Preface

Performance is an important consideration for any programming language. In spite of an initial lag in performance of the Java** language, its performance is improving, and this improvement is expected to continue. As the Java environment evolves, a concerted effort to increase performance includes the techniques and tools to identify improvement opportunities, and the resulting development of algorithms and structures to achieve higher levels of Java performance.

Performance affects many aspects of the software life cycle, from requirements definition to use by customers. In the context of the Java language, new considerations are introduced because Java is still relatively young, and time is needed to learn how to apply familiar techniques to a Java environment. In addition, the Java language itself introduces fresh concepts and challenges. In this issue, eleven papers discuss various topics on Java performance, beginning with application experience and benchmarks and continuing with tools and techniques for performance analysis. Attention is then focused on the performance of the implementation of the Java virtual machine, including new directions for implementations that promise to deliver even higher levels of performance for Java virtual machines and applications. We are indebted to R. F. Berry of the IBM Network Computing Software Division in Austin, Texas, for his considerable effort in organizing, developing, and coordinating this issue.

Christ et al. view Java performance from an application development perspective. A cost-effective approach for developing Java applications is made possible through the use of specially designed frameworks such as SanFrancisco*. The authors describe the issues associated with ensuring adequate performance for large-scale applications built on such frameworks, given the performance of the underlying Java virtual machine implementation.

The Java language is being increasingly accepted beyond the business community. However, for numerically intensive computation, some critical performance deficiencies remain. Naive algorithm implementations suffer from both the heavyweight Java multidimensional implementation and the important, but expensive, language compliance constraints. Moreira et al. discuss this problem in their paper. They introduce a new Java Array package that presents array computations in a natural and familiar way, based on FORTRAN 90 and ideally suited to compiler (e.g., just-in-time compiler) optimization. Their results are compelling, resulting in performance nearly like FORTRAN for certain benchmarks.

Baylor et al. present an overview of Java benchmarks—applications developed to elicit fundamental differences between competing Java virtual machine implementations or between software systems employing those implementations (e.g., a Web application server implementing servlet support). Benchmarks help system developers identify opportunities for improving the performance of a particular virtual machine implementation. Benchmarks help application programmers learn where a particular virtual machine implementation may have weaknesses.

Viswanathan and Liang present a new Java virtual machine interface specifically designed to support performance monitoring and analysis of both applications and the virtual machine itself. The Java Virtual Machine Profiler Interface, or JVMPI, is an important and promising direction for Java analysis, since it frees performance tool authors from the requirement of detailed knowledge of the internals of any particular Java virtual machine implementation. With JVMPI, portable tools for Java application and Java virtual machine analysis become a viable alternative.

The Java language and environment are excellent for the development and deployment of distributed applications. Kazi et al. highlight the problem of assessing the performance of complex distributed Java applications that employ remote method invocation for communication. The authors develop new techniques for instrumenting and analyzing performance data from multiple Java virtual machines, all cooperating in the execution of a single unit of work.

Alexander et al. describe the application of a technique particularly well-adapted for Java application and Java

virtual machine performance analysis. The technique, named arcflow, is a scalable approach to recording widely differing types of performance data in a common data model, and then generating a core set of generic reports. Although not restricted only to the Java language in its application, the technique is well-suited to highly structured environments, such as the Java virtual machine and applications written in Java.

Initially, the primary focus for Java performance activity was the client. This focus mirrored the early perception of the Java language as a client technology, a view fostered by the then-widespread use of Java as a means to produce flashy animations on Web pages. Early benchmarks concentrated on measuring graphical and some computational characteristics of a virtual machine implementation. These early benchmarks have been used quite successfully for tuning graphical and core Java virtual machine activity. Gu et al. explore the use of benchmarks and tools for obtaining improved Java virtual machine performance on the client.

A notable shift in focus from client-side Java performance to the performance of Java on the server has occurred. The value of Java in making server code portable is now recognized, and server-side deployment of business logic written in the Java language is growing. For example, Web application servers allow businesses to develop code in Java that would formerly have been written in COBOL, C, or Perl, and deploy it in the form of servlets. The increasing use of the Java language in the computer science curriculum of universities has also accelerated the importance of delivering high-performance Java for servers. Dimpsey, Arora, and Kuiper discuss the challenges with, and techniques for delivering, server-side Java with competitive performance.

The power of the Java language that enables code to be written once and run anywhere was initially diminished by concerns for the expected poor performance of an interpretive language. When early implementations were compared unfavorably with the C language, many apprehensions about Java's performance were realized. Fortunately, great advances in just-in-time, or JIT, compilation have been made. In a paper by Suganuma et al. the architecture, algorithms, and capabilities of IBM's high-performing Java JIT compiler technology are described.

Dillenberger et al. describe successful efforts associated with combining the benefits of the Java language with the benefits of large-scale enterprise data processing systems. The OS/390* (Operating System/390) hardware and software architecture delivers a highly scalable, highly reliable platform for high-volume enterprise transaction processing. Blending the benefits of Java with the benefits of this mature operating environment is a significant challenge. Achieving satisfactory performance in this environment is an important objective, but still greater opportunity exists. The paper closes with a compelling discussion of a new Java virtual machine architecture that is specifically designed for supporting high-volume transaction processing.

Alpern et al. describe the Jalapeño virtual machine for servers. Jalapeño represents a new direction and possible future for Java virtual machine implementations. The Java virtual machine is implemented almost entirely in Java itself, thus eliminating some of the problems associated with porting across platforms. Since it is written in the Java language, Java-oriented optimizations tend to improve the performance of the Java virtual machine.

As the *Journal* begins its 39th year, we would like to acknowledge the support given by readers, authors, and referees that have sustained its publication. We are grateful for your support and encourage you to continue your interest and participation. It also seems appropriate to state a few facts that may escape us as we focus on the contents. First, this quarterly publication is a refereed technical journal, which means that the integrity of each paper is ensured by a process that depends on peer reviews of content, currency, and value by recognized experts within and outside IBM. Second, it is intended for the software and systems professional and applied research community worldwide. The papers are written for a technically aware readership, and we welcome submissions by knowledgeable authors from around the globe, within and outside IBM. Third, the *Journal* has around 50 000 subscribers worldwide. Of those, approximately two-thirds are technical professionals and researchers outside IBM and one-third are IBM employees; two-thirds are in the United States and one-third are elsewhere.

The next issue of the *Journal* will feature papers describing insights gained through experiences in using the SanFrancisco frameworks, along with several individual papers.

Gene F. Hoffnagle
Editor

*Trademark or registered trademark of International Business Machines Corporation.

**Trademark or registered trademark of Sun Microsystems, Inc.