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

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

[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], Micaiah Chisholm[1], Namhoon Kim[1], Shrinivas Panchamukhi[2], Xing Pan[2]

NC State University

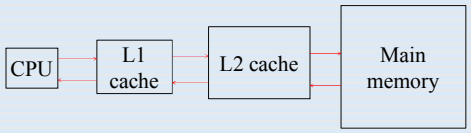
Motivation

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

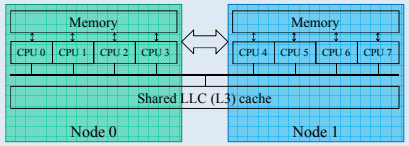
➢Worst-case execution time (WCET) analysis for RTS with shared hardware resources is often so pessimistic that the extra processing capacity of multicore systems is negated.

➢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.



➢Shared memory shows timing unpredictability. (1) The latency of accessing remote node is significantly longer than local node. (2) conflicts between shared-bank accesses result in unpredictable memory-access latencies.
➢Sharing last-level caches (LLCs) results in timing behaviors that are exceedingly difficult to characterize for WCETs without excessive pessimism.

Solution

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

➢ **Controller-Aware Memory Coloring:**

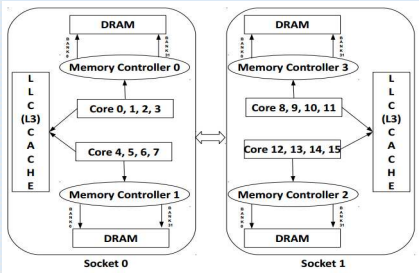
- Design a heap allocator that "colors" memory pages with locality affinity for controller and bank-awareness.
- Avoid memory accesses to remote node.
- Reduce conflicts among banks.

➢ **Cache and Bank Isolation:**

- Provide criticality-aware isolation in LLC and DRAM bank.
- Provide an optimized LLC allocation technique based on linear programming.
- Isolate the higher-criticality tasks from the operating system (OS).
- Enable schedulability gains by integrating MC analysis and hardware management.

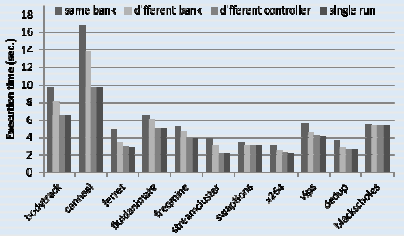
Controller-Aware Memory Coloring – Solutions & Results

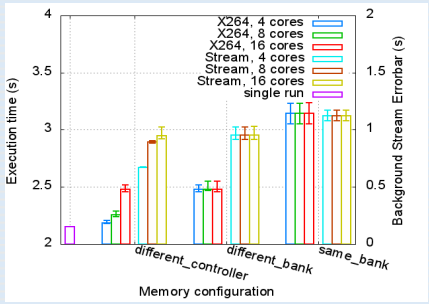
Design and Implementation



- char * A = (char*) mmap(color id, length, prot | 0x40000000, flag, fd, offset);
- Third parameter triggers our coloring allocator.
- First parameter indicates the memory coloring information.

Results

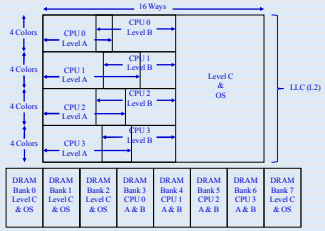




- System performance for Parsec code is enhanced by our controller-aware memory coloring scheme since it can avoid remote access penalty and reduce shared bank conflict.
- The "different_controller" of our approach is a policy that provides single core equivalence.

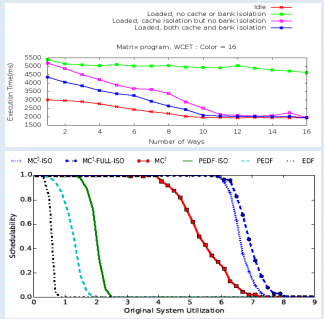
Cache and Bank Isolation – Solutions & Results

Cache and DRAM Bank Allocation



- Criticality-aware optimization techniques based on linear programming are used for allocating LLC.
- Tasks at Level C and OS shares the same partition.

Results



- We devised optimized LLC allocation techniques.
- Isolation reduced WCETs by up to 242%.
- MC provisioning improved schedulability for 68% of the scenarios.
- LLC and bank isolation improved schedulability for all scenarios under partitioned earliest-deadline-first (PEDF).
- Combining both approaches enables us to schedule one to two cores' worth of additional utilization in most cases.
- In run-time experiments, there were no deadline misses at levels A and B. At level C, the largest deadline-miss ratio was 1.40%, which is acceptable as level-C is soft real-time.

Conclusions

➢ Designed controller-aware memory coloring techniques to support memory page allocations.

➢ Designed criticality-aware LLC optimization techniques.

➢ Provided OS and task isolation in terms of LLC and DRAM banks.

➢ Avoid remote memory node access.

➢ Evaluated on Intel and AMD 16 core platforms and I.MX6 quad-core platform.

➢ Conducted experiments using synthetic benchmark, Parsec and Data intensive systems (DIS) benchmark.

➢ Conducted a large-scale overhead-aware schedulability study.

➢ Conducted run-time experiments to validate our assumptions.