Supporting Mixed Criticality Applications in a Ravenscar-Java Environment

Hao Cai and Andy Wellings

Department of Computer Science, University of York, UK
{haocai, andy}@cs.york.ac.uk

Abstract. Ravenscar-Java is a subset of Java augmented by a subset of the
Real-Time Specification for Java. It is targeted at high integrity real-time sys-
tems, however, currently only a single integrity level is supported. This paper
proposes extensions to Ravenscar-Java to allow it to support multiple criticality
applications within the same virtual machine on a single processor. A real-time
isolate is defined which supports both temporal and spatial firewalling.
Communication mechanisms are provided to allow controlled interaction
between high and low-level integrity applications. The implementation in a
Ravenscar-Java environment is discussed. Byte code verification and analysis is
performed offline to ensure the robust, predictable, scalable, efficient and safe
execution of Ravenscar-Java applications. A temporal deterministic runtime
architecture of the Ravenscar-Java is proposed to achieve temporal and spatial
isolation between applications, and also improve the scalability by safely
sharing the runtime data structures as much as possible with the help of the
offline analyzer.

1 Introduction

For many people, Java is the antithesis of a high integrity programming language
[6,7]. Ironically, many of the features that have led to its success as a programming
language for general applications are problematic when used in high integrity sys-
tems. Its combination of object-oriented programming features, garbage collection
and its poor support for real-time multithreading are all seen as particular drawbacks.
The Real-Time Specification for Java [3] has introduced many new features that help
in the real-time domain. However, the expressive power of these features means that
very complex programming models can be created, necessitating complexity in the
supporting real-time virtual machine. Consequently, Java, with the real-time
extensions as they stand, is too complex for confident use in high integrity systems. A
high integrity real-time Java profile, called Ravenscar-Java, has been defined [6,7] to
address these problems and work is underway to implement the required real-time
virtual machine [4].

Executing multiple computations (called “isolates” [8]) in a single instance of a
safe language virtual machine can improve performance and overall platform
scalability, and realize the full potential of language-provided safety and protection
[8,18]. It is also desirable for high integrity systems where multiple applications of
different integrity levels may exist on the same platform without having to
engineering everything to the highest integrity level. The overall objective of this paper is to propose an extension to Ravenscar-Java so that it supports real-time isolates. The paper is organized as follows. Section 2 introduces Ravenscar-Java Isolates. Section 3 proposes an architecture for their implementation. Section 4 discusses related work. Section 5 gives the current status and future work.

2 Real-Time Isolates in Ravenscar-Java

Ravenscar-Java Isolates [15] are defined to improve the overall system scalability, and allow a mixture of different integrity level applications to share the same computational platform. In this section, the overall model of a Ravenscar-Java Isolate is introduced; the approach to spatial and temporal isolates is discussed; the communications between isolates along with the communication integrity policy are also specified.

2.1 Overall Model

A Ravenscar-Java system consists of a fixed number of mixed integrity level isolates. An isolate consists of a fixed number of schedulable objects (real-time threads and asynchronous event handlers) that are created immediately after the program begins its execution. There are two execution phases: an initialization phase and a mission phase, illustrated in Figure 1:

Fig. 1. Ravenscar-Java Isolates.