Abstract

Single-language runtime systems, in the form of Java virtual machines, are widely deployed platforms for executing untrusted mobile code. These runtimes provide some of the features that operating systems provide: inter-application memory protection and basic system services. They do not, however, provide the ability to isolate applications from each other, or limit their resource consumption. This paper describes KaffeOS, a system that provides these features for a Java runtime. The KaffeOS architecture takes many lessons from operating system design, such as the use of a user/kernel boundary.

The KaffeOS architecture supports the OS abstraction of a *process* in a Java virtual machine. Each process executes as if it were run in its own virtual machine, including separate garbage collection of its own heap. The difficulty in designing KaffeOS lay in balancing the goals of isolation and resource management against the goal of allowing direct sharing. Overall, KaffeOS is up to 11% slower than the freely available JVM on which it is based, which is an acceptable penalty for the safety that it provides. KaffeOS is substantially slower than commercial JVMs, but exhibits much better performance scaling in the presence of uncooperative code.