Abstract

Language-based extensible systems such as Java use type safety to provide memory safety in a single address space. Memory safety alone, however, is not sufficient to protect different applications from each other. Such systems must support a *process model* that enables the control and management of computational resources. In particular, language-based extensible systems must support resource control mechanisms analogous to those in standard operating systems. They must support the separation of processes and limit their use of resources, but still support safe and efficient interprocess communication.

We demonstrate how this challenge can be addressed in Java operating systems. First, we describe the technical issues that arise when implementing a process model in Java. In particular, we lay out the design choices for managing resources. Second, we describe the solutions that we are exploring in two complementary projects, Alta and GVM. GVM is similar to a traditional monolithic kernel, whereas Alta closely models the Fluke operating system. Features of our prototypes include flexible control of processor time using CPU inheritance scheduling, per-process memory controls, fair allocation of network bandwidth, and execution directly on hardware using the OSKit. Finally, we compare our prototypes with other language-based operating systems and explore the tradeoffs between the various designs.