CES Research Report
Southern Illinois University Carbondale

Towards Predictable Execution of Safety-Critical Tasks on Mixed-Criticality Multi-Core Platforms.

Dimitri Kagaris, Harini Ramaprasad.
Students (graduate): Aishwarya Vasu, Ashish Choudhari.

Project Description
Multi-core architectures are a natural choice for integrating multiple independent functionalities into a single node in a cost- and space-effective manner. However, for systems with multiple levels of criticality, transporting highly safety-sensitive (HSS) applications (such as those for avionics systems with Design Assurance Level (DAL) of grade from DAL-C to DAL-A) onto a multi-core platform and sharing the benefits of the new computing environment with other less safety-sensitive (LSS) applications that have lower assurance levels but may be computation-intensive presents challenging problems in ensuring predictability/determinism of the HSS applications while still maintaining acceptable Quality of Service (QoS) for the LSS applications. The dominant approach towards isolating HSS and LSS tasks is the use of a virtualization environment (hypervisor) on top of the underlying multi-core platform. In prior work, extensive experimentation was conducted on the Freescale P4080 platform to characterize the behavior of HSS tasks in the presence of LSS tasks when executing them on different, statically derived partitions residing on separate cores. In the completed phase of this on-going project we investigated the use of P4080 hardware mechanisms such as cache locking, cache partitioning and message passing among partitions to maintain determinism of HSS applications under regular and overload situations while maintaining QoS for the LSS applications.

Objectives
The P4080 platform involves 3 levels of cache. Cache levels L1 and L2 are private to each core, while L3 is common to all cores. In this phase of the project, the objective was to examine the effect of cache locking at level L1.

Industrial Relevance
While cache locking, partitioning and partition management mechanisms are not new concepts, there is no study/research applying to mixed-criticality workloads executing in virtualized environments on the Freescale P4080 platform.

Project Outcomes
We created a benchmark that creates two arrays AR1 and AR2 and performs random reads on both arrays. Different array sizes ranging from 1K to 8K are used to perform the experiments and the execution times are recorded. Array AR1 is locked in the cache for the case “L1 cache locking”. Each experiment is executed after a cold reset of the target system. Each experiment is executed several times and the average values are obtained. We developed three kernel modules for the following cases.

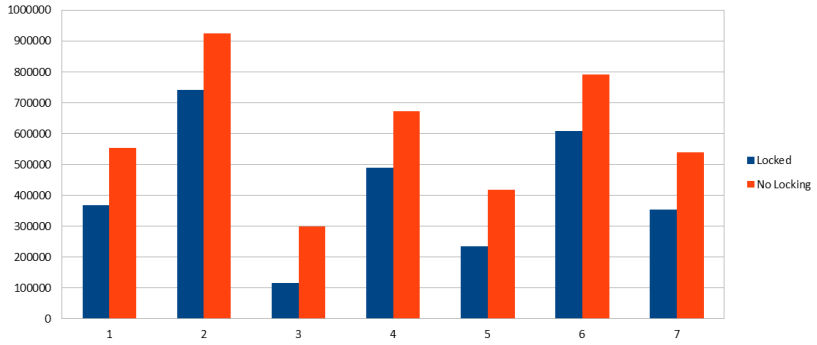
- “No L1 Cache” --A kernel module which disables L1 cache altogether. This is done to see if there is any effect on the execution time of the benchmark. Ideally, this must result in higher execution times.

- “L1 Cache Locking” - A kernel module which accepts an array address and the number of bytes to lock in L1 cache. The array1 is created in the user program and array sizes ranging from 1K to 8K are loaded and locked in the L1 cache by the module. As the data are locked in the cache, the execution

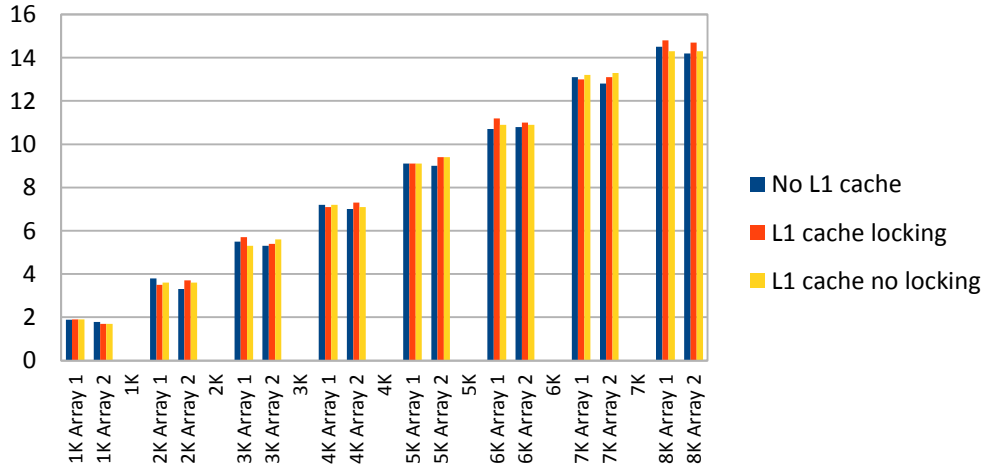
times of the benchmark for various array sizes for array1 are expected to be smaller compared to other cases.

- "L1 cache no locking" - A kernel module which enables L1 cache but does not lock any data in L1 cache. The execution times of the benchmark are expected to be smaller when compared to the case "No L1 cache" but slightly will be higher than for the case "L1 cache locking"

The figure below shows results on the execution time when array AR1 is the only array used in the benchmark under the "L1 Cache Locking" and the "L1 cache no locking" scenarios. The size of AR1 for this experiment was set to 8K. The results are shown for 7 different runs.



The behavior is as expected with lower execution times for the locked scenario. However, when AR1 is not the sole array in the benchmark but the benchmark contains an additional array AR2 that is never locked, the results are not always consistent. The following graph displays the average execution times for all the three cases "No L1 cache", "L1 cache locking" and "L1 cache no locking". The execution times for all array sizes for array AR1 under the case "L1 cache locking" are expected to be the least. But this is not the case. We have verified the register settings and the procedure to lock L1 cache lines with Freescale engineers. We have been trying through a variety of different setups to find out what causes this unexpected behavior.



Research Team

Dr. Dimitri Kagaris, Professor, Dr. Harini Ramaprasad, Assistant Professor, Aishwarya Vasu, PhD. student, Ashish Choudhari, PhD student.