EXPERIMENTAL EVALUATION OF DISTRIBUTED MIDDLEWARE WITH A VIRTUALIZED JAVA ENVIRONMENT

Author* NUNO A. CARVALHO
Supervisor: José Pereira
* nuno@di.uminho.pt

University of Minho
School of Engineering
Computer Science and Technology Center

Introduction

The correctness and performance of large scale service oriented systems depends on distributed middleware components performing various communication and coordination functions. It is, however, very difficult to experimentally assess such middleware components, as interesting behavior often arises exclusively in large scale settings, but such deployments are costly and time consuming. We address this challenge with MINHA, a system that virtualizes multiple JVM instances within a single JVM, reproducing the concurrency, distribution, and performance characteristics of the actual distributed system. The usefulness of MINHA is demonstrated by applying it to the WS4D Java stack, a popular implementation of the Device Profile for Web Services (DPWS) specification.

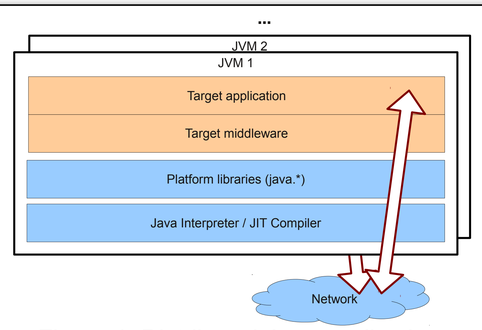

Figure 1: Distributed Java application.

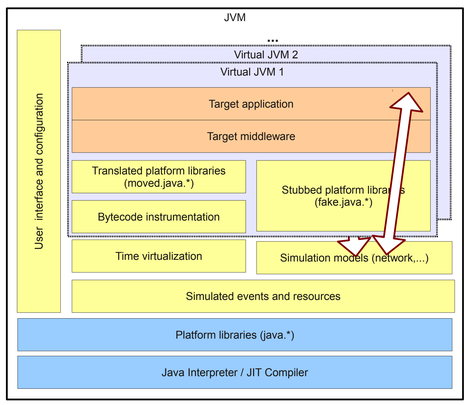

Figure 2: Simulation of a distributed Java application.

Overview

The experimental evaluation of some middleware component usually requires the architecture outlined in Figure 1. Briefly, multiple instances of an application that makes use of the target middleware component are deployed in multiple JVMs. Distributed interactions are then initiated by the application using the middleware, that makes use of platform's libraries and of the underlying Java bytecode execution mechanism.

MINHA allows reproducing the same distributed run within a single JVM as shown in Figure 2. In detail, the application and middleware classes for each instance are loaded by a custom class loader that replaces simulation models for references to the underlying platform, namely, libraries and synchronization bytecode. Some of these simulation models are developed from scratch while others are produced by translating native libraries themselves. The resulting code makes use of the simulation kernel and time virtualization to run. Multiple instances are loaded under the control of a command line user interface and configuration loader, which has the following advantages.

Global observation without interference: Once the whole process is centralized, it is possible to get a global observation of all operation and system variables.

Simulated components: When in a real execution, still in the development phase, it is necessary to evaluate the system for different environments and software components. With MINHA, such environments and software models can be replaced by simulation models, and incorporated in a standard test harness to be run automatically as code evolves.

Large scale: Large scale applications, that require a huge amount of resources to be deployed in the real environment, make this development more complicated. Testing during the development phase may require a big share of the system already in operation which requires high costs already in these phase.

Automated "What-if" analysis: By resorting to simulated components and running the system with varying parameters, the impact of extreme environments can be assessed, exploring even conditions that are not yet possible in practice.

Fault-injection: MINHA can also be extended to perform various fault-injection operations, in space and time domains

Results

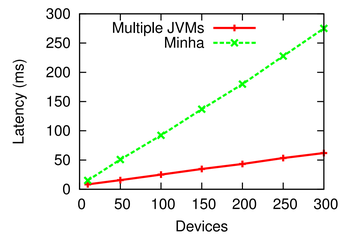

Figure 3: Latency.

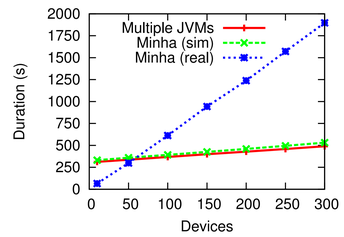

Figure 4: Duration.

Execution mode	RAM (GB)
Multiple JVMs	25.4
MINHA	5.7

Figure 5: Average memory usage for 300 devices.

Conclusion

This poster introduces MINHA, a simulation platform that allows multiple virtual JVM instances that run off-the-shelf Java middleware components and distributed applications within a single host JVM, much as an hypervisor enables multiple virtual machines within a single server. In contrast with common hypervisors, MINHA manages a simulated timeline which is updated using accurate measurements of the time spent executing real code fragments, hence reproducing the performance of multiple physically independent hosts. Finally, it includes simulation models of networking components for realistically mimicking a distributed execution.
The usefulness of MINHA is demonstrated by showing how the WS4D Java stack, an off-the-shelf middleware component, is evaluated on a large scale system with hundreds of simulated devices in single host using 5 X less RAM than needed for separate JVMs. In fact, MINHA avoids the error introduced by running all devices within a single host, competing for the same CPU resources, and provides a more truthful approximation of a distributed system.