Performance Analysis of Distributed Embedded Systems

Lothar Thiele
ETH Zurich, 8092 Zurich, Switzerland
thiele@tik.ee.ethz.ch

Categories and Subject Descriptors: C.3 SPECIAL-PURPOSE AND APPLICATION-BASED SYSTEMS: Real-time and embedded systems

General Terms: Performance

Keywords: Embedded Systems, Distributed Systems, Performance Analysis.

Tutorial Background

Often, the constraints imposed by a particular application domain require a distributed implementation of embedded systems. In this case, a number of software or hardware components communicate via some interconnection network. We find this structure on various levels of granularity, starting from multiprocessor systems on a chip via distributed embedded control units (ECU) in automotive applications and ending at large-scale sensor networks.

For example, architectural concepts of heterogeneity, distributivity and parallelism can be observed on single hardware components, as they are often implemented as so-called systems-on-chip (SoC) or multiprocessor-systems-on-a-chip (MPSoC). A collection of memories and heterogeneous computing resources are implemented on a single device, and communicate using networks-on-chip (NoC) that can be regarded as dedicated interconnection networks involving adapted protocols, bridges or gateways.

Embedded systems are typically reactive systems that are in continuous interaction with their physical environment to which they are connected through sensors an actuators. Examples are applications in multimedia processing, automatic control, automotive and avionics, and industrial automation. Therefore, many embedded systems must meet real-time constraints, i.e. they must react to stimuli within a time interval dictated by the environment. It becomes apparent that heterogeneous and distributed embedded real-time systems as described above are inherently difficult to design and to analyze because of the tight interaction between computation, communication and the available resources.

Part of this difficulty is caused by the fact that the functional and extra-functional behavior of the system is influenced by interferences on shared resources such as processors, memory or communication devices. Packet streams or tasks may prevent each other from using these resources, even if they belong to independent parts of the application. As a result, resource sharing strategies influence the system behavior to a large extend. In addition, the system environment which continuously interacts with the embedded system may vary and cause an additional degree of non-determinism.

During the system level design process of an embedded system, a designer is typically faced with questions such as whether the timing properties of a certain system design will meet the design requirements, what architectural element will act as a bottleneck, or what the on-chip memory requirements will be. Consequently it becomes one of the major challenges in the design process to analyze specific characteristics of a system design, such as end-to-end delays, buffer requirements, or throughput in an early design stage, to support making important design decisions before much time is invested in detailed implementations. This analysis is generally referred to as system level performance analysis. If the results of such an analysis is able to give guarantees on the overall system behavior, it can also be applied after the implementation phase in order to verify critical system properties.

One of the major requirements for models, methods and tools is their support for a modular, component-based design. This aspect covers as well the composition of the underlying hardware platform as well as the software design. Because of the import role of resource interaction, these components not only need to talk about functional properties but also about resource interaction.

Tutorial Program

In the tutorial, we will cover the following aspects of system level performance analysis of distributed embedded systems:

- Approaches to system-level performance analysis (simulation-based vs. analytic methods, review of existing methods and tools). Requirements in terms of accuracy, scalability, composability and modularity.
- Modular Performance Analysis (MPA): basic principles, methods and tool support.
- Examples that show the applicability, the embedding into design space exploration, and a comparison to simulation-based approaches.

For further information see http://www.tik.ee.ethz.ch/pisa and http://www.mpa.ethz.ch.

EMSOFT'07, September 30–October 3, 2007, Salzburg, Austria.
ACM 978-1-59593-825-1/07/0009.