Load Testing

The Key to Keeping Business Applications Running
Executive Summary

Every business depends on applications to automate its core business processes. These applications must be constantly available for partners, customers, employees, and other stakeholders. But the reality is often very different. When applications are under peak demand, they can often be inaccessible or perform unacceptably. This issue can have dramatic effects: orders may be lost, customers abandoned, and reputations damaged.

The challenge of dealing with peak loads has increased in recent years. Today’s applications are growing in complexity. The example of rich Internet applications (RIAs) illustrates that end users demand functionality that can be very resource intensive. This trend, combined with the rising volume of usage, means that demands placed on applications are greater than ever, and will continue to grow.

This white paper examines the importance of load testing to address these challenges: when in the software development process to load test, and how proper load testing leads to optimal performance. It also looks at the various strategies for implementation and investigates the real benefits load testing returns to organizations. Finally, it identifies the proper strategy for realizing the full benefits of load testing, including how to diagnose issues in your applications and how to leverage the power of the cloud.

How Load Testing Helps Your Business

Why load test?

BUSINESS LOSS

For retailers, a website outage can be catastrophic for business. Recently, a major online ticketing agency lost access to its web-based storefront, and the resulting loss of business was immense. But the full cost of a failure goes beyond missed revenue opportunities. To fully recover from a failure, businesses must “clean up” after an outage, a process that can cost many times the lost revenue. Besides the obvious hard costs, there are also softer costs such as brand image, customer satisfaction, competitive gain, and other factors.

If an internal application, such as an ERP system, goes down, employees lose productivity immediately. If the employees affected are dependent on the application to perform their jobs, the costs of downtime quickly grow.

In addition to lost productivity, it’s important to consider other opportunity costs to the business. For example, if high loads cause an order processing system to go down at the end of a quarter, orders might not be processed and revenue may not be recognized.

External customers are more likely to abandon your business process if dissatisfied with the performance or availability of your applications. Internal end users who aren’t given a choice of services may simply refuse to use an application rather than tolerate sub-standard performance. It follows that optimizing software performance is vital to satisfy customer usability needs—whether internal or external customers—and keep up with (or surpass) competitors.

INCIDENT-SPECIFIC COSTS

If a business fails to meet service level agreements or is otherwise obliged to provide a certain level of service to its customers, there can be financial or legal costs. If the failure becomes public knowledge, PR and marketing budgets may have to be realigned to defend against negative press or attacks by competitors.

LOWERING COSTS FOR THE APPLICATION INFRASTRUCTURE

Unnecessary investments can be avoided with adequate load testing through application tuning and capacity planning. As an example, one well-known provider of accounting software was able to triple application performance utilizing its existing infrastructure—and thus significantly save expenses it would have incurred to purchase additional hardware—as a direct result of load testing activities.

Best Practices for Load Testing

Load testing, implemented properly and performed throughout the development cycle, will significantly reduce the number of application failures. The number of failures avoided will vary according to application
complexity, developer skill, and application usage, but if a tool prevents even a handful of failures, it can pay for itself several times over. This is especially true if the application is mission-critical or customer facing and used to generate revenue or interact with an external audience.

**Approaches to Load Testing**

Load testing helps pinpoint and diagnose problems that affect an application’s performance, scalability, and reliability. There are many kinds of load testing, each of which addresses a specific problem area:

- **Performance testing** establishes the speed of a system under a given load. To do so, you load the maximum number of anticipated users and measure the response times of specific pages or critical business transactions. This approach helps you determine whether you can meet SLAs and deliver business value.

- **Stress testing** determines the maximum volume a system can handle. A high enough load is generated to stress the application to its limit, and beyond. The stress test continues until the application or the supporting infrastructure breaks down. This helps you plan for potential overages, for instance, if a major new marketing initiative brings in a large surge of clients.

- **Scalability testing** ensures that the current system scales to the intended load. Results of scalability testing can be used to plan for future growth and avoid over-investment in hardware.

- **Stability testing** determines whether an application will remain serviceable over an extended time span. It determines, for example, whether the resource consumption of the application will remain stable or whether there are other technical faults that may compromise the stability of the application. A common example for a stability issue would be a slowly increasing memory leak.
Volume testing validates whether a system can handle large amounts of data, calculations, and processing. This type of testing may involve pushing large volumes of data through the system with high processing demands, testing with huge database sizes, large data or log files, and requesting many processes simultaneously. This ensures that the system can process business needs in the real world.

When investigating solutions for load testing, it is important to ensure that you have the flexibility to apply each of these models. Otherwise, potentially threatening gaps can be left in the stability of your applications.

When to Load Test?
Software defects are up to 100 times more expensive to correct at the end of a software development lifecycle than at the beginning. So the earlier that load testing can be conducted, the sooner that defects or architectural issues can be found and costs minimized. Today’s typical multi-tier applications have separate tiers for presentation, business, and data logic, plus potential legacy integrations. So load testing can be applied at different phases in the lifecycle to maximum benefit.

COMPONENT-LEVEL TESTS
Unit testing is a widely accepted way to verify distributed software components’ functionality early in the development cycle. However, multiple clients often simultaneously access components that reside on the server (e.g., EJBs) that host business logic or access legacy applications. Here classic unit testing is insufficient. Only a stress test can cost effectively identify typical issues like deadlocks; memory leaks; or performance, synchronization, or architectural issues.

INFRASTRUCTURE LOAD TESTS (BENCHMARKING)
The infrastructure that an application is built on directly influences the performance, scalability, reliability, and cost of deployment. So all available infrastructure options should be carefully evaluated, weighing both performance and cost. Official benchmark numbers are often unhelpful as they generally measure single components and don’t account for architectural variance. Early load tests and their results can fill this gap by providing insight into which configuration to use. They can also influence application architecture decisions (e.g., .NET or J2EE). Understanding the effects of the various system configurations also provides insight to aid tuning and to benchmark future tests.

ARCHITECTURAL LOAD TESTS (BENCHMARKING)
Early load tests on an application’s architecture can verify that components in various tiers work together as expected. In an “all-tier” prototype that includes a small subset of the complete functionality, early tests can quickly detect design flaws. Even design alternatives, like the distribution and replications of logic, can be evaluated.
END-TO-END LOAD TESTS
End-to-end load tests analyze the entire application under various realistic end user workload scenarios that may last several hours or even several days. These tests answer questions like:

- Are there functional errors that only occur under load conditions?
- What system capacity is needed across all tiers?
- Will the application meet defined service levels?
- Is the application tuned for optimal performance?
- Has the introduction of new functionality impacted performance?
- Is the application ready for full deployment?

Addressing the Challenges of Load Testing
The benefits of load testing are readily apparent. Some common issues may arise when you attempt to implement load testing, here’s how you can address them.

WEB 2.0
Today’s web applications are more interactive, responsive, and content-rich than ever. These new applications behave more like modern desktop applications without the deployment burdens. The gains in user experience come with significant testing challenges, especially related to performance and load.

Different rich Internet application (RIA) technologies like Adobe Flash and Flex, Microsoft Silverlight, and AJAX execute much of the functionality on the client side. The rise of new AJAX frameworks, most of them with differing technical approaches, increases testing complexity still further. As a result, load testing approaches must address these needs—and one size doesn’t fit all. Testing tools that provide multiple approaches, each tailored to the Web 2.0 technology used by the application are now needed.

TESTING EARLY AND OFTEN
As previously mentioned, leaving load testing until late in the development lifecycle can lead to extreme risks. A methodology that establishes quality and performance-related activities early in the application lifecycle helps to mitigate the risk of project failure, reduces overall project costs, and increases the application’s quality and performance.

Component load testing can be executed against business logic components as soon as they are ready, even without fully developed UI or other software components. With SOA-component load testing, early load testing becomes even more critical. The earlier load tests for components of a system are developed, the sooner it is possible to find regressions of performance when these components change.

Each system change can introduce functional, performance, scalability, and stability regressions. Focusing only on functional tests to address these regressions leaves performance issues undetected until the final system load tests. Integrating load tests as part of a regression test suite avoids the danger of detecting technical problems only when they become business problems.

Sigma-Aldrich is a leading life science and high technology company serving customers in more than 150 countries from its operations in 36 countries and has 7,600 employees providing excellent service worldwide. Its website generates 39% of its revenue and must bear 100,000 visits per day. Clearly, this website must perform under intense pressure.

Sigma-Aldrich uses performance testing to uncover bottlenecks and performance issues early in the development process. Because its product catalog changes frequently and users’ behavior often varies, tests need to be highly adaptable. So ease of scripting and reuse of scripts is a critical need for this company.

MANAGING THE TEST PROCESS
This best practice necessitates another. Users should mandate a test management solution that supports manual and functional testing and integrates with a performance testing tool. Easy management of test plans, scheduled test executions, and automatic results evaluations are importance for “continuous” performance testing.

Trend reports of performance-related metrics like response times provide a quick comparison of test execution over time or builds. For detailed comparison of a particular load-test run with an “ideal” baseline test, a cross-load test report can help identify which metrics have improved due to tuning activities or other changes to the system.

INFRASTRUCTURE COMPLEXITY MAKES PROBLEM DIAGNOSIS A CHALLENGE
Today’s business critical applications continue to become more complex as new technologies are added on top of existing systems. New application development trends like SOA and EAI address this need, leading to highly distributed and heterogeneous multi-tier architectures. These models are complex to manage and difficult to test and tune.
Further, large numbers of frameworks like Java EE, .NET, Hibernate, and Struts allow faster development. As business functionality can only be supported by the interoperation of many systems, messaging services, remote protocols, and web services begin to play a central role as well. While these technologies may decrease the initial cost of development, the resulting applications become significantly more complex to test and tune. And when these applications are hosted in virtual environments or in the cloud, an added layer of complexity emerges.

Cloud-based tooling on its own cannot analyze the internal behavior of the application under test. This can mean that applications may be incompletely repaired following tests, increasing the risk of a real-world failure. It is more effective to combine the power of cloud-based testing with “on the ground” diagnostics of an application’s performance.

This complexity can lead to applications not performing as expected. As a result, efficient performance diagnosis and tuning become essential to profit from the lowered development costs. The key question becomes how much time and effort is needed to reach the performance goals?

For today’s complex architectures there must be close collaboration between developers, DBAs, architects, system engineers, and QA. An integrated performance testing and diagnostics solution can bridge this gap. It improves communication between development and QA and allows them to quickly resolve issues without dispute after a single load test run.

This is because an integrated suite enables engineers to readily document the root cause of issues down to the offending line of code—even in highly distributed applications.

Müller, a major German pharmacy, experienced this challenge. Its ERP and CRM systems are in constant demand from customers and employees and are a critical front-line for servicing customers. They simply must withstand peak loads, especially as there is a seasonal nature to Müller’s business.

Testers at Müller simulate thousands of simultaneous users working with multiple computing environments and interacting with various distributed application environments such as Web, SOA, client/server, Citrix, legacy, or ERP/CRM systems—on one or more test machines.

After every test run, managers receive a comprehensive overview of the application’s performance during peak times. This enables management to determine the risk of releasing an application and to dedicate resources to correcting issues early in the process.

REAL-WORLD TESTING

End-to-end load tests must ensure that applications are “battle ready” and can withstand massive, global usage. This requires simulating large loads, but often the infrastructure in place is insufficient. End-to-end load tests are frequently performed in staging environments, which simulate the production system at a reduced scale. Also, the hardware resources necessary to run the virtual users for a peak load test require significant investments and may not be available right on time.

This may lead to “under testing” of applications by testing with lower volumes of peak loads, in turn introducing the risk that applications will fail in real-world conditions. Alternatively, it may lead to delaying testing until the necessary computing resources can be assembled, stalling the delivery of business requirements and impairing an organization’s agility.

Cloud-based infrastructures are extremely well-suited to the kinds of peak demands required by performance testing. The sheer size of cloud data centers ensures that sufficient computing power is available as you scale from 50,000 to 100,000 to 200,000 virtual users, and beyond. Of course not all cloud-based load testing providers can scale to this degree, and not all testing solutions can effectively harness these kinds of resources—care should be taken when selecting vendors.

Peak load testing via the cloud also takes advantage of the ability to run tests virtually on-demand—simply schedule time for a test and resources are automatically provisioned. This avoids testing bottlenecks and prevents long delays as internally managed hardware is acquired and set-up.

Thomson Sweet & Maxwell is a leading provider of information and solutions to the legal and regulatory markets in the UK and Ireland. The quality of the user experience delivered through the company’s services is a business-critical priority directly related to customer satisfaction and retention.

The IT organization must also comply with numerous service-level agreements related to the online user experience. One top-line SLA requires that all searches on the company’s flagship product be successfully performed in less than eight seconds. Additionally, pages that do not function as searches must load within three seconds.

The company created real-world test scenarios to mirror load, performance, and stress testing for over 10,000 concurrent users. As new applications are rolled out, the company creates realistic test scenarios
that mimic demanding access circumstances that result from spikes in use, and then address potential issues prior to deployment. For instance, the team uses performance test tooling to pinpoint multi-threaded access problems that might occur with users concurrently logging into its online systems with similar user names and similar scenarios.

As a result, the company saves up to $10,000 per project through the automation of its performance test efforts. Last year, the team completed performance testing on over 15 projects—resulting in a significant cost saving, through more efficient and faster application performance testing. Further, they are able to reduce project risk by finding performance bottlenecks before they happen. This eliminates costly and frustrating rework and reduces the impact of performance defects on customers.

**GLOBAL USERS**

Load tests should include a “true” end-user perspective by testing outside of the lab in different geographies. A global company needs to ensure that its clients worldwide can access an application, but testing applications with Virtual Users from only a single geography fails to provide a complete picture of performance. The appropriate levels of infrastructure in the right locations are required to truly replicate production-like states and ensure that applications perform.

Cloud-based infrastructure and testing solution providers may have centers in Europe, Asia, and North America to provide this ability. For any global business, this need is imperative. This global approach to load testing lets testers place Virtual Users in a variety of locations to test international performance. It is no longer necessary to maintain test hardware in multiple countries. Not all cloud providers or test solutions can provide this capability, so it is key to evaluate if global readiness is a requirement for you.

**Testing Web and Native Applications for Mobile Devices**

**MOBILE USERS**

In today’s highly connected and mobile world, it’s not surprising that the sale and usage of mobile devices, such as tablets, smartphones and laptops, has just exploded in recent years. Therefore, it is becoming mission critical that your organization is able to support performance testing of mobile web applications and mobile native applications that communicate with servers over HTTP. Due to smaller screen sizes and different input methods (touch screen) of mobile devices, many web applications look different when loaded on a mobile device compared to a full site.

From an application performance testing perspective, such mobile versions of a web page need to be treated as separate applications, even though they might share some components on the back end. This all comes down to the fact that a variety of mobile devices are driven by a range of operating systems that include Android, iOS, Windows Phone, and Blackberry, thus your testing solution must enable you to record test scripts from a PC, an emulator, or a mobile device. Moreover, it must simulate the bandwidth limitations of mobile network connections and support all existing and upcoming mobile phone standards like GPRS, EDGE, UMTS, HSDPA, HSPA+, and LTE.

As mobile users are travelling, it is also important to test to emulate traffic of mobile devices from different parts of the world.

**Conclusion**

Peak load testing is not simply a development need. It is a business need. Key stakeholders rely on your applications to be available and perform when they need them. This means that your applications must be able to support thousands or even hundreds of thousands of users. Micro Focus delivers a solution through its Silk Performer™ technology. Silk Performer enables you to ensure that your applications can perform through:

- **Silk Performer Web 2.0 Support.** Fast and accurate load testing for Web 2.0 technologies
- **Silk Performer Diagnostics.** Integrated root cause diagnosis that tunes application performance after one load test
- **Continuous Performance Testing.** Early and often performance testing ensures quality as applications are being built
- **Silk Performer CloudBurst.** First comprehensive load testing solution for simulating peak demands from the cloud
- **Mobile Testing.** Simulate massive traffic of mobile devices from different parts of the world and validate the performance of your mobile applications.