

# JavaSplit: A Runtime for Execution of Monolithic Java Programs on Heterogeneous Collections of Commodity Workstations

Michael Factor	Assaf Schuster	Konstantin Shagin
IBM Research Lab in Haifa Haifa University Campus Haifa 31905, Israel factor@il.ibm.com	Computer Science Department Israel Institute of Technology Technion City Haifa 32000, Israel assaf@cs.technion.ac.il	Computer Science Department Israel Institute of Technology Technion City Haifa 32000, Israel konst@cs.technion.ac.il

## ABSTRACT

This paper describes the design and presents the preliminary performance evaluation of *JavaSplit*, a portable runtime for distributed execution of multithreaded Java programs. *JavaSplit* *transparently* distributes threads and objects of an application among the participating nodes. Thus, it gains augmented computational power and increased memory capacity without modifying the Java multithreaded programming conventions, allowing the programmer to be unaware of the distributed nature of the underlying environment.

*JavaSplit* works by rewriting the bytecodes of a given parallel application, transforming it into a distributed application that incorporates all the runtime logic. Each runtime node carries out its part of the resulting distributed computation using nothing but its local standard (unmodified) Java Virtual Machine (JVM). This is unlike previous Java-based distributed runtime systems, which use a specialized (modified) JVM or utilize unconventional programming constructs. Since the proposed runtime is *orthogonal* to the implementation of a local JVM, it achieves portability across any existing platform and allows each node to locally optimize the performance of its JVM, *e.g.*, via a just-in-time compiler (JIT).

The *JavaSplit* runtime is designed to be highly *scalable*, never requiring global cooperation of nodes and using an efficient and scalable fine-grain distributed shared memory (DSM) protocol.

**Keywords:** Java, Network-based Distributed Computing, Single-System Image, Bytecode Instrumentation, Portability.