EMBARCADERO
TECHNOLOGIES®

Tech Notes

Managing Java EE Performance with Embarcadero's J Optimizer Request Analyzer

Al F. Mannarino, Embarcadero Technologies

June 2008

Corporate Headquarters
100 California Street, 12th Floor
San Francisco, California 94111

EMEA Headquarters
York House
18 York Road
Maidenhead, Berkshire
SL6 1SF, United Kingdom

Asia-Pacific Headquarters
L7. 313 La Trobe Street
Melbourne VIC 3000
Australia

The iterative use of performance tools throughout the development process is critical for keeping performance and reliability problems under control as well as for producing fast, scalable Java EE and Java SE applications. Java™ technology is great for accelerating time-to market, but performance and reliability risks can become a serious challenge. Because Java technology provides a higher level of abstraction, it affords developers only a limited understanding of—and control over—the way their code is executed. Embarcadero J Optimizer offers a four-pronged attack at improving Java application performance. Three of those four prongs address developer performance considerations and are focused on individual developer productivity, identifying performance risks and reliability issues, and helping developers correct problems such as memory leaks, performance bottlenecks, and multi-threading issues. Within J Optimizer there are three profiling tools which help address these areas.

- Profiler—examine memory and CPU use; identify memory leaks, inefficient temporary storage issues, CPU bottlenecks, and unit test performance regressions.
- Thread Debugger—identify and analyze thread contention issues, thread-starvation, excessive locking, deadlocks, and other thread related issues.
- Code Coverage—identify and analyze the classes, methods and lines of code that are being executed; remove dead code, improve quality, and improve the application's footprint.

All of these tools offer productivity improvements to the development team and risk reduction for management. These J Optimizer tools are extremely valuable for discovering and diagnosing computational performance, RAM footprint, startup time and perceived performance problems. However the information that is collected by these tools and presented to the user is at a level of abstraction that is most suitable for Java developers who are actively benchmarking or profiling an application with which they are intimately familiar.

The fourth prong addresses system testing performance considerations and focuses on the collection and presentation of diagnostic information that is suitable for other members of the Application Lifecycle Management (ALM) team such as QA staff.

J Optimizer's Request Analyzer is the proper tool for performance management and performance analysis of JEE protocols which can affect the scalability and performance of an entire system. Developers writing system code that involves using Java EE standardized services such as JMS and JNDI find it very critical and efficient to detect Java EE system-level performance issues while performing local system integration testing on their development workstation. This step provides them a snapshot of how these services and other key Java EE components such as Servlets and JSPs™ are performing. This gives developers a 360 degree view of how their JEE system is performing and gives confidence of building high performance JEE applications before they are formally tested by test engineers.

The full functional and load testing of a JEE application is performed by QA engineers or testing teams who also measure performance and detect any bottlenecks. While the load test is running, it is absolutely essential to see and analyze the performance behavior of the Java EE application. At this time the QA and the test teams require simplified performance views of system components like JSP, JDBC™, Servlets and JMS explaining how they are performing overall in terms of the response times, CPU time and relative system-wide bandwidth consumed in real-time. The performance data on the JEE components should then be compared to initial goals setup during the design phase of implementing the application.

Stringent application performance requirements mean that collection and analysis of performance behavior data must be accomplished with low impact on the overall execution of the JEE application under test. On the other hand, low overhead must be balanced with the need for real-time data collection in order to enable tactical performance monitoring while simultaneously providing comprehensive data needed or detailed root-cause analysis and strategic planning. The ever-present dichotomy of low overhead vs. comprehensive collection of detailed data is exacerbated on Java EE systems.

Eliminating Java EE Performance Bottlenecks Using J Optimizer Request Analyzer

Performance requirements for any JEE system can be best met by capturing performance metrics for the overall system and each JEE component using the Request Analyzer in J Optimizer. This paper uses the Java Blueprints—Java EE Pet Store application from Sun Microsystems—to illustrate how JEE performance can be benchmarked and refined.

Overall System Performance

The Java EE architecture is designed for developing multi-tiered distributed applications that provide interconnections between first tier presentation components (e.g. JSPs, Servlets, etc.) middle tier business logic components (e.g. EJBs, etc.), and backend enterprise information systems such as transactional DBMS's (JDBC™), asynchronous messaging systems (JMS) and directory services (JNDI). This corresponds to the information presented in the J Optimizer Request Analyzer J2EE System Dashboard (see Figure 1).

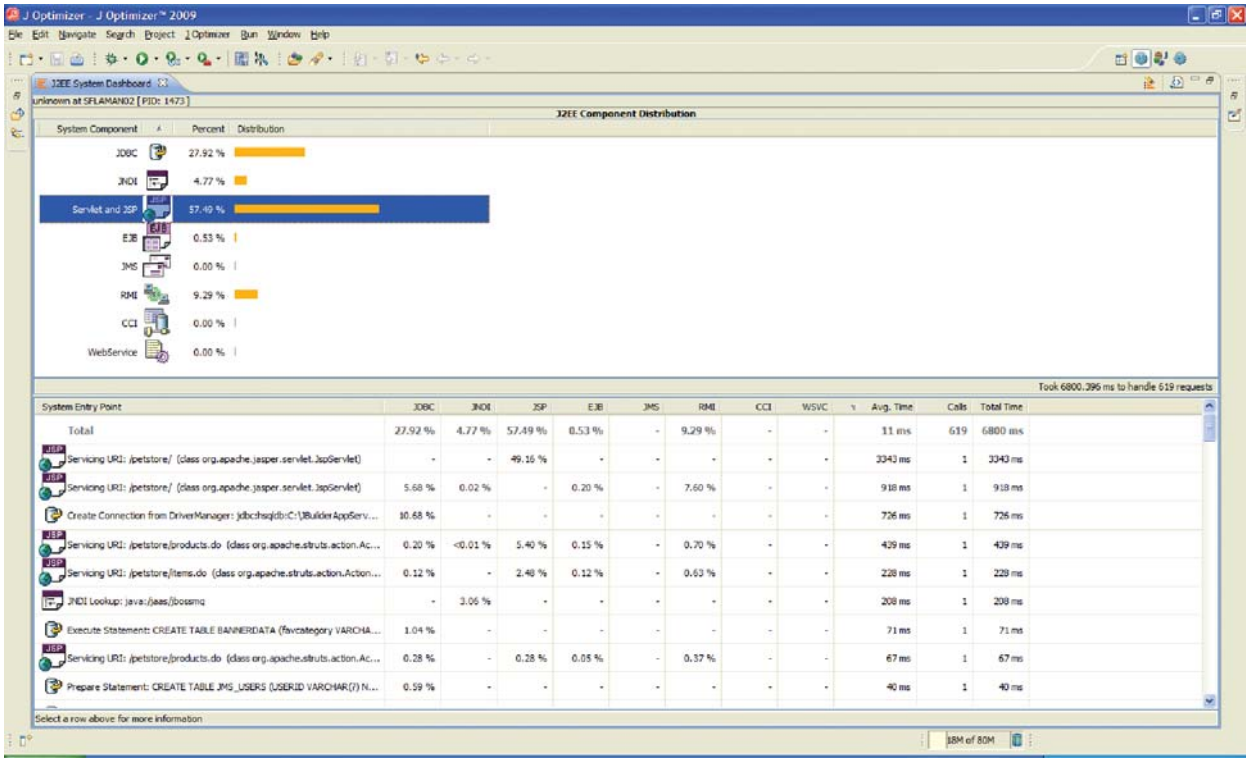

Figure 1 J Optimizer Request Analyzer J2EE System Dashboard

Performance management should start by benchmarking JEE system performance at the highest levels of abstraction. This corresponds to the information presented in the J Optimizer Request Analyzer J2EE System Dashboard. This J2EE Application Server Summary View presents an upper pane that displays the aggregate time spent by the application in code belonging to each of the following eight JEE APIs: JDBC, JNDI, JSP, Servlets, EJB, JMS, CCI and WebService. The lower pane presents all of the Java EE system entry points with the amount of time taken by all spawned events broken out by activity type. The nature of the J2EE system under test will dictate the next steps in your performance management efforts. For instance, a J2EE system that is designed to predominantly serve static content to its users should not indicate that most of its time is taken by JDBC calls. Here we see about 28% of the time is spent on JDBC calls. Double click the JDBC bar graph to activate the JDBC view and begin isolating any JDBC related performance problems.

Automatic Application Quality Analyzer—A unique capability to prevent J2EE deployment disasters

J Optimizer Request Analyzer provides development teams with this unique ability to generate predictive error reporting automatically as a bi-product of their performance analysis. The Automatic Application Quality Analyzer View provides insight into conditions that will limit the scalability and perceived performance of the profiled application. Where information technology departments once threw exponentially increasing amounts of hardware at a system in order to scale, there is now the option to run J Optimizer Request Analyzer. J Optimizer identifies classes of different J2EE API usage errors that will impair an application's scalability. Selecting an error from the Application Quality Analyzer pane allows you to get a thorough diagnosis of the predictive performance error condition including information required to identify the causes, number of occurrences and solution to the problem. The benefit of predictive performance error reporting goes directly to the total cost of ownership for the J2EE application. By running Request Analyzer repeatedly throughout the application development lifecycle, software development teams ensure that money spent on hardware capacity goes directly and only to delivering a fast, reliable user experience.

Conclusion

Java EE systems are large and complex requiring the participation of various software development professionals including business analysts, architects, designers, developers, testers, and administrators. In order to work together effectively, these professionals need to be able to share information effectively. The entire ALM process is geared towards improving the efficiency of these interactions.

Embarcadero J Optimizer, and specifically the Request Analyzer, radically streamlines performance management during testing in preparation for deploying applications for the JEE platform. Test teams can pinpoint performance bottlenecks at the component level more easily using advanced capabilities such as the J2EE System Dashboard and Automatic Application Quality Analyzer. Building a powerful bridge between test teams and developers, the Request Analyzer accelerates application-testing cycles by delivering constructive, unambiguous feedback. It provides specific diagnostic information as to the exact location and cause of performance issues that developers can act upon.

J Optimizer allows architects, designers, developers and especially test and QA professionals to collaborate more effectively throughout the application life cycle. By using J Optimizer Request Analyzer early and often, throughout the entire development process, a J2EE

development team is able to identify real and potential performance problems so that they are addressed well before deployment. J Optimizer Request Analyzer is designed to simplify the management of complex, mission-critical J2EE applications, resolving performance problems and avoiding deployment disasters with an easy-to-use, cost-effective approach. A key component of Embarcadero J Optimizer, the Request Analyzer allows development teams to easily ensure that deployed J2EE applications exhibit optimal performance and near perfect scalability, effectively minimizing the total cost of ownership of their applications.

About the Author

Al Mannarino is currently a Lead Systems Engineer and Evangelist for Embarcadero Technologies, Inc. Before joining Embarcadero, his last three years was spent at CodeGear from Borland and the prior five years was spent as a Lead Systems Engineer with Borland helping to sell ALM/SDO/Developer Tools solutions. Al has over 25 years of Software Development experience including OOAD and the responsibility for developing and deploying production applications. Prior to Borland, Al was an SE for Objectivity, Versant, Red Brick Systems, Information Builders, and an Electrical Engineer with Grumman Aerospace where he performed real application implementations on complex electrical-mechanical systems. Al has a BS in Electrical Engineering from Manhattan College.

Embarcadero Technologies, Inc. empowers application developers and database professionals with award-winning tools to design, build and run software applications in the environment they choose. With the acquisition of CodeGear in 2008, Embarcadero now serves more than three million professionals worldwide with tools that are both interoperable and integrated. From individual software vendors (ISVs) and developers to DBAs, database professionals and large enterprise teams, Embarcadero's tools are used in the most demanding vertical industries in 29 countries and by 90 of the Fortune 100. The company's flagship tools include: Change Manager™, CodeGear™ RAD Studio, DBArtisan®, Delphi®, ER/Studio®, JBuilder® and Rapid SQL®. Founded in 1993, Embarcadero is headquartered in San Francisco, with offices located around the globe. For more information, visit www.embarcadero.com.