M = \{m_0, m_1, ..., m_K\} \qquad \hat{M} = \{\hat{m}_0, \hat{m}_1, ..., \hat{m}_{N-1}\}$$

Where,

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

Inter-task Cache Eviction (Cont.)

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

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

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

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

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

Inter-task Cache Eviction (Cont.)

- An example

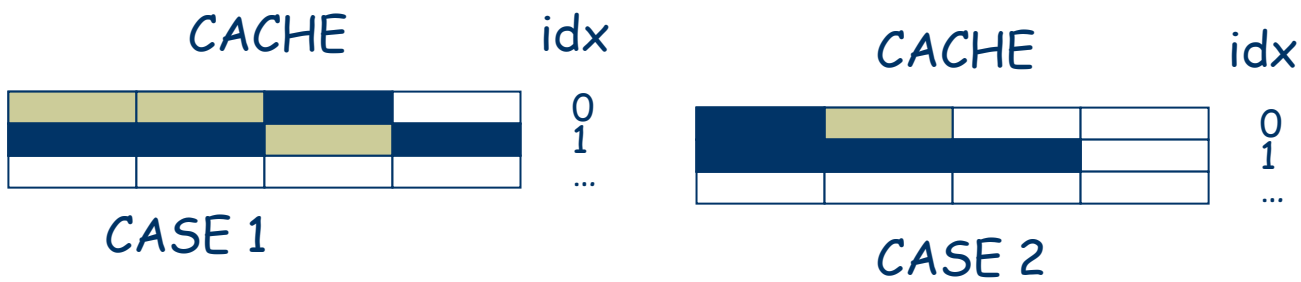
A 4-way SA cache with 16 sets, each line has16 bytes.

Two Sets of Memory Blocks:

$$M_1 = \{0x700; 0x800; 0x710; 0x810; 0x910\} \quad \hat{M}_1 = \{\hat{m}_0, \hat{m}_1\} = \{\{0x700; 0x800\}, \{0x710; 0x810; 0x910\}\}$$

$$M_2 = \{0x200; 0x310; 0x410; 0x510\} \quad \hat{M}_2 = \{\hat{m}_0, \hat{m}_1\} = \{\{0x200\}, \{0x310; 0x410; 0x510\}\}$$

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



Inter-task cache eviction (Cont.)

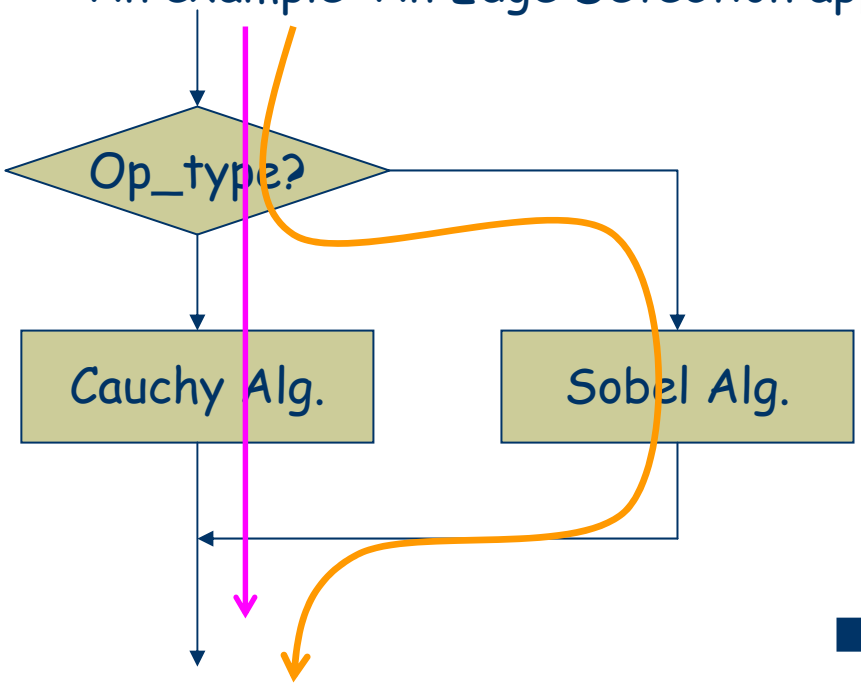
- Inter-task cache eviction cost
  - Fixed cache miss penalty
  - Two tasks T1 and T2, T2 has a higher priority than T1.
  - Memory trace M1 and M2

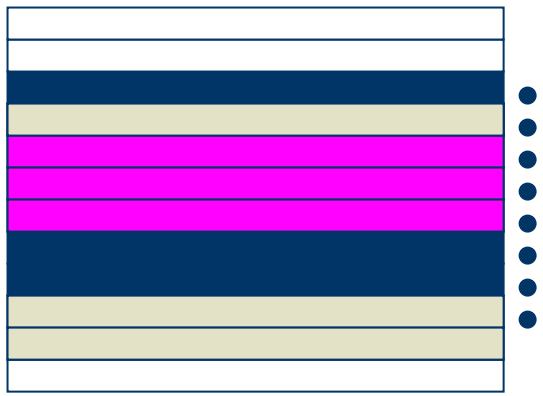
$$C_{pre}(T_1, T_2) = S(M_1, M_2) \times C_{miss}$$

Path Analysis

- A real application may have multiple feasible paths.

An example: An Edge Detection application with two algorithms.





Cache lines used by another task

Path 1: 4 lines overlapped

Path 2: 2 lines overlapped

No path analysis: 5 lines overlapped

Path Analysis (Cont.)

- Two tasks: T_i T_j

M_i The set of all memory blocks that can possibly be accessed by T_i .

T_j has multiple paths, Pa_i^k.

M_j^k The set of memory blocks accessed by T_j when it runs along path Pa_i^k

Cost of a path in T_j

$$C(Pa_j^k) = S(M_i, M_j^k) = \sum_{r=0}^{N-1} \min(L, |\hat{m}_i|, |\hat{m}_j^k|)$$

The problem is converted to find the longest path in T_j , $Pa_{longest}$

Currently, we search all possible paths in T_j

Path analysis (Cont.)

- We apply path analysis to the preempted task.
  - Two tasks T_i T_j
  - T_i has a higher priority than T_j

The cache reload overhead caused by T_i preempting T_j

$$C_{pre}(T_j, T_i) = C(Pa_{longest}) \times C_{miss}$$

WCRT Analysis

- WCRT Analysis without cache

T_i All tasks in the system.

C_i WCET of T_i P_i Period of T_i

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

R_i Response time of T_i

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

WCRT Analysis (Cont.)

- WCRT with Cache
  - Iterative calculation

$$R_i^0 = C_i;$$

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

...

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

Twice Context Switch: one for preemption and one for resuming

Schedulability

- The tasks are schedulable if:
  - The iteration above converges.
  - The WCRT of all tasks are less than their periods.

Experiment

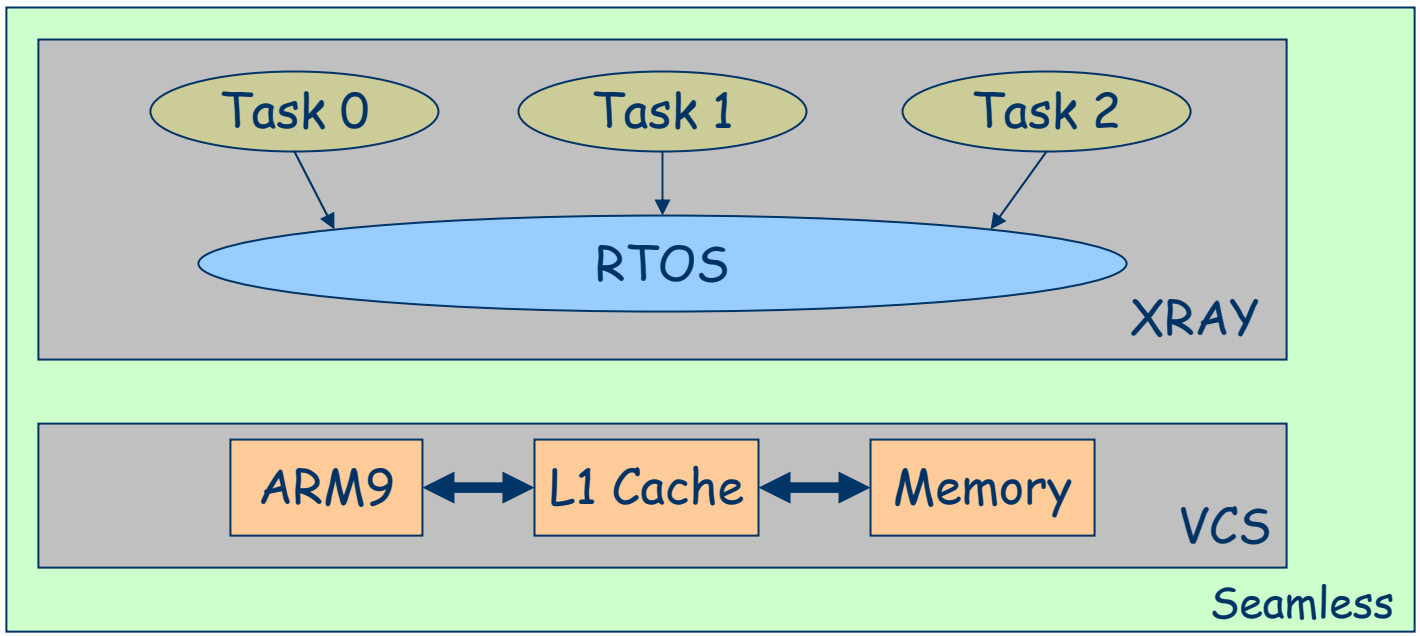
- A mobile robot application with three tasks
  - Edge Detection (ED)
  - Mobile Robot control (MR)
  - OFDM for communication

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

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

- Three approaches
  - App1 (Busquests-Mataix's method): All cache lines used by preempting tasks are reloaded for a preemption.
  - App2 (no path analysis): Only lines in the intersection set of lines used by the preempting task and the preempted task are reloaded for a preemption. Inter-task cache eviction method proposed in this paper is used here.
  - App3: Path analysis for the preempted task is added to Approach2.
- Three types of preemption
  - MR preempted by ED
  - MR preempted by OFDM
  - ED preempted by OFDM

Experiment Results

- Estimate of the number of cache lines to be reloaded

	App. 1	App. 2	App. 3
OFDM by MR	245	134	111
OFDM by ED	254	172	135
ED by MR	245	82	77

Table 2. Number of cache lines to be reloaded

Experiment Results

- WCRT estimates

C_{miss}(cycles)	Task	App. 1	App. 2	App. 3	ART
10	OFDM	9847	9350	9207	6113
	ED	2567	2404	2399	2382
20	OFDM	12510	10096	9810	6211
	ED	2812	2486	2476	2400
30	OFDM	23501	12174	10413	6255
	ED	3057	2568	2553	2426
40	OFDM	45216	16700	12390	6362
	ED	3302	2650	2630	2525

Table 3. Comparison of WCRT estimate

Improvement of App3 over App1:

Task	Cache Penalty (cycles)			
	10	20	30	40
OFDM	6.5%	21.6%	55.7%	73%
ED	6.5%	11.9%	16.5%	20.4%

Table 4. Comparison of results

Conclusions

- Inter-task cache eviction affects greatly WCRT estimates of tasks in a preemptive multi-tasking system.
- Applying path analysis can reduce the WCRT estimates further.

Future Work

- Consider integrating intra-task cache eviction analysis (Lee's work) in our approach.
- More than two level memory hierarchy.