Bringing the Multicore Revolution to Safety-Critical Cyber-Physical Systems

[1]University of North Carolina Chapel Hill [2]North Carolina State University
PIs: Dr. James Anderson[1] & Dr. Frank Mueller[2]
Students: Bryan Ward[1], Jonathan Herman[1], Namhoon Kim[1], Shrinivas Panchamukhi[2]

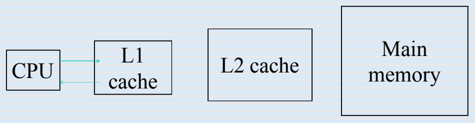
Motivation

➢Shared hardware like caches & TLBs introduce timing unpredictability for real-time systems (RTS).

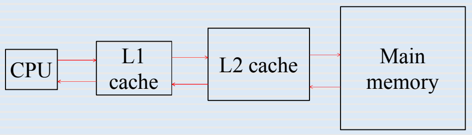
➢Worst-case execution time (WCET) analysis for RTS with shared hardware resources is often so pessimistic that the extra processing capacity of multicore systems is negated.

➢Different levels of assurance are required for different criticality tasks.

Problem



➢a) Memory ref hits in L1 cache – Access latency: 1-4 cycles.



➢b) Memory ref misses L1 & L2 cache – Access latency: 40-100 cycles.
➢c) Memory ref misses in TLB – Access latency: +1000 cycles.

➢Tighter WCET estimates can be established if we know which references hit in the cache and which do not.

➢Other shared resources like TLBs show similar timing unpredictability.

Solution

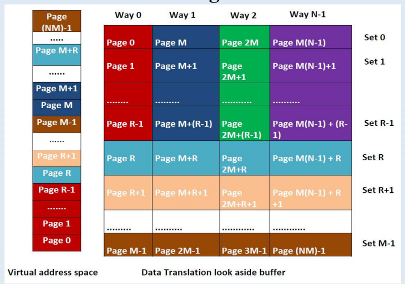
➢ Our solutions focus on two shared resources: shared caches and TLBs.

➢ **TLB Coloring:**
- Control the allocation of memory so that real-time tasks will not interfere with one another in terms of DTLB conflicts.
- Make DTLB misses more predictable.
- Enable static timing analysis tools to compute tight bound on WCET.

➢ **Cache Management:**
- Leverage the fact that only highly critical components require conservative provisioning.
- Provide "temporal isolation" across criticality levels.
- Apply a multiprocessor real-time synchronization protocol to manage cache lines.
- Enable schedulability gains by reducing WCET.

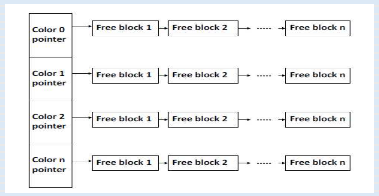
TLB Coloring – Solutions & Results

Design



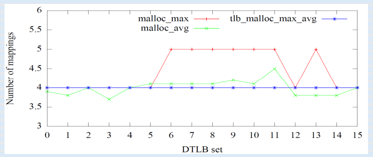
- Assign the same color to pages that map to the same DTLB set.
- To support greater than page size allocations, R sets are reserved. (Max contiguous allocation = PAGE_SIZE * R)

Data Structures and Implementation

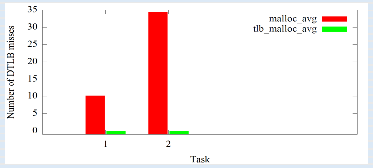


- tlb_malloc_init()
  - Sets aside huge virtual address space
- tlb_malloc(size, color)
  - Allocates memory region of size bytes of a particular color
  - Serves allocations from the pool set aside by tlb_malloc_init
- tlb_free(color)
  - Deallocates memory
  - Adds memory back to the pool

Results



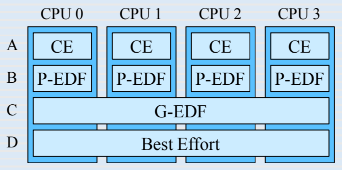
- malloc - non uniform utilization of DTLB sets
- tlb_malloc - predictable



- Zero misses if tasks use tlb_malloc

Cache Management – Solution & Results

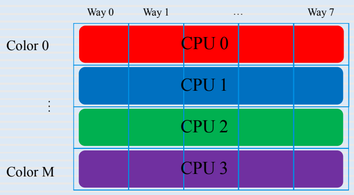
Mixed-Criticality System



MC[2] (mixed-criticality on multicore) architecture

- Assign colors to pages to control the mapping address of memory pages.
- Higher-criticality tasks should not be affected by lower-criticality tasks.
- Request tokens to reserve cache lines before loading pages.

Cache Management



- Partition caches to eliminate interference across processors for Levels A and B.
- Global tasks at Level C require tokens to reserve cache lines.

Ongoing Work

- Implement MC[2] scheduler in LITMUS^{RT}.
- Enable budget enforcement using container abstraction within the OS.
- Trap page fault when a job accesses a page for the first time to request token.
- Determine whether cache bypass can be selectively applied.
- Devise needed schedulability analysis.

Conclusions

➢ Designed TLB coloring techniques to support contiguous page allocations.
→ eliminate DTLB misses.

➢ Evaluated on Intel 16 core platform.

➢ Conducted experiments using synthetic benchmark, Malardalen benchmark, and MiBench.

➢ Provided task isolation in terms of DTLB conflicts.

➢ Re-implemented a mixed-criticality scheduler in LITMUS^{RT} to support cache management.