How to Achieve Best-in-Class Performance Monitoring for Distributed Java Applications

July / 2012
Introduction

Critical Java business applications have been deployed for some time. However, enterprise IT support staff members still face difficulties in ensuring application health and performance.

The scale and complexity of deployments have grown, and customer-facing apps are experiencing more traffic than ever before. For example, for the first time last year, the banking industry reported that online banking transactions exceeded traditional in-branch transactions.

This growth in usage was accompanied by technology trends such as Service Oriented Architectures (SOA) and cloud-based models for storage, infrastructure, and application hosting – making applications more dynamic and agile than ever before.

Monitoring tools have become very good at collecting performance metrics for these new architectures. These tools have become so good at generating performance data that, according to Gartner, IT staff have to try to analyze 300% more metrics than just four years ago. There can be hundreds, or even thousands of metrics for just one application running on a single server. And enterprise scale applications can consist of hundreds or thousands of servers. It’s not surprising then that many of today’s application performance management (APM) tools have failed to evolve their analytics enough to help users address this “Big Data” problem in APM.

What is required is a solution that will help users model and understand the volume, velocity, variety, and complexity of the new dynamic application ecosystem and, as much as possible, automate the process of digesting and understanding all this performance data.

What You Will Learn

In this white paper, we will look at two developments that will help IT and application support staffs better understand and manage the performance of distributed Java applications. Specifically, you will learn the following:

- How application performance monitoring has changed, and how a new APM paradigm is best suited to help support staff understand the performance of distributed Java applications.

- How analytics technologies have evolved to automate performance management and analysis of applications and their supporting infrastructures.

Original Approaches to Application Performance Monitoring

Original approaches to APM focused almost entirely on metrics related to resource consumption at the application server (app server) level. These metrics include CPU and memory utilization, counts of active threads, JDBC connections, and HTTP connections. While this perspective is still useful today, it is an infrastructure-centric approach that provides little insight into how applications are performing. It also does not account for the fact that today’s application servers are more powerful and capable of running multiple applications concurrently.

As a result, performance monitoring tools evolved to become more application-centric, providing deep insight into a complex stack starting with a physical or virtual server hosting applications running inside application servers (WebLogic, WebSphere, JBoss, Tomcat, .NET, etc.) Many APM tools can provide detailed performance metrics or key performance indicators (KPIs) for the application server and the applications they host. These metrics can be correlated with operation system (OS) metrics for the underlying server, Java Virtual Machine (JVM), and application metrics to provide a basic understanding of the Java application stack that is a core “unit” in APM.
With each of these “layers” generating hundreds of metrics, and with the number of applications running in today’s distributed SOA environments, it is easy to appreciate the daunting task of the APM user who is trying to manage performance by processing and understanding all the interrelated behavior of components in this stack.

APM Evolves to Handle Transactions in Distributed Environments

In some ways true application performance management didn’t begin until APM tools began to collect metrics related to the performance of individual components. But APM tools truly show their value in distributed environments that process transactions through the interaction of multiple Java components.

Today’s sophisticated APM tools are capable of telling us how each software object in a Java application performs in its environment. We can see metrics that tell us how long it takes for a servlet or Java Server Page (JSP) to process a request from a browser. We can also see how long it takes for an Enterprise Java Bean (EJB) to handle inbound requests from the servlet or JSP, and response times when EJBs make outbound calls to backend systems like databases, web services, and other backend systems.

In today’s dynamic and distributed Java applications, correlating incoming requests to and outbound requests from software components is more critical than ever to understand whether or not an application performance problem lies in its intrinsic business logic, or if the issue is further downstream in the transaction.

Another mistake to avoid when trying to diagnose performance problems is focusing only on individual transaction traces. This approach may be useful in analyzing why or when a transaction failed, but to understand why an application is “slow,” application support teams typically need to analyze performance metrics averages and statistics that characterize the holistic behavior of the application. These metrics typically fall into three categories (illustrated below):
1. Metrics that help characterize transaction throughput and peak performance, such as the average number of transactions or the peak number of concurrent transactions executed over a period of time.

2. Metrics that tell you transaction response times—either end-to-end or decomposed by the steps of the transaction as it traverses the infrastructure.

3. Metrics that tell you the number of errors or the error rates in specific parts of the transactions.

This model for application performance certainly has business relevance in that it tracks metrics that, once aggregated, matter from a user perspective. The model also provides resource metrics that are useful in troubleshooting problems due to constraints such as insufficient memory or threads—which is symptomatic of a sporadic problem that can cause an application to be "slow."

Yet despite the detailed metrics, this model does not represent the distributed nature of modern Java applications. A user has no clear visibility into the number of servers that an application will be deployed on, for redundancy and performance purposes. Today’s best-in-class performance models for distributed Java applications also have to take into account how applications and the underlying IT infrastructure perform in distributed clusters.

APM for Distributed Applications Using Application and App Server Clusters

Many large scale applications today are deployed in Java application clusters. To truly understand distributed application performance, one must have visibility into how an application operates across the multiple servers in a cluster.

Application clusters are interesting to support staff from two differing perspectives.

- Application throughput and response time for the entire cluster
- The transaction workload balance across application servers in the cluster

Application support administrators care about the user experience so they focus primarily on application throughput and response time.
A Web Application Cluster perspective can provide visibility into total application throughput. In this view, users are not interested in the response times of transactions through a specific server, or even what resources the server is consuming. Instead, a user wants to see calculated metrics that indicate how the cluster is performing as a whole – like the average response time or total number of transactions for all transactions processed through the cluster in a given amount of time.

The other clustering perspective of interest is the App Server Cluster, which is the infrastructure supporting the application. IT operations staff are generally more interested in this view, as their concern is how the application workload is balanced among multiple application servers (or app servers).

The person managing the cluster wants to ensure that the workload is distributed among cluster nodes by a weighted average (even though the weighted average may mean uneven distribution, since the same resources may not be available to all elements of the cluster). If there is a workload imbalance, the user will either want to investigate the applications running on an app server or drill down into the server that runs the app server.
How Behavior Learning Helps to Understand Application Performance

We've discussed how performance models for applications can have infrastructure, user experience or transactional oriented views, as well as “cluster” views. Taken as a whole, these models provide IT operations and application support staff with a very complete picture of performance for distributed Java applications.

Collecting and understanding performance metrics in these models allows users to better detect and quickly address potential performance problems in the application or underlying infrastructure in order prevent or mitigate performance degradations or outages.

This is no easy task. APM can consist of monitoring hundreds of individual and aggregate metrics per software or hardware components, for thousands components. For one application alone this can mean that millions of performance metrics have to be analyzed in near-real time.

So how do you digest all application performance information? You will need answers to the following questions:

- What’s “normal” behavior for the application at any given time?
- How can I automate threshold management for the application?
- How is the performance of the application correlated to the underlying infrastructure?

To address these challenges, a recent Gartner webcast highlighted the role of analytics powered by “Behavior Learning Technology” as a necessary part of your APM strategy. Netuitive is the leader in this space with solutions deployed to monitor critical applications in some of the world’s largest banks and mobile communications companies.

Netuitive’s solution uses advanced mathematics to provided automated insight into the correlated “normal” behavior of atomic and composite elements in the Java application ecosystem. Netuitive’s cross-domain analytics also correlate application behavior with that of the underlying IT infrastructure or business KPIs. This approach is crucial to deliver proactive performance management, where application and IT support staff can isolate and address developing performance issues before they impact business performance.

Summary in Bullets

- Java application architectures today are larger and more complex than ever for enterprise class applications.
- Basic APM tools provide visibility into the Java application stack (server, application server, application).
  - Key application metrics include metrics focusing on average response time, responses per interval, and error rates.
  - Key application server metrics focus on resource utilization (CPU, memory, threads, JDBC connections, and HTTP session)
- To get real visibility into distributed Java applications, you need to have a model that tracks an application as it is clustered across multiple servers for redundancy and/or performance. This model tells you how the distributed application is performing as a whole.
- Another essential model is clustering the application server infrastructure that hosts the application, to ensure that resources are properly utilized across the cluster.
- One challenge is the amount of data generated, and the need for analytics tools to help IT operations, application support, and line of business management understand how their applications are performing.
- Behavior Learning Engines are well suited as a technology to model and solve this problem –as recommended by leading analyst firms such as Gartner.

Learn More

- Watch a replay of a Gartner webcast on behavior learning technologies and analytics for APM
- See a demo of Netuitive in action
About Netuitive


Netuitive provides predictive analytics software for IT. Netuitive replaces human guesswork with automated mathematics and analysis to help companies visualize, isolate, and proactively address application performance issues before they impact quality of service. Hundreds of customers, including eight of the 10 largest banks, rely on Netuitive to proactively manage the performance of their critical applications and underlying IT infrastructures - physical, virtual and cloud. Industry recognition includes the 2011 “CTO Award for Innovation” from Morgan Stanley, the 2011 CODiE Award for “Best Systems Management Solution,” the 2010 EMA Award for “Best Analytics,” and “Best of VMworld” Awards in 2007, 2009 and 2011. For more information, visit www.netuitive.com.

Contact Netuitive to learn more about our solutions for proactive IT performance management, or to schedule an on-site demonstration.

© Netuitive, Inc., 2012. No material in this document can be reproduced without the expressed written consent of Netuitive, Inc.