SYS-CON

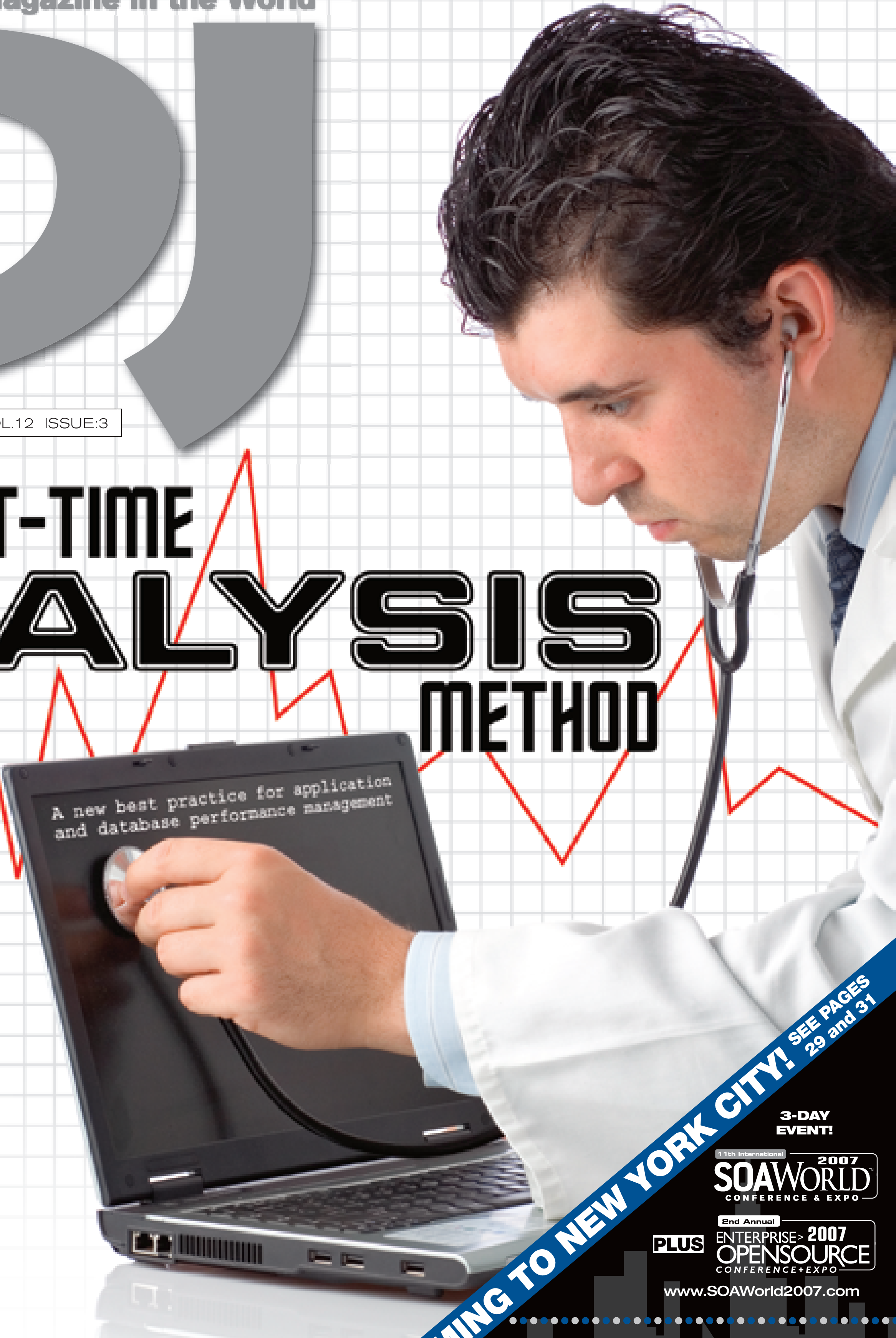
DELIVER COMPOSITE APPS WITH JAVA, WS-BPEL & SOA PAGE 12

No. 1 *i*-Technology Magazine in the World

JDJ

JDJ.SYS-CON.COM VOL.12 ISSUE:3

WAIT-TIME ANALYSIS METHOD



A new best practice for application and database performance management

PLUS...

- Top Cluster Considerations
- Looking Inside Stuck Threads

RETAILERS PLEASE DISPLAY
UNTIL MAY 31, 2007

$5.99US $6.99CAN

0 09281 01751 6 04

Desktop
Core
Enterprise
Home

JAVA DEVELOPER'S JOURNAL

To submit a proposal for an article, go to
http://jdj.sys-con.com/main/proposal.htm

Subscriptions

For subscriptions and requests for bulk orders, please send your letters to Subscription Department:

888 303-5282
201 802-3012
subscribe@sys-con.com

Cover Price: $5.99/issue. Domestic: $69.99/yr. (12 Issues) Canada/Mexico: $99.99/yr. Overseas: $99.99/yr. (U.S. Banks or Money Orders) Back Issues: $10/ea. International $15/ea.

Editorial Offices

SYS-CON Media, 577 Chestnut Ridge Rd., Woodcliff Lake, NJ 07677
Telephone: 201 802-3000 Fax: 201 782-9638

Java Developer's Journal (ISSN#1087-6944) is published monthly (12 times a year) for $69.99 by SYS-CON Publications, Inc., 577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677. Periodicals postage rates are paid at Woodcliff Lake, NJ 07677 and additional mailing offices. Postmaster: Send address changes to: Java Developer's Journal, SYS-CON Publications, Inc., 577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677.

Worldwide Newsstand Distribution
Curtis Circulation Company, New Milford, NJ
For List Rental Information:
Kevin Collopy: 845 731-2684, kevin.collopy@edithroman.com
Frank Cipolla: 845 731-3832, frank.cipolla@epostdirect.com

Newsstand Distribution Consultant
Brian J. Gregory/Gregory Associates/W.R.D.S.
732 607-9941, BJGAssociates@cs.com

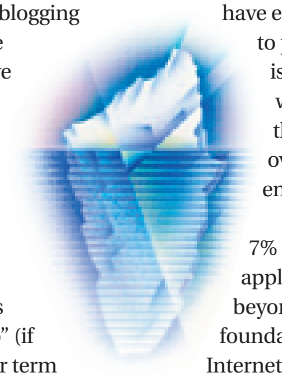
Is Blogging Just the Tip of the Co-Technology Iceberg?

Jeremy Geelan

Gartner says that the total number of bloggers will peak during the first half of this year at around 100 million, causing John R. Patrick to ask rhetorically whether spring 2007 truly is The Peak of Blogging?

It isn't, he says, in answer to his own question. "Blogging," he writes, "is just beginning!"

The significance of blogging is not the word "blog," whether used as a verb or a noun, but its role as a harbinger of the game-changing Web-as-platform revolution. In particular, the migration of blogging from the individual toward the enterprise represents a massive validation of those like Professor Andy McAfee who argue that Enterprise Web 2.0 is already a reality.



Put crudely, by embracing blogging, corporate America gave a big wet kiss to Web 2.0 (if you prefer Dale Dougherty's handy term). To the "New Web" (if you prefer a somewhat simpler term and one that no one owns).

One part of the Gartner report that made me think of my earlier "Are We Blogging Each Other to Death?" editorial (http://java.sys-con.com/read/157184.htm) and that was the bit about how less than 2% of all Internet users are frequent contributors to content on the Web. This is the kind of statistic that always exercises anyone involved with building Web-based applications, because it's a classic full-glass/empty-glass situation: do we rejoice at the fact that 98% of the marketplace may yet come round, or bemoan that only 2% have seen the light?

Wikipedia provides us with a good working example. According to the John Musser/Tim O'Reilly report "Web 2.0 Principles and Best Practices," it's known what percentage of the registered Wikipedia users are contributors – around 7%. If like me you view Wikipedia as an application rather than a destination site, that means that 70,000 people or more are actively using the Wikipedia application. That's a lot of users for a fairly sophisticated app. Kudos to EMC's Cornelia Davis for nailing one very simple way of realizing how significant a number: "that is more than twice the number of people working for my employer (EMC has around 31,000 employees)."

So the challenge for today's software developers is to achieve for their apps the same kind of buy-in that already exists out there on the New Web. Because unless a company has the same kind of percentage of its intranet users actively contributing content, my contention is that it will swiftly be overtaken by those companies that do. Small wonder then that Sam Palmisamo (who famously has his own avatar) is rumored to have enjoined his fellow IBM execs to participate in Second Life. He is probably interested in seeing whether the 7% figure operates there, too; if it did, there would be over 25,000 Big Blue avatars by the end of this year!

As for the wider New Web, a 7% participation rate in any one application on a global scale would, beyond a shadow of a doubt, be the foundation of the next Microsoft. Total Internet users are estimated at 1.1 billion, so the 100 million people who blog (if we go by Gartner's figures) takes us far above the 7% mark (77 million). But no one company owns blogging, any more than any one company owns Web mail, or Instant Messaging, or photo sharing.

What kind of app is missing from the New Web landscape? What kind of app would attract 77 million users? One that adheres to all the tried and true principles of first-rate co-technology (my turn to coin a term now), plus adds functionalities and leverages emergent methodologies like automatic semantic video tagging, audio search, and social bookmarking, but above all, one that solves a problem not already being solved; or solves one already being solved, but 10 times better.

("100% Spam Free E-mail" would be one obvious example of the latter; "Google for Memories" would be a less obvious example of the former.)

For those many Webpreneurs and innovators who believe they're on to exactly that, I say only this: Stay with it. *The i-Technology world can do way, way better than mere blogging. I just know it.*

Jeremy Geelan is Sr. Vice-President, Editorial & Events of SYS-CON Media. He is Conference Chair of the AJAX-World Conference & Expo series and of the "Real-World Flex" One-Day Seminar series.

jeremy@sys-con.com

Achieve Point-and-Click SOA™

exteria™
Business Automation Platform

Get on the SOA fast track!

NEXT STOP: extentech.com

FREE Exteria Downloads

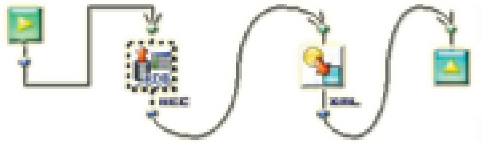
Demos and Whitepapers

We're addicted!
Now that we are going to output to the Web, ExtenXLS has saved us a tremendous amount of time and work."

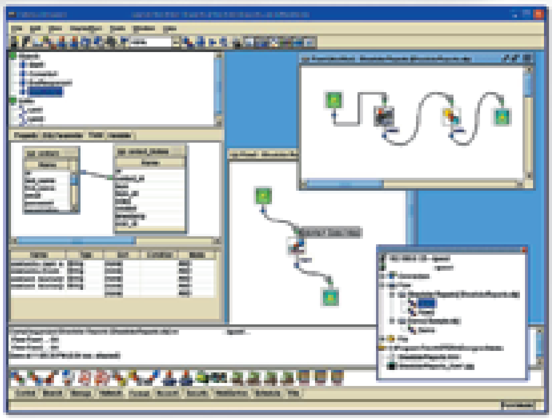
-Steve T, System Architect,
B2B Application Development
Pfizer, Inc.

High-Speed Productivity

Exteria is the ideal modeling tool and web services platform for rapid SOA implementations.



Model data flows graphically in the code-free designer then publish with a click to XLS, PDF, SOAP, XML, and more -- all without writing any code!



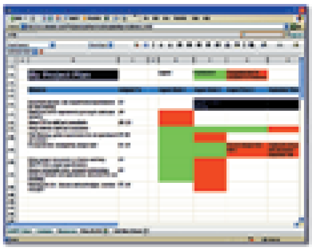
The Exteria data flow modeler features dozens of connectors including LDAP, SOAP & XML

Spreadsheet Automation

ROI is all about re-use. There is no need to re-code the business logic already available in spreadsheets. Leverage this asset by mapping data flows to formula cells and publishing them as RSS feeds, REST API and SOAP web services, or to a web spreadsheet.

works with:
sheetster™

Integration with Sheetster.com web spreadsheets enables realtime data feeds and collaboration.



includes:
EXTENXLS 5™
JAVA SPREADSHEET TOOLKIT

Includes the best Java Spreadsheet Automation available... ExtenXLS 5.1!

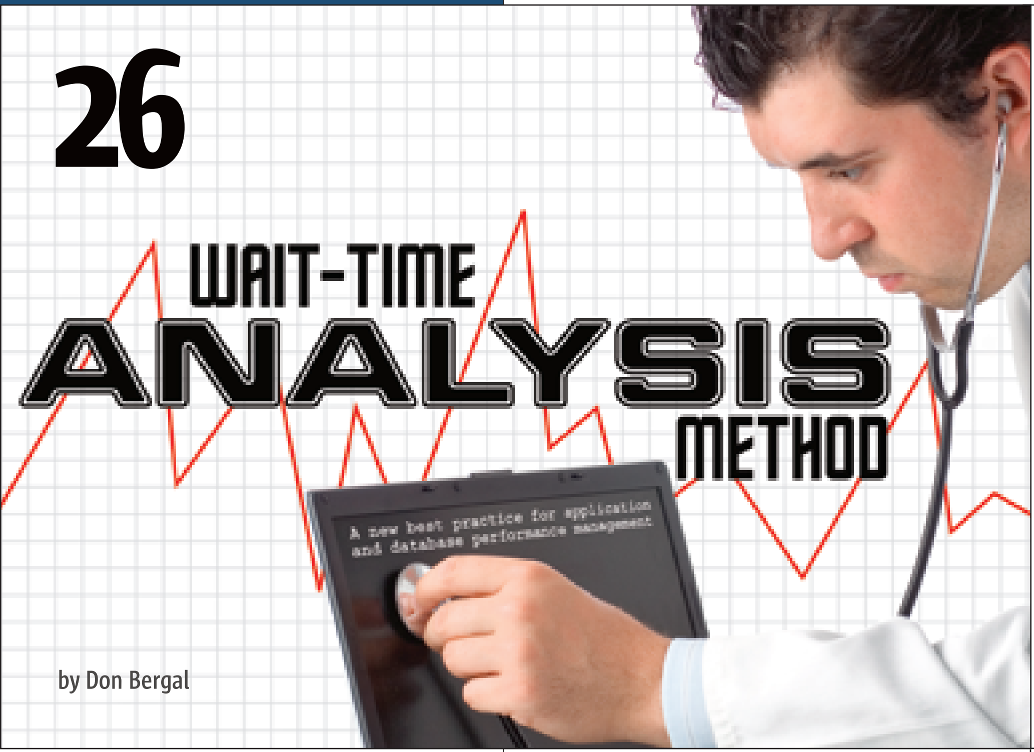
extenTECH™
exceed expectations

sales@extentech.com
call 415.759.5292
http://extentech.com/itsg

MARCH 2007 VOLUME:12 ISSUE:3

JDJ contents

JDJ Cover Story


A new best practice for application
and database performance management

Feature

JDJ (ISSN#1087-6944) is published monthly (12 times a year) for $69.99 by SYS-CON Publications, Inc., 577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677. Periodicals postage rates are paid at Woodcliff Lake, NJ 07677 and additional mailing offices. Postmaster: Send address changes to: JDJ, SYS-CON Publications, Inc., 577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677.

JDJ Industry News

Sun Microsystems Upgrades Java Enterprise System

(*Santa Clara, CA*) – Sun Microsystems, Inc., has announced the availability of the latest version of the Sun Java Enterprise System (Java ES). Java ES 5.0 is a unique and comprehensive set of subscription-based services that work in conjunction with top enterprise operating systems(OS) like Solaris 10 OS to combine software, support, and professional and educational services in a single package, for one single price.

The new Java ES v5.0 is now modular and provides greater choices for enterprises to consolidate their IT infrastructure on an integrated, open, and standards-based software system. The new system allows customers to sign-on for the core Java ES 5.0 Base and then maximize the value of their IT spending by easily adding on subscriptions to Sun's industry-leading Sun Java System Identity Manager or Sun Java Composite Application Platform Suite (Java CAPS). This unique flexibility gives customers greater choice, better value and lower TCO.

AdventNet Updates Applications Manager

(*Pleasanton, CA*) – AdventNet, Inc. has unveiled the next update of its Applications and Systems management solution. ManageEngine Applications Manager now provides in-depth support for BEA WebLogic Integration Servers. With this enhancement, enterprises can ensure that the critical infrastructure that helps unlock data from legacy and packaged applications, runs efficiently.

ManageEngine Applications Manager also adds deep support for Java runtime Monitoring. This capability helps users to monitor their custom Java Applications and ensure high uptime.

Applications Manager provides in-depth monitoring of

- Application Servers (WebLogic, WebLogic Integration, WebSphere, JBoss, Tomcat, Microsoft .NET, Oracle AS)
- Databases (SQL Server, Oracle, IBM DB2, MySQL)
- Servers (Windows, Linux, Solaris, IBM AIX, HP Unix, FreeBSD, Tru64 Unix, Mac OS)
- J2EE Web Transactions, Java Runtime, mail servers, Web sites and custom Java, JMX, J2EE applications with performance management, fault management and notification capabilities.

Sun Announces Support for the Next Generation Mobile Java Platform Across Entire Mobile Product Line

(*Santa Clara, CA*) – Sun Microsystems, Inc., has announced full support for Mobile Service Architecture (MSA), the next generation Java platform for mobile phones, is available immediately across its mobile product line and the NetBeans(TM) Integrated Development Environment (IDE).

Sun, along with 13 other companies, including operators, OEMs and software vendors, participated in the development of MSA through the Java Community Process (SM). The goal of MSA is to provide greater handset platform functionality to application developers and reduce implementation variation through the creation of standard Application Programming Interfaces (APIs) and specific component clarifications. The new capabilities of MSA allow for the creation of compelling mobile applications and services. Examples of these capabilities include: 3D graphics, personal information management, Bluetooth, animation, Web services, location services and payment services. More information about MSA can be found at http://java.sun.com/javame/reference/docs/msa_datasheet.pdf.

Reuters and IBM Collaborate to Give Financial Markets Customers Better Trading and Integration

(*London / New York*) – Reuters and IBM are collaborating to make it simpler for financial markets customers to integrate Reuters' core software platform, the Reuters Market Data System (RMDS), with IBM WebSphere Software. Joint customers can now access and integrate the broadest array of market data with existing information to manage massive data volumes better and improve trading.

The move will allow the content and services on RMDS to completely integrate with applications that use IBM's messaging software. It has been made possible by the introduction of three new products from Reuters that will extend the reach of RMDS into clients' front, middle and back office applications – the RMDS Multicast Distributor, the RMDS Enterprise Distributor (eDist), and the RMDS Internet Finance Platform (iFP).

SYS-CON MEDIA

President and CEO:
Fuat Kircaali fuat@sys-con.com

President and COO:
Carmen Gonzalez carmen@sys-con.com

Senior Vice President, Editorial and Events:
Jeremy Geelan jeremy@sys-con.com

Advertising

Vice President, Sales and Marketing:
Miles Silverman miles@sys-con.com

Advertising Sales Director:
Megan Mussa megan@sys-con.com

Associate Sales Manager:
Corinna Melcon corinna@sys-con.com

Events

Events Manager:
Lauren Orsi lauren@sys-con.com

Editorial

Executive Editor:
Nancy Valentine nancy@sys-con.com

Production

Lead Designer:
Tami Lima tami@sys-con.com

Art Director:
Alex Botero alex@sys-con.com

Associate Art Directors:
Abraham Addo abraham@sys-con.com
Louis F. Cuffari louis@sys-con.com

Web Services

Information Systems Consultant:
Robert Diamond robert@sys-con.com

Web Designers:
Stephen Kilmurray stephen@sys-con.com
Richard Walter richard@sys-con.com

Accounting

Financial Analyst:
Joan LaRose joan@sys-con.com

Accounts Payable:
Betty White betty@sys-con.com

Customer Relations

Circulation Service Coordinator:
Edna Earle Russell edna@sys-con.com

Top Cluster Considerations

by Ruth Willenborg and Stan Cox

Design your application to work in a cluster and be sure to test it carefully to avoid cluster gotchas

One of the greatest strengths of Java Platform 2, Enterprise Edition (J2EE) application servers is the ability to scale solutions to meet increased performance and availability demands. The inherent clustering and failover capabilities built into products such as the IBM WebSphere Application Server Network Deployment Edition take care of most of the dirty work; however, there are important application considerations that can't be overlooked. If an application isn't designed for cluster awareness, functional or performance issues can surface when deploying your application to a cluster.

Expecting your application to run only in a single server Java Virtual Machine (JVM) is self-limiting, and with a little more upfront work, you can ensure cluster readiness, and be more confident that your application will scale when the need arises. This paper discusses design and implementation considerations for cluster awareness. While many of these best practices are the tried-and-true best practices for any J2EE application, you'll see how essential they become when you move to a cluster. In addition, there are more subtle considerations – often realized the hard way – that cause an application to fail in a cluster.

When an application is moved into a clustered environment, multiple versions of the application are now executing, with some type of workload management distributing and balancing the work. The application may be running on different JVMs on the same physical server or across different servers. Data, resource sharing, and timing have to be carefully designed. Subsequent requests to the same application may end up executing on different servers.

This article discusses the following areas that require special consideration when designing your application for cluster awareness.

1. Follow user session state best practices
2. Design configuration data access to avoid local files and in-memory updates
3. Design application and database for data concurrency and idempotence
4. Maximize local object accesses within tiers
5. Leverage messaging to enable background and/or batch processing
6. Design and partition caches
7. Configure shared resources appropriately
8. Use application server provided workload management and replication
9. TEST, TEST, TEST

Follow User Session State Best Practices

HTTP is a stateless protocol requiring the user state to be stored on the server side using HTTP sessions and other J2EE components such as Stateful Session EJBs. When your application is moved into a clustered environment, different requests from the same user may go to different application servers. This means that all application servers need to have access to these types of session data to maintain availability across user requests. Application servers like WebSphere provide a sophisticated infrastructure to support sharing HTTP session data across servers. This includes performance optimizations in the workload routing and caching so that under normal operating situations, the HTTP session object is actually resident on the server that the request is sent to, eliminating the need for the runtime to repopulate the session object from the database or shared memory.

Though your application server may provide algorithms that minimize the need for a session to be repopulated from the database, any HTTP session replication and failover capability requires adherence to several best practices to perform effectively:

1. A session should be small and avoid complex object graphs
2. Session objects must be serializable
3. References to unnecessary objects that are dependent on the current runtime environment should be avoided
4. Session wrappers must call the set attribute appropriately

Some techniques that can be helpful in keeping your session small:

- Make sure it's state data, not history
- Make sure it's user state data, not application data that's not user-specific
- Keep a key to the data
- Re-create the data on failover rather than storing it in the database

An example of a technique using transient variables to enable WebSphere to serialize objects selectively and re-create data is discussed by Brown and Botzum in "Improving HttpSession Performance with Smart Serialization."[1]

Design Configuration Data to Avoid Local Files and In-Memory Updates

Another type of data that's common is application configuration data or properties. Simple single-server solutions often read configuration data from a local file and store it in a Java object. This is commonly a set of static fields or a singleton. The configuration data is then accessed and updated by the application and may be persisted back to the local file.

These techniques don't work in a cluster. Remember, the configura-

Ruth Willenborg is a senior technical staff member in IBM's WebSphere Technology Institute working on virtualization. Prior to this assignment, Ruth was manager of the WebSphere performance team responsible for WebSphere Application Server performance analysis, performance benchmarking, and performance tool development. Ruth has over 20 years of experience in software development at IBM. She is co-author of *Performance Analysis for Java Web Sites* (Addison-Wesley, 2002).

rewillen@us.ibm.com

> “To run a cluster successfully, an application has to be tested in a clustered environment to expose any subtle problems that can arise, including edge cases such as simultaneous logins from the same user, machine, or application server”

tion data needs to be accessible by the application running on different servers, and any changes have to be visible to all running copies of the application.

Common techniques that work in a cluster include using a database or LDAP server to keep the configuration data. If your design uses a file system, you'll have to set up file replication techniques using ftp or a shared file system. Remember, if you use any of these techniques, you'll need to document the mechanism your application requires since these are not capabilities supplied by the application server runtime.

Design Application and Database for Data Concurrency and Idempotence

Once an application is clustered, there are now multiple copies of the application working against the same shared runtime data. This will usually increase the amount of concurrent access to data. It can also mean that the same user can login and make requests simultaneously from multiple browsers. This increased data concurrency can lead to invalid requests and potential data corruption. When you design your application for database access, appropriate levels of locking within transactions are required to prevent data corruption while also minimizing the lock levels to prevent deadlocks. Concurrent access deadlock problems may not show up until you move to a cluster.

For example, Message Driven Beans (MDBs) that access data in a random order with exclusive locks will work in a single application server environment, but will often be deadlocked when moved to a cluster. MDBs need to have fixed order for any data or tables that are locked exclusively. This can be done by sorting the data before accessing. Another alternative when record processing order doesn't matter is to partition the queues, allowing multiple consumers of the requests.

Another challenge for a clustered application is ID generation for database keys. If high performance is needed, one common design pattern is to add a table to the database that's used to allocate pools of keys for each cluster.

Maximize Local Object Accesses Within Tiers

When an application is running as a single instance all accesses are local. Moving to a clustered environment means, however, that accesses to EJBs, Web Services, and other “remotable” components may no longer be local. Remote object invocations are expensive compared to local calls, requiring request and return object serialization and de-serialization.

Care should be taken to ensure as much local object access as possible:

1. Deploy all application components in the same tier, when feasible. For example, Web and EJB components in the same tier will ensure local access to EJBs when feasible. Sometimes, security or other requirements preclude this.
2. Use the EJB “local” APIs to avoid a request and response object pass by value semantics (requiring serialization and de-serialization).
3. Design coarse grain APIs between components to reduce application “chattiness”, but at the same time ensure that only the minimum required data is sent and retrieved between components.

Leverage Messaging to Enable Background and/or Batch Processing

Performance is often the requirement that motivates moving from a single instance to a clustered environment. To further improve performance and reduce the load on contentious resources the application can be modified to perform some work asynchronously. For example, an order processing application may choose to process inventory updates in the background. When a new order is being processed, updating the inventory data during the order transaction isn't essential. Processing the inventory data synchronously actually increases the response time for the user and further burdens the database and other resources in the transaction.

In J2EE, JMS can be used in a transaction to create a durable and reliable message for background processing. MDBs are usually used for message processing and provide a high-performing mechanism for asynchronous updates. Often MDBs allow work to pile up and do updates and other processing in “batch” mode (many updates at once) to improve performance further. At times this can even be done when there's a reduced load on the application.

Design and Partition Caches for Cluster Awareness

Another important area to consider for clustered applications is caching. Caching heavily used data can significantly improve response time, as well as reduce processing time and the load on resources such as databases. Most application servers provide caching capabilities in various flavors such as EJB “read-only beans,” Java object caching, Web Services caching, servlet/JSP/JSP tag caching, etc.

Some application servers support these caching facilities across a cluster. This requires complex mechanisms that provide key features such as invalidating cached data across systems when it's updated, sharing cached data across a clus-

J. Stan Cox is a senior engineer with IBM's WebSphere Application Server performance group. In this role, he has worked to significantly improve WebSphere performance for J2EE applications including Web applications, Enterprise Java Beans, and Web Services. His current focus is WebSphere foundation performance. Stan holds a BSCS from Appalachian State University and a master's degree in computer science from Clemson University.

stancox@us.ibm.com

"When designing your application, there is a set of best practices that should be followed so your application works and performs when moved to a cluster environment"

ter for data that's very expensive to retrieve, etc. Using the capabilities provided by your application server is the preferred approach. Leveraging these facilities can greatly improve performance and reduce interruption in processing due to failovers.

In addition, there are special considerations the designer must be aware of when leveraging application caching in a cluster. As data is updated and invalidated in one cache, there are policies governing when and how often the data is updated across the cluster of caches. Common examples include EOS (end of service) updates and TBW (time-based writes). With an EOS policy, the data is synchronized at the end of a request. With TBW multiple updates are synchronized in a batch mode at certain time intervals. This of course leads to potentially longer time frames where the data isn't synchronized. TBW provides a higher-performance alternative for application data, such as a product catalog that doesn't have high integrity requirements and isn't updated very frequently. Customer or Order data, on the other hand, would generally require a more timely approach such as EOS. So the application design must take into account the caching policies to achieve the desired behavior and performance.

Data partitioning mechanisms are provided by some application servers such as WebSphere Extended Deployment (XD). These facilities can be used to partition data access from HTTP, EJBs, JDBC, etc. to dedicated systems in the cluster. For example, in the order processing example given above, the inventory database can be partitioned and accessed by a dedicated set of EJBs. This type of partitioning can significantly reduce data concurrency and locking while providing extended options and performance for caching. Be aware that leveraging WebSphere XD data partitioning will require changes to your application.

Consider Cluster Impacts on Shared Resources

At deployment time, you have to make sure that the shared resources are configured appropriately to handle the additional load. For example, if you've configured a single application server instance to have 20 pooled connections to the back-end database for optimal performance, this number will be multiplied by each server in the cluster. Thus a cluster of five application servers would result in 100 pooled connections. This is probably too many. Be aware of this multiplicative effect for all shared resources and pools.

Another potential gotcha in a cluster with multiple physical machines is the server clock time. It's important to synchronize the system clocks to avoid unwanted behavior due to session timeouts on failover, data time-stamping, or other application-specific time processing.

Use Application Server-Provided Workload Management and Replication

Application servers like WebSphere provide capabilities for cluster scalability, including workload management and failover, HTTP session affinity, HTTP session and cache replication services, etc. As we discussed in the sections on HTTP session and caching, these services are both complex to implement and can have a significant impact on overall cluster performance. Our recommendation is to leverage the platform cluster capabilities over trying to roll-your-own solutions.

Additionally, conflicts may arise if you try to roll-your-own workload management and affinity techniques and then combine them with your application server capabilities. Except in highly unusual circumstances, use the platform capabilities.

TEST, TEST, TEST

To run a cluster successfully, an application needs to be tested in a clustered environment. This will help expose many of the subtle problems that can arise such as the ones discussed above. It's also important to test edge cases such as simultaneous logins from the same user, machine, or application server. Additionally, you'll want to test failover scenarios as described by David Purcell in "Moving to a Cluster."[2]

Summary

When designing your application, there is a set of best practices that should be followed so your application works and performs when moved to a cluster environment. Your application server platform may provide powerful capabilities such as workload management and session replication but if your application session data isn't serializable, or your application writes configuration data to the local file system, your application is going to fail in a cluster. So, design your application to work in a cluster and be sure to test it carefully to avoid the cluster gotchas.

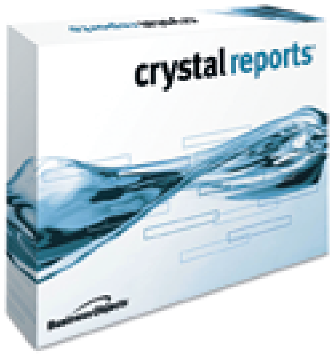
Acknowledgements

The authors would like to thank Andrew Spyker, Tom Alcott, Matt Hogstrom, Ken Hygh, and Tony Tuel for sharing their first-hand experiences in how to design applications to work in a clustered environment.

References

[1] *Improving HttpSession Performance with Smart Serialization*, Kyle Brown and Keys Botzum, IBM developerWorks, 11/23/03; http://www-128.ibm.com/developerworks/websphere/library/bestpractices/httpsession_performance_serialization.html

[2] Moving to a Cluster, David Purcell, *Java Developers Journal*, December 2004, http://jdj.sys-con.com/read/47354.htm


crystal reports

Deliver Composite Applications with Java, WS-BPEL & SOA

Supporting the complete lifecycle

by Kevin Schmidt, Prabhu Balashanmugam, and Gopalan Raj

Java is an outstanding language for building components, services, and many applications that are vendor and platform neutral. The vast adoption of Java technology by the industry in the past decade is a testament to the power of Java. Development of new applications, services, and components using Java is not going away, but many organizations have progressively moved to the next phase in maturing their IT Infrastructure. This phase is driven by many factors including how businesses operate today, having to constantly adjust to market trends, and that IT has moved from being a support organization to being the backbone of business and, hence, needs to keep pace with the organization. Continuous and faster alignment with changing business needs, time-to-market, and cost are the factors that determine success in this phase.

There are some technologies that are starting to play a critical role in this phase, for example, service-oriented architecture (SOA) is a key enabler. Java EE technology is a natural service-enabler of existing applications, thereby forming the foundation of SOA. Service-enabled applications create the opportunity to compose functions from disparate and cross-functional applications to model business processes that transcend application and enterprise boundaries. Web Services Business Process Execution Language (WS-BPEL) provides a faster way to compose and orchestrate services by reuse. Java and WS-BPEL complement each other perfectly and provide a solid foundation for integrating services and delivering composite applications.

This article will briefly explain what these technologies are and how they can work together to improve developer productivity and business agility.

Kevin Schmidt is the director of product management for SOA and Business Integration Software at Sun Microsystems, Inc. He has 17 years of experience in the software industry in roles that include product management, professional services, pre-sales, and development. Most recently he has focused on composite applications using Java, Web services, SOA, BPM, and related technologies.

kevin.schmidt@sun.com

The Technologies – Java, WS-BPEL, and SOA

There are no globally accepted definitions for the technologies that this article will explore. So let's clarify them now as a common understanding is needed before getting into the details.

Service-oriented architecture is a technical pattern for implementing cohesive and loosely coupled business and technical functions with well-defined interfaces. Such services are consumed through the details specified in the interface and without any knowledge about the implementation.

While the SOA-based infrastructure model shown in Figure 1 has a few, but well-defined, layers of services, in reality there may be many more layers, as the services are reused and composed to create coarser-grained services. As illustrated in the diagram, services can consume each other to provide layers of services and such a model can be implemented using any language including Java. The service is not technology-dependent as long as it can be consumed through the well-defined interface. Java EE 5 is a set of coordinated technologies that enable solutions for developing, deploying, and managing server-centric applications. WS-BPEL is an XML-based execution language that can be used to compose the coarse-grained services into broader services or complete applications.

Tough Decisions

Technology without the right set of tools often does more damage than good. Identifying the right set of tools for a technology is as important as choosing the technology.

The science of delivering composite applications becomes more of an art when architects try to understand when to switch from Java to WS-BPEL. This decision often determines the agility of the composite application.

The Right Set of Tools

Sun Microsystems provides a bundle of tools and servers that we will be using for this article to build our composite application, specifically the Java EE 5 SDK Update 2 Tools Bundle. This can be

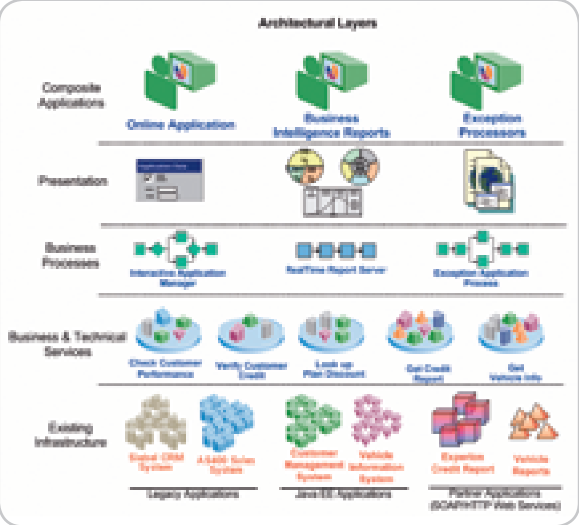


Figure 1 SOA-based infrastructure

found at http://java.sun.com/javaee/downloads/ and includes:

- Java EE 5 Application Server
- NetBeans 5.5 with NetBeans Enterprise Pack 5.5
- Java Business Integration (JBI) runtime and WS-BPEL Service Engine

This bundle provides the integrated tooling and runtime environments needed to build, deploy, and manage composite applications using Java and WS-BPEL. The tooling, in addition to a full Java SE/EE development environment, includes a WS-BPEL editor and debugger, WSDL and XSD editors, and a project system for creating services and composing them into applications. The runtime provides the Java EE 5 application server but also the WS-BPEL engine needed to execute the business processes created in the WS-BPEL editor.

In effect this bundle provides the capability to support the complete life cycle to develop, deploy, and manage applications and business processes, composing services from Java EE applications and Web services.

Putting the Technology into Action

Use Case

To put the technology and the tools in context, let's take a scenario that you must be very familiar with, a scenario in which a new external-facing application is being developed iteratively by reusing functions from existing Java EE applications and external Web services.

In the architecture depicted in Figure 2, there are three existing Java EE Applications: 1) the *Performance Evaluator* – provides evaluations on the fiscal performance of existing customers; 2) the *Vehicle Information Server* – evaluates market conditions and vehicle history and serves a detailed report on the worthiness of the vehicle; and 3) the *Financial Index Server* – provides the current running rates and values that should be used to compute the loan details.

The new application should provide a service to receive loan applications over the Web and respond with a decision. The loan decision should be arrived at by applying predefined processing rules on the following data: 1) applicant's financial worthiness, 2) vehicle's value, 3) loan amount, and 4) current policies and indices for loans.

Design

The new application could be completely implemented in Java technology with just servlets, JSPs, EJBs, and POJOs. However, WS-BPEL is a more suitable language for service composition and business process execution. As mentioned earlier, the decision about when to start composing services using WS-BPEL can be tricky. The general rule is that Java can be used for extending existing services or composing fine-grained Java EE services to create more services that may not yet provide a cohesive and complete business or technical function. Should the new processing unit implement business or technical logic, which is more than invoking services and basic data transformations between invocations, that's an indication that it may be more suitable to implement Java. However, when the application or composition needed is process-centric and involves invoking services and hooking them together with flow logic and simple mappings, WS-BPEL may be the best way to implement this layer of the application.

The new application requires the development of the following services: 1) *Applicant Evaluator* – a service that aggregates an internal report if the applicant is an existing customer with an external credit report; 2) *Loan Approval Processor* – a service that takes an applicant's fiscal performance, vehicle report, and current indicators and comes up with a decision; and 3) the *Vehicle Loan Application Process* that will be exposed to the external world. This process will receive an incoming credit application, invoke the Applicant Evaluator Service, Vehicle Information Service, and Financial Index Service in that order and pass the results from all these services to the Loan Approval Processor service and return the decision to the caller.

The Applicant Evaluator service will be implemented as a stateless session bean annotated to be exposed as a Web service. This service will invoke an external Web service to get a credit report and also invoke another stateless session bean – Performance Evaluator – to get an internal credit report, if one is available.

The Vehicle Information Server and Financial Index Server Stateless Session Bean functions will be exposed as Web services. The Loan Approval Processor service will be implemented as a stateless session bean and annotated to be exposed as a Web service.

The Vehicle Loan Application Process will be implemented as a business process that invokes the Applicant Evaluator, Vehicle Information Server, Financial Index Server, and Loan Approval Processor services in that order (see Figure 3).

Implementation

Java EE technology and the tools provided in NetBeans allow users to adopt bottom-up or top-down approaches or a combina-

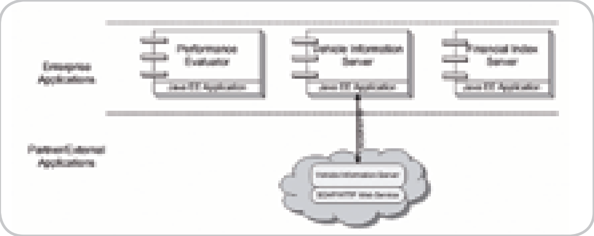


Figure 2 Existing applications, functions, and technology

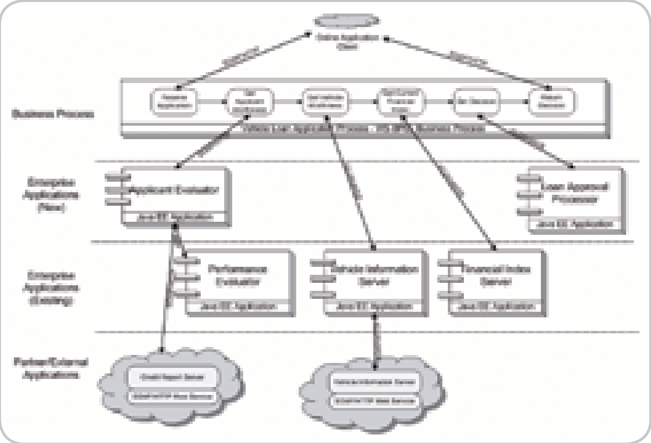
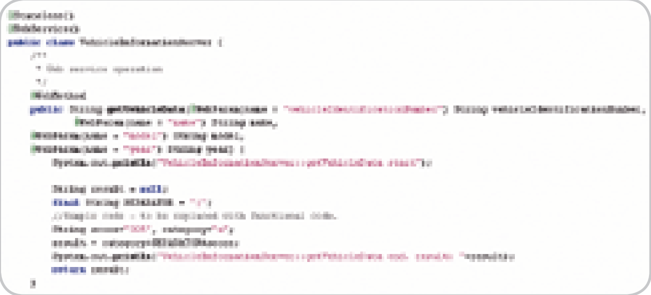


Figure 3 Proposed applications, functions, and technology



```
@Stateless()
@WebService()
public class VehicleInformationServer {
    /**
     * Web service operation
     */
    @WebMethod
    public String getVehicleData(@WebParam(name = "vehicleIdentificationNumber") String vehicleIdentificationNumber,
            @WebParam(name = "make") String make,
    @WebParam(name = "model") String model,
    @WebParam(name = "year") String year) {
        System.out.println("VehicleInformationServer::getVehicleData start");

        String result = null;
        final String SEPARATOR = "|";
        //Sample code - to be replaced with functional code.
        String score="[illegible]", category="[illegible]";
        result = category+SEPARATOR+score;
        System.out.println("VehicleInformationServer::getVehicleData end. result: "+result);
        return result;
    }
```

Figure 4 Annotated Vehicle Information Server EJB

Prabhu Balashanmugam is a product manager for SOA and Business Integration Software at Sun Microsystems, Inc. He is currently responsible for managing Business Process Management and Data Management products. Prior to joining Sun Microsystems, he has implemented and managed various software solutions for companies across the globe.

prabhu.balashanmugam@sun.com

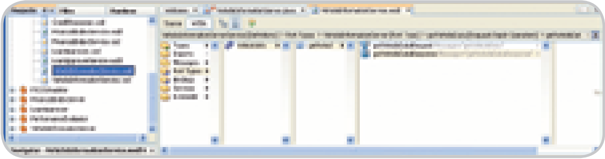


Figure 5 Generated WSDL for the annotated EJB

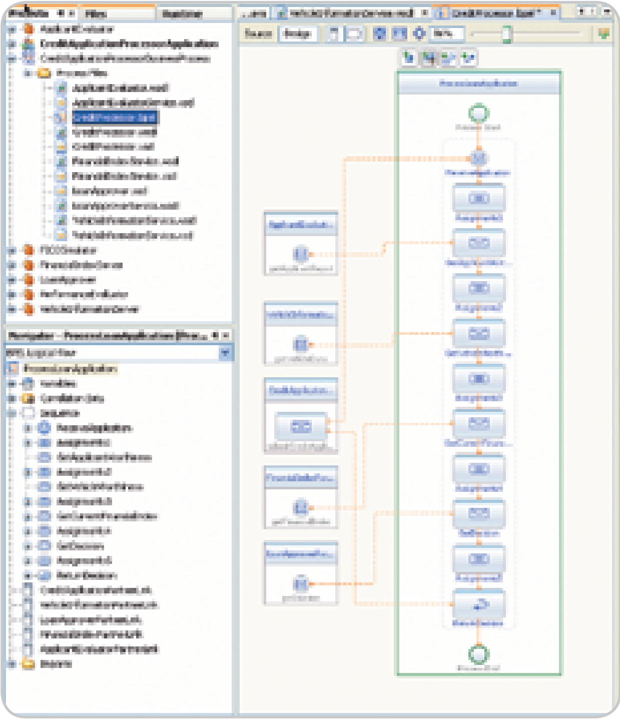


Figure 6 Loan application evaluation process

tion of both to deliver composite applications. In this example we will follow a bottom-up development approach. We'll start by building Java services, followed by invoking the services in a business process. We'll then compose the services in a composite application, deploy the application on Sun Application Server 9.1, and test the composite application.

NetBeans Enterprise Pack 5.5 provides the necessary capability to perform all the above tasks from within the IDE without the need to use any other tool or editor.

The Java Services

Java EE 5 enables functionality in an existing Java EE application to be easily service-enabled by annotating the Java classes. The creation of new Java EE applications that are Web-service enabled can also be done easily using the EJB Module Creation Wizard in NetBeans. In the first phase, the EJB Modules Vehicle Information Server and Financial Index Server are service-enabled by adding the appropriate annotations to the classes.

In the second phase, two new services are created. The Applicant Evaluator service is a stateless session bean also exposed as a Web service. This service aggregates results from an external Web service, FICOSimulator, and the Performance Evaluator stateless session bean. The FICOSimulator Web service simulates an external Web service that returns a credit report (see Figure 4).

The Loan Approval Processor is another service developed as a stateless session bean, also exposed as a Web service. This service combines reports on the applicant's credit worthiness, the vehicle's value, the loan indicators, and returns a decision based on predefined business rules.

Composing the Services

The Business Process Editor in NetBeans Enterprise Pack 5.5 can be used to compose services. The Business Process Editor enables users to model business processes graphically in a visual environment and the WS-BPEL code is automatically generated, corresponding to the visual model. However, the user can choose to make changes directly in the generated WS-BPEL code and the visual model is automatically synchronized to the graphical view. The services invoked by business processes are defined as WSDLs (see Figure 5). Users can create new WSDLs using the WSDL and XSD Editors directly in NetBeans. However, users can also import existing WSDLs and XSDs and edit them using the editors.

A BPEL Project, CreditApplicationProcessorBusinessProcessor, is created and the WSDLs and XSDs for the following services are included: 1) Applicant Evaluator, 2) Vehicle Information Server, 3) Financial Index Server, and 4) Loan Approval Processor Service. Once the WSDLs are imported, Partner Link Types are created for the Port Type that will be invoked from the business process. Please note that the credit application process is going to be exposed as a Web service, so, another WSDL is created, CreditProcessor.wsdl, that represents the interface to the business process .

Partner Link Types specify the role that will be played by the service defined in the Port Type. For example, the Partner Link Type CustomerDataServicesPartners below specifies that the service will play the role of a CustomerReportProvider when the function getCustomerReport is invoked.

```
<partnerLinkType name="CustomerDataServicesPartner">
<role name="CustomerReportProvider"
portType="getCustomerReport"/>
</role>
</partnerLinkType>
```

Once Partner Links Types are created in the WSDLs for the PortTypes, they can now be invoked from a business process. The user can simply drag and drop the WSDL on the Business Process Editor canvas and the Business Process Editor automatically recognizes the available Partner Link Types and shows a wizard for configuring the Partner Links for the business process. The wizard allows the user to determine if the business process will be a consuming partner or a providing partner of the service defined in the Partner Link Type.

The WS-BPEL 2.0 specification provides various types of activities for modeling complex and real-world business processes. The business process can receive and reply to messages or it can just receive messages without responding. Business processes can also receive messages asynchronously from external sources. The key constructs supported in the WS-BPEL specification include the ability to invoke external services, handle exceptions, process compensation, and error condition logic. It also allows for modeling the concurrent and conditional execution of activities. The sample used in this article uses only a few of the constructs, namely, Receive, Reply, Invoke, and Assign activities. The Invoke activity is used to consume the services and the Assign activity is used to set and get values to the input and from the output messages of the invoked service. The Assign activity opens up a mapper that allows users to view all the variables in the business process and graphically get and set values between them. The mapper also provides data processing and transformation functions that

Gopalan Suresh Raj, an architect at Sun Microsystems, Inc., is a member of Sun's research and architecture team. His expertise spans enterprise architectures and distributed computing. He is the author of a number of technical books and articles. You can read and participate in discussions with him at http://blogs.sun.com/gopalan.

gopalan.raj@sun.com

can be used in conjunction with the assignments. The Receive activity is used to receive an incoming message that creates an instance of the business process at runtime. The reply activity is used to send a response message back to the caller and the business process instance is then discarded.

The user starts by creating a template business process by laying out the activities and then configuring them one at a time. The user drags and drops the CreditProcess WSDL and configures the business process as the providing partner of the service. The user configures the Receive and Reply activities to implement the PortType specified in the CreditProcess WSDL. The user then drags and drops all the other WSDLs and configures the business process as a consuming partner of those services. The user uses one invoke activity per service consumed and configures the invoke activity to link it with the appropriate Partner Link. Once configured, the business process looks similar to Figure 6.

Create and Deploy Composite Applications

NetBeans provides the capability to group more than one business process available under different projects into one composite application. This allows you to group multiple logical business processes and manage them as one logical unit during deployment. NetBeans also provides the necessary runtime components for deploying composite applications. The runtime components include Java Application Server 9.1, WS-BPEL Service Engine, and HTTP/SOAP Binding Components. The composite applications can be easily created by selecting and adding components from different projects in the environment. NetBeans also allows users to build and deploy composite applications with a single click.

In the sample used in this article, the user adds the business process module from the project CreditApplicationProcessorBusinessProcess to the Composite Application project. The composite application is then deployed at runtime.

Testing

The need to write a client application just to test a newly deployed composite application can be daunting. NetBeans eliminates this need by providing a powerful test facility. This facility is automatically added under the Composite Application projects. The users can create any number of test cases to test various functions of the deployed composite application and specify the success criteria for these tests. The input and output messages for the test cases are also automatically generated when the user selects the WSDL that corresponds to the business process to be tested. The user can then customize the messages as needed in the editor shown in Figure 7. The test facility also maintains a log of all the test results. In case of system errors, the server and application logs can be viewed from NetBeans. This facility allows users to immediately test just-deployed composite applications.

The user adds a test case for the application discussed in the article. The output message can be customized before the first run or the output message from the first run of the test case can be used as the expected output message.

Okay, What's Next?

Java EE 5 SDK provides the complete set of tools and environments required for composing services using WS-BPEL. Invoking all services as SOAP/HTTP Web services could bring out the obvious concern about performance. WS-BPEL can be used to directly invoke Enterprise JavaBeans to compose functionality from Java EE applications. Java EE 5 SDK can be used to model business processes that can invoke internal and external Web services and Enterprise JavaBeans directly, as shown in Figure 8.

Business processes can invoke the services available in those technologies including REST Style Web services. If you are interested in finding out what other options are available for composing services and processing

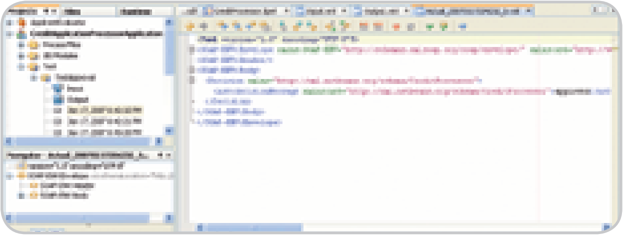


Figure 7 Composite Application Testing Facility

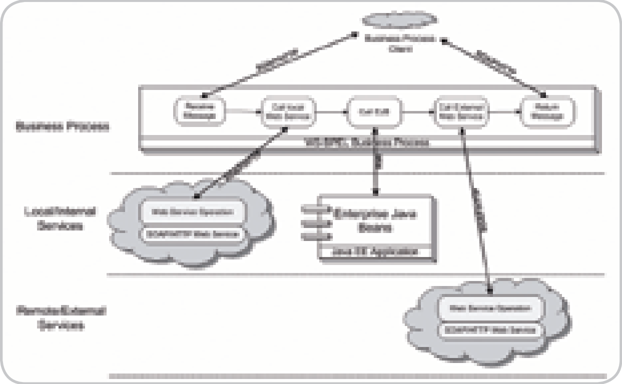


Figure 8 WS-BPEL invoking SOAP/HTTP Web services and Enterprise JavaBeans

business events, check the following open source projects.

- NetBeans: http://netbeans.org/
- GlassFish: http://glassfish.dev.java.net/
- Open ESB: http://open-esb.dev.java.net/

You can get the latest version of the products and updates on the enhancements that are currently being worked on. You can also join and contribute to these projects if you're interested.

The source code for the sample projects used in this article can be downloaded from the online version of this article at http://jdj.sys-con.com.

Summary

This article has discussed and shown the many capabilities in the Java EE 5 SDK Tools Bundle for delivering composite applications using Java, WS-BPEL, and SOA technologies.

First, this article discussed the benefits of these two technologies and how they can work together to enable the development of next-generation applications.

Second, this article demonstrated how NetBeans provides a truly integrated development environment that allows users to extend existing Java services, create new Java services, and compose Java services with a business process. This article has also shown how NetBeans can be used to iteratively develop, build, deploy, and test composite applications seamlessly, thereby reducing the overall turnaround time and effort required to deliver composite applications.

Resources

- Java EE 5 SDK: http://java.sun.com/javaee/downloads/
- NetBeans IDE & Enterprise Pack: http://www.netbeans.org/
- Project Open ESB: http://open-esb.dev.java.net/
- WS-BPEL 2.0 Specification: http://www.oasis-open.org/committees/download.php/14616/wsbpel-specification-draft.htm
- Technical Blog: http://blogs.sun.con/gopalan/
- Technical Blog: http://blogs.sun.com/kevinschmidt/
- Technical Blog: http://blogs.sun.com/theaquarium/

What Is SDO?

Part 2 – A standardized approach to data programming particularly well suited to SOA

by Geoffrey Winn
and Kelvin Goodson

In Part 1 of this article we introduced SDO, looking at its origins and then reviewing some of its features through a scenario based on medical investigations. In Part 2 we'll go on to examine further features of SDO in such a way as to add property values to a data object beyond those explicitly defined by its type, the ability to record changes made to the data graph, a more elegant way to create types programmatically, and the ability to extend datatypes at runtime.

Our examples will depend on two XML Schema definition files that we'll reproduce here. The full details of the scenario they represent were presented in Part 1. These listings are available at http://res.sys-con.com/story/dec06/313547/source.html.

Part 1 of this article was published when version 2.01 of the SDO specification was current. Since then v2.1 has been released, however, in this article we'll continue to use version 2.01 for consistency.

Adding Open Content

The scenario begun in Part 1 now unfolds a little further. In particular, we'll examine the addition of values to a DataObject by virtue of the DataObject's type being open; that is to say, the property values in addition to those defined by its type. Joe Johnson has been found to have the rare condition, Panarr Syndrome. The designers of the test's data model have made use of the generic People model described in the first XML Schema from Part 1 and this design decision was only possible because the designer of the People model made the Person type open.

Note how in the schema for the medical test there are two global elements defined, one for the test itself and one for the medical condition. When the schema is read into the SDO runtime, these elements are translated into SDO global properties, scoped in the namespace URI of the schema. They're global properties, because they're not logically part of a type. It's this kind of global property that can be used to add information to an instance of an open type.

```
    Property conditionProperty = XSDHelper.
INSTANCE.getGlobalProperty(
        "www.example.org/MedicalTest", "condi-
tion", true);
    DataObject condition = DataFactory.
INSTANCE.create(
        "www.example.org/MedicalTest",
«Condition»);
    condition.setString("name", "Panarr
Syndrome");
    List conditions = person1.getList(conditi
onProperty);
    conditions.add(condition);
```

In this code, SDO's XSDHelper has been used to look up the global property, and the DataFactory has been used to create a DataObject of the appropriate type. The interesting part of this code snippet is that when asked for the list of values associated with the "condition" property of the person DataObject, even though that property doesn't belong to the Person type and was never associated with the Person instance, the call to getList(conditionProperty) resulted in an empty list being created and returned. If the type of Person hadn't been open, this call would have resulted in an exception being thrown. Now we can simply add the condition to the list and any further serialization of this person instance will look like this:

```
    <person gender="male" id="1" name="Joe
Johnson Snr.">
      <Test:condition name="Panarr Syndrome"/>
    </person>
```

It's worth noting that the SDO 2.1 specification adds to the SDO's capability for handling open content metadata and instance data.

Complex Graphs and Non-Containment References

In part 1 we examined one kind of relationship between data objects, that of *containment*. Every data object other than the root has exactly one parent data object that's its container. However, what if we want to express other relationships? In SDO these are represented by *non-containment* references.

We've already seen how SDO handles containment. We'll now explore adding complexity to the data graph by using non-containment links. Given the likelihood of members of Joe's family exhibiting Panarr Syndrome we'll add a list of Joe's relatives to the test set:

```
    DataObject relatives = test.createDataObje
ct("relatives");
    DataObject person2 = relatives.createDataO
bject("person");
    person2.setString("id", "2");
    person2.setString("name", "Joe Johnson
Jnr.");
    DataObject relation = person1.createDataOb
ject("relative");
    relation.set("target", person2);
    relation.set("relationship", "child");
```

This code snippet demonstrates the building of a complex graph using a non-containment link. We've created a new person in the test's set of relatives and made a link to that person from a new "Relative" property value for person1. So now our test looks like this:

```
<Test:Test xmlns:Test="www.example.org/
MedicalTest"
  <referrals/>
  <patients>
    <person gender="male" id="1" name="Joe
Johnson Snr.">
      <relative relationship="child" tar-
get="2"/>
      <Test:condition name="Panarr Syndrome"/>
    </person>
  </patients>
  <relatives>
    <person gender="male" id="2" name="Joe
Johnson Jnr."/>
  </relatives>
</Test:Test>
```

Observe how the target attribute of the relative element has a value of "2," which is the id attribute value of the person "Joe Johnson Jnr" thus establishing the relationship from father to son. Looking back

Geoff Winn is based at IBM Hursley in the UK, as part of the Open Source SOA team. He is a member of the SDO specification group and currently works on development of the Apache Tuscany C++ implementation of SDO. He has degrees in Mathematics and Computation, and has previously worked in the areas messaging and brokering middleware.

gwinn@uk.ibm.com

at the schema for Person, note how the use of the XML Schema “ID” attribute in the definition of the Person type and the “IDREF” attribute in the “Relative” type’s target element lets us make this non-containment link. Having made the link, SDO needs a bit of help from the schema author to know what kind of value to expect the non-containment link to resolve to and so the “target” attribute is qualified by an sdoxml:propertyType attribute to indicate that any referenced objects will be the type Person.

Monitoring Change

A major capability of SDO is being able to provide a summary of the changes made to a data graph. In SDO 1 this behaviour was the responsibility of a special class, the DataGraph. In SDO 2 you can get this behaviour through the DataObject interface by including a property of the ChangeSummaryType in your type definitions. This new feature offers flexibility in choosing the scope of change monitoring. For example, you might want to monitor the changes to the body of a Web Services message without monitoring its containing envelope. Here we’ll focus on the new style of achieving this result. For this we must return to the original XML Schema for the medical test and add an element to the “Test” type.

```
 <complexType name=”Test”>
   <sequence>
     <element name=”referrals” type=”people:
PersonSet” />
     <element name=”patients” type=”people:
PersonSet” />
     <element name=”relatives” type=”people:
PersonSet” />
     <element name=”changes” type=”sdo:
ChangeSummaryType” />
   </sequence>
 </complexType>
```

The SDO specification defines the rules about the definition of types that include ChangeSummaryType properties. It also defines rules about assembling a data graph that includes data objects of such a type. These rules are in place to avoid the possibility of a graph having multiple change summaries describing overlapping areas of the graph.

Once you have a data object containing a change summary property you can access the change summary using the getChangeSummary() method (on any data object within the scope of the change summary). You can then use this change summary object to alter the logging state or query the changes that have occurred.

Imagine that we had been using the above type definition throughout our example and immediately after creating the test instance DataObject we turned on change monitoring as follows:

```
   DataObject test = DataFactory.INSTANCE.
create(“www.example.org/MedicalTest”, “Test”);
   test.getChangeSummary().beginLogging();
   test.set(“referrals”, referrals);
```

Having done this, at any subsequent point in the program we can make use of the interface to the ChangeSummary object to examine the alterations in state that have occurred since the change monitoring was started.

As an example, imagine a data entry panel for adding and editing patient data. The ChangeSummary’s undoChanges() method can be used to roll back all the changes made since logging began, and could therefore be used to underpin a “cancel” operation in the data entry panel. Knowing that the changes made to a graph opens up many other possibilities, for example, the above data entry panel could highlight all the changes made as the data is updated, or it could provide for a “commit” phase, where the operator is given an opportunity to review the changes before committing them. It would also be possible to use the change summary information to reduce inter-process communication by serializing and transmitting only the changed objects between the application layers.

Creating Types Programatically

In Part 1 of this article we first adoped an Apache Tuscany implementation-specific approach to creating types programmatically so we could introduce the concept of types and then we relied on the XML Schema-to-SDO Type conversion to do the bulk of our type system creation. The reason for starting with the non-standard Tuscany approach to building types was that the official method requires knowledge of SDO so we have a “chicken and egg” situation where you need types to build data objects and you need to understand how to build data objects to create your own types. We are now equipped to revisit type creation and do it the official way.

SDO has a TypeHelper interface that can be passed descriptions of the desired types and will produce those types and make them available for lookup later. The format for description of the types is a graph of data objects. There are two TypeHelper methods that let you define types; in the first you can submit a data graph describing a single type to the type helper, while the second lets you submit a list of data graphs to be processed all together and thereby generate a list of types. This is more than just a convenience and we’ll explain later why there’s a compelling reason for using this second option.

In preparing the description of our types we’ll be building a number of similar data objects into data graphs so we’ll create a little helper method to create a basic description:

```
 private DataObject createTypeDescription(
     String uri, String name) {
   DataObject typeDesc = dataFactory.create(
     “commonj.sdo”, “Type”);
   typeDesc.set(“name”, name);
   typeDesc.set(“uri”, uri);
   return typeDesc;
 }
```

We now have enough understanding of the SDO API to know that this method builds a little one-node data graph with a root data object of the “Type” type with two properties, name and uri, having been set to the values supplied to the method. The “Type” type is built into the SDO runtime and is there specifically to build type descriptions. Note that this data object isn’t itself a type, just a description of a type.

The first step in creating our type system programatically is:

```
   DataObject personTypeDesc = createTypeDe-
scription(
      “www.example.org/people”, “Person”);
   List typeDeclarations = new ArrayList();
   typeDeclarations.add(personTypeDesc);
```

At this point we have a fairly empty type description that’s been added to a list that we’ll build up as we go along so we can submit the list of type descriptions to the TypeHelper’s define method later. Adding property descriptions to type descriptions is something we do a few times so we’ll create another little utility method to do the basics for us:

```
 private DataObject addPropertyDescription(
     DataObject containerTypeDescription,
     Object propertyType,
     String propertyName
 ) {
   DataObject property =
     containerTypeDescription.createDataObje
ct(“property”);
```

Kelvin Goodson is based at IBM Hursley in the UK as part of the Open Source SOA team. He is a committer to the Apache Tuscany incubator project, and works primarily on development of the Tuscany Java implementation of SDO. He gained a Ph.D. in image analysis and artificial intelligence in 1988, and has previously worked in the areas of medical imaging, weather forecasting and messaging middleware.

kelvin_goodson@uk.ibm.com

```
    property.set(“type”, propertyType);
    property.setString(“name”, propertyName);
    property.setBoolean(“containment”, true);
    return property;
  }
```

This method takes our new unpopulated type description and adds to it a basic description of a new property that we want the type to have. For example:

```
    Type stringType = typeHelper.
getType(“commonj.sdo”, “String”);
    addPropertyDescription(personTypeDesc,
stringType, “name”);
```

The personType now has a property “name” that has a type of the pre-existing SDO String type. We could go on adding new properties to the description using existing types; but note how the second argument for adding property descriptions to our utility method, “propertyType,” used the abstract Object Java class. This is because the method is suitable as it stands to accept a type described by a data object (as above), rather than a real Type instance.

Here's an example of why this is important. In our schema definition a person has a list of relatives. A relative is described by a relationship name and a reference to another person. So the Person type has a dependency on the Relative type and the Relative type has a dependency on the Person type. If we weren't able to use a description of a type in this way (rather than a real type instance) then we wouldn't be able to represent these commonly occurring circular dependencies.

A point of note about the TypeHelper interface's define methods is that the process of creating a true type instance is logically atomic, i.e., once a Type is created you can't go adding properties to it. This is helpful because it ensures that once a data object has been created, you can be sure its type won't be altered. The upshot of this is that for types such as Person and Relative that are interdependent we have to create them in one call to TypeHelper.define() so we have to build a list of type descriptions and submit it to the variant of the TypeHelper's define method that takes a list of descriptions and produces a list of types in one method call.

Leaving aside the remaining simple typed properties of the type Person, here's how we complete the type description:

```
    DataObject relativeType = createTypeDescrip-
tion(
    peopleURI, “Relative”);
  DataObject rp = addPropertyDescription(
    personTypeDesc, relativeType, “relative”);
  rp.setBoolean(“many”, true);
  personTypeDesc.set(“open”, Boolean.TRUE);
```

First we create a new unpopulated description of the Relative type that we can use as a “forward declaration” of the type. This lets us add a “relative” property description to the Person type. The “relative” property is modified to allow multiple values and finally the Person type is described as being open just as we saw in the earlier XML Schema definition of the type system to permit things such as a medical condition to be added to an instance of the Person type.

The rest of the simple typed properties of the Relative type are added to its description, and all the other types we require are described this way. Eventually, when the description is complete, and all the descriptions are collected in the list, we submit the list of descriptions to a TypeHelper for compilation into a type system:

```
  List types = TypeHelper.INSTANCE.define(type
Declarations);
```

The types are returned as a list but more importantly they're known to the environment. So returning to the first line of code of our scenario following the creation of the type system:

```
  DataObject person1 = DataFactory.INSTANCE.
create(
      “www.example.org/people”, “Person”);
```

So the example can be executed using a type system that was created programatically, rather than by using the XML Schema-to-SDO conversion process.

A TypeHelper can be regarded as an instrument for controlling the scope for a set of types. There are known issues with the singleton pattern and so the recently published SDO 2.1 specification has addressed this issue by introducing a new HelperContext class. The HelperContext provides access to a consistent set of SDO helpers all referencing a specific TypeHelper instance.

Generating Java Classes to Represent the Type System

So far in our example we've only been using the dynamic SDO API, but in many environments, when the type system can be defined at application development time, it's better to use generated Java classes to represent the data objects in a data graph. When using generated classes, the performance is better and the code is more natural; and yet the full power of the dynamic API is available to be used on the generated classes if needed.

The SDO API defines a mapping from the XML Schema-to-Java classes. There's no current specification to define the tooling for generating SDO classes. In the Apache Tuscany implementation of SDO this is done with the XSD2JavaGenerator class:

```
java org.apache.tuscany.sdo.generate.
XSD2JavaGenerator  MedicalTest.xsd
```

This creates an interface and implementation packaged for each of the namespaces in the XML Schema and any imported schemata. An interface class is generated per XML Schema type, and a Factory interface for crating instanes of the types is created along with the correponding implementations (see Figure 1).

And now we can begin coding in an altogether different style, using the property-specific getter and setter methods of the generated classes:

```
      Person p1 = PeopleFactory.INSTANCE.cre-
atePerson();
      p1.setId(“1”);
      p1.setName(“Joe Johnson Snr.”);
      p1.setGender(“male”);
      XMLHelper.INSTANCE.save(
          (DataObject)p1,
          “www.example.org/people”,
          “person”, System.out);
```

Summary

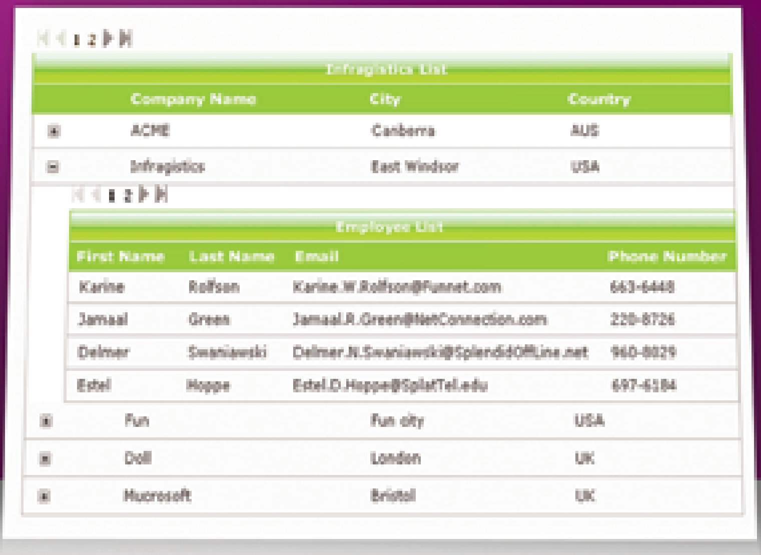
In this two-part article we've seen how SDO provides a standardized approach to data programming that's particularly well suited to Service Oriented Architectures, presenting a single consistent view of data obtained from a variety of sources such as relational databases and XML. Version 2.1 of the SDO specification was released at the end of 2006 and work has already started on the next version.

References

- G. Winn and K. Goodson. “What Is SDO?” *Java Developer Journal*. Vol. 11, Issue 12.
- “Service Data Objects for Java Specification Version 2.1.0.” November 2006. Available from http://www.osoa.org/

Speed. Simplicity. Style.

Build better UIs with
our JSF components.



1 2

Infragistics List

	Company Name	City	Country
⊞	ACME	Canberra	AUS
⊟	Infragistics	East Windsor	USA

1 2

Employee List

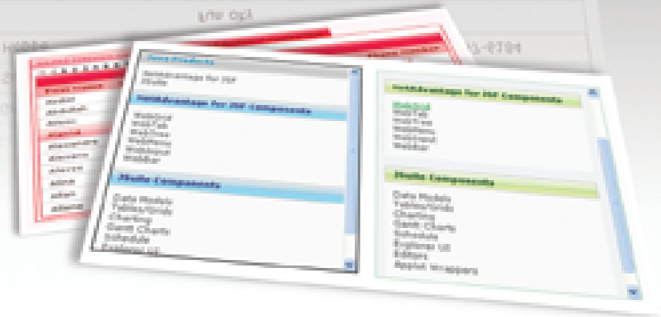
First Name	Last Name	Email	Phone Number
Karine	Rolfson	Karine.W.Rolfson@Funnet.com	663-6448
Jamaal	Green	Jamaal.R.Green@NetConnection.com	220-8726
Delmer	Swaniawski	Delmer.N.Swaniawski@SplendidOffLine.net	960-8029
Estel	Hoppe	Estel.D.Hoppe@SplatTel.edu	697-6184

	Company Name	City	Country
⊞	Fun	Fun city	USA
⊞	Doll	London	UK
⊞	Mucrosoft	Bristol	UK

NetAdvantage®

for JSF 2006 Volume 2

AJAX-enabled JavaServer™ Faces components



Simplify Complex Data – Our All-New Hierarchical Grid easily organizes and displays data in nested grids

Maintain Readability – Fixed Columns keep critical column data in view while your users scroll

Built-in Flexibility – Our APIs allow incredible interactive experiences on the web

Great User Experience – Our AJAX-enabled components turbo-charge your web applications for a rich client UI experience

learn more: infragistics.com/jsf

Infragistics Sales - 800 231 8588 **Infragistics Europe Sales** - +44 (0) 800 298 9055

Infragistics®
Powering The Presentation Layer

WINDOWS® FORMS ASP.NET WPF JSF

grids scheduling charting toolbars navigation menus listbars trees tabs explorer bars editors

Looking Inside **Stuck Threads**

The transmigration of Java threads

by Patrick Yeh

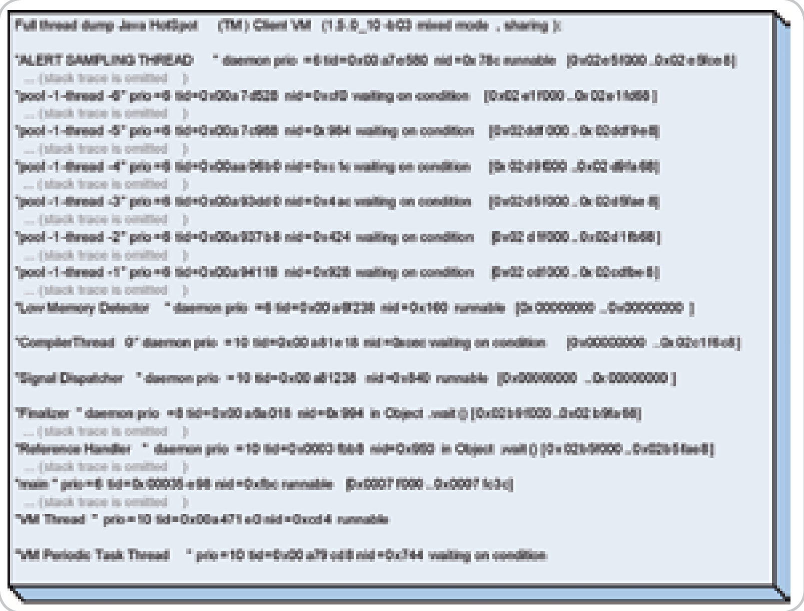
Thread pooling is a common technique that modern application servers adopted to run Java applications efficiently. Even application servers not implemented by Java share the concept of using system resources more compactly to maximize overall throughput. Besides the underlying programming mystery of native OS threads, a Java thread object encapsulates some hurdles to easy-to-use and flexible synchronization at the programming level. JDK 5.0 has built-in thread pooling classes in its 'java.util.concurrent' package to facilitate programming the thread pool quickly. If we're using a J2EE application server, the container inherently enforces thread synchronization from its runtime nature. That means we don't have to fight difficult threading issues day and night, but it doesn't mean we can dismiss them. Instead, we should attend to the thread issues inside the code and the architecture. If we don't, system performance will degrade. A once well-running system will gradually become slower and slower, then application throughput will be blocked and external requests start to queue up. There's some degree of denial of service. In most commercial production environments like telecom, e-commerce and banking, this situation impacts the business and can create unplanned system outages.

While the server operator calls for help, an experienced engineer often asks something outstanding of the application environment. During the incident, we may see either ultra-high or ultra-low CPU usage at the OS level along with applications hanging and threads sticking at the JVM level from a three-dimensional viewpoint. How does one disclose the bottleneck and abnormality at the JVM level? The answer is: When the problem is reproducible then a commercial productive profiling tool or remotely debugging the JVM is an option. But taking copies of the thread dumps is widely used because it's straightforward and instantaneous. And it involves the least overhead.

Thread dumps provide a snapshot of the JVM internals at a special point at a minimal cost. We may give the JVM hosting the applications a signal SIGQUIT with the JVM process ID (PID) on a Unix-like system (e.g., kill -3 xxxxxx; where 'xxxxxx' is the JVM PID) or have a control-break on the Windows Java console ask the JVM to output its thread information in detail to a standard output *when the JVM didn't start in company with a '-Xrs' option before.* Due to the importance of the thread dump, it's best to redirect the standard output to a file or pipe the information to a utility that can store and rotate the standard output to log files (see Figure1).

A JVM has a complementary function that enables it to get the thread dump at the undocumented C API level. (We can look at the Java source code that Sun released recently under the GPL to see this feature.) We may utilize this API for a simple debugging framework to address many common issues inside the application. But it requires a JNI implementation in C because there's no pure Java API to force the JVM to generate the thread dump, though we may get similar thread stack traces in JDK 5.0 via the 'getAllStackTraces()' API. Despite this tricky function, we're interested in a snapshot of the thread dump *while we have identified the stuck threads* (see Figure 2).

With copies of the thread dump collected at intervals of seconds, we may identify the stuck threads from the running state of each thread in the thread pool. Fortunately, some application servers do an automatic health check on the application thread pools. In fact, it acts like a watchdog that periodically check the last running statistics on the threads in the thread pools. Once the threads have run *for fixed long-running seconds*, it will print out the execution information on the stuck threads either in standard output or log files. Second, some platform JDK vendors have out their diagnostic utilities in the public domain to aid us in detecting stuck threads (e.g., HP's JMeter and IBM's thread analyzer). However, once we isolate the stuck threads, we'll have to figure out why



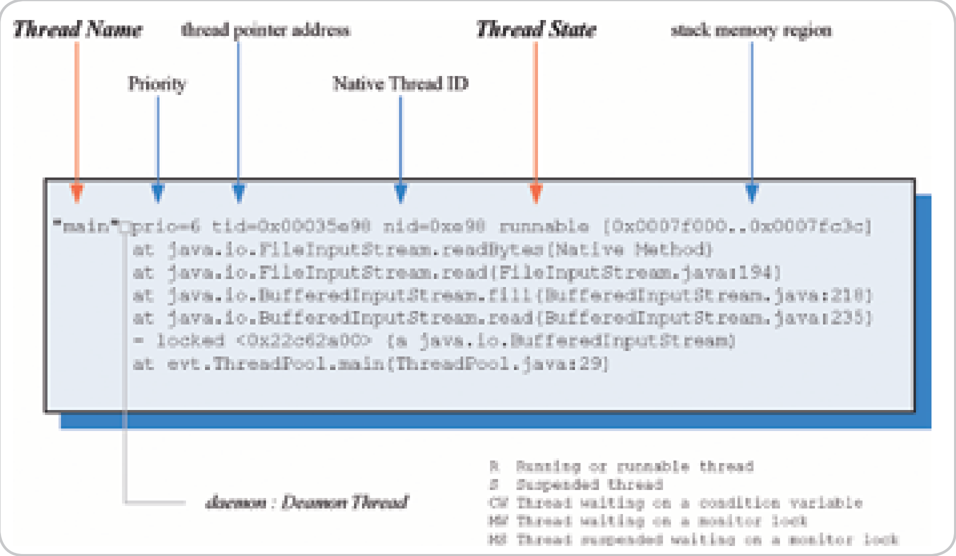
```
Full thread dump Java HotSpot  (TM) Client VM  (1.5.0_10-b03 mixed mode , sharing ):

"ALERT SAMPLING THREAD   " daemon prio =6 tid=0x00a7e580 nid=0x78c runnable  [0x02e5f000..0x02e5fce8]
  ... ( stack trace is omitted   )
"pool-1-thread-6" prio=6 tid=0x00a7d528 nid=0xcf0 waiting on condition   [0x02e1f000..0x02e1fd68]
  ... ( stack trace is omitted   )
"pool-1-thread-5" prio=6 tid=0x00a7c988 nid=0x984 waiting on condition   [0x02ddf000..0x02ddf9e8]
  ... ( stack trace is omitted   )
"pool-1-thread-4" prio=6 tid=0x00aa96b0 nid=0xc1c waiting on condition   [0x02d9f000..0x02d9fa68]
  ... ( stack trace is omitted   )
"pool-1-thread-3" prio=6 tid=0x00a93dd0 nid=0x4ac waiting on condition   [0x02d5f000..0x02d5fae8]
  ... ( stack trace is omitted   )
"pool-1-thread-2" prio=6 tid=0x00a937b8 nid=0x424 waiting on condition   [0x02d1f000..0x02d1fb68]
  ... ( stack trace is omitted   )
"pool-1-thread-1" prio=6 tid=0x00a94118 nid=0x928 waiting on condition   [0x02cdf000..0x02cdfbe8]
  ... ( stack trace is omitted   )
"Low Memory Detector   " daemon prio =6 tid=0x00a8f238 nid=0x160 runnable  [0x00000000..0x00000000 ]

"CompilerThread0" daemon prio =10 tid=0x00a81e18 nid=0xcec waiting on condition   [0x00000000..0x02c1f6c8]

"Signal Dispatcher   " daemon prio =10 tid=0x00a81238 nid=0xb40 runnable  [0x00000000..0x00000000 ]

"Finalizer  " daemon prio =8 tid=0x00a8a018 nid=0x994 in Object .wait () [0x02b9f000..0x02b9fa68]
  ... ( stack trace is omitted   )
"Reference Handler   " daemon prio =10 tid=0x0003fbb8 nid=0x950 in Object .wait () [0x02b5f000..0x02b5fae8]
  ... ( stack trace is omitted   )
"main " prio=6 tid=0x00035e98 nid=0xfbc runnable   [0x0007f000..0x0007fc3c]
  ... ( stack trace is omitted   )
"VM Thread  " prio=10 tid=0x00a471e0 nid=0xcd4 runnable

"VM Periodic Task Thread   " prio=10 tid=0x00a79cd8 nid=0x744 waiting on condition
```

Figure 1 A sample thread dump on Sun Java 5.0 Client VM

Patrick Yeh (WEN-PIN, YEH) A senior technical consultant of BEA Systems, Taiwan for solving the critical production issues. The core value of this position is to provide the solid technical power on problem solving and to reduce the customer's downtime losses that may have a critical impact on their business (+4 years).

wenpin@gmail.com

they got stuck from the information (i.e., the stack traces of these stuck threads) about what they were doing when they got stuck. This way we can improve code quality and tier architecture in the next iteration (see Figure 3).



```
"main" prio=6 tid=0x00035e90 nid=0xe90 runnable [0x0007f000..0x0007fc3c]
        at java.io.FileInputStream.readBytes(Native Method)
        at java.io.FileInputStream.read(FileInputStream.java:194)
        at java.io.BufferedInputStream.fill(BufferedInputStream.java:218)
        at java.io.BufferedInputStream.read(BufferedInputStream.java:235)
        - locked <0x22c62a00> (a java.io.BufferedInputStream)
        at evt.ThreadPool.main(ThreadPool.java:29)
```

```
R  Running or runnable thread
S  Suspended thread
CW Thread waiting on a condition variable
MW Thread waiting on a monitor lock
MS Thread suspended waiting on a monitor lock
```

Figure 2 A sample stack trace of a thread in the thread dump

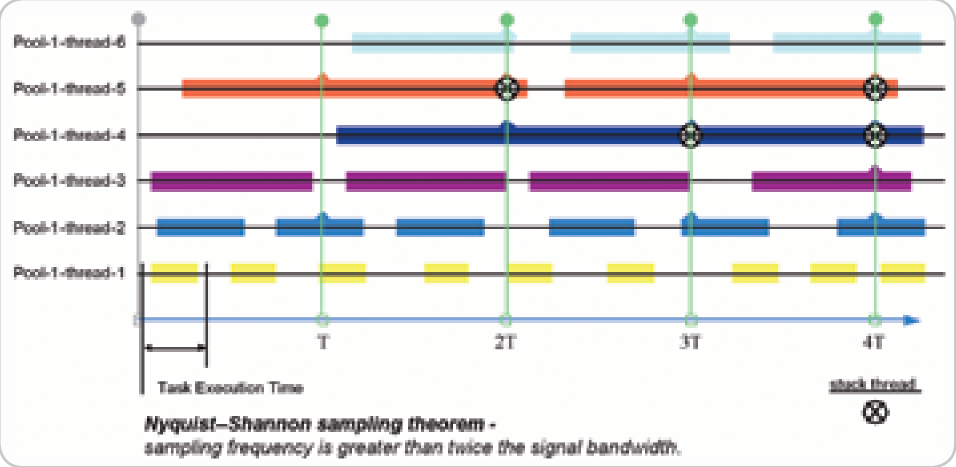


Figure 3 Time chart for stuck thread detection

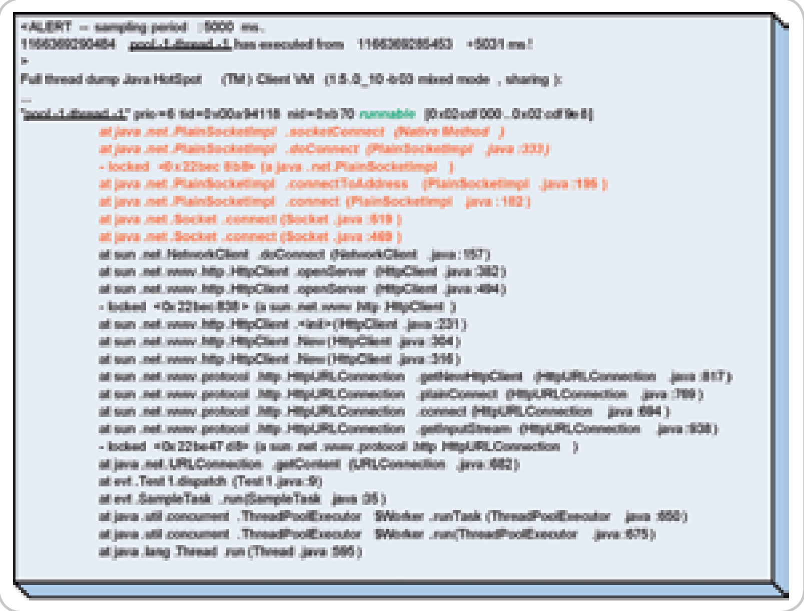
A stuck thread means a thread is blocked and can't return to the thread pool smoothly in a given period of time. When an application thread is blocked unintentionally, it means it can't quickly complete its dispatch and be reused. In most of production situations, the root cause of these stuck threads is also the root cause of bad system performance because it interferes with regular task execution. [It's also a performance issue for producers and healthy consumers. < 1] (request frequency) < (healthy thread count for request execution/ average measured request execution time per healthy thread.]

Blocking without specifying a network connect or read timeout is the most frequent reason we have seen. When we don't manually configure a timeout for each method call involving networking, *it will have a potential blocking behavior by the underlying physical socket read/connect characteristic*. While waiting infinitely for the response from the other side, the native OS networking layer probably throws an I/O exception. By default this behavior takes an unexpectedly long time (e.g., 240 seconds). Modern distributed systems need to factor in this situation (especially, Web Services invocations). Though we may set timeouts for well-known protocols via some system properties (e.g., sun.net.client.defaultConnectTimeout and sun.net.client.defaultReadTimeout), the newer version of JDK might provide a generic mechanism to explicitly configure each default timeout value for those whose methods call socket connect/read as a security policy file. For example, com.sun.jndi.ldap.read.timeout (http://java.sun.com/docs/books/tutorial/jndi/newstuff/readtimeout.html) wasn't available prior to JDK 6.0 for LDAP service provider read timeout. Otherwise, when the problematic code isn't under the control of end users, it usually needs to restart the application to temporarily reset the abnormal phenomenon propagated from the other side. In addition, we should take into account whether the service we called is idempotent while analyzing this kind of issue in the design phase because we don't know whether the service at the other end keeps executing when the thread has ended its invocation after a timeout (see Figure 4).

The unexpectedly long execution time of a SQL statement is a common condition that causes a stuck thread. In the thread dump we collected, we can see that the stuck thread was running a network socket read for a long time without changes and the thread's stack trace contains many JDBC driver classes. Under these conditions, we can also check the status of the database it connected with and set the query timeout for all application code using a JDBC statement setQueryTimeout method. (Most JDBC drivers support this feature but we'd have to read the JDBC driver's release note first.) According to the different nature of every SQL query, it would be better to segregate the programs that have a longer execution time in another thread pool and tune the database table with indices for faster access. We would also need to check whether the JDBC driver is certified with the connected database. A sub-issue is the accessed table locked by other processes so the threads for the JDBC query couldn't continue because of table locking.

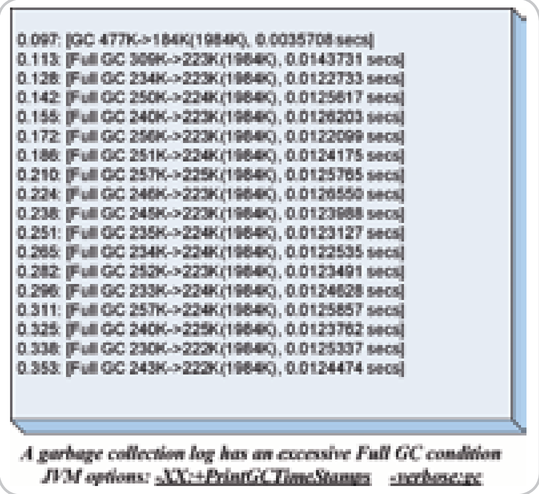
Resource contention is an issue that's hard to find if we don't get the entire thread dump to analyze. Basically, it's an issue of producers and consumers. Any limited resources on the system (JDBC connections, socket connections, etc.) will impact this issue. The best thing to do is look at the thread dump, get the stuck thread name from the log, and find the bottleneck that's causing the stuck thread.

File descriptor leaking is an issue that causes this phenomenon (Note that a Unix socket implementation requires a file descriptor). So the JVM should have enough file descriptor numbers to host our applications. Generally, we can adjust the open file limit with the Unix shell



```
<ALERT -- sampling period :5000 ms.
1166369290484 pool-1-thread-1 has executed from 1166369285453 +5031 ms!
>
Full thread dump Java HotSpot (TM) Client VM (1.5.0_10-b03 mixed mode , sharing ):
...
"pool-1-thread-1" prio=6 tid=0x00a94118 nid=0xb70 runnable [0x02cdf000..0x02cdfe8]
            at java.net.PlainSocketImpl.socketConnect (Native Method )
            at java.net.PlainSocketImpl.doConnect (PlainSocketImpl .java :333)
            - locked <0x22bec8b8> (a java.net.PlainSocketImpl )
            at java.net.PlainSocketImpl.connectToAddress (PlainSocketImpl .java :195 )
            at java.net.PlainSocketImpl.connect (PlainSocketImpl .java :182)
            at java.net.Socket.connect (Socket .java :519 )
            at java.net.Socket.connect (Socket .java :469 )
            at sun.net.NetworkClient .doConnect (NetworkClient .java :157)
            at sun.net.www.http.HttpClient .openServer (HttpClient .java :382)
            at sun.net.www.http.HttpClient .openServer (HttpClient .java :494)
            - locked <0x22bec838 > (a sun.net.www.http.HttpClient )
            at sun.net.www.http.HttpClient .<init> (HttpClient .java :231)
            at sun.net.www.http.HttpClient .New (HttpClient .java :304)
            at sun.net.www.http.HttpClient .New (HttpClient .java :316)
            at sun.net.www.protocol.http.HttpURLConnection .getNewHttpClient (HttpURLConnection .java :817)
            at sun.net.www.protocol.http.HttpURLConnection .plainConnect (HttpURLConnection .java :769)
            at sun.net.www.protocol.http.HttpURLConnection .connect (HttpURLConnection .java :694 )
            at sun.net.www.protocol.http.HttpURLConnection .getInputStream (HttpURLConnection .java :938)
            - locked <0x22be47d8> (a sun.net.www.protocol.http.HttpURLConnection )
            at java.net.URLConnection .getContent (URLConnection .java :682)
            at evt.Test1.dispatch (Test1.java :9)
            at evt.SampleTask .run(SampleTask .java :35 )
            at java.util.concurrent .ThreadPoolExecutor $Worker .runTask (ThreadPoolExecutor .java :650)
            at java.util.concurrent .ThreadPoolExecutor $Worker .run(ThreadPoolExecutor .java :675)
            at java.lang.Thread .run (Thread .java :595 )
```

Figure 4



```
0.097: [GC 477K->184K(1984K), 0.0035708 secs]
0.113: [Full GC 309K->223K(1984K), 0.0143731 secs]
0.128: [Full GC 234K->223K(1984K), 0.0122733 secs]
0.142: [Full GC 250K->224K(1984K), 0.0125617 secs]
0.155: [Full GC 240K->223K(1984K), 0.0126203 secs]
0.172: [Full GC 256K->223K(1984K), 0.0122099 secs]
0.186: [Full GC 251K->224K(1984K), 0.0124175 secs]
0.210: [Full GC 257K->225K(1984K), 0.0125765 secs]
0.224: [Full GC 246K->223K(1984K), 0.0126550 secs]
0.238: [Full GC 245K->223K(1984K), 0.0123988 secs]
0.251: [Full GC 235K->224K(1984K), 0.0123127 secs]
0.265: [Full GC 234K->224K(1984K), 0.0122535 secs]
0.282: [Full GC 252K->223K(1984K), 0.0123491 secs]
0.296: [Full GC 233K->224K(1984K), 0.0124628 secs]
0.311: [Full GC 257K->224K(1984K), 0.0125857 secs]
0.325: [Full GC 240K->225K(1984K), 0.0123762 secs]
0.338: [Full GC 230K->222K(1984K), 0.0125337 secs]
0.353: [Full GC 243K->222K(1984K), 0.0124474 secs]
```

A garbage collection log has an excessive Full GC condition
JVM options: -XX:+PrintGCTimeStamps -verbose:gc

Figure 5 Turn on garbage collection log to observe GC activity

'ulimit' command for the current shell. And we can list the open files with the public domain 'lsof' tool. It's intensely interesting that many developers don't explicitly use the 'close()' method in the final block when an object inherently provides a 'close()' method and want JVM to release these unclosed objects when garbage is collected. We should keep firmly in mind that *that act is bad without closing the system resource after use*. A special case is when the socket connections in the application don't close properly while still being underdeployed and then the application begins to throw an IOException with a 'Too many open files' message after repeated application redeployment.

Connection leaking is like the resource contention. However, we have to get rid of any connection leaking in our application code. If the bad application code isn't removed, the contention leaking issue will be endless and destructive. Though there is usually an inactive connection timeout option in an application server's connection pool, it doesn't have great advantages here when the leaking connection number is repeatedly increasing. Please get used to explicitly closing unused JDBC connections and associated JDBC objects (Connection, ResultSet, Statement, etc.) in a final block in correct sequences.

Thread blooming is a divergent phenomenon when running a thread pool. In general, in any computation environment, we should avoid this uncommon condition because too many threads in the JVM will cause expensive thread context switching inside the OS kernel. It also lowers the CPU usage that each thread can have. An ordinary case happens when these additional threads are introduced by a third-party class library rather *than the code we did directly*. A concrete example is when we use the LDAP API without using an LDAP connection pool. Due to this potentiality, *orderly evaluate the total thread number in every copy of the thread dump that we have gotten.* Some OS utilities may show the total number of threads at the process level. (For example, HP Glance, Solaris prstat, etc.) It's very important that the EJB specification *restricts not managing threads in an EJB implementation* (the Enterprise JavaBeans specification 2.0, Section 25.1.2 – Programming Restrictions). If we violate any of these restrictions without compliance, it will mean our efforts are in vain. Furthermore, a thread blooming condition could originate from the auto-expansion of the thread pool when its threshold meets whether it's busy or blocked. If we've set the thread-increasing number to greater than zero in the configuration of a thread pool, *don't set too high a value for the maximum thread count that a system can offer and carry*.

Excessively Full GC pause time is another severe phenomenon. It happens when application code/library is extensively called 'System.gc()' or has a memory leak condition that's going to run out of the remaining Java heap. We sense that system performance is slow, and we often can't find anything obvious from the thread dump except a poor garbage collection log. But CPU usage is higher than usual. If we analyze the breakdown of CPU usage by thread, we may find the top usage is always led by the VM thread. During the pause for garbage collection, the JVM suspends the application threads for internal housekeeping. A full GC takes longer than a minor GC, maybe hundreds of milliseconds to seconds, because it's doing heap sweeping and compaction. When we see a lot of contiguous full GCs in the GC log and the interval between two adjacent GCs is short and each pause takes a high percentage during the GC interval, we should check to see if the thread dump has threads whose stack traces were calling 'System.gc()' simultaneously *to have forcible garbage collections incessantly*. A unique case is the use of RMI programming. It will bring up a GC daemon thread in the JVM and periodically have a 'System.gc()' method call in a 60,000ms default. It's tunable by setting a pair of system properties (sun.rmi.dgc.server.gcInterval and sun.rmi.dgc.client.gcInterval). We can disable the programmatic GC by a special JVM option that all JVM vendors provide (e.g., '–XX:+DisableExplicitGC' for the Sun JVM). When a JVM owns an insufficient heap size or there's a memory leak condition that keeps decreasing the free heap size, it will try to reclaim space *with GCs by best efforts*. A strange full GC condition is caused by the violation of the famous young-generation guarantee

JavaOne℠

Sun's 2007 Worldwide Java Developer Conference℠

May 8-11, 2007

The Moscone Center
San Francisco, CA

JavaOne™ Pavilion: May 8-10, 2007

IT'S AN EXCITING TIME FOR THE ENTIRE JAVA™ TECHNOLOGY COMMUNITY.

With the evolution of the Java platform, new opportunities are emerging for developers, thought leaders, and entrepreneurs. That's why for 2007 the Conference is featuring an expanded program that embraces technologies outside the core Java platform while keeping Java™ technology at the center. You'll get the same in-depth content we have always focused on, plus more topics and issues relevant to today's evolving market.

LEARN MORE ABOUT:*

> Java Technology
> Scripting
> Open Source and Community Development
> Integration and Services-Oriented Development
> Web 2.0
> Compatibility and Interoperability
> Business Management
> And More

Java™

PLATINUM COSPONSORS

intel ORACLE

GOLD COSPONSOR

NAVTEQ

SILVER COSPONSORS

InterSystems PARASOFT. We make software work.

SAVE $200**

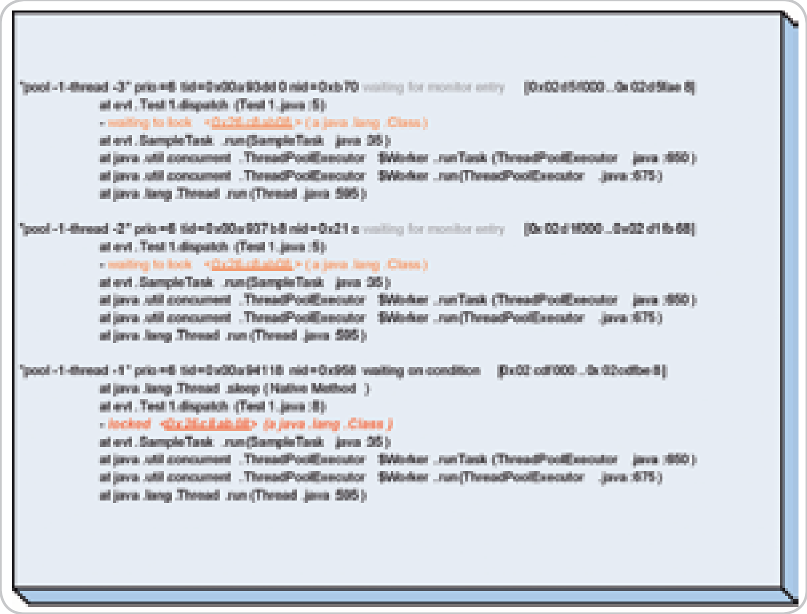
REGISTER BY
APRIL 4, 2007

Please use priority code: J7PA1JDJ

*Content subject to change.
**Offer not available on site.

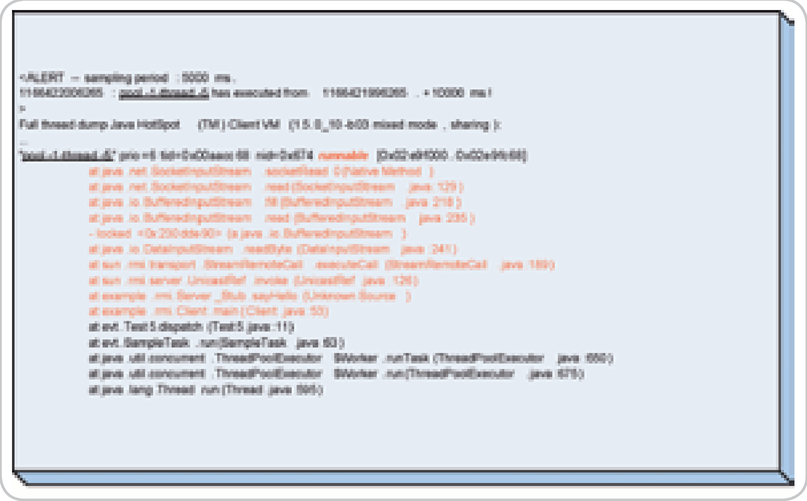
Register Today @ java.sun.com/javaone

Sun microsystems



```
"pool-1-thread-3" prio=6 tid=0x00a93dd0 nid=0xb70 waiting for monitor entry    [0x02d5f000..0x02d5fae8]
        at evt.Test1.dispatch (Test1.java:5)
        - waiting to lock   <0x26c8ab08> (a java.lang.Class)
        at evt.SampleTask .run(SampleTask   java:35)
        at java.util.concurrent .ThreadPoolExecutor   $Worker .runTask (ThreadPoolExecutor   java:650)
        at java.util.concurrent .ThreadPoolExecutor   $Worker .run(ThreadPoolExecutor   .java:675)
        at java.lang.Thread .run (Thread .java :595)

"pool-1-thread-2" prio=6 tid=0x00a937b8 nid=0x21c waiting for monitor entry    [0x02d1f000..0x02d1fb68]
        at evt.Test1.dispatch (Test1.java:5)
        - waiting to lock   <0x26c8ab08> (a java.lang.Class)
        at evt.SampleTask .run(SampleTask   java:35)
        at java.util.concurrent .ThreadPoolExecutor   $Worker .runTask (ThreadPoolExecutor   java:650)
        at java.util.concurrent .ThreadPoolExecutor   $Worker .run(ThreadPoolExecutor   .java:675)
        at java.lang.Thread .run (Thread .java :595)

"pool-1-thread-1" prio=6 tid=0x00a94118 nid=0x958 waiting on condition    [0x02cdf000..0x02cdfbe8]
        at java.lang.Thread .sleep (Native Method  )
        at evt.Test1.dispatch (Test1.java:8)
        - locked  <0x26c8ab08> (a java.lang.Class )
        at evt.SampleTask .run(SampleTask   java:35)
        at java.util.concurrent .ThreadPoolExecutor   $Worker .runTask (ThreadPoolExecutor   java:650)
        at java.util.concurrent .ThreadPoolExecutor   $Worker .run(ThreadPoolExecutor   .java:675)
        at java.lang.Thread .run (Thread .java :595)
```

Figure 6 Calls blocked on a synchronized method



```
<ALERT -- sampling period  : 5000 ms.
1166422306265  : pool-1-thread-5 has executed from   1166421996265  .. + 10000 ms!
>
Full thread dump Java HotSpot   (TM) Client VM  (1.5.0_10-b03 mixed mode , sharing ):
...
"pool-1-thread-5" prio=6 tid=0x00aacc68 nid=0xd74 runnable  [0x02e9f000..0x02e9fc68]
        at java.net.SocketInputStream   .socketRead0 (Native Method  )
        at java.net.SocketInputStream   .read (SocketInputStream   .java: 129 )
        at java.io.BufferedInputStream   .fill (BufferedInputStream   .java 218 )
        at java.io.BufferedInputStream   .read (BufferedInputStream   java 235 )
        - locked  <0x230dde90> (a java.io.BufferedInputStream   )
        at java.io.DataInputStream   .readByte (DataInputStream   java :241 )
        at sun.rmi.transport .StreamRemoteCall   .executeCall (StreamRemoteCall   .java 189)
        at sun.rmi.server.UnicastRef  .invoke (UnicastRef  .java :126)
        at example .rmi.Server_Stub .sayHello (Unknown Source  )
        at example .rmi.Client .main (Client .java 53)
        at evt. Test5.dispatch (Test5.java :11)
        at evt. SampleTask  .run(SampleTask  java 63 )
        at java.util.concurrent  .ThreadPoolExecutor   $Worker .runTask (ThreadPoolExecutor   java 650)
        at java.util.concurrent  .ThreadPoolExecutor   $Worker .run(ThreadPoolExecutor   java 675)
        at java.lang.Thread  .run (Thread java 595)
```

Figure 7 Blocked on a remote method call

(see http://java.sun.com/docs/hotspot/gc1.4.2). Increasing the young generation becomes counterproductive at half the total heap or less (whenever the young-generation guarantee can't be met). Here the JVM continues crazy full garbage collections because of the violation against the guidelines of the Sun Hotspot JVM on most of garbage collectors (see Figure 5).

The internal transaction process with a slow resource manager condition will reveal itself when the stuck thread has a transaction manager implementation class in the thread dump. We have to review the log for transaction timeout errors and transaction exceptions to get the root cause. The cause might be an incompatible resource manager (e.g., a non-XA JDBC driver and database), a too short value for the abundant timeout of a two-phase commit or other deadlock issues on the resource manager. Carefully check the transaction log about transaction timeout rollbacks and application rollbacks; it may help us to rule out this nuisance.

A huge object serialization effort in the network or 'call by value' semantic in applications will cause stuck threads too. The RMI programming that current J2EE components widely use to serialize a big object graph is also a big performance penalty for an application. If we see the stack trace of a stuck thread is doing serialization, it means either the networking is slow or the serialization cost is very expensive. It's better to minimize the serialization cost by using data transfer objects and J2EE core patterns to convey objects in a coarse-grained granularity manner and reduce the serialization cost for remote calls and transactions between the client and server.

Incautiously synchronized method modifiers in the application running on a thread pool will cause a serious disaster. Once threads in a thread pool *blocked by other threads acquire the object lock***,** it means that the threads waiting for monitor entry are doing nothing. While a thread gets the object lock, other threads subsequently executing the relevant synchronized methods become blocked sequentially because only one of them can exclusively acquire the lock they're waiting for. So, improperly synchronized statements will weaken system performance. We can look to see whether the running state of the stuck thread is 'waiting for the monitor entry' and whether the object is 'waiting to lock' to know which piece of code is doing synchronization.

There is also the philosophy of avoiding using synchronized methods on an application implementation running on a J2EE server. J2EE specifications are trying to alleviate the pain of multithreaded programming *from its design*. That means server code has enforced the proper synchronization mechanism for applications (e.g., the servlet's single thread model). For example, 'DriverManager.getConnection()' method is a class synchronized method and we should use a DataSource object to get the JDBC connection. A bad synchronized method call could let the threads in the thread pool fall into an unrecoverable or time-consuming condition *for waiting the lock*. By the way, notice the safe use of many core Java classes like System, Thread and Runtime. For example, calling 'System.exit(0)' will end the JVM silently if the JVM doesn't enable a security manager to check it's runtime permission (see Figure 6).

A remote invocation hang or a slow response to a remote call from other servers usually happens on an n-tier architecture. If a remote call (e.g., calling an EJB remote method) doesn't respond in a timely fashion, it will cause the client to queue and block. It's possible that the symptom will propagate to the front-end like a chain reaction. RMI calls provide a way to time out the communication through the 'RMISocketFactory' class. However, we should be aware if the service we're calling is idempotent or transactional *in the essence of the service*. The other way is trying to adjust the invocation logic from synchronous to asynchronous *that may have a form of benefit* (see Figure 7).

Slowness while a huge I/O is writing via a slow media is a common issue. It depends on the client network/media speed to complete the I/O work.

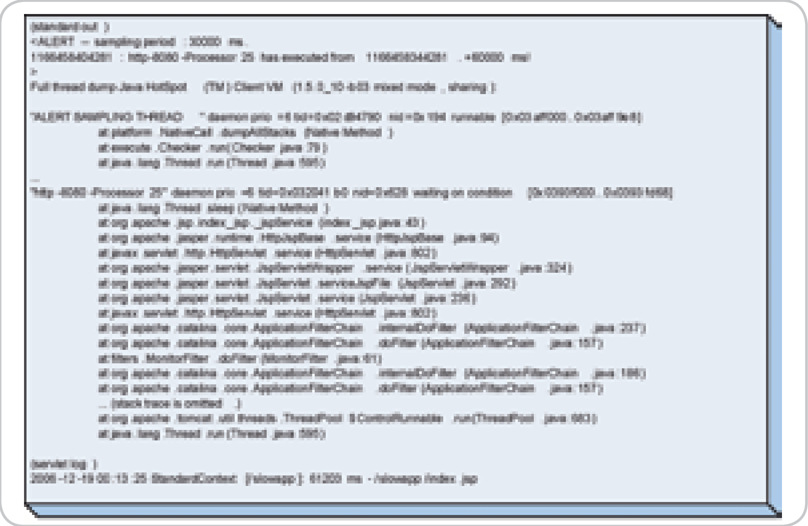
When the media speed is slow, it causes the I/O to be bounded. For example, we have to transmit a very complex data stream to a slow remote client through a 19.2 Kbps modem; it will consume a lot of execution time on the I/O work. Another one is using too detailed a log level and it doesn't filter out the unnecessary messages generated for debugging, the log size increases tremendously and floods the disk space. A more particular case is when the network bandwidth is unexpectedly eaten by malicious software and leaves a small bandwidth for normal operation.

Blocking with loop conditions will exhaust all the CPU power and other threads won't have a normal schedule to run. We may find the system utility to get the thread usage first. Rely on other system tools to find the mapping between native process threads and Java threads to get the suspected thread stack trace in the thread dump. One bad example is the 'java.awt' package repeatedly drawing complex images that consume a lot of CPU usage. An interesting thing is that we occasionally find the root cause is server code rather than application code.

A premature version of the JVM during bytecode optimization or garbage collection may cause stuck threads. Once we find the thread dump isn't trustworthy because it doesn't contain the information about the *thread locked the object* that other threads were waiting to acquire. It's best to check the JVM and OS-supported configuration and its successors. A bad thing is that running the OS doesn't install the required kernel patches that the current JDK version needs. This negligence can cause instability.

A JNI call can certainly cause stuck threads. On some EAI systems, a Java application tends to connect the existing legacy systems through a JNI call with native libraries. When the system being called has a small capacity, stuck threads often occur. In this situation, it requires some isolation to shield the stability from a JVM crash and system hang.

In this article, we provided a small utility that can be embedded in our Web applications to monitor stuck threads. It implements the 'javax.servlet.Filter' interface and uses a daemon thread to check the entry timestamp that the threads stored. To pick out the stuck threads efficiently, the sampling time should be lower than half the execution time the stuck thread took (this is the Nyquist-Shannon sampling theorem). This filter class uses the Checker.register() and Checker.unregister() methods to detect stickiness and logs the execution time that the currently accessed Web URI spent for further analysis. As an experiment, we can put the 'informer.jar' file including the 'stuck.Checker' class in Tomcat's 'common/lib' directory and put the required JNI shared library that triggers the JVM thread dump on a correct path (java.library.path). Subsequent things are to adjust a proper sampling period for our own Web applications on the start-up script and configure the web.xml to install this filter class (see Figure 8).



```
[standard out ]
<ALERT -- sampling period : 30000 ms.
1166458404281 : http-8080-Processor 25 has executed from 1166458344281 ..+60000 ms!
>
Full thread dump Java HotSpot (TM) Client VM (1.5.0_10-b03 mixed mode , sharing )

"ALERT SAMPLING THREAD" daemon prio =6 tid=0x02 db4790 nid=0x 194 runnable [0x03 aff000..0x03aff fe8]
          at platform .NativeCall .dumpAllStacks (Native Method )
          at execute .Checker .run(Checker .java 79 )
          at java .lang .Thread .run (Thread .java 595)
...
"http-8080-Processor 25" daemon prio =6 tid=0x032841 b0 nid=0x628 waiting on condition [0x0350f000..0x0350 fd68]
          at java .lang .Thread .sleep (Native Method )
          at org.apache .jsp.index_jsp._jspService (index_jsp.java 43)
          at org.apache .jasper .runtime .HttpJspBase .service (HttpJspBase .java 94)
          at javax .servlet .http.HttpServlet .service (HttpServlet .java 802)
          at org.apache .jasper .servlet .JspServletWrapper .service (JspServletWrapper .java 324)
          at org.apache .jasper .servlet .JspServlet .serviceJspFile (JspServlet .java 292)
          at org.apache .jasper .servlet .JspServlet .service (JspServlet .java 236)
          at javax .servlet .http.HttpServlet .service (HttpServlet .java 802)
          at org.apache .catalina .core.ApplicationFilterChain .internalDoFilter (ApplicationFilterChain .java 237)
          at org.apache .catalina .core.ApplicationFilterChain .doFilter (ApplicationFilterChain .java 157)
          at filters .MonitorFilter .doFilter (MonitorFilter .java 61)
          at org.apache .catalina .core.ApplicationFilterChain .internalDoFilter (ApplicationFilterChain .java 186)
          at org.apache .catalina .core.ApplicationFilterChain .doFilter (ApplicationFilterChain .java 157)
          ... [stack trace is omitted ]
          at org.apache .tomcat .util.threads .ThreadPool $ControlRunnable .run(ThreadPool .java 683)
          at java .lang .Thread .run (Thread .java 595)

[servlet log ]
2006-12-19 00:13:25 StandardContext [/slowapp ]: 61203 ms - /slowapp /index .jsp
```

Figure 8 A small utility to detect stuck threads caused by the webapp deployed on Apache Tomcat application server

By introducing the anti-patterns that cause stuck threads, we should intrinsically strengthen application quality and prevent stuck threads when involved in a thread pool implementation. We may get the JVM internals from a thread dump when we're focused on administering and maintaining a production environment. The most important thing we have learned is to prevent the system from hanging by root cause identification when we found a stuck thread. Many dim syndromes are usually infected by little stuck threads in the beginning phase and they become lethal when their states are irrecoverable. So let's free them from the referred scenarios that we may avoid it in advance.

Wait-Time Analysis Method

A new best practice for application and database performance management

by Don Bergal

Until recently, tuning IT application performance has been largely a guessing game. This is both surprising and unacceptable considering the relentless focus IT organizations put on cost-efficiency and productivity. The traditional approaches to database and application tuning that involve collecting large volumes of statistics and making trial-and-error changes are still in widespread use. Today, most server management and monitoring tools deliver "server-oriented" statistics that don't translate to concrete end-user benefits.

The landscape is changing, however. The current thinking of leading consultants, DBAs, and training organizations is focusing on performance tuning practices that are tied directly to end-user service levels and improvements in operating efficiency.

Wait-Time analysis is a new approach to application and database performance improvement that allows users to make tuning decisions based on the optimal service impact. Using the principles of Wait-Time analysis described here, DBAs, developers, and application owners can align their efforts with the service levels desired by their IT customers. Wait-Time analysis lets IT find the root cause of the most important problem impacting customers and identify which critical resource will resolve it.

What Is Wait-Time Analysis?

Measure Time

If you were trying to shorten your commute to work, what would you measure? Would you count the number of tire rotations? Would you measure the car's temperature? Would these statistics have any meaning in the context of your goal? All that really matters is what impacts the time for your trip. All the other statistics are distractions that don't help your mission. Wait-Time analysis gets to the root of the problem to achieve the end business result. Although this seems obvious, common IT practices suggest that other practices hold the answer. Rather than immediately focusing on the time to complete requested services, IT tools barrage the user with detailed statistics that count the number of many different operations. So while the DBA should really be looking at how long it took for the database to return the results of a query, typical tools display the number of input/output (I/O) operations and locks encountered.

Get the Details

Under the trial-and-error approach, what level of detail do you need to actively improve your commute time? If the only statistic you have is that the trip took 40 minutes, you can compare one day to the next, but there's not enough data to help improve the situation. What you need is detailed insight into how long you spent at each stoplight, which stretches of road have the most stop-and-go traffic and how long you waited there. This detail is essential to making the exercise useful.

The same concept applies to IT performance systems. When Wait-Time is typically measured, a "black box" approach is taken, where the user sees how long a server took to respond to a request. However, no indication is given as to which of the thousands of steps performed by the server were actually responsible for the delay. As will be shown here, it's important not just to measure Wait-Time but to break it down into sufficient detail so that you can take action.

Wait-Time analysis for IT applications is the singular focus of measuring and improving the service time to

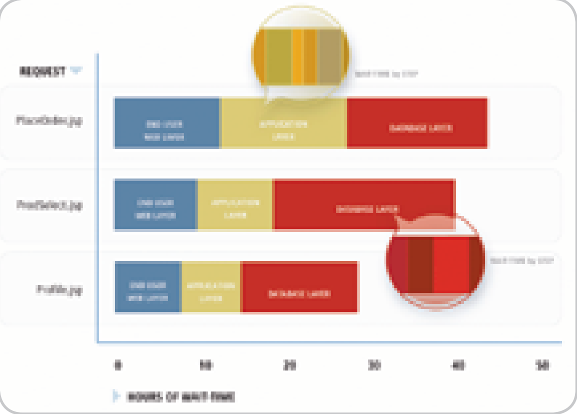


Figure 1 Ideal analysis – exactly where is every request spending time? Identifying every step inside each layer for every request exposes the true sources of end-user Wait-Time

Don Bergal is the chief operating officer of Confio Software and is responsible for overseeing all of the company's sales, marketing, product management, and business development initiatives. Don has over 15 years experience in the software, services, and data communications industries. He earned a BS in engineering from the University of Michigan and an MBA from the Harvard Business School.

DonBergal@confio.com
or see *www.confio.com*

“Wait-Time analysis allows DBAs to find the root cause of the most important problem impacting customers and identify what critical resource must be addressed to resolve it. Used in heterogeneous Oracle environments, Wait-Time analysis is the singular focus on measuring and improving service time to end-users”

the IT customers. By identifying exactly what contributes to longer service time, IT professionals can focus not on the thousands of available statistics, but on the most important bottlenecks that have direct and quantifiable impact on the IT customer.

Wait-Time Analysis for Service Level Management

Because Wait-Time analysis measures the collective time delays causing end users to wait for an information request, it's the measurement technique most closely matched to end-user service levels. For organizations focused on Service Level Management (SLM) techniques, or those bound by Service Level Agreements (SLAs), Wait-Time analysis techniques allow the IT department to measure the performance that is most relevant to achieving the stated service level goals. Service level management typically identifies technical metrics that define whether performance is adequate, and Wait-Time data is the basis for evaluating those metrics.

The Problem with Conventional Statistics

There are so many management tools gathering thousands of statistics from IT systems. Don't these provide the same answer as Wait-Time methods? Why are they not effective?

Traditional approaches to database tuning and performance analysis introduce the same errors identified in the driving example above.

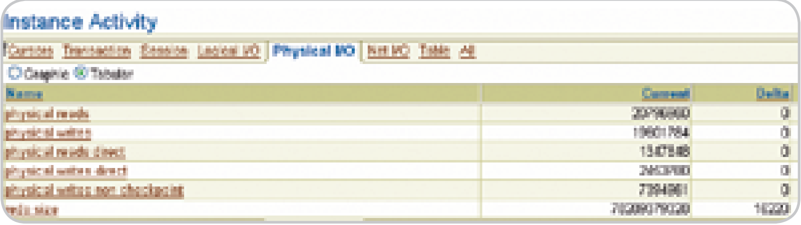
1. Event Counters versus Wait-Time Methods

Typical tools count the number of **events**, but don't measure **time**. These statistics are numerous and easy to capture, so they tend to flood management dashboards. But, are they useful?

Broad management dashboards have sophisticated displays of monitored data, but counting events or calculating ratios doesn't indicate or predict better performance for database customers. In fact, this approach can have the effect of covering up, rather than exposing, the real service level bottlenecks.

The example is an excerpt from a long summary of counted statistics. Clearly there's much detail and technical accuracy. But where would you go to begin your diagnosis? Do these raw numbers reveal a performance problem? Is the value for “*physical writes direct*” in the table too high or too low? There's no indication of impact on the end-user service level to make that judgment.

On the other hand, Figure 3 ranks individual SQL requests by Wait-Time. The statement with the highest Wait-Time is at the top of the list. Its relative impact on overall user service is reflected in the length of the bar – measuring how much time users experience waiting on this request. Without counting how many times an operation occurred, this is a much more meaningful measure of end-user service.


Instance Activity

Cursors Transaction Session Logical I/O **Physical I/O** Net I/O Table All

○ Graphic ● Tabular

Name	Current	Delta
physical reads	20796960	0
physical writes	19601784	0
physical reads direct	1347848	0
physical writes direct	2463700	0
physical writes non checkpoint	7394961	0
redo size	70280079028	16220

Figure 2 Typical event counter statistics. Too high or too low? Many standard tools supplied by database vendors focus on counting events, not identifying Wait-Time

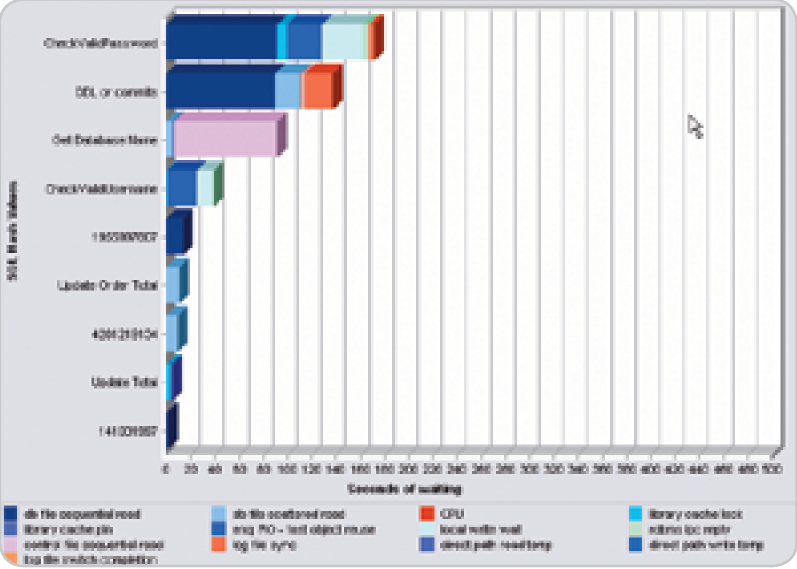
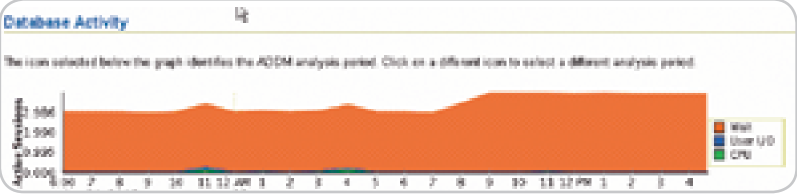


Figure 3 Wait-Time display immediately identifies the problem. Wait-Time analysis, like the example shown here, tells the DBA or system manager: “This is where you need to focus, here is the exact problem”


Database Activity

The icon selected below the graph identifies the ADDM analysis period. Click on a different icon to select a different analysis period.

Active Sessions
2.596
1.596
0.596
0.000
6:00 7 8 9 10 11 12 AM 1 2 3 4 5 6 7 8 9 10 11 12 PM 1 2 3 4
Wait
User I/O
CPU

Figure 4 Performance averaged across the entire database. System-wide averages aren't actionable, since they don't identify the exact point of the bottleneck. Conventional vendor-supplied tools use system-wide averages

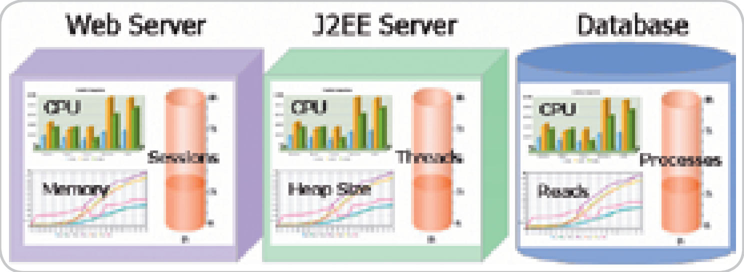
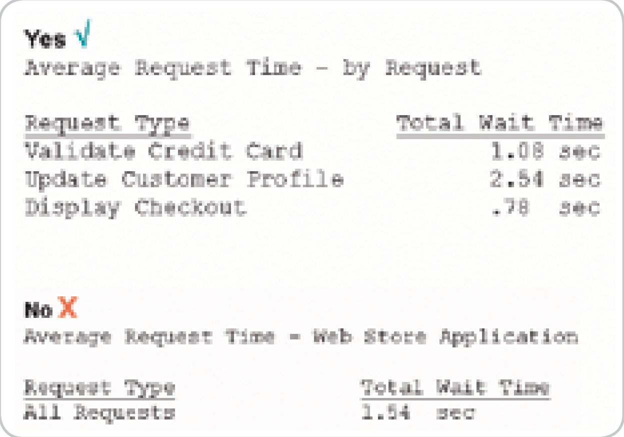
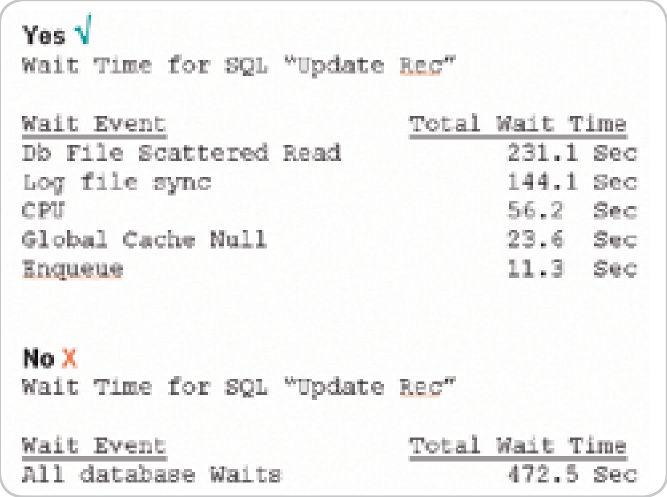


Figure 5 A traditional silo-based measurement of server statistics. Focusing on individual silos without connecting the result to end-user service time results in a system optimized for the system admin, not the customer



```
Yes √
Average Request Time – by Request

Request Type                 Total Wait Time
Validate Credit Card                1.08 sec
Update Customer Profile             2.54 sec
Display Checkout                     .78 sec


No X
Average Request Time – Web Store Application

Request Type              Total Wait Time
All Requests              1.54   sec
```

Figure 6 Seeing individual requests tells where customer service is slow. A system average doesn't identify the problem



```
Yes √
Wait Time for SQL "Update Rec"

Wait Event                    Total Wait Time
Db File Scattered Read              231.1 Sec
Log file sync                       144.1 Sec
CPU                                  56.2 Sec
Global Cache Null                    23.6 Sec
Enqueue                              11.3 Sec


No X
Wait Time for SQL "Update Rec"

Wait Event                    Total Wait Time
All database Waits                  472.5 Sec
```

Figure 7 Seeing every step individually shows which ones are bottlenecks. An average of all the steps doesn't identify the problem

2. System-Wide Averages

Typically statistics are gathered across an entire system, rather than on a basis that applies to an individual user request. When averaging performance across all requests, it becomes impossible to tell which requests are the most critical resource drains and which resources are impacting service levels.

Vendor-supplied database tools, for example, typically display data across the entire database without breaking it down into specific user requests. As a result, there's no indication which end-user functions were impacted.

3. Silos versus End-to-End Analysis

Another key problem with typical IT monitoring tools is the creation of individual information "silos" that localize statistics for a single type of system, but don't expose an end-user's view of performance.

Because of the differing technical skill sets, separate groups manage databases, application servers, and Web infrastructure. Each group has a primary focus – to optimize the performance of their box. And typically they use the most common and convenient statistics to measure and improve performance. For an application server, this often means watching memory utilization, thread counts, and CPU utilization. For a database, this is a count of the number of sessions, number of reads, or number of processes.

The problem is that there's no view of the end objective – minimizing service time for the customer – and no collaboration across these groups focusing them beyond their individual server operations. In reality, the database bottlenecks are a direct result of the application procedure calls while the application responds to Web requests. All of these combine to have a direct impact on end-user service. Without the ability to track the flow of transactions across the multiple systems, each IT group can only try to optimize its own statistics, not of the response time to the customer.

4. Finger Pointing

The real trouble starts when a cross-functional group assembles to try to respond to a customer-reported problem. With each department watching their own server-oriented statistics, the result is a "finger-pointing" session where blame is deflected from one group to the next.

Without a performance measurement system that identifies in exact detail the root cause of the performance bottleneck, finger pointing becomes inevitable. By contrast, by measuring end-user Wait-Time with the

> "Another key problem with typical IT monitoring tools is the creation of individual information 'silos' that localize statistics for a single type of system, but don't expose an end-user's view of performance"

> “Wait-Time analysis is an approach to application performance management that captures and delivers data in a way that enables business decisions that have optimal service impact”

recommended detailed granularity, management can identify exactly where in the IT value chain the bottleneck lays and who is really responsible. Using Wait-Time techniques to pinpoint the source of the problem helps eliminate the finger pointing.

Key Requirements for Wait-Time Analysis

Wait-Time analysis is an approach to application performance management that captures and delivers data in a way that enables business decisions that have optimal service impact.

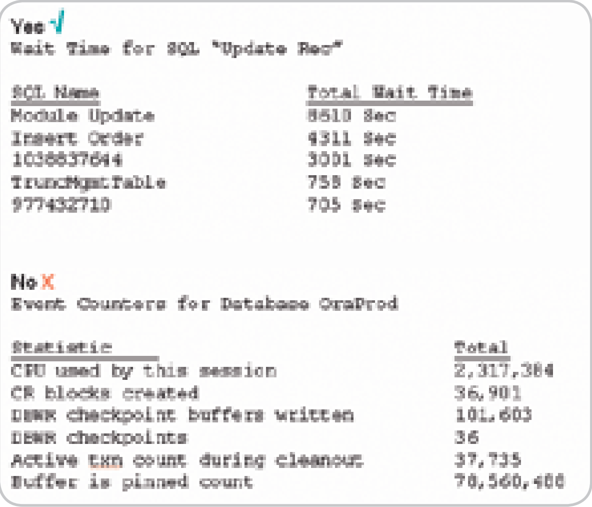
The foundations of Wait-Time analysis are three requirements – measuring **User Requests**, measuring **Every Step**, and measuring accumulated **Time**.

Requirement One: Every User Request – Individually

This requirement states that all IT performance statistics must correspond to specific user requests, not averages across the entire system. Individual SQL statements or Web user screens must be tracked individually as they pass through the respective servers. In a database or application server, mixing data across all requests has the effect of averaging all responses and hiding any unique information about the request of interest. To effectively identify the problem, each SQL or Java application screen must be monitored and optimized separately.

Requirement Two: Every Step

To be actionable, every user request must be measured with sufficient granularity to identify each step taken along the path from the end user through the database. This requires more detail than simply designating the database layer as the source of delay. It requires measuring each of the individual processes along the execution path. For an Oracle or SQL Server database, these steps correspond to hundreds of individual Wait-Events. For a Web application, the steps can be Java methods that are executed on an application server.



```
Yes √
Wait Time for SQL "Update Rec"

SQL Name                          Total Wait Time
Module Update                     8610 Sec
Insert Order                      4311 Sec
1038837644                        3001 Sec
TruncMgmtTable                    758 Sec
977432710                         705 Sec


No X
Event Counters for Database OraProd

Statistic                                   Total
CPU used by this session                    2,317,384
CR blocks created                           36,901
DBWR checkpoint buffers written             101,603
DBWR checkpoints                            36
Active txn count during cleanout            37,735
Buffer is pinned count                      78,560,488
```

Figure 8 Measuring Wait-Time on the individual steps shows the most serious problem. Counting the number of steps processed has no bearing on end-user service

You can’t take action if all you know is that your request waits on Java or Oracle. But if you know that your request is hung in a specific “getCreditCard.do” method or “Enqueue” database lock then you have sufficient detail to productively work the problem.

Requirement Three: Measure Time

The most important requirement is measuring the time spent on a request, not counting how often a computing resource was used. The principle follows logically from the business purpose of the information system, which is to process requests and deliver output as quickly as possible. Counting events provides no indicator of how long a database user must wait for a response, or how long a request must wait for the execution of a Java method.

In the Wait-Time service-oriented performance approach, time is the most important resource to measure.

Conclusion: Practical Considerations for Wait-Time Analysis

The Wait-Time approach to performance monitoring described here is only practical if it can be implemented efficiently in a performance-sensitive production environment. While basic tools to extract Wait-Time from databases on an individual session basis were the first step in this type of analysis, more efficient approaches have now been developed that meet ease of use, low impact, and continuous monitoring requirements. Beyond the database, it’s now possible to employ Wait-Time analysis efficiently in end-to-end production application environments.

With increased focus on service levels as the most important measure of IT productivity, Wait-Time analysis has come to the forefront as the monitoring technique that ties IT practice to overall IT goals. This movement is aided by the combination of superior results experienced by leading consultants and trainers as well as the availability of excellent packaged tools. Wait-Time analysis tells the IT organization exactly where the problem lies, who should fix it and how it impacts the customer. Unlike traditional methods that barely deliver clues, Wait-Time, implemented in sufficient detail, delivers answers.

> “Businesses that ignore the potential of SOA will find themselves outpaced by rivals who improve their agility and transform themselves into new kinds of enterprises
>
> — Yafim Natis, Gartner Analyst

3-DAY EVENT!

SOAWorld 2007

Plus

Enterprise OpenSource Conference & Expo 2007

TOPICS INCLUDE:

SOA Web Services

- AJAX and SOA
- Web 2.0
- Universal SOA
- Protecting Web Services
- Troubleshooting SOA
- Governance
- Open-Source SOA
- XBRL
- Service Virtualization

Open Source

- Open Source Business Models
- Open Source ESB
- OpenAjax Alliance
- SaaS and Open Source
- Spring, Hibernate and Eclipse
- Seam
- Open Source Penetration
- Monetizing Open Source
- Open Source Databases
- AMQP
- Open Source Middleware

June 25-27, 2007

Roosevelt Hotel / New York City

Register Online! www.SOAWorld2007.com

SOAEOSCONFERENCE.SYS-CON.COM

REGISTER ONLINE TODAY

SAVE $200!

(HURRY FOR EARLY-BIRD DISCOUNT)

11th International SOAWORLD™ 2007 CONFERENCE & EXPO

2nd Annual ENTERPRISE> 2007 OPENSOURCE CONFERENCE+EXPO

2007 is to many industry insiders shaping up to be a major inflection point in software development and deployment, with SOA, Web Services, Open Source, and AJAX all converging as cross-platform and cross-browser apps become the rule rather than the exception.

Accordingly the 11th International SOA Web Services Edge 2007 again seeks to offer comprehensive coverage and actionable insights to the developers, architects, IT managers, CXOs, analysts, VCs, and journalists who'll be assembling as delegates and VIP guests in The Roosevelt Hotel in downtown Manhattan, June 25-26, 2007

Co-located with the 2nd Annual Enterprise Open Source Conference & Expo, the event will deliver the #1 i-technology educational and networking opportunity of the year. These two conference programs between them will present a comprehensive view of all the development and management aspects of integrating a SOA strategy and an Open Source philosophy into your enterprise. Our organizing principle is that delegates will go away from the intense two-day program replete with why-to and how-to knowledge delivered first-hand by industry experts.

Visit soaeosconference.sys-con.com for the most up-to-the-minute information including... Keynotes, Sessions, Speakers, Sponsors, Exhibitors, Schedule, etc.

BROUGHT TO YOU BY:

SOA Web Services Journal focuses on the business and technology of Service-Oriented Architectures and Web Services. It targets enterprise application development and management, in all its aspects.

Enerprise Open Source Magazine EOS is the world's leading publication showcasing every aspect of profitable Open Source solutions in business and consumer contexts.

SYS-CON EVENTS For more great events visit www.EVENTS.SYS-CON.com

Exhibit and Sponsorship Info:

Call 201-802-3020 or email events@sys-con.com

Joe Winchester
Desktop Java Editor

E-mail – Problem Solved or Created?

At the annual Alan Turing memorial lecture given by Grady Booch in London last month, he chose as his subject, *The promise, the limits, and the beauty of software*. It was an excellent address in which one of the themes was that for each of the incredible advances that software has brought to our lives, there is an almost Newtonian opposite effect that is negative and destructive. One such example given was e-mail: while making us able to communicate instantly with our peers, allowing effective and immediate information sharing, it brings its own set of problems. Issues with information theft, virus attachments, phishing, worms, and privacy are well documented and are very real threats although, to a certain extent, these are merely mirrors of real-world phenomena that e-mail merely amplifies and concentrates. The question that interests me the most is whether e-mail actually increases or decreases communication effectiveness.

The success of humans as a species over other animals can almost solely be attributed to our ability to communicate with each other. While this may have evolved as a mechanism to simply aid survival in the wild, it allows one generation to capture and record knowledge and information that it can share with its offspring. They in turn don't have to rediscover the fundamentals of how to perform a particular task, such as hunting, weaving, or building homes, and can enjoy a more efficient existence than their forefathers, and as a result can increase the body of acquired knowledge for their offspring. Discourse between individuals took the form of face-to-face encounters, where body language, speech inflexion, emotions, etc., are all present and the physical nature of a meeting constrains its length and duration. Telephones and, to a certain extent, instant message clients, extend the distance over which this exchange of ideas and information can occur. Printing, the act of recording words in text, was designed primarily as one-way flow of information between the writer and the reader. Twenty-first century e-mail is something that challenges us because, being text based, it lies in the family of printed media; however, its usage is less for publishing information and more as a forum for exchanging ideas and opinions and performing long-distance discussions.

Joe Winchester is a software developer working on WebSphere development tools for IBM in Hursley, UK.

joewinchester@sys-con.com

One problem this brings is that once there are more end points and inputs available, the phenomenon *information overload* occurs. This describes a state in which a person isn't able to make a decision or informed choice about a particular topic because he or she has too much data to digest. It occurs because e-mail often creates a low signal-to-noise ratio where important content is buried in a swath of reply lists and carbon-copied exchanges that the recipient has to trawl through, often following external links to further citations that then link to more and more sources of information, making the information-gathering process potentially endless and unbound. A *New Scientist* report, http://www.newscientist.com/article.ns?id=dn7298, found that the net effect of e-mail information overload was to lower the recipient's IQ by 10 points; the same drop that occurs by missing a full night's sleep. Interestingly, men did twice as worse as women in the research, although whether this is due to the ability to process the volume of information more effectively or organize ones time more efficiently isn't concluded by the study.

E-mail addiction is another disturbing problem that has its own advice centers and, ironically, mailing lists you can subscribe to and receive helpful advice in your in-box http://www.netaddiction.com. Behavioral traits to watch for include answering messages during meals or social engagements. The reason this is problematic is because the urge to open the mail isn't because the sender is expecting it to occur; it's because the recipient's urge to know why and by whom he is being contacted is more interesting than his current surroundings. It reminds me of the old adage, "*How do you keep a fool in suspense?*", to which the answer is, *"I'll tell you tomorrow."*

Anthroposemiotics, the study of group dynamics, identifies that e-mail, like driving a car, gives people a sense of anonymous power. The same emotion that gives rise to road rage can cause people to exhibit aggressive behavior with e-mails where the language and tone of the exchange is one that would never occur if the conversation occurred face to face. Senders of flame mail are disconnecting themselves from reality by using their computer as an alternative reality to which their alter ego belongs. Just as video game fights and conflicts allow people to vent aggressive frustrations in imaginary virtual worlds, the computer as a machine is our medium in which people can reinvent themselves and engage foes with seemingly physical impunity.

The ineffectiveness of e-mail has even prompted various companies to ban it on certain days of the week, forcing employees to resort to more traditional communication means, such as having a chat in the hallway or talking to a colleague over the phone. Others have gone even further – including complete corporate-wide e-mail bans http://news.zdnet.co.uk/itmanagement/0,1000000308,39116502,00.htm. It seems draconian that such measures need to be taken, analogous to avoiding food poisoning by going on a hunger strike.

The problem with e-mail ineffectiveness can almost always be solved by applying simple time management rules to its usage and only visiting the in-box at sensibly placed intervals in each day, rather than operating in a reactive high-interrupt mode. While this does work to make one's day more effective, it does put the onus of responsibility back onto the user rather than the creator of the software. A disturbing trait I've encountered throughout my career as a software engineer is one in which programmers blame users for incorrect operation, with quips such as "they should have read the manual" or "user error, they were stupid enough to take the wrong option that deleted their data." Software is now so prevalent in our lives that it has become part of the fabric of society, and as engineers we should apply ourselves to thinking about the negative consequences of its usage and try to understand and prevent the extremes of its usage and abusage; whether intended or accidental. If nothing else, we owe it to the generation that follows us.

JAVA DEVELOPER'S JOURNAL Advertiser Index

Advertiser	URL	Phone	Page
AjaxWorld East Conference 2007	www.ajaxworldexpo.com	201-802-3022	29
Altova	www.altova.com	978-816-1600	Cover II
Business Objects	www.businessobjects.com/devxi/misunderstood		11
Extentech	http://extentech.com/itsg	415-759-5292	4
Infragistics	www.infragistics.com/jsf	800-231-8588	19
InterSystems	www.intersystems.com/ja2p		7
Java Developer's Journal	www.jdj.sys-con.com	888-303-5282	33
JavaOne	www.java.sun.com/javaone		23
Jinfonet Software	www.jinfonet.com/live	240-477-1000	Cover IV
Northwoods Software Corp.	www.nwoods.com	800-434-9820	25
SOA and EOS 2007 Conference & Expo	www.soaworld2007.com	201-802-3020	31
Software FX	www.softwarefx.com	561-999-8888	Cover III

General Conditions: The Publisher reserves the right to refuse any advertising not meeting the standards that are set to protect the high editorial quality of ***Java Developer's Journal***. All advertising is subject to approval by the Publisher. The Publisher assumes no liability for any costs or damages incurred if for any reason the Publisher fails to publish an advertisement. In no event shall the Publisher be liable for any costs or damages in excess of the cost of the advertisement as a result of a mistake in the advertisement or for any other reason. The Advertiser is fully responsible for all financial liability and terms of the contract executed by the agents or agencies who are acting on behalf of the Advertiser. Conditions set in this document (except the rates) are subject to change by the Publisher without notice. No conditions other than those set forth in this "General Conditions Document" shall be binding upon the Publisher. Advertisers (and their agencies) are fully responsible for the content of their advertisements printed in ***Java Developer's Journal***. Advertisements are to be printed at the discretion of the Publisher. This discretion includes the positioning of the advertisement, except for "preferred positions" described in the rate table. Cancellations and changes to advertisements must be made in writing before the closing date. "Publisher" in this "General Conditions Document" refers to SYS-CON Publications, Inc.

This index is provided as an additional service to our readers. The publisher does not assume any liability for errors or omissions.

JDJ
ARE PORTALS THE MAGIC BULLET OF WEB APPLICATION DEVELOPMENT?
IN THIS ISSUE...
PLUS...

JCP.org Web Site Gets a New Life

Onno Kluyt

It's well known in Java community neighborhoods that the jcp.org site has been the communication lifeline for the JCP program since its inception. It's been the virtual meeting place for members, the primary source of information and updates about Java standards, and the spot where key interactions for the development of JSRs have occurred.

This month the JCP program has a new site. The redesigned jcp.org provides for broader and deeper interaction through personalized content, user-friendly navigation, and other newly introduced capabilities.

Don't take my word for it though – go to the new jcp.org. I'm sure your experience with it will be richer and your time on the site more effectively spent.

Take, for instance, the ability to customize content. Users can now build their own My JSR Page with a list of specifications that you choose to watch. Or take the update icons that alert you to changes on the JSR page and make it easier to spot and click through them.

The log-in procedures have been streamlined too: each user has only one sign-on name and password. In the past, users were required to have several logins for each level of involvement.

You'll notice from the beginning that while the redesigned site includes almost the same content as before, the flow of menus is more logical, making it easier to navigate to the preferred information.

You'll also be pleased with the improved search engine, which now allows you to look up JSRs by number, name, or keyword descriptions. The membership sign-up procedures are simpler too; applicants can now input their data directly into an online form. Gone are the days of faxed-in forms and handwritten illegible account information.

As for enhanced participation, communication, and transparency, the new jcp.org takes a leap forward and inaugurates message boards. Be they for Spec Leads, Expert Group members, or general users – the message boards allow users to interact through e-mail or via a Web interface. I expect this will give a new impulse to the open exchange of ideas among participants.

There's something better for everyone on the new jcp.org. The site enables everyone to do more than was previously possible. For instance, if you're a general user, you can take advantage of the opportunity to personalize your view of the site, access message boards with the public's comments included, and do quick searches for JSR pages. In addition to these features, if you're a Spec Lead, you'll find it easier to edit JSR pages and update JSR community pages. Stepping into what was formerly PMO territory, Spec Leads will be able to make announcements and update the JSR proposal detail page with new information, such as an anticipated schedule of releases. If you are an Executive Committee member, you'll have your own message board and a single login process to enter the balloting area so you can vote on JSRs.

However (there's always a "however"), to take advantage of all the site has to offer, there is one prerequisite: go register! It's easy and free.

Expect additional improvements to be included in the second roll out of the JCP site, such as the ability to search for articles, success stories, case studies, tutorials, bios, and other content, so bookmark it. Until then I encourage you to e-mail your feedback to pmo@jcp.org after you spent some time on the new site and tried out its new features.

“Expect additional improvements to be included in the second roll out of the JCP site, such as the ability to search for articles, success stories, case studies, tutorials, bios, and other content, so bookmark it”

Onno Kluyt is the director of the JCP Program at Sun Microsystems and Chair of the JCP.

onno@jcp.org

Experience JReport Live™

Bring Your Reports to Life

JReport Live is the only solution to empower your Java application with operational business intelligence. Now your application can support both operational reporting and analytics – making users and developers happy.

Powerful Analytics Based on Dynamic Cube Technology

For the first time your users can enjoy ad hoc reporting right from the application, working with the most current operational data. Users can now slice-and-dice data, expand/collapse groups, pivot and drill up/down/across any report. The power of JReport Live is based on our Dynamic Cube Technology. JReport Dynamic Cubes are easy to define and are instantly created with no overhead. Information is always current and presented in the context of the application.

An Enterprise-class Foundation for Operational BI

JReport Live is built on our powerful operational reporting solution, delivering all of the benefits of scalability, performance and security of JReport 8. Plus, you can combine reports into report sets for better performance, easy scheduling and fast deployment.
The pipeline mode allows users to start viewing reports before generating a full report. JReport 8 can integrate with any Java EE application and any security scheme. In addition, JReport runs reports on demand, on a schedule or by event triggers.

Build Complex Precise Reports Quickly and Easily

Only JReport 8 is not constrained by a rigid report layout, it lets you mix and match report components and control precisely how they are presented on the page. With multimedia objects, Web controls and Web forms to further enhance reports, your users will be happy to work with the highly functional, interactive JReport Live reports.

Download your version of JReport 8 with JReport Live or call 240-477-1000 today.

Bring your Reports to Life with JReport Live Server
For more information, visit www.jinfonet.com/live

JReport®
JINFONET SOFTWARE