Supporting Mixed-Criticality Avionics Workloads on Multicore Platforms

The Challenge

This project is directed at the development of software design and validation methods for supporting mixed-criticality workloads in avionics systems. The primary target of this research is **unmanned air vehicles** (**UAVs**). Next-generation UAVs will be much more sophisticated in terms of their planning and decision-making capabilities than current designs. This will result in a significant increase in the computational workload that must be hosted on board. It would clearly be desirable to host this workload on as few processors as possible, to reduce weight and volume. This can be partially accomplished by co-hosting applications of different criticalities on the same computing platform. The overall hardware infrastructure can be further reduced, while enhancing processing capabilities at the same time, by using multicore processors. For such reductions to be possible, operating-system (OS) infrastructure must be developed that allows co-hosted subsystems of different criticalities to be "isolated" from one another. The major goal of this project is to produce this needed infrastructure, with an emphasis on providing isolation from a real-time perspective.

The Approach

The needed OS infrastructure will be provided by developing a container framework that allows subsystems of different criticalities to be encapsulated and isolated from one another. In its most conventional sense, a container is simply a task group that is managed in such a way that tasks inside the group are isolated from tasks outside the group. We seek to expand the concept of "isolation" so that the timing correctness of subsystems can be validated at *different criticality levels*. Components of higher criticality require more conservative provisioning and have more stringent timing requirements than those of lower criticality.

The research to be undertaken consists of both algorithmic and implementation components. Algorithmic research is focusing on the development of scheduling and synchronization algorithms for supporting real-time containers of different criticalities, and associated analysis techniques for validating timing constraints.

In the implementation component of the project, the scheduling and synchronization algorithms that are produced are being implemented in LITMUS^{RT}, a UNC-produced real-time variant of Linux, and evaluation studies are being conducted using avionics workloads. The container framework being developed will allow resources to be dynamically re-provisioned at runtime (which may be needed to enable a UAV to react to events in its environment). Because high-criticality components are conservatively provisioned, they will (almost) never overrun their allocation budgets. The same is not true of low-criticality components. A key focus of this project is the development of policies for dealing with overloads at low criticality levels and for allowing unused processing capacity at high levels to be redistributed to low levels.

Significance

In current avionics systems, multicore processors are not used to host highly-critical components (more precisely, if such processors are used, then all but one core is turned off). This is partially because work on real-time resource allocation has not matured to the point where complex interactions across cores can be effectively analyzed. In this project, we are seeking to remedy this situation by devising resource-allocation mechanisms that can be efficiently applied and analyzed in this context.

Project Members

James Anderson, professor (PI)
Sanjoy Baruah, professor (PI)

Research Sponsor

For More Information

Dr. James Anderson
Department of Computer Science
University of North Carolina at Chapel Hill
CB#3175, Frederick P. Brooks, Jr. Building
Chapel Hill, NC 27599-3175
Phone: (919) 962-1757
Fax: (919) 962-1799
E-mail: anderson@cs.unc.edu

http://www.cs.unc.edu/~anderson/projects/ngc.html