CMP
United Business Media

Bonus Coverage! Windows & .NET

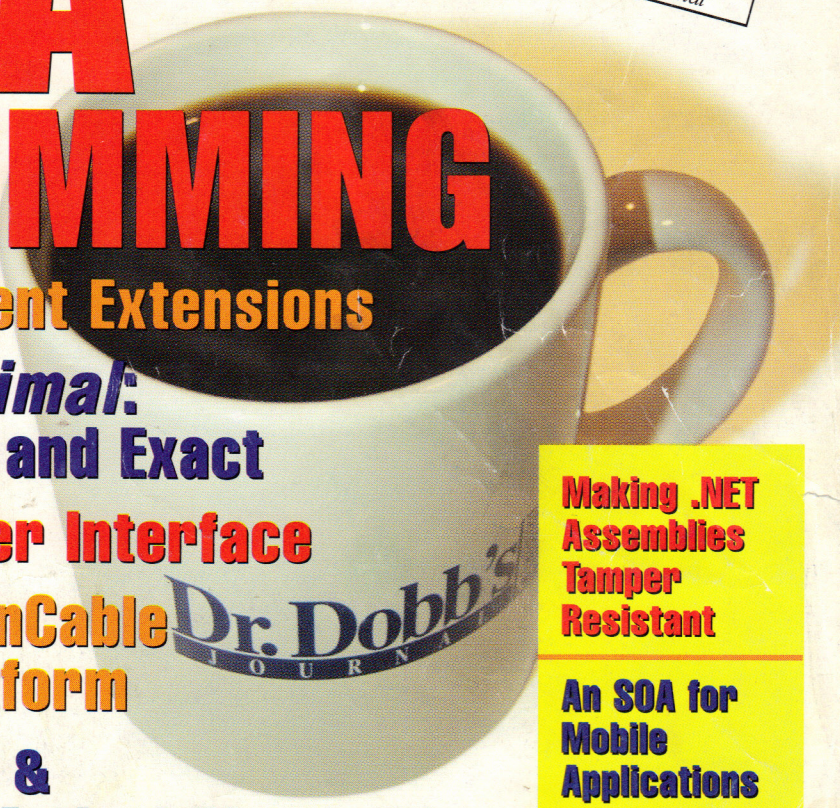
#362 JULY 2004

Dr. Dobb's JOURNAL

SOFTWARE TOOLS FOR THE PROFESSIONAL PROGR

http://www.ddj.com

JAVA PROGRAMMING



- Java Management Extensions
- Java's *BigDecimal*: Fixed, Floating, and Exact
- The JVM Profiler Interface
- Java & The OpenCable Application Platform
- Java Reflection & Smalltalk-Like Method Dispatching
- GC Is the Cygnus Native Interface
- Double-Checked Locking
- 2003: Into the Future
- cting Source-Code Plagiarism
- g ActiveX with Java
- Putting Web Services into Context

Making .NET Assemblies Tamper Resistant

An SOA for Mobile Applications

Ed Nisley Programming In the Small

Jerry Pournelle XP Service Pack 2 Release Candidate 1

CONTENTS

JULY 2004 VOLUME 29, ISSUE 7

FEATURES

EMBEDDED SYSTEMS

COLUMNS

FORUM

RESOURCE CENTER

As a service to our readers, source code, related files, and author guidelines are available at http://www.ddj.com/. Letters to the editor, article proposals and submissions, and inquiries can be sent to editors@ddj.com, faxed to 650-513-4618, or mailed to *Dr. Dobb's Journal*, 2800 Campus Drive, San Mateo CA 94403.

For subscription questions, call 800-456-1215 (U.S. or Canada). For all other countries, call 902-563-4753 or fax 902-563-4807. E-mail subscription questions to ddj@neodata.com or write to *Dr. Dobb's Journal*, P.O. Box 56188, Boulder, CO 80322-6188. If you want to change the information you receive from CMP and others about products and services, go to http://www.cmp.com/feedback/permission.html or contact Customer Service at the address/number noted on this page.

Back issues may be purchased for $9.00 per copy (which includes shipping and handling). For issue availability, send e-mail to orders@cmp.com, fax to 785-838-7566, or call 800-444-4881 (U.S. and Canada) or 785-838-7500 (all other countries). Back issue orders must be prepaid. Please send payment to *Dr. Dobb's Journal*, 4601 West 6th Street, Suite B, Lawrence, KS 66049-4189. Individual back articles may be purchased electronically at http://www.ddj.com/.

NEXT MONTH: In August, we focus on testing and debugging.

DR. DOBB'S JOURNAL (ISSN 1044-789X) is published monthly by CMP Media LLC., 600 Harrison Street, San Francisco, CA 94017; 415-905-2200. Periodicals Postage Paid at San Francisco and at additional mailing offices. SUBSCRIPTION: $34.95 for 1 year; $69.90 for 2 years. International orders must be prepaid. Payment may be made via Mastercard, Visa, or American Express; or via U.S. funds drawn on a U.S. bank. Canada and Mexico: $45.00 per year. All other foreign: $70.00 per year. U.K. subscribers contact Jill Sutcliffe at Parkway Gordon 01-49-1875-386. POSTMASTER: Send address changes to *Dr. Dobb's Journal*, P.O. Box 56188, Boulder, CO 80328-6188. GST (Canada) #R124771239. Canada Post International Publications Mail Product (Canadian Distribution) Sales Agreement No. 0548677. FOREIGN NEWSSTAND DISTRIBUTOR: Worldwide Media Service Inc., 30 Montgomery St., Jersey City, NJ 07302; 212-332-7100.

The Java Virtual Machine Profiler Interface

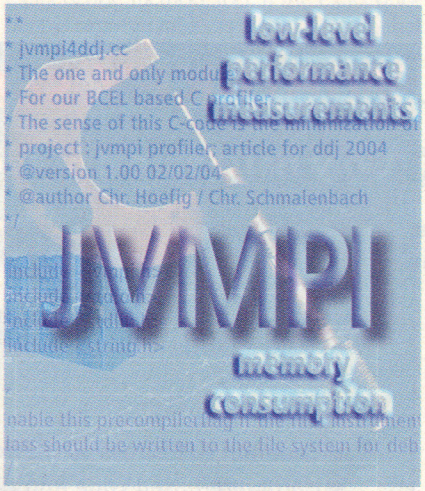
Low-level performance measurements for Java apps

Christof Schmalenbach and Christian Höfig

Java is often considered the technology of choice for highly distributed intranet and Internet applications. One reason for this is that complex issues such as security, transaction control, and data persistence are encapsulated by standardized APIs within the J2EE specification. Moreover, these issues are addressed and implemented through Java-based application servers such as IBM's WebSphere Application Server, BEA's WebLogic Server, and the open-source Jboss.

But successful e-business applications do not just provide infrastructures for development and runtime systems. They also define, monitor, and guarantee quality-of-service standards. Just as developers are supported by sophisticated IDEs, standardized monitoring APIs, and protocols support system management to meet service-level agreements.

Since 1998 and Java SDK 1.2, Sun has offered a standardized monitoring API—the Java Virtual Machine Profiler Interface (JVMPI)—for low-level performance measurements involving memory consumption, bytecode of classes being loaded, parameters of methods, and the like. Performance requirements may differ significantly between applications, but with JVMPI, there is a broad range of values to select from. A CPU-bound application (an XML parser, for instance) may best be analyzed by looking at time usage of methods or even single lines of code. JVMPI offers the *getCurrentThreadCpuTime()* function for analysis such as this. Meanwhile, database-driven web applications have other analysis needs. Application servers, for instance, often act as mediators between web servers and database back ends, and performance analysis must focus on time consumption of transactions—specifically SQL statements with long response times. In such cases, you can use JVMPI to patch (or instrument) the database JDBC driver class and extract SQL statements information on the fly.



Although intended originally for tool vendors, JVMPI can be a great tool for developers because it gives you a free view of what is going on under the hood.

Performance measuring in heavily distributed systems is complex. Often, several physical machines and application services are involved, all acting together to provide, for example, web-based transaction processing. When end users complain about an application's bad response times, low throughput, or rejected requests, getting to the cause of the problem may be difficult. Analysis must take into account all the layers between users and servers, where the business logic executes. Items to look at include web server load, J2EE server connection pools, and database index organization; or coding problems at the web layer, business layer, or in utilities, libraries, and helper classes.

Consider a typical J2EE example such as a travel agency where users request bookings. A service chain starts from the http server to a servlet, delegating to a Session Bean, activating one or more Entity Beans, and finally communicating with the database. With the exception of the web server to the application server channel, all services are directly under the control of Java processes and can be looked at with Java monitoring tools.

As an ad hoc solution, Java developers often start measuring via logging statements, putting them around code that does something interesting (like the crossing of a transaction layer). For example, a database statement call may be surrounded by timestamp information and domain-specific parameters.

Measuring performance this way has several drawbacks. You can slow down your application by including too many measuring points and spending too much time writing to log files, distorting the values you are getting. Another disadvantage of this approach is that you may end up constantly adding/removing logging statements to/from your code, cluttering it and creating extra efforts for code management and deployment. A better solution is monitoring using JVMPI.

JVMPI

JVMPI is a two-way API between the JVMPI and profiler agent (http://java.sun.com/products/j2se/1.3/docs/guide/jvmpi/jvmpi.html). With JVMPI, your profiling agent has a way to specify to the JVM what kind of events you are interested in. The JVM, in turn, can tell you (the agent) that interesting events occurred.

The agent code is a platform-specific native library that runs in the same process as the JVM. The agent code can initially (or at any later time) register with

Christof is IT Architect at IBM Business Consulting Services in Germany. He can be contacted at cschmale@de.ibm.com. Christian is a consultant and can be contacted at http://www.harung.de/.