Automate performance testing to predict system behaviour and improve application performance

Business white paper

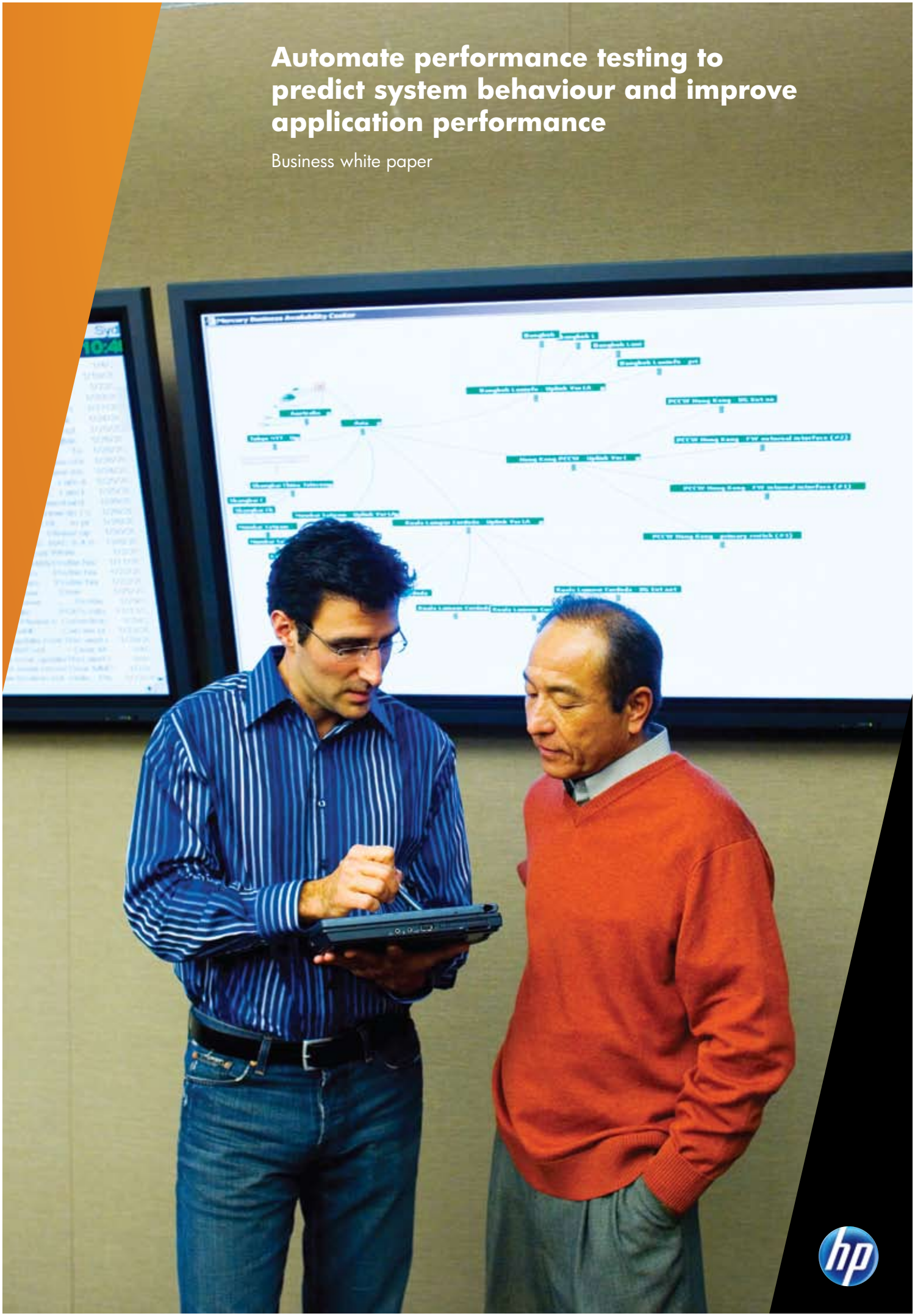


Table of contents

The incredible rate of change and the explosion of software complexity due to application modernisation introduce tremendous risk into the software development process.

Executive summary

Successful businesses rely on specialised software applications to drive efficiency and productivity throughout the enterprise. These applications can provide a more effective medium for collaboration and information sharing, and have become the primary channel for business-critical information sharing as well as transaction processing. Today's software applications – from email to customer relationship management (CRM) and enterprise resource planning (ERP) to Web 2.0 and service-oriented architecture (SOA) applications – run the business.

While software development technologies have changed and matured tremendously during the past few years, the complexity of modern applications has also exploded. Applications may use hundreds of separate components to do work once done with paper or by hand. Each individual component is a link in the chain. And the chain is as strong as its weakest link. This complexity directly correlates to more potential points of failure in a business process, and more difficulty in isolating the root cause of a performance problem.

Moreover, software applications do not work like a car. They do not have permanent parts that are only replaced when they wear out. Whether they are designed to deliver competitive advantage or to respond to changing business conditions, software applications are evolving weekly, monthly and annually. This stream of changes introduces yet another set of risks that companies have to manage.

The incredible rate of change and the explosion of software complexity introduce tremendous risk into the software development process. Rigorous testing is the only strategy to both quantify and reduce this risk to the business. The question for developers, quality assurance (QA) teams and management alike is how to accurately and thoroughly validate system performance before going live – without breaking the IT budget.

By automating performance testing, you can verify if applications meet the needs of your business before they go live. This translates into fewer deployment surprises, the ability to quantify the impact of changes on the end-user experience and the ability to pinpoint failing components and rapidly resolve them. Yet the prospect of automating performance testing raises new issues. Before undertaking an automated testing program, you should understand:

- What is performance testing? What should it accomplish?
- Why should it be automated?
- What are the right processes for performance testing?
- Who should be involved in a good performance test to achieve success?
- Who needs to see the results and how can results be reported to quantify return on investment (ROI)?
- Which features are essential when comparing automated load testing solutions?

This paper presents a brief overview of the benefits of automating performance testing and covers how best to approach performance testing. It also summarises the key functions and benefits of HP LoadRunner software, the industry-standard automated load testing solution that is part of HP Performance Center software.

What is performance testing?

Performance testing is a discipline you can use to leverage people, processes and technology to reduce the risks involved in application deployment, upgrades or patch deployments. At its core, performance testing consists of applying production workloads to pre-deployment systems while simultaneously measuring system performance and end-user experience. The goal is to identify and fix any performance issues under expected production levels. A well-constructed performance test should be able to answer questions like:

- Does the application respond quickly enough for the intended users?
- Will the application handle the expected user load and more?
- Will the application handle the number of transactions required by the business?
- Is the application stable under expected and unexpected user loads?
- Will users have a positive experience (for example, fast response time) on go-live day?

By answering these questions, performance testing helps you quantify in business terms, the impact of a change. This quantification in turn clarifies the risks of deployment. An effective automated performance testing process can help your enterprise make more informed release decisions and prevent system downtime and availability problems.

Why automate performance testing?

Performance testing accurately tests the end-to-end performance of a system prior to going live. However, this is not a process that can be done manually.

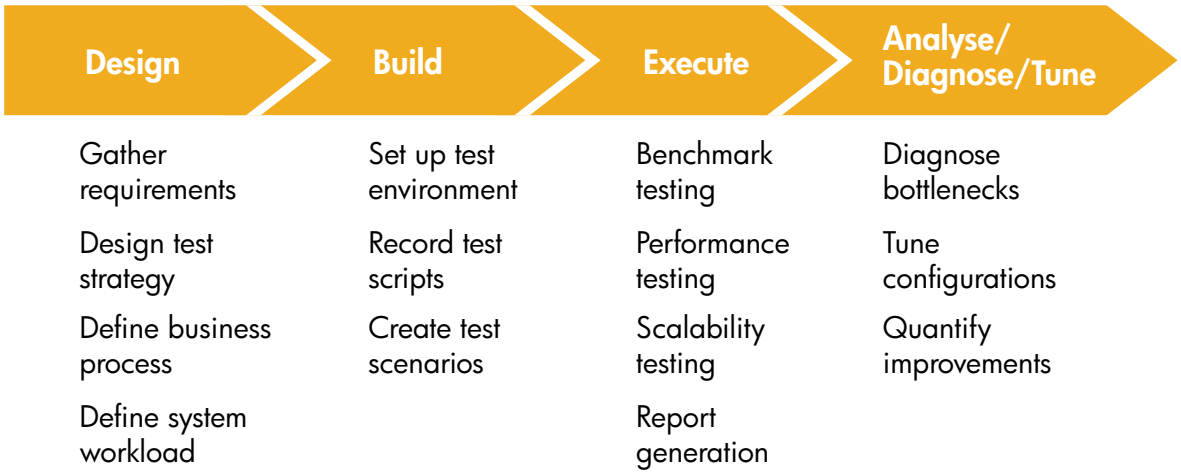
Performance testing solutions should be able to:

- Emulate hundreds or thousands of users interacting with the system without large hardware demands
- Repeat the load in a consistent way
- Measure end-user response times
- Monitor system components under load
- Provide robust analysis and reporting engines

Effective automated testing solutions typically use five major components to build and run tests. These include:

- A virtual user generator to capture end-user business processes into automated scripts
- A controller to organise, drive and manage the load test
- Load generators to run the virtual users during execution
- Monitors to monitor the various system components
- An analysis engine to view, dissect and compare results

Figure 1: The four phases of an effective automated performance testing process



The automated performance testing process

Organisations that have successfully implemented automated performance testing have done so by breaking the process into discrete phases. While specific implementations may differ, performance testing can be broadly described as having four phases – Design, Build, Execute and Analyse/ Diagnose/Tune. Each of these phases has specific tasks, involving different participants, which should be completed before moving to the next phase. At the highest level, the four phases can be described as follows:

- The Design phase involves defining the business processes to be tested, the business process mix of an average or peak production hour and the overall user and response-time goals for the system.
- The Build phase involves setting up and configuring the test system and infrastructure, and using the automated performance testing solution to build both test scripts and load scenarios.
- The Execute phase consists of running the load scenarios and measuring system performance.
- The Analyse, Diagnose and Tune iterative phases go beyond measuring system performance and take load testing to another level. Here, the focus is on pinpointing problems to help resolve them rapidly and tuning system parameters to enhance performance in an ongoing fashion.

The following sections will drill down another level to examine the tasks necessary to make each separate phase of the automated performance testing process successful.

The Design phase

The Design phase is the primary time when your performance testing team works with the line of business (LOB) managers to gather performance requirements. Requirements can be thought of as falling into four buckets – business, technical, system and team requirements. Business requirements are generally gathered by meeting with subject-matter experts (SMEs). These may be business analysts or end users.

A comprehensive set of business requirements exists when the following are in place:

- **An application overview:** Create a demo of system usage to allow your performance team to understand at a high level how the application is used.
- **A business process list:** Generate a list of the key business processes to reflect the activities that end users perform on the system. Experience shows that 20% of the total business processes are often responsible for 80% of the load on the system.
- **Business process flows:** Create Microsoft® Word documents to detail the exact steps/screens of each business process.
- **Business process diagrams:** Prepare business process flowcharts to illustrate branching conditions in the business process flows.
- **A transaction list:** Compile a list of the key activities within the business process that need to be measured under load, such as Login or Transfer Funds.

You can gather technical requirements by meeting with the system administrators and database administrators (DBAs). These individuals may be part of the enterprise's development group, or operations division or both. You will have a comprehensive set of technical requirements when the following are completed:

- **Environment walkthrough:** Conduct a walkthrough of the testing architecture with the systems or infrastructure team
- **Systems scope meeting:** Hold a meeting to discuss and agree on what pieces of the system should be excluded from the test process
- **Production diagram:** Create a diagram of the production infrastructure to flag deltas that may impact performance during the migration from QA to production

It is vital to gather system requirements. These are the high-level goals for the system that govern the pass/fail status of the load testing process. These generally are agreed upon by working with the project manager from the LOB. System requirements include answers to the following:

- How many users must the system support at normal and peak periods?
- How many transactions per second must the system be able to process?
- What are the minimum and maximum acceptable response times for all business-critical transactions?
- How do the user communities connect to the system?
- What system workloads are experienced in production? What is the transaction mix?

Finally, you should iron out team requirements before moving to the Build phase. This consists of determining which performance team members participate in the upcoming load test. Initially, you can determine this automatically (for example, when there is a team of just one person). However, if performance testing becomes part of a Center of Excellence (CoE), you should handle resource allocations and internal logistics in the Design phase.

Gathering a complete set of business, technical, system and team requirements up front lays the foundation for an effective and successful load test.

The Build phase

In the Build phase, you turn the business processes and workloads identified in the Design phase into automated components that you can leverage to drive a repeatable, realistic load. You can break this into two areas of focus: automation setup and environment setup. Automation setup consists of a set of sequential tasks by a performance engineer:

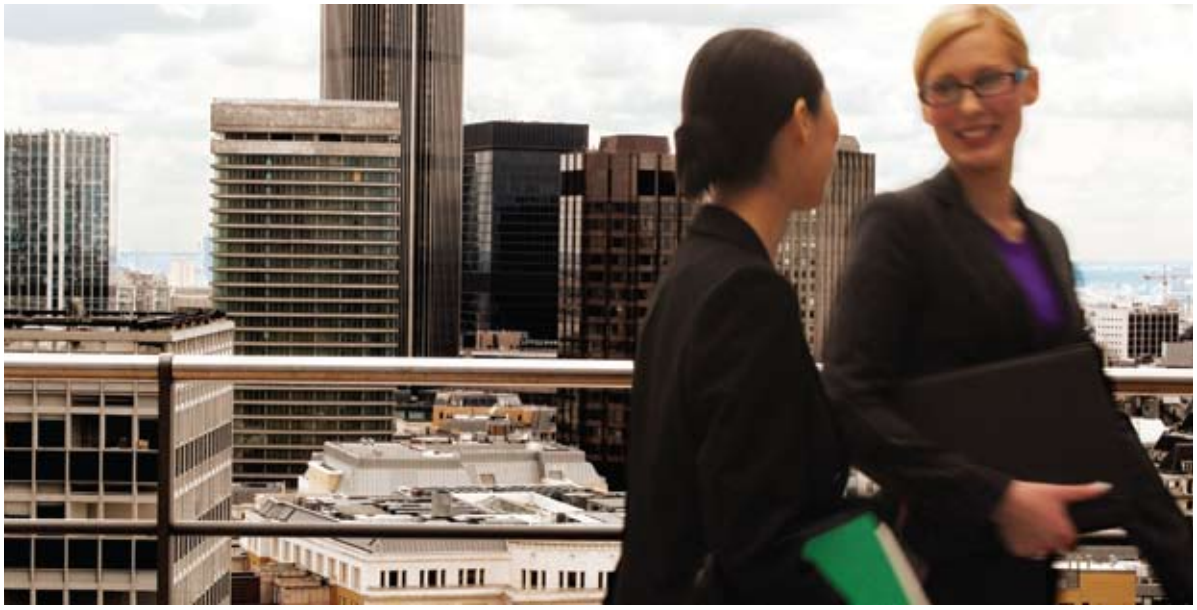
1. **Scripting:** Record the documented business processes into automated scripts
2. **Transactions:** Insert timers to produce the logical timings desired by the business
3. **Parameterization:** Replace all input data, such as log-in IDs and passwords, with a pool so each virtual user accesses the application using unique data
4. **Scenarios:** Create production workloads by assigning to different groups of users different scripts, connectivity and user behaviour
5. **Monitors:** Decide which servers or machines to monitor under load

Environment setup consists of assembling the hardware, software and data required to execute a successful, realistic load test. This may involve working with the systems, DBA, operations and business teams. The end result of the Build phase is a set of automated assets that you can execute at will on an available, configured environment.

The Execute phase

Among those new to performance testing, there is often a misconception that execution is a single event. In fact, it is a multi-step process consisting of several types of performance tests. Each type of test provides information necessary to understanding the business risk of releasing the application. The different types of load tests include the following:

1. **Baseline tests** verify that the system and its surrounding environments function within reasonable technical parameters. Performance tests are run with only five to ten users to baseline end-user transaction performance. These tests should be executed at the start and end of the performance testing process to measure absolute response-time improvement.



2. **Performance tests** simulate a load on an environment to provide information about how many users the system can handle. These tests should emulate average and peak-hour production usage. They should apply real-world user behaviours such as think time, modem emulation and multiple browser types for maximum accuracy. All monitors and diagnostics should be run to achieve maximum visibility into system degradation and bottlenecks.
3. **Benchmark tests** are designed to measure and compare the performance of each machine type, environment or build of the application in an ideal situation. You can run these tests after verifying the system's scalability to understand the performance impact of different architectures.
4. **Soak tests** are designed to run the system for a long period of time under load and examine how well the system performs.
5. **Peak tests** are designed to simulate a peak load on the system for a period of time to help demonstrate that the application and the underlying hardware can handle a high load for reasonable periods of time.

The Analyse, Diagnose and Tune iterative phases
After you complete the Design, Build and Execute phases of a load test, the project advances to the Analysis, Diagnosis and Tuning phases. These functions are ongoing and are conducted iteratively. The load testing solution should provide a comprehensive view of end-user, system-level and code-level performance data and identify the likely causes of system slowdown. Such a view enables you to determine if performance goals have been met – and if not, why not and who owns the problem.

How to determine the ROI of performance testing
The ROI of a good performance testing solution consists of two components:

- Risk mitigation helps a project go live with the right system scalability and performance. Risk mitigation is classic performance testing. You should be able to report back data to the development/project teams that provide clear, quantifiable information on how well the system should scale and perform in production.
- Performance optimisation quantifiably improves the performance of the system as measured by improved end-user response time or by reducing the overall hardware infrastructure that is required.

How to improve performance:

During and after a performance test, a wealth of information surfaces that can improve system performance. You can uncover key information during monitoring, analysing, diagnostics and tuning.

1. **Monitoring:** Monitoring during a performance test shows what is happening at each tier of the infrastructure, offering more clarity on the performance of the database server, Web server, application server or individual application or processes during the test. Monitoring may quickly flush out valuable information, for example, that the central processing unit (CPU) at the application server is being pegged at 100 per cent with 200 users, well short of the goal of 300 users. (This would point to requiring more application server capacity – or to enhancing the application itself.)
2. **Analysis:** After a load test is completed, you can correlate metrics – such as virtual users against CPU or application server CPU to Web server – to uncover additional information about application behaviour.
3. **Diagnostics:** An effective performance testing solution should provide performance engineers with a single, unified view of how individual tiers, components and SQL statements impact the overall performance of a business process under load conditions. Performance engineers should be able to see all components touched by an end-user transaction and then determine how much processing time each component uses and how many times it is called. With this information, project and QA managers can focus resources to improve the end-user experience by targeting the most significant Web, application and database server bottlenecks.
4. **Tuning:** Many companies conduct automated performance testing before, during and after application deployment. Some automated performance testing solutions can systematically identify and isolate infrastructure performance bottlenecks, then resolve them by modifying the system configuration settings. By iterating through the process of resolving infrastructure bottlenecks, you can establish an enhanced configuration for go-live.

Who should be involved in an effective performance test?

A successful performance testing project requires contributions from many individuals. Some roles that should be included in a testing program include:

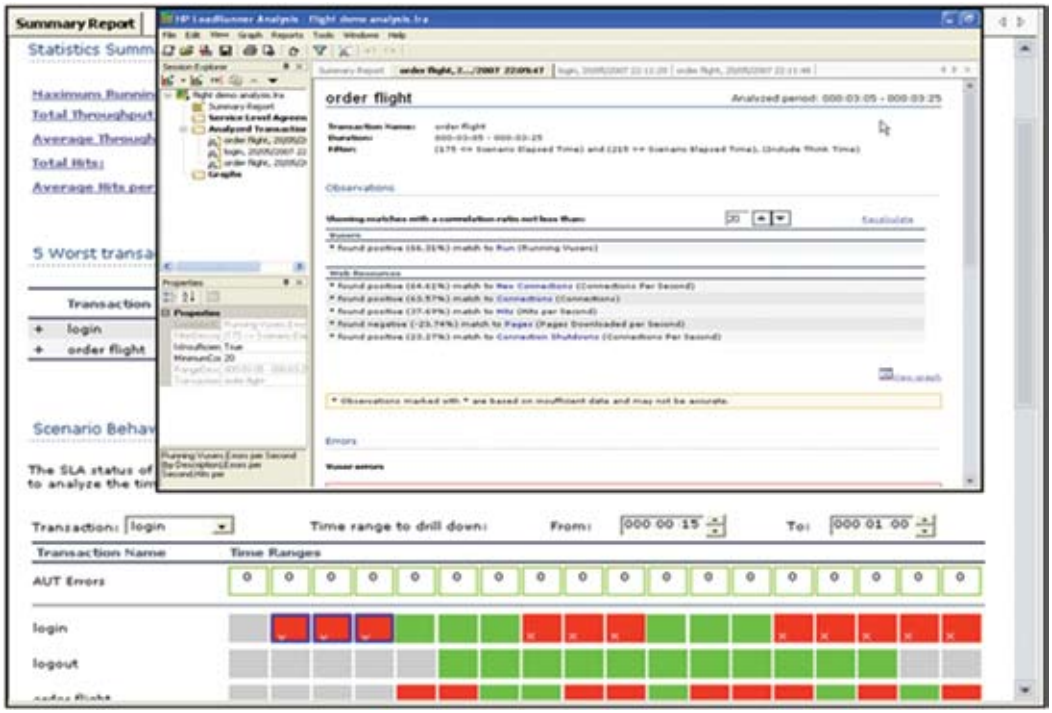
- Project manager: Co-ordinates multiple performance projects, manages testing schedules, acquires necessary hardware and/or software and handles resource and funding issues
- Business analyst: Accountable for review and sign-off of system performance from a business standpoint; assists in the development of the transaction mix and the expected timings for performance testing
- Performance manager: Accountable for coordinating the efforts of the performance support team, acting as the point of contact for the group; responsible for managing day-to-day activities of the performance effort
- Performance testers: Responsible for creating and executing the automated tests and for gathering test results
- Application architect: Receives information from diagnostics and analysis of a load test to improve application performance or to solve performance bugs
- Infrastructure specialists (DBAs, network administrators, system architects): Receive information from tuning and analysis of a load test to improve system performance or to solve performance bugs

Why HP LoadRunner software?

HP software has a large per cent of the performance testing market share. HP LoadRunner helps you prevent costly performance problems in production by detecting bottlenecks before you deploy a new system or upgrade. Your enterprise can help verify that new or upgraded applications deliver intended business outcomes before go-live, preventing overspending on hardware and infrastructure.

With HP LoadRunner, your enterprise can measure end-to-end performance and diagnose application and system bottlenecks – all from a single point of control. It supports a wide range of enterprise environments, including Ajax, Flex, Web services, J2EE and .NET.

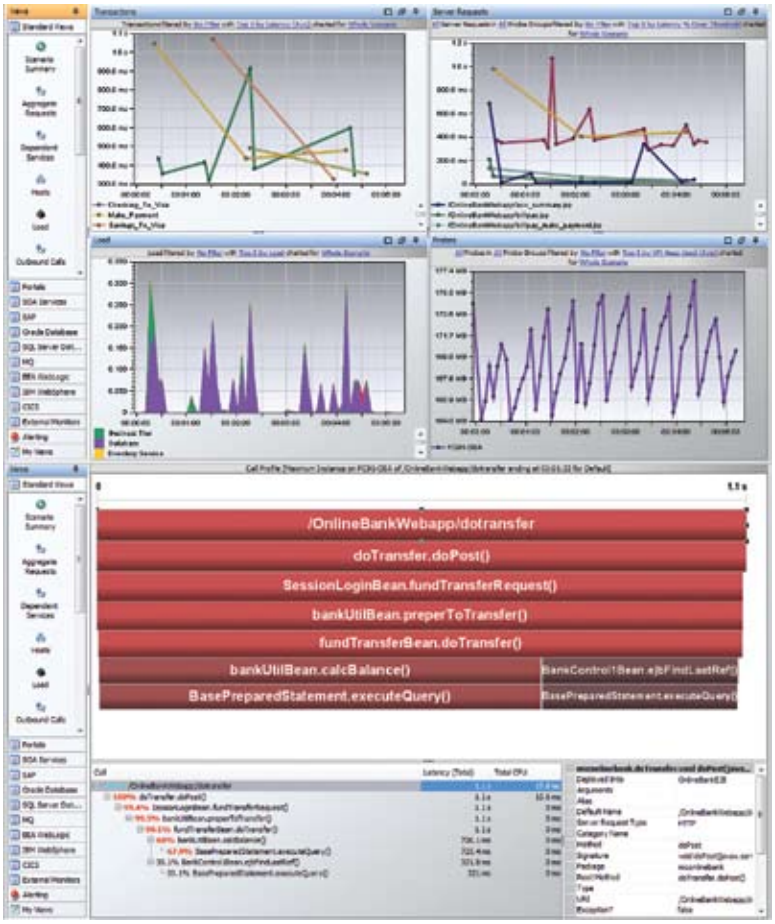
Figure 2: Bubble-Up Analysis in HP LoadRunner



The following characteristics describe the key capabilities of HP LoadRunner that make it the industry leader in this space:

1. **On-demand production workloads:** HP LoadRunner can drive hundreds and thousands of virtual users, executing different business processes, to emulate the production conditions a deployed application is likely to face. This helps you uncover performance and scalability bottlenecks before going live that would otherwise surface in production. It helps you to greatly reduce production downtime and poor performance, making it easier for you to meet service-level and uptime requirements.
2. **Broad environment coverage:** HP offers an extensive testing environment with support for many protocols and platforms. HP LoadRunner supports legacy as well as modern applications and protocols including Web 2.0, J2EE, .NET, XML, SAP, Oracle, Microsoft Remote Desktop, wireless, Citrix and client/server applications. As the types of applications being deployed change from client/server to Web to Java™, the same tool can be used for performance testing. It offers one consistent tool and one set of employee skills even if applications change over time. It also enables a lower total cost of ownership (TCO).
3. **Ease of use:** HP LoadRunner is built from the ground up for QA users. It provides Visual Scripting Language, Data and AutoCorrelation wizards and ActiveScreen technology to make scripting and running of load tests as easy as possible. This results in short ramp-up time, faster ROI and easier performance testing within weeks of training.
4. **Service Level Objective (SLO) definitions:** HP LoadRunner provides the ability to define SLOs for transactions as part of the load test. This helps to identify quickly, which transactions and tests failed to meet the business requirements.
5. **Built-in monitoring:** HP LoadRunner includes HP SiteScope for pre-production systems. HP SiteScope offers non-intrusive, real-time performance monitors that provide detailed metrics on all parts of the system under test. This includes Web servers, application servers, databases, enterprise resource planning (ERP) and CRM systems, firewalls and load balancers. HP LoadRunner can identify hardware limitations and software configuration issues that might otherwise go undetected.
6. **Easy and comprehensive analysis:** The HP LoadRunner Bubble-Up Analysis automatically digests all the monitoring and diagnostics data and calculates the top causes of performance degradation. It also quickly pinpoints the transactions that did not meet the pre-defined business requirements, so that users can make informed decisions. Converting performance testing results in actionable, precise data for the development team, dramatically reduces time to resolution and allows for more testing cycles. This helps you put into production a high-quality application.

Figure 3: HP Diagnostics drills down into problem areas



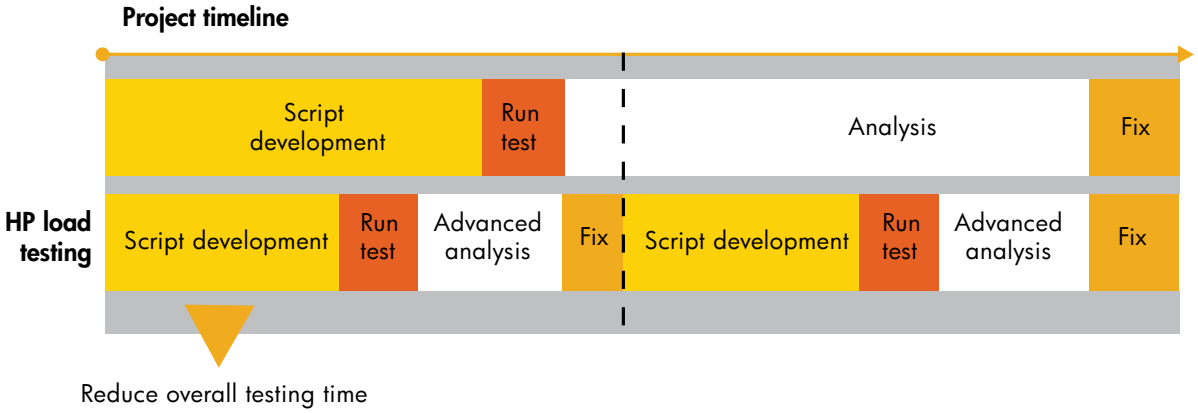
7. **Integration with HP Diagnostics:** HP LoadRunner has a seamless integration with HP Diagnostics, which can trace transactions on the application tier to identify the root cause of application bottlenecks. Within LoadRunner, you can drill down from a slow end-user transaction to the bottlenecked method or SQL statement causing the slowdown. You can also identify the exact cause of memory leaks. This granularity of results helps every load test provide development personnel with actionable results, reducing the cost and time required to enhance J2EE, .Net, SOA, SAP and Oracle deployments.
8. **Scalable solution:** HP LoadRunner requires low CPU and memory resources per virtual user, for high scalability with limited hardware. This helps to diminish hidden hardware costs in implementation.
9. **Unified scripting engine:** HP LoadRunner has the same scripting engine as HP Business Availability Center software. This helps you reduce training costs, scripting costs and TCO for HP software.

Use HP diagnostics software to resolve performance problems

HP Diagnostics software integrates with HP LoadRunner and HP Performance Center to help you address the unique challenges of testing complicated applications across the application lifecycle. HP Diagnostics isolates application performance problems and reduces the mean time to resolution (MTTR) of application performance bottlenecks. It provides actionable information to help you resolve performance problems. HP Diagnostics provides you with the ability to:

- Find and solve more problems earlier in the lifecycle
- Raise the quality bar for your applications by finding the most common application problems before they go live
- Collect concrete data to support decisions surrounding the go-live point for an application
- Manage and monitor applications after they have gone live with role-based visibility to quickly solve problems

Figure 4: Improve cycle time efficiency using HP LoadRunner



During a performance test, HP Diagnostics traces J2EE, .NET, SOA or ERP/CRM business processes from the client side across all tiers of the infrastructure. The modules then break down each transaction response time into time spent in the various tiers and within individual components. Your performance testing team gains:

- An intuitive, easy-to-use view of the way individual tiers, components, memory and SQL statements impact the overall performance of a business process under load conditions. For example, during or after a load test, testers can not only point out to the application team that the application is not scaling, but also provide them with actionable data.
- The ability to effectively triage and find problems with business context. This enables your team to focus on problems impacting business processes.
- The ability to more easily find components relevant to a specific business process under test. Because J2EE and ERP/CRM applications potentially use thousands of components, this can be a challenge. HP Diagnostics automatically detects which components are 'active' when a given transaction is executed and collects data on them for analysis. Components untouched by the business process are filtered out, enabling your team to focus on getting the job done, not on configuring the system.

Summary

Your business cannot gamble on the performance of its mission-critical applications. By automating performance testing, your enterprise can reduce costly performance problems in production by detecting bottlenecks before new systems or upgrades are deployed.

HP LoadRunner is a load testing solution for predicting system behaviour and performance – providing comprehensive integrated load testing, service testing and diagnostics. With HP LoadRunner, you can measure end-to-end performance, execute asynchronous performance testing and diagnose application and system bottlenecks – for a far better final performance result.

Contact information

To find an HP Software sales office or reseller near you, visit **www.managementsoftware.hp.com/buy**

To understand how performance testing helps you improve application performance and reduce risk of application deployment process, please visit: **www.hp.com/go/performancevalidation**

Share with colleagues

Get connected

www.hp.com/go/getconnected

Get the insider view on tech trends, alerts, and HP solutions for better business outcomes

© Copyright 2007, 2010 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

Java is a U.S. trademark of Sun Microsystems, Inc. Microsoft is a U.S. registered trademark of Microsoft Corporation. Oracle is a registered trademark of Oracle Corporation and/or its affiliates.

4AA1-4227EEW, Created August 2007; Update April 2010, Rev.1