Analyzing Timing of Multicore-Software Scheduling—A New Way that Makes It Simple

Technical Point of Contact:
Dr. Bjorn Andersson, Cyber-Physical Systems and Ultra-Large Scale Systems Group (CPSULS),
baandersson@sei.cmu.edu

For general information about the SEI
For information about the SEI and its products and services, contact Customer Relations
P: 412-268-5800
F: 412-268-6257
customer-relations@sei.cmu.edu
www.sei.cmu.edu

Multicore processors are the norm today in desktop computing, servers, and mobile computing. Many software developers are increasingly interested in using multicore processors in critical applications—for example, in automotive, medical devices, avionics, space, and defense—to achieve more computing power with lower power consumption, smaller physical size, and higher reliability. But these applications pose additional hard real-time requirements on the software's design and operation.

Satisfying hard real-time requirements is typically needed when the software interacts with its physical world. For example,

- an airbag in a car, needs to be inflated at the right time;
- a braking system in a car needs to apply the right braking force at the right time;
- counter-measures, in an aircraft, need to be deployed at the right time; and
- pacemakers need to deliver stimuli at the right time.

In such applications, where hard real-time requirements must be satisfied, one of the main challenges that software developers face is that the timing of software depends on how low-level hardware resources, such as the memory system, are allocated when requested to support the software's execution. The one resource that is easiest to understand is the memory bus. If two processes, one on each processor, experience memory-cache misses simultaneously, both processes will request access to memory simultaneously, resulting in two requests to use the same memory bus. Unfortunately, both cannot be served simultaneously; hence one of these requests will be queued up and will experience an increase in execution time. For memory-intensive applications, we and those with whom we have collaborated have experienced anywhere from a 300% to 1,100% increase in execution time because of the sharing of resources in the memory system. Clearly, if such resource contention is not accounted for in the analysis at design time, timing requirements may be violated at runtime, resulting in mission failure or disaster in which interaction with the physical world does not occur at the right time.

The challenge of analyzing the timing of resource contention in the memory system of multicore processors is an issue in both "greenfield" development of software for multicores and in the transitioning of legacy code to multicore processors. Figure 1 illustrates one such challenge. Figure 1a shows a legacy software system using a single hardware processor with two software processes where the timing requirements are satisfied. However, Figure 1b shows that when the same software is migrated to a multicore processor with two hardware processors on a chip, a timing requirement is violated. Hence, care must be taken when migrating legacy real-time software to a multicore. This is crucial because we can expect all major software systems to undergo this transition to multicore processors as the commercial marketplace evolves.

Analyzing Timing of Multicore-Software Scheduling—A New Way that Makes It Simple

The problem of analyzing the timing of contention for resources in the memory system of multicore processors is challenging for several reasons. One is that the number of schedules (or interleavings) of the processes is large, and it is not possible to manually check all of them. Another is that the timing analysis of memory contention depends on operation of the memory system, and constructing models to mimic that operation is often not possible because hardware makers typically do not disclose the internals of the memory controller.

Consequently, the SEI has developed an automatic method for verifying timing requirements. The method does not model the actual hardware; instead, it models the effect that one process has on another—how much longer the execution time of one process is when it executes in parallel with another process. With this model, it is possible to prove that all timing requirements are satisfied for all schedules that the system can generate.

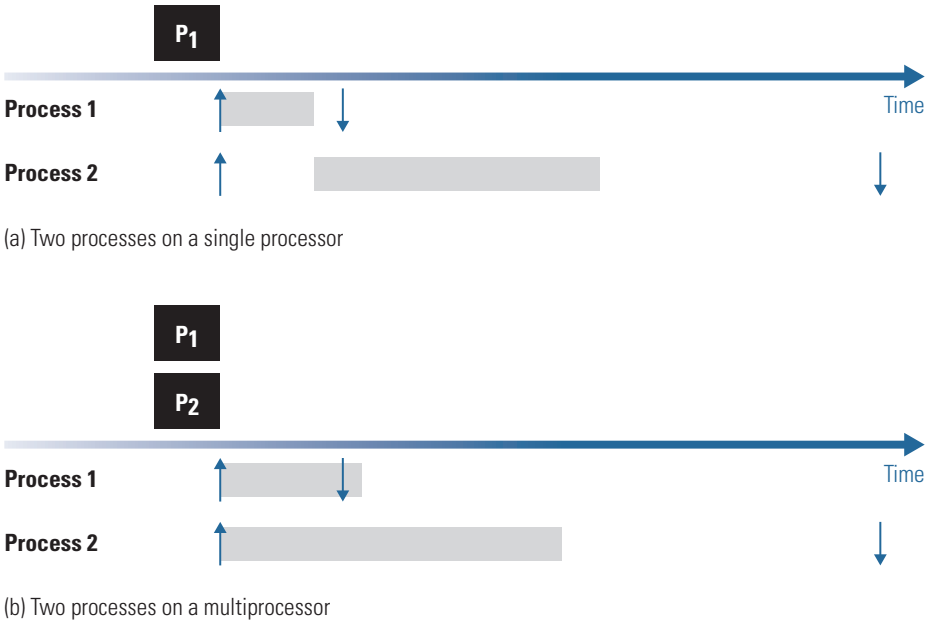

Figure 1. Migrating a software system to a multicore can cause a violation of timing. The execution time of processor 1 is longer in figure (b) than in figure (a) because of contention on resources in the memory system in the multicore processor. An arrow pointing up indicates the time when a process is requested to execute and an arrow pointing down indicates the deadline of a process.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM-0001319