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

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

- 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

- 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)

Cache Management – Solution & Results

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

- Zero misses if tasks use tlb_malloc

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 Millicast.
- Provided task isolation in terms of DTLB conflicts.
- Re-implemented a mixed-criticality scheduler in LITMUSRT to support cache management.

Ongoing Work

- Implement MC2 scheduler in LITMUSRT.
- 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.