ABSTRACT

In singlecore processors timing analysis involves a step of Execution Time Analysis at task level that yields an Execution Time Bound (ETB) for the task, and one of schedulability analysis, where the scheduling attributes of the individual tasks, including the ETB, are studied from a system level perspective. Response Time Analysis serves as a compositional bridge between those two steps by accounting for the contention that arises from resource sharing.

In this paper, we show that the advent of multicore processors challenges the viability of this two-step approach. This stems from the fact that inter-task interference effects in a multicore are much more intricate in nature than what can be compositionally captured in response time analysis by widening the tasks' ETB with the time intervals during which tasks cannot progress while actually holding the CPU; we also show how contention in the access to hardware shared resources creates a circular dependence between the task ETB and its actual scheduling at run time. Finally, we show how various degrees of time composability can help breaking this knot.

1. INTRODUCTION

The timing analysis of a real-time system involves deriving an Execution Time Bound (ETB) for each application program. That ETB is computed assuming that the program runs in isolation. All system-level effects such as interrupts, blocking times when arbitrating access to software-level resources, and the preemptions that result from priority-driven scheduling are taken into account afterwards, compositionally, during the Response Time Analysis (RTA) stage. RTA is fed with the ETB of each task, which is assumed given. In that process, the ETB computed considering tasks in isolation is augmented with the time duration in which the task, though notionally running, cannot actually use the CPU to progress. This two-step timing approach in which ETB is simply enlarged to account for system-level effects has shown to work sufficiently well in singlecore architectures.

In COTS multicore systems, task scheduling affects task preemption much like in singlecore systems. Unlike in singlecore systems, however, scheduling in multicore systems also determines the tasks that may potentially run at the same time in the processor, whose execution may cause conflict on the access to hardware shared resources, which is the source of a potentially huge impact on tasks' timing behavior (i.e. their ETB). This creates a dependence between execution time analysis and task scheduling.

Attempting to compositionally inflate the ETB obtained for individual programs obtained in isolation, with the effects of inter-task conflicts on access to hardware shared resources is impractical: tasks may conflict so frequently (e.g. on every access to the on-chip bus) and for so many resources (caches, buses, buffers, etc.) that the amount of knowledge required to characterize the hardware and application behavior rapidly gets out of control.

In this short paper we analyze the dependence between execution time analysis and task scheduling in COTS multicore and we also discuss how various flavors of time composability can help breaking such circular dependence.

2. BREAKING THE TIMING ANALYSIS - SCHEDULING DEPENDENCE

Several works in the state of the art propose hardware support to either eliminate or control inter-task conflicts on access to shared hardware resources. Proposals include: TDMA or UBD in buses [4]; partitioning for caches [4]; or real-time aware memory controllers [1, 3]. Those solutions simplify timing analysis since the effect of inter-task interference is null (when the hardware eliminates inter-task interference) or can be easily derived (when the hardware bounds inter-task interference). This ultimately enables the use of the said two-step analysis process. However, to the best of our knowledge, no current commercial multicore processor provides complete isolation or full control of inter-task interference. As a consequence, the execution time of a software program may be inordinately affected by the load that its co-runners place on the hardware shared resources. For instance, some studies report that the conflicts in resources such as memory or the processor bus, have a considerable magnifying effect (>2x) on the observed execution times [2].

Time composability stipulates that the timing behavior of a software component determined in isolation, does not vary on composition with other software components. When applied to a multicore processor, this can be understood to mean that the timing behavior of a task is not affected by the activity of its co-runners.

Time composability is generally considered as an all-or-nothing metric. If the ETB of a task varies depending on which co-runners it may encounter, then that ETB is non-time composable and vice-
versa. The implication here is that when deriving (fully) time-composable ETB for a task, the worst-case execution conditions must be considered, in which the worst possible contention is generated by any other potential co-runner task. Although using fully time-composable ETB breaks the circular dependence between timing analysis and scheduling, the resulting values often are so much larger than the observed behaviour to cause massive over-provisioning.

2.1 Generalizing Time Composability

Our proposal to attack the scheduling - time analysis dependence consists in determining Partially Time Composable (pTC) ETB for each task. The idea behind this approach is to reduce the search space for the execution conditions to consider in the computation of pTC ETB. The resulting pTC ETB remain valid as long as the execution conditions for which they have been derived are preserved during operation. The execution conditions for a task are meant to cover all hardware factors of influence on its timing behavior. In the case of multicores, the execution conditions for task $\tau_i$ are determined by the load that its co-runners place on the hardware shared resources, which ultimately determines the extent of hardware contention that $\tau_i$ suffers while running.

Under our partially time composable approach, each task $\tau_i$ is associated multiple ETB, one for each of the possible set of execution conditions being considered. For that approach to be usable, the scheduling of tasks at run time must ensure that every task encounters the execution conditions for which its ETB was derived, so that schedulability analysis can safely use that value.

We assume a task set $T = \{\tau_1, \tau_2, ..., \tau_n\}$. Under partitioned scheduling, individual tasks are statically assigned to one of $m$ groups $\Phi = (\varphi_1, \varphi_2, \cdots, \varphi_m)$, where $m$ stands for the number of cores. In contrast with global scheduling, partitioned scheduling simplifies timing analysis by limiting the number of co-runners that a task may have.

In this short paper we develop four definitions of time composability with increasing isolation, i.e. time composability – properties.

- **Full TC (fTC) ETB**. fTC considers the worst possible inter-task interference from hardware contention that a task may suffer from any other co-runner task, not limited to those in the application task set (worst execution conditions). This requirement, which may sound inflictive, serves the purpose of allowing fTC ETB values to be independent of the co-runners, hence breaking the scheduling - timing analysis dependence.

- **Task-set TC (tsTC) ETB**. Under global scheduling, the ETB estimate $C_i$, derived for $\tau_i$, is time-composable with respect to any scheduling of the other tasks in the task set. That is, $C_i$ is determined considering the execution conditions (i.e interference on contention for hardware shared resources), that can occur when any of the $m-1$ tasks of the task set can simultaneously run with $\tau_i$.

- **Allocation TC (aTC) ETB**. Under partitioned scheduling, the ETB estimate, $C_i$, derived for $\tau_i$, is time-composable with respect to a given allocation $\Phi$. Under $\Phi$ the other tasks running in the same core where $\tau_i$ is – i.e $\tau_i$’s group-mate tasks, cannot run simultaneously with $\tau_i$, hence the execution conditions induced by $\tau_i$’s group-mates are not considered when deriving ETB for $\tau_i$.

- **Non time composable (nTC) ETB**. In this weakest form of time composability, the ETB workload computed for task $\tau_i$ under a given set of co-runners is only valid for that particular set of co-runners. We regard nTC ETB as too weak to be used in practice because any variation in the alignment of tasks at run time varies the execution conditions and invalidates the ETB. Thus, we do not consider this case further.

2.2 Putting it all together

Each TC approach incurs a different pessimism in derived ETB and affects incremental verification.

- **fTC**, which can be attained for global or partitioned task scheduling alike, cleanly breaks the dependence among timing analysis and scheduling since the ETB derived for a task upper bounds the effect of contention that any other task can generate. Hence fTC fully enables incremental verification in the time domain. However, the resulting ETB can be overly pessimistic since they have to contemplate the worst-case execution conditions.

- **tsTC** is meant for global scheduling and in general introduces less overestimation than fTC, though this depends on the use of resources made by the tasks in the task set rather than by any ideal task. However, if any task in the task set changes, timing analysis must be repeated for all the tasks as well as schedulability analysis. This drawback makes this approach less viable for incremental development.

- **aTC** focuses on partitioned scheduling. It provides tighter ETB than tsTC in general, but only covers the execution conditions for a particular allocation of tasks. It places no constraints on how the tasks in each core are scheduled.

- **nTC** provides no time composability making more difficult incremental verification. It is also unclear how it could be used with global or partitioned scheduling.

Overall, the particular TC approach to use depends on the wanted balance between pessimism in the ETB and time composability. It is noted that each TC approach may incur different overheads in the timing analysis tool to derive ETB under a specific set of execution conditions. Enabling different TC approaches on real COTS considering the benefits and drawbacks of each of them is part of our future work.

Acknowledgments

This work has been supported by the Spanish Ministry of Science and Innovation under grant TIN2012-34557 and under the HiPEAC Network of Excellence. Jaume Abella has been partially supported by the MINECO Ramon y Cajal fellowship RYC-2013-14717.

3. REFERENCES


