5 Ways to Improve QA Through Transaction Tracing

Improving Application QA and Delivery With End-to-End Transaction Tracing

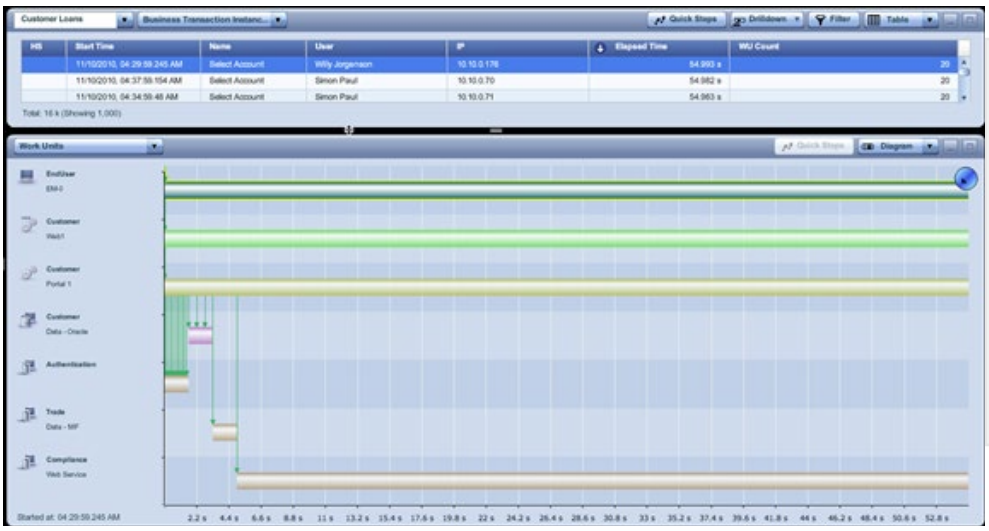
The business case for getting things done right the first time is simple: time and money. A task executed properly on the first go-round doesn't require future time or money to rectify. As a matter of fact, the cost of fixing a defect in production is more than 700% higher than during testing. And that figure doesn't even include the negative impact on the business. Yet, the number of defects escaping QA continues to be a substantial issue.

Today's complex and elaborate technology environments certainly contribute to this development. It's undeniable that demands from the business, agile development, and multi-tier application environments put added pressure on QA teams. Moreover, when transactions do fail in functional, integration or performance testing, QA often lack granular visibility into the root cause. Not only does this make it difficult for developers to re-create and resolve issues, but it also limits the number of issues that can be fixed during each test cycle.

QA needs greater visibility into the full spectrum of transactions and interdependencies between tiers and applications in order to more effectively identify and resolve application issues. The next generation of transaction management solutions provides just that: end-to-end transaction tracing that can tag-and-follow every request from the user -- across the application -- to any backend tier. This amplified visibility can help QA teams quickly identify and isolate problems to improve application performance during testing cycles.

Enhance End-to-End Transaction Visibility

The growing complexity of modern applications makes it difficult to efficiently test and determine issues and bottlenecks. The interdependencies between applications, portals, web services, ESBs, and other backend services create significant challenges to pinpointing precise application glitches. Further compounding the matter is the fact that QA teams sometimes spend days, weeks, or even months correlating log files and backend metrics to determine a root cause. Transaction tracing alleviates this problem by providing granular visibility into where and how your transactions flow, specifically which tiers they execute on and exactly how much time is being spent on each transaction, in each tier.



An example transaction tracing instance for a user request that includes the username, type of transaction, all of the tiers traced for this request, and the time spent in each tier. We can see that the Compliance Web Service is consuming 90% of the time for this transaction.

OpTier

Understanding Application Architecture and Interdependencies

Developers often write code based on business and functional requirements. They may, however, not have complete visibility into how their code impacts resource utilization or other components of the application in question. Conversely, infrastructure managers, like DBAs, understand their systems and performance, but don't have visibility into the type of requests being made and the upstream user and transactional context. Comprehensive transaction tracing combines both the stakeholder perspective and the business request into the context of the user.

One problem that end-to-end transaction tracing can immediately shed light into is "chatty transactions," or transactions that open large number of connections to other systems. These transactions often lead to unnecessarily high resource consumption and can also have significant impact on overall application performance. In one case, a banking application had a "chatty" transaction that was being run several hundred times a second. Identifying and rewriting this single transaction decreased system utilization by 40% and saved hundreds of thousands of dollars in additional infrastructure.

Granular, Correlated Diagnostics

Comprehensive transaction tracing can also help improve QA by providing correlated details about the backend tiers being executed during a load test. Performance testing and certification are often conducted with tools to test the scalability of the application from a synthetic user perspective, but transaction tracing can significantly complement this user dimension with specific details including tier KPIs, memory leaks, code, and method level performance, SQL, and web services. This data can quickly pinpoint bottlenecks and be used as a feedback loop to developers and infrastructure managers to continuously improve application performance and scalability.

Comparative Advantages

As new versions of code are tested and released, it is important to evaluate changes in transaction response time and system utilization in order to validate incremental application improvements. Transaction tracing measures response time and resource consumption that can be used for comparisons between releases. During a load test, for example, we can measure the volume and CPU utilization of every transaction that was executed in an effort to understand if improvements in response time, availability and resource utilization were actually realized. An example of application comparison can be seen below.

We can see that the Select Account transaction CPU has dropped from 398 µs to 77.99 µs, an improvement of 80% utilization, and response time has dropped from 13s to 2.4s, an improvement of 81%.

OpTier

Improve Communication Across the Enterprise Lifecycle

Transaction tracing can actually provide non-technical benefits as well. A recent study from EMA in 2013 indicated that a significant challenge in supporting cross-team communications (DevOps) is the lack of integration between tools used in Development and Operations. Fortunately, effective transaction tracing can address this challenge by providing granular data in both production and testing environments at very low overhead (<2% CPU). Traditional tools have high overhead and are not suited for production environments, while production monitoring tools often focus on infrastructure and don't provide details at the code level. End-to-end transaction tracing overcomes both of these limitations and can help:

- Provide details about a specific user and transaction instance which can be relayed from production to development teams to investigate a specific issue
- Assess the real mix of production transactions and volumes so they can be tracked to more accurately model testing scenarios
- Developers understand how their code functions and scales in the production environment
- Infrastructure managers see how the performance of systems affects applications and business transactions
- Developers, Operations, infrastructure managers, and others, identify bottlenecks in the infrastructure (down to the SQL and method level) to understand where improvements can be targeted for tangible business improvement.

There is little debate that end-to-end transaction tracing can provide a myriad of benefits for QA teams looking to be more productive, effective, and impactful. Knowledge is power in the Information Age and arming QA managers with deep infrastructure visibility and contextual application information can help ensure new code releases provide tangible value. More importantly, transaction tracing can empower QA to be more strategic and help provide invaluable enterprise alignment.

OpTier, headquartered in New York, is the leader in Application Performance Management, IT Operations Analytics and Big Data Analytics, analyzing billions of business transactions daily. With transactions in its DNA, OpTier offers an analytics-driven enterprise class APM solution, an industry-leading Big Data Analytics platform, and empowers businesses to uncover the truth behind their transactional data. This actionable intelligence provides global 1000 enterprises – such as Aflac, Deutsche Bank, HSBC, Morgan Stanley, O2 and Wells Fargo –with insights that drive business growth, operational productivity and customer satisfaction. To learn more about OpTier, visit www.optier.com.

OpTier