Java and Distributed Simulation

[email dated 6 Aug 1996]

Geoffrey Fox

Distributed computing is a powerful tool in many high performance applications but it is particularly relevant when the underlying problem is itself distributed. The best known examples come from tactical military simulations as implemented in SIMNET and successor programs such as DSI—Distributed Simulation Internet. However there are many other distributed organizations and/or applications which can naturally use this paradigm—in the commercial arena, finance and interdisciplinary (inter-organization) manufacturing are two examples.

Distributed simulation is characterized by some key features—geographically distributed but relatively loosely coupled components. These components can be pure simulation modules but can also include "people in the loop" or active sensors and instruments ("Machines in the loop"). In the military case, one has linked simulated vehicles (say aircraft and tanks) with real vehicles in the field and with military commanders and pilots/tank drivers all in the same simulation.

We suggest that the Web and Java are very suitable for such applications. The Web itself naturally supports a distributed set of loosely coupled applications coordinated by linked web servers. The session management function of current collaborative tools needs to be generalized to support the event driven simulation paradigm underlying typical distributed simulation applications. VRML provides the natural client technology to support simulated virtual environments for "people in the loop"—here we use the latest VRML 2.0 enhancements to provide dynamic behavior. Web-linked databases and an overall Dataflow (WebFlow) computing model provide the necessary high level information and computing services.

Java is the natural implementation language at all levels of such a distributed computing environment. Some arguments for the use or Java are quite general and reflect my prejudice that Java will become a dominant general purpose language. The "web-ready" nature of Java and its excellent support for graphics output are some such general features. Looking at particular capabilities, event driven simulations are usually built as a set of objects interacting via messages. This appears to be naturally implemented using Java's object-oriented features. Java (as in MIT's Jigsaw and NCSA's Habanero) is getting used in a growing

number of Web server and collaboration systems. This illustrates that Java will be excellent base control software for a distributed simulation environment. Note that we expect that Java will soon be supported in three distinct but co-existing modes with different performance-functionality tradeoffs. There is the current semi-interpreted (Universal bytecode) Applet model; JavaScript (being integrated with Java by Netscape) and the VRMLScript discussions illustrate that a fully interpreted version of Java will be supported. This is essential in many applications. Finally there will be native Java compilers which will give Java (native) classes with performance competitive with C and FORTRAN code. Distributed simulation needs all three modes in different parts of its implementation.

Java and the Web automatically build in support for the coordination of loosely coupled simulation modules which is the major form of parallelism needed in most distributed simulations. However one can expect to need data parallel execution of some tightly coupled components. For example, these could be an airflow simulation component in a distributed manufacturing system or a military weather simulation or image processing application. We have finished a preliminary study of data parallelism in Java and believe that techniques developed in HPF and HPC++ can be adapted for Java. Interpreted Java front ends can invoke high performance Java wrapper classes which link to native FORTRAN (HPF) or compiled Java programs.