

IMPROVE THE EFFICIENCY AND RESPONSIVENESS OF MISSION-CRITICAL JAVA APPLICATIONS

Why are so many organizations turning to Java? Because it addresses today's key IT challenge — agility. Java provides the integration and expansion capabilities that existing technologies require to meet dynamic business needs.

Java allows organizations to develop new applications in the IBM WebSphere Application Server and leverage legacy applications and CICS transactions, as well as access DB2 data. The fact that complex Java code is usually developed under tight deadlines makes performance analysis essential. Developers need a tool that quickly identifies the exact location of inefficient code and provides detailed reports on CPU usage and delays contributing to poor performance. Developers need Compuware's Strobe for Java™.

Strobe for Java™ is part of the Strobe product family. These performance management tools help IT organizations deliver efficient and responsive z/OS applications that perform optimally throughout the life cycle. Strobe for Java™ captures critical Java application performance data, pinpointing the Java packages, classes and methods that are most resource-consumptive or that contribute to delays. Strobe for Java™ also reports on time spent in resource-consumptive system services and shared objects, and reports on the use of the IBM zSeries Application Assist Processor (zAAP), if applicable.

CI62SQLJ: Java Call Stacks Find Print Threshold: 0.1 Apply

Target method	CPU %		Wait %		
	Solo	Total	Page	Total	
SQLJ2.main(java/ lang/ String[])	60.58	61.09	0.00	0.00	
Called method	SQL	Solo	Total	Page	Total
SQLJ2.SQL_MinList(Ctx)	Y	26.60	26.62	0.00	0.00
SQLJ2.SQL_MaxList(Ctx)	Y	26.33	26.35	0.00	0.00
SQLJ2.SQL_InnerJoin(Ctx)	Y	1.74	1.75	0.00	0.00

Call stack navigation	Call stack total		CPU %		Wait %	
	Solo	Total	Page	Total		
Select stack: 1 of 2	Stack and SQL	26.59	26.61	0.00	0.00	
View Stack View SQL	SQL only	26.59	26.61	0.00	0.00	

Call stack

- SQLJ2.main(java/ lang/ String[])
- SQLJ2.SQL_MinList(Ctx)
- COM/ ibm/ db2os390/ sqlj/ runtime/ DB2SQLJRResultSet.next()
- COM/ ibm/ db2os390/ sqlj/ jdbc/ DB2SQLJDBCCursor.fetch()
- COM/ ibm/ db2os390/ sqlj/ jdbc/ DB2SQLJDBCCursor.native_SQLSelect(int)

The Target (top) frame contains the Target method and its Called methods. The Navigation (middle) frame contains the stack and SQL statement totals and stack navigation controls. The Stack (bottom) frame contains the call stack detail, in this case the first of two call stacks under MinList, and its attributed SQL statements. In this example, the View Stack mode is in effect. In this mode, the forward and backward controls in the Navigation frame enable users to step from the displayed stack to the next or previous stack.

By implementing Strobe *for Java*[™], IT organizations can easily:

- understand and improve the performance of Java, WebSphere, batch, CICS and DB2 applications
- identify activity in Java applications by transaction, package, class and method, as well as the full file path that identifies the package
- view any SQL statements that were invoked on behalf of Java code
- determine whether Java code was executed via a just-in-time (JIT) compiler or whether the Java code ran interpretively
- view call stack information
- determine if the zAAP is being used effectively (if available)
- improve the performance of SQL statements using SQL analysis and predicate analysis functionality.

A SIMPLE WAY TO MEASURE AND ANALYZE APPLICATION PERFORMANCE

Measuring application performance is easy with Strobe *for Java*[™], as no changes to the application or environment are required to collect performance information. In addition, because of its low-density sampling technology, Strobe *for Java*[™] can be used with confidence in production and throughout the application life cycle.

TARGET YOUR AREAS OF INTEREST, THEN RUN YOUR REPORTS

Using ISPF panels, the user can also target specific packages, classes or methods of the Java program on which they want to focus attention and prioritize at the top of the call stack.

Once a target area has been established, Strobe *for Java*[™] performance information is presented in a hierarchy of reports

detailing where and how time is spent during a Java application's execution, so Java developers can easily locate and eliminate sources of excessive resource demand.

Strobe *for Java*[™] reports include:

- Java CPU usage and Wait Time by Called Method report, which
 - presents execution information at the package, class and method level; processing and wait are related back to either the user-targeted method or the initial method run
 - shows the package, class and methods (up to a total of eight in iStrobe) invoked by the user-targeted or initial method; invoked DB2 SQL statements are also shown
- Java CPU usage and Wait Time by Executing Method report that shows the method or service routine actually processing at the time when Strobe *for Java*[™] collects data
- Java Targeting report that shows names of search arguments used by the Java targeting function
- Java Environment report that shows class path information so developers can uniquely identify which JVM under CICS is used
- CICS API and Non-API Transaction Profile report that shows Java methods invoked under CICS.

The Program Section Usage Summary, Program Usage by Procedure and CPU, and Wait Time attribution reports also reflect Java activity, if it is detected.

The Measurement Session Data report shows the percentage of eligible Java activity that ran on the zAAP, if available. This information allows you to determine if you are maximizing the performance benefit and cost savings of the zAAP.

CI62SQLJ: Java Activity by Executing Method				
Expand all Collapse all Find Print				
Threshold: 0 Apply				
Package/Class	CPU %		Wait %	
	Solo	Total	Page	Total
Totals	1.18	1.20	0.00	0.00
Package/Class	Solo	Total	Page	Total
com/ibm/jvm/ExtendedSystem	0.01	0.01	0.00	0.00
java/util/Stack	0.01	0.01	0.00	0.00
java/util/jar/Attributes\$Name	0.03	0.03	0.00	0.00
java/io/ObjectStreamClass	0.01	0.01	0.00	0.00
java/text/resources/DateFormatZoneData	0.04	0.04	0.00	0.00
java/util/Hashtable	0.01	0.01	0.00	0.00
java/util/GregorianCalendar	0.01	0.01	0.00	0.00
java/lang/ClassLoader	0.03	0.03	0.00	0.00
java/net/URLClassLoader\$ClassFinder	0.01	0.01	0.00	0.00
java/util/ResourceBundle	0.01	0.01	0.00	0.00
Package/Class	Solo	Total	Page	Total
java/io/BufferedInputStream	0.06	0.06	0.00	0.00
java/io/ObjectInputStream	0.10	0.10	0.00	0.00
java/io/ByteArrayInputStream	0.03	0.03	0.00	0.00

The Java Activity by Executing Method report presents CPU and Wait activity observed in a measurement at the most deeply nested level of Java method invocation. The top level of the report shows the Package/Class totals.

Package/Class	Solo	Total	Page	Total
▼ java/lang/System	0.01	0.01	0.00	0.00
Method				
arraycopy(java/lang/Object,int,java/lang/Object,int,int)	J	0.01	0.01	0.00
Package/Class				
▼ java/util/jar/Attributes	0.03	0.03	0.00	0.00
Method				
read(java/util/jar/Manifest\$FastInputStream,byte[])	M	0.03	0.03	0.00
JITted total	J	0.01	0.01	0.00
Interpreted total	I	0.02	0.02	0.00
Package/Class				
▼ java/lang/Character	0.04	0.04	0.00	0.00
Method				
<clinit>()	I	0.01	0.01	0.00
toUpperCase(char)	I	0.03	0.03	0.00
Package/Class				
▼ java/util/Vector	0.01	0.01	0.00	0.00
Method				
addElement(java/lang/Object)	I	0.01	0.01	0.00

The detail levels of the Java Activity by Executing Method report contain the Method totals and, in the case of Mixed Execution mode (M), the JITted and Interpreted subtotals.

The Java Activity by Called Method report presents the CPU and Wait activity attributed to the Target (initial Java caller) and Called (initial Java callee) methods observed on the Java Call Stack when the JVM is active during a measurement session. The top level of this report shows the CPU and Wait activity by Package/Class. The detail level of the Java Activity by Called Method reports shows the CPU and Wait activity by method. Users click the stacks icon to the left of a method to invoke the Java Call Stacks pop-up window. The Called Method report page remains on the desktop along with the pop-up window.

CI62SQLJ: Java Activity by Called Method						Expand all	Collapse all	Find	Print
						Threshold:	0.1	Apply	
						CPU %		Wait %	
						Solo	Total	Page	Total
Totals						64.61	65.14	0.00	0.00
Package/Class		Solo	Total	Page	Total				
▼ com/ibm/ics/server/Wrapper		2.56	2.56	0.00	0.00				
Method									
<clinit>()	SQL	2.42	2.42	0.00	0.00				
main(java/lang/String[])		0.14	0.14	0.00	0.00				
Package/Class									
▼ java/lang/ClassLoader		1.18	1.18	0.00	0.00				
Method									
loadClass(java/lang/String)	SQL	1.07	1.07	0.00	0.00				
getSystemClassLoader()		0.12	0.12	0.00	0.00				
Package/Class									
▼ SQLJ2		60.72	61.24	0.00	0.00				
Method									
main(java/lang/String[])	SQL	60.72	61.24	0.00	0.00				

APM PROBLEM SOLVER SERVICE

The Application Performance Management (APM) Problem Solver service assists clients in identifying and resolving specific performance problems in their mainframe-centric, business-critical applications.

Using Compuware's industry-leading products, our experienced Delivery Consultants work closely with a client's IT personnel to measure an application's performance, identify performance improvement opportunities and make recommendations for implementing solutions.

With the APM Problem Solver services, organizations can not only resolve problems quickly and effectively, but also gain the skills necessary to prevent future application performance degradation.

“Our current monitor basically reflects response time, alerting us when the system runs slowly and when we're not meeting service levels. Strobe actually tells us how the resources are used and what resources are used.”

— Joe Reimann, IT Project Coordinator, Comerica

Compuware's Delivery Consultants are experts in managing APM projects. They have the latest knowledge of APM methodology and technologies. They average ten or more years experience in OS/390 and z/OS application or system programming, database administration and/or application performance tuning.

Ask your sales representative for information on our Mainframe APM offerings that support your operating environment.

To learn more about Compuware Strobe, visit:
compuware.com/strobe

Mainframe APM: Strobe Products
Z/OS OPERATING ENVIRONMENT <ul style="list-style-type: none">• Strobe• iStrobe• AutoStrobe
SUBSYSTEM AND DATABASE ENVIRONMENTS <ul style="list-style-type: none">• Strobe <i>for CICS</i>• Strobe <i>for DB2</i>• Strobe <i>for IMS</i>• Strobe <i>for WebSphere MQ</i>• Strobe <i>for WebSphere Application Server</i>• Strobe <i>for CA IDMS</i>• Strobe <i>for ADABAS/NATURAL</i>• Strobe <i>for UNIX System Services</i>
LANGUAGES <ul style="list-style-type: none">• Strobe <i>for Java™</i>• Strobe <i>for COBOL</i>• Strobe <i>for C/C++</i>• Strobe <i>for PL/I</i>• Strobe <i>for FORTRAN</i>• Strobe <i>for CA Gen</i>

Compuware Corporation, the technology performance company, provides software, experts and best practices to ensure technology works well and delivers value. Compuware solutions make the world's most important technologies perform at their best for leading organizations worldwide, including 46 of the top 50 Fortune 500 companies and 12 of the top 20 most visited U.S. web sites. Learn more at: compuware.com.

Compuware Corporation World Headquarters • One Campus Martius • Detroit, MI 48226-5099

© 2011 Compuware Corporation

Compuware products and services listed within are trademarks or registered trademarks of Compuware Corporation. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

11.30.11 20117sd

