Vorteile eines Produktunabhängigen Monitorings komplexer J2EE Anwendungen

Matthias Zieger

Borland GmbH
Robert-Bosch-Strasse 11
63225 Langen
matthias.zieger@borland.com

Abstract. Im Vortrag geht es darum, wie man prozessorientiert die Komplexität bestehender J2EE Anwendungen durch Monitoringwerkzeuge aufbrechen kann. Dabei werden sowohl die benötigten Prozesse als auch die verfügbaren Werkzeuge vorgestellt. Anhand eines sehr bekannten Beispieles aus der J2EE-Welt werden die einzelnen Schritte live gezeigt werden.

Introduction

The Java™ 2 Platform, Enterprise Edition (J2EE™) has become a de facto standard for supporting mission-critical business applications. According to Giga, between 65% and 75% of enterprises worldwide are now using J2EE technology for application development. (IT Trends 2002). At the enterprise level, J2EE technology offers a complete, stable, and secure Java™ platform that has the capability to reduce time-to-market and increase the value of enterprise applications. Selecting J2EE as a platform for development and deployment is fueled by important technology benefits inherent to the mature J2EE specification. J2EE standards have been engineered to evolve and adapt to changing requirements and evolutions in the industry, providing a flexible and stable framework that allows enterprises to deliver world class applications, moving into the future without abandoning the past.
In a fiercely competitive environment, performance and reliability of mission-critical J2EE applications are essential business requirements for the enterprise today. Business customers and end users increasingly expect 24x7 availability for enterprise-class applications without incurring the frustrations of system bottlenecks and sluggish response times. Organizations are demanding that their application investments meet performance, scalability, and reliability demands, while having flexibility built in to allow changes as the business evolves.
In tandem with the acceptance of J2EE technology as a mainstream platform for building enterprise-class applications, the need to deliver high-performance in J2EE solutions is more critical than ever.

Borland supports this critical business requirement for achieving high performance of J2EE applications in deployment with Optimizeit™ ServerTrace, a solution that delivers end-to-end visibility into the performance of enterprise-class Java™ systems,
This paper discusses the importance of J2EE performance and the challenges that organizations encounter in trying to meet business-driven performance thresholds. The paper outlines the requirements for implementing an effective strategy for J2EE performance management, looking at how Optimizeit ServerTrace fulfills these needs and verifies that performance standards are met, reducing the risk of under-performing applications in deployment.

The importance of J2EE™ performance

In light of today's compressed development cycles, multi-tiered architectures, and complex multi-layered technologies, many organizations are challenged to get stable enterprise applications out the door in a timely manner. For project teams already challenged with escalating development and deployment costs, unexpected J2EE performance issues serve only to exacerbate an already fraught situation, threatening to delay time to market and ultimately lead to missed business opportunities caused by downtime. A recent Zona Research study indicated that more than $25 billion is lost in potential business due to poor Web performance.
Meeting and exceeding customer expectations is too critical to compromise with inadequately performing applications. Yet the problem of achieving even minimum performance requirements is frequently compounded by the difficulty of accurately diagnosing and locating performance problems. Both at the technical and organizational level, problem resolution is slow often due to both the complexity of understanding J2EE system performance and the difficulty of effectively communicating the location and cause of performance issues between test and development teams. The Newport Group Inc recently published findings that the average time of resolving a performance problem is 25.8 hours.
Indeed, Kevin Gallagher, vice president of research and reporting at Newport Group Inc., goes on to add that "In a large company, it takes a lot of pull and tug to make the changes of uncovering and resolving problems,".
This inefficient feedback loop between teams is an important factor contributing to project delays and escalating development costs.
With a myriad of conflicting priorities and overwhelming project management demands to cope with, performance is often unintentionally treated as an afterthought during the late stages of the application lifecycle, resolved only when issues present themselves as an uncompromising obstacle that threaten the entire project. A common solution is to try and conceal them with additional servers or more powerful and expensive hardware. However this only serves to mask architectural, logical, and programming issues and in the long-term is not a cost-effective, sustainable solution. More often than not, it just delays application failure until a time when a higher traffic threshold is reached, while in the meantime increasing deployment and operational

costs. Achieving J2EE performance standards needs to be a design consideration that impacts every step of the application lifecycle from requirement analysis, through the design and build stages, and right through to testing and deployment.

The challenge of managing J2EE™ system performance

The strength of Java and the J2EE platform is the high level of abstraction, re-use of objects, and insulation from layers of processing and system interpretations. This type of encapsulation delivers significant benefits in that it shields developers from lower-level complexities, dramatically simplifying and expediting application development of missioncritical enterprise solutions. With portability,standard, modular and reusable components, J2EE makes it simpler and easier for an enterprise to maintain existing applications and systems and also expand these resources to support enterprises today.

However an important side effect is that multi-tiered systems increase the likelihood of performance bottlenecks. The complex interactions that occur within J2EE systems have a low tolerance for performance problems and can quickly result in application slow-downs that heavily impact response times.

Revisiting the need for comprehensive J2EE performance management, the challenge becomes clear: the black-box effect caused by the higher level of abstraction makes understanding application behavior extremely difficult, impeding the process of finding, isolating, and fixing systemwide performance problems before they manifest themselves in deployment as disasters.

Traditional performance tools focus on monitoring application problems, but treat J2EE systems as a black box, limiting the ability of project teams to obtain sufficient diagnostic information about the cause and location of system performance bottlenecks as they provide little or no insight into where a problem originates.

Understanding systemwide performance is a challenging problem, one that requires visibility into the performance of the J2EE systems and how it interacts and performs.

Optimizeit ServerTrace addresses this need providing comprehensive insight into J2EE systems, pinpointing performance bottlenecks at the component level, and delivering specific diagnostic information as to the exact location and cause of performance issues. Optimizeit ServerTrace simplifies the management of complex, mission-critical J2EE applications, resolving performance problems and avoiding deployment disasters with ease-of-use and an effective communication framework.

Managing J2EE™ applications throughout the application lifecycle

In order to effectively understand how Optimizeit ServerTrace streamlines J2EE system
performance testing. it is important to look at how this process currently works in most enterprise organizations.
Consider the testing phase of the application development lifecycle.

? Unit testing J2EE system
? integration
? testing
? J2EE final QA
? testing
? J2EE
? pre-production/staging

The testing process can be broken down into clearly identifiable stages, each with a clear function and responsibility.

Unit Testing.
By testing J2EE applications for performance during the Unit Test phase,
developers can locate and correct inappropriate design decisions early in the application lifecycle. Tools such as Borland Optimizeit™ Suite are frequently used at this stage to address performance issues at the code-level, looking inside the Virtual Machine to optimize performance of time and memory, analyzing method calls, and delivering very detailed, granular information about code performance. Performance tuning at this early unit level is an essential aspect of meeting performance requirements to increase code maintainability and extensibility addresses the need to catch memory and speed related weaknesses in the code
before the application is assembled.

System Integration Testing.
As the application components move in to System Integration,
testing unveils new performance vulnerabilities based around the complex interactions of J2EE components. Critical at this stage is the need to gain clear insight in to where vulnerabilities occur between the respective components and to provide actionable diagnostic information back over the wall to the developers for precise and accurate resolution.

Final QA Testing:
As the application moves in to final regression testing, QA engineers are looking to make “go /no-go” decisions regarding J2EE system performance and scalability.
This decision needs to be based on an objective, informed evaluation as to whether or not the application meets minimum performance requirements. As with system

integration testing, problems uncovered need to be passed back to the appropriate development team with relevant diagnostics for quick resolution.

Pre-production/Staging:
The last chance of pre-empting performance vulnerabilities is during pre-deployment staging which is typically comprised of scalability and performance engineers assigned the task of testing J2EE applications in a production-like environment prior to deployment. At this late stage, only fine-tuning of system performance should be required, as the application is stress-tested under load, using a load-testing tool.
Test engineers stand out as the ideal group to take responsibility for verfying that performance goals are met. One step removed from the inner workings of the application, test teams hold the unique position of being the first group to lay their hands on the newly assembled system. This lends itself as the primary opportunity to detect problems early before they impact deployment.
Delivering fast, scalable, high-performance applications clearly needs to be part of a best practice approach that is owned and upheld by diagnostic experts. Delaying the identification of problem areas until too late in the process often leads to the realization that expensive and risky late-stage application redesign is required. Taking a proactive approach pre-empts just this type of rushed endgame rework.
Yet each of these testing phases share the common need to flag system performance problems and communicate unambiguous, actionable diagnostic information back to the appropriate development group. However, it is often at this stage that the process breaks down. With scant information as to the cause of the problem, developers spend costly hours frantically reproducing the condition, and hunting for the root cause. Futile finger pointing is not uncommon as teams struggle to assign responsibility, and so the blame-game escalates.

The solution: Optimizeit™ ServerTrace

As we have seen in most testing environments, if an application is deemed to perform poorly or fail during pre-deployment testing, it will be thrown back over the wall to the development team. Unable to provide information about the nature and location of problems, test teams are frequently perceived to be a roadblock and find themselves at the center of fingerpointing.
Developers need to be armed with real knowledge that enables them to solve problems quickly and effectively, pinpointing the exact cause of performance. Assuring J2EE system performance requires a solution with the capability to provide a high-level systemwide view in to application behavior.
Optimizeit ServerTrace radically streamlines performancemanagement during testing in preparation for deploying applications for the J2EE platform. Building a powerful bridge between test teams and developers, Optimizeit ServerTrace 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.

Optimizeit ServerTrace 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. Many test engineers rely on load-testing tools for regression and acceptance testing. While load testing tools provide the effective means to flag system problems by highlighting bottlenecks when the system is stress tested, they give little or no insight into where the problem originates. However, coupled with Optimizeit ServerTrace, the benefits of testing an application under load become increasingly valuable as clear insight into system performance identifies where problems originate once they have been detected. The more visibility that can be achieved into system performance, the more value that system testing can add to achieving application business requirements.

Once a performance issue has been isolated, the next stage is enabling development teams to immediately begin work solving specific problems instead of manually trying to locate the vague problems previously identified by test engineers. Optimizeit ServerTrace not only identifies the problem but also isolates them beyond the component level, down to transactions and specific lines of code. The concise information helps to cross the boundaries between development, QA, and staging teams with an effective, iterative process for solving performance problems.

In closing the loop back to the developer, an effective performance management solution must integrate with the organization's existing development tools, quality assurance framework, and IT infrastructure. The need to integrate with IDEs and profilers is vital.

Optimizeit SeverTrace delivers this capability through tight integration with Borland JBuilder,® the leading Java development environment, and Optimizeit Suite, a detailed Vmlevel performance solution designed to tackle performance problems throughout development.

J2EE™ performance best practices

The purpose of a best practice is to improve the application development process both from the standpoint of efficiency and quality in order to deliver a mature application that meets business requirements. From performance management standpoint the objectives are clear-cut:

- Reduce the risk and cost of developing, deploying, and operating J2EE systems.
- Get applications to market faster by tightening testing cycles.
- Maximize ROI in hardware and software investments.
- Consistently meet or exceed SLA performance targets by delivering high performance J2EE systems.

Interrupting service to re-deploy a production application will commonly happen if an application fails or is performing poorly. This results in missed business opportunities and costly firefighting cycles. The ability to identify when performance is sub-prior to deployment is crucial to avoid such production crises.

Capturing performance criteria early in the business requirements establishes a baseline to achieve minimum performance standards for J2EE applications which influences design, build, and deployment decisions and sets minimum standards for acceptance testing. At this stage, implementing a framework for end-to-end performance testing will verify that performance thresholds are met.
To achieve a best practice approach to performance management for J2EE systems, both testers and developers need to have the capability to scan the entire system and pinpoint exactly where performance vulnerabilities occur. By allowing test engineers to talk to developers within a common framework, both development and test cycles can be accelerated and high-performance J2EE applications consistently delivered on time and within budget.

In short, there are 3 simple steps that businesses can apply to meet performance requirements:

1. Invest in performance management software.
2. Implement performance management throughout the application lifecycle.
3. Tighten up application lifecycle processes.

Summary

By finding J2EE system performance problems earlier in the application lifecycle, the business benefits are considerable:

Reduce the risk and cost of developing, deploying, and operating J2EE systems

Optimizeit ServerTrace shortens development and test cycles by finding performance problems quickly and early in the application lifecycle.

Increase value of testing groups and the information they deliver

Using Optimizeit ServerTrace, test teams are able not only to identify problems, but isolate them beyond the component level, down to transactions and specific lines of code. This concise information allows development teams to immediately solve problems and to communicate with test engineers using a common framework.

Maximize ROI in hardware and software investments.

Investments in hardware and deployment software are expensive and IT executives have a responsibility to maximize the return on investment. Optimizeit ServerTrace ensures that the highest performance is extracted from these investments as well as leveraging investment in other tools, such as load testing tools for concurrent load and performance testing.

Meet performance expectations.

Most J2EE systems are subject to Service Level Agreement, either internally or externally. Optimizeit ServerTrace provides a framework for comprehensive, end-to-end performance management solution, consistently delivering SLA targets.