EXPERTISE IN PERFORMANCE TESTING

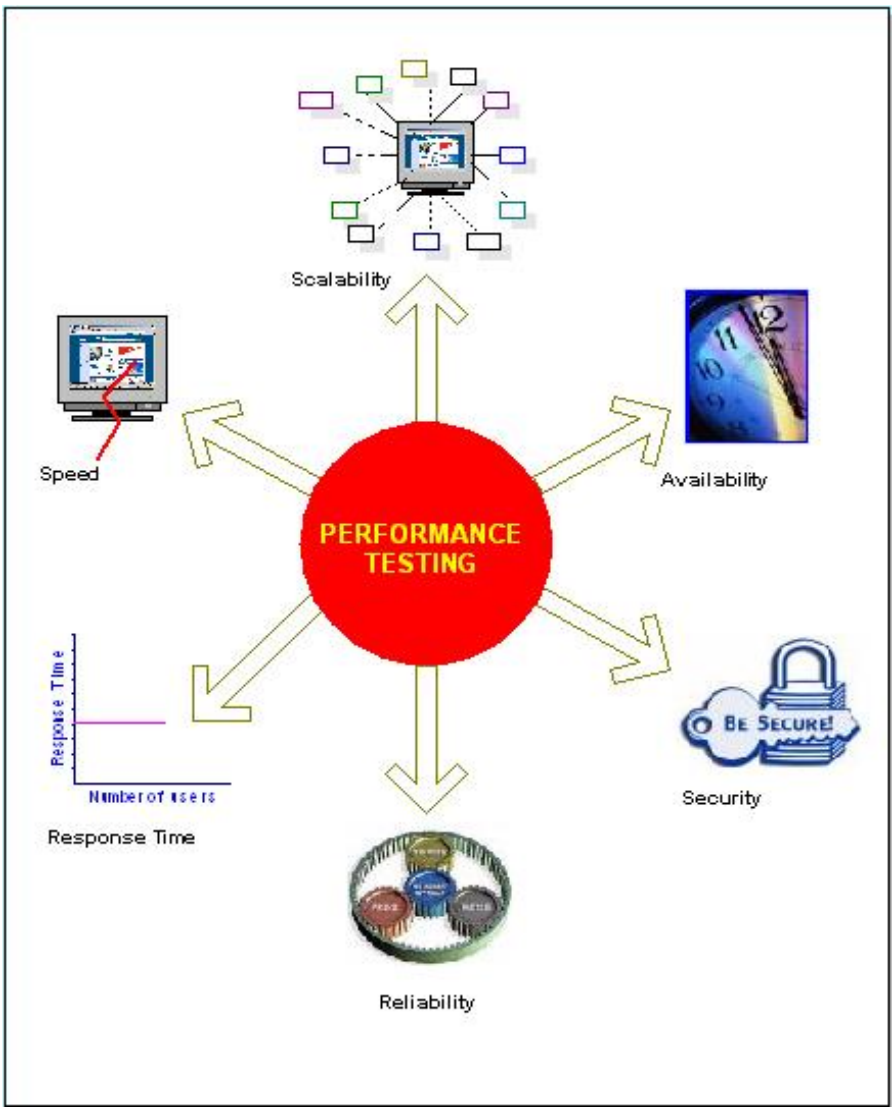


Table of Contents

List of Figures

Advantage Disha

- One of the largest test focused companies world-wide specializing in Product testing
- Over 1500 man-years of testing experience
- 800 high-caliber test professionals
- Domain expertise in:
  - Networking and Operating Systems
  - Infrastructure Services
  - Web Services
  - Enterprise Applications
  - Storage
  - Mobility and Telecom
  - Embedded Systems
- Tested more than 1000 software products
- Tested some of the most complex products in the world
- Strategic test partner to some of the largest product companies in the world as well as start-ups, mid-sized companies

Disha Background

Founded in 1997, Disha Technologies is a pioneer in independent testing and provides STQE services to ISVs. We have built a strong, efficient and dedicated team comprising test analysts and strategists, automation developers, test engineers, project managers and service delivery experts. We are committed to helping customers build high quality software and hardware solutions by providing world-class engineering and consulting for testing and quality assurance.

We have test engineering labs in Pune, Hyderabad, Bangalore, and Bellevue (Washington). We have sales presence in USA and Europe. Disha is an independent subsidiary of Aztec Software and Technology Services Limited.

Introduction

In addition to core product testing, Disha offers a number of specialized expert STQE services. This document describes the Performance Test practice through which we have provided Performance Test services to numerous product companies. The following paragraphs describe the methodology, tools, people skills, and sample case studies related to performance testing.

Introduction to Performance Testing

Performance Testing attempts to make sure that the application provides a fast, reliable user experience when faced with typical as well as exceptional design loads in production settings. It attempts to make sure that application responds to loads beyond design loads gracefully. It helps answer questions such as:

- Is the application prepared for as many users as expected?
- Is the application prepared for increasing load over the month/years?
- Does the application survive a massive peak of users (e.g. if it is mentioned on national TV)
- How many users can the application handle before users get error messages or timeouts?
- Is the internet connection bandwidth sufficient?
- Is the hardware too small or too powerful?
- Is the application correctly tuned (OS parameters, DB parameters etc.)
- How well does this application compare with earlier versions of the application?
- How well does it compare with other applications in this genre'? (e.g. ESD and Amazon)

Performance is typically perceived by customers as one of the following:

- **Speed –** Does the application respond within acceptable time?
- **Scalability –** Is the application architected to accommodate constantly increasing load expectations?
- **Reliability –** Will the application be stable under normal loads?
- **Robustness –** Does the application deal with unexpected loads in a reasonable manner?

Why Performance Testing

In a highly competitive environment with rapidly changing technologies and shortage of skilled workforce, the biggest challenge for all the software organizations is the success of the application developed. No matter how rich the product is functionally, if it fails to meet the performance expectations the product will be considered a failure.

Performance testing helps in preparing the application deal with real life as well as unusual load conditions, thus reducing the number of failures related to performance and availability. It helps with capacity planning, and helps reduce application deployment infrastructure cost. It helps improve customer experience thus minimizing the risk of customer abandonment (due to service unavailability and poor response times).

Today's software product landscape is dominated by N-tier Web-based applications which present unique challenges for performance testing. Disha's current focus is on performance testing of N-tier Web Applications.

Challenges Involved in Performance Testing of N-tiered Applications

N-tiered applications present unique performance test challenges. Some of these challenges are listed below.

- Short release cycles
- Continually changing technologies
- Large number of users
- Large variety of user operating environments, e.g. Windows, Linux, Mobile
- Expectation of 24x7 availability due to the global nature of the Web
- An N-tier application is made up of a large number of components residing on multiple machines. This increases the scope and complexity of testing.
- N-tier applications are typically multi-threaded. This introduces additional hard-to-find problems such as deadlocks, race conditions, improper handling of shared resources, etc.

Types of Performance Testing

There are various types of performance tests, each of which performs a specific function. We broadly classify these groups into following groups:

- **Normal Load Testing**: These are tests that involve subjecting the application to design or less-than-design loads. The purpose of these tests is to verify whether the application performs adequately when subjected to design loads.
- **Heavy Load Testing**: These are tests that attempt to break the application by subjecting it to unreasonable load levels while denying adequate resources (memory, disk capacity processing power etc.) necessary to handle the load. The idea is to stress a system to the breaking point in order to find bugs that will make that break potentially harmful. The system is not expected to process the overload without adequate resources, but to behave (e.g., fail) in a decent manner (e.g., not corrupting or losing data).

Normal Load Testing

Under normal load testing we measure the performance of the system when subjected to the ***expected real life*** load. This kind of testing involves understanding of the expected usage patterns, simulating these patterns and measuring how well the system performs in terms of response time and throughput.

Normal load testing includes the following tests.

- Load Test
- Scalability Test
- Benchmarking Tests

Load Test

Load tests are generally carried out to measure response time and throughput of the application while subjected to ***statistically representative load***. This involves:

- understanding expected usage patterns
- simulating these patterns
- measuring how well the system performs (response time, throughput etc.).

Scalability Test

Scalability tests are executed to answer questions like,

- Does the application support large numbers of concurrent users?
- How does response time degrade when the number of users goes up?

Benchmarking Test

Benchmarking tests are executed to answer questions like,

- How well does this application compare with its earlier versions?
- How well does it compare with other applications in this genre?

Heavy Load Testing

Heavy load tests are executed at greater than expected user loads, generally far heavier than a system is ever expected to have to handle. They are used not to determine if a system will fail, but where it will fail first, how badly, and why. Answering the "why" question can help determine whether a system is as stable as it should be. The majority of significant deficiencies in the system will already have been identified during the execution of "Normal Load" tests, so this phase deals more with assessing the impact on performance and functionality under an unexpectedly heavy load. These tests are designed to find subtle performance issues, such as minor memory leaks, caching, and database locking. Heavy-load scenarios will also identify system bottlenecks not previously noticed, which may be found to be partially responsible for earlier identified problems.

These tests are generally only executed after several rounds of tuning.

Heavy load testing involves the following tests.

- Stress Test
- Hammer Test
- Spike Test
- Long-haul Test

Stress Test

Stress test helps determine the largest number of users that the application can successfully handle. It also verifies whether for loads exceeding this limit the system fails and recovers gracefully. Some of the bugs discovered in this test may not be fixed.

Hammer Test

Hammer tests bear little or no resemblance to real-world loads. These tests take all existing load-generation scripts and methods, eliminate user think times, and increase load until failure occurs. These tests are designed to find the weak spots in a system so that appropriate risk-mitigation strategies can be developed for those weak spots. There are no pass/fail criteria for hammer tests, as the intent is to make the system fail.

Spike Test

Spike tests use real-world load but with extremely fast ramp-up and ramp-down times. For example, it is common to execute spike tests that ramp up to 100% or 150% of expected peak user-load in 10% of the normal ramp-up time. Situations like this can cause inadvertent denial of service, e.g. a website being mentioned on national TV or an email marketing campaign sent to prospective customers that asks them to visit the website.

Long-haul or Reliability Test

Long-haul testing is a method of improving software quality by running a set of tests, which represent common user scenarios, over an extended period of time. This type of testing uncovers bugs that might be missed by the normal STQE effort which consists of functional and standard performance testing.
Long-haul tests help to find the following kinds of bugs.

- Resource leak
- Timing-related
- Hardware-related
- Counter overflow

Why Disha

Proven Performance Test Methodology

We begin every performance test project with the exercise of building a test strategy which is a joint effort with the product development team to come up with a detailed understanding of the application's performance requirements and an approach to address those requirements. The test strategy asks questions like:

- What is the application's usage pattern?
- What are the business transaction volume and mix?
- What are the performance goals of the application? For example, When subjected to a load of 1000 concurrent users, 80% of all page responses shall take less than 5 seconds.

The Test Strategy then:

- proposes an approach to test the application for the performance goals.
- lists sample test scenarios for various types of performance tests.
- evaluates and recommends tools to use.
- identifies data generation requirements (for example reference data of product inventory for a marketplace application).
- identifies infrastructure requirements.
- identifies the required interactions among various groups, like Test, Dev, PM, Systems etc, to ensure smooth execution of the project.
- recommends the execution process for the project.
- time and cost estimate of the overall effort.

The next phase involves execution of the test strategy which includes the following activities.

- Setting up the infrastructure necessary for executing the Performance Testing project. This would include:
  - Setting up the test labs with the requisite hardware and software
  - Network provisioning
  - Any other setup
- Designing test cases to implement high-level scenarios identified in the planning phase.
- Developing scripts and generating data for various test cases.
- Test Execution. In this phase we measure application performance characteristics specified in the Test Planning phase (response times, throughput, and percentage of failed requests).
- Monitor system parameters of the servers being tested (CPU utilization, memory availability and so on.)
- Monitoring key parameters of the software entities making up the Application being tested (web-server, app-tier and database).
- Preparing reports according to the Test Plan.
- Analyze the observed data and offer suggestions to do further investigations.
- identify bottlenecks.

Performance Test Lab

The infrastructure required for performance testing is highly specific to each application, and is usually an expensive part of the total effort. It is a challenging task to design the required infrastructure for a performance testing project. It includes creating a multi-machine setup, and high bandwidth and complex topology network configuration.

Disha has a deep understanding of the complex issues involved in designing and setting up performance test labs. We have developed a standard infrastructure configuration which can be customized for a variety of commonly used N-tier applications in the small and medium business segment. This includes a mix of servers, workstations, network topology, and software tools. We also have dedicated network bandwidth for doing PT projects.

Multiple Testing Configuration Models

We offer different models including,

- Application hosted by customer can be tested remotely by the Disha team.
- Application to be hosted by Disha and testing executed in-house.

Expertise in Solving Problems Unique to Performance Testing

Performance testing involves some unique and often unexpected problems. Through experience we are intimately familiar with such problems and are therefore well equipped to solve them. Examples of such challenges are:

- Creation of large quantities of reference data.
- Creation of large number of user accounts.
- Collating information from multiple log files located on multiple test machines and generating reports.
- Analysis of huge amounts of data.
- Simulation of complex topologies.
- Remote monitoring of servers.
- Error and exception handling.

Tool Selection Capabilities

Tool selection is a critical activity in Performance Testing projects. Choosing the wrong tool can set a project back by months when it is discovered to be inadequate. We have an intimate understanding of the trade-offs in using various tools and can help customers identify the right set of tools for the job at the start of an engagement rather than later.

Understanding of Various Performance Test Tools

Disha has expertise in using the most popular and powerful tools available in the world today. We have a good level of expertise using commercial as well as freeware tools. We can deliver the cost-effective solution to our customer using a combination of open-source and specially designed testing tools. These include the following.

Open Source Tools

- Open STA
- JMeter
- Funnel Web Analyzer

Commercial Tools

- Microsoft's ACT
- Load Runner
- Silk Performer
- WebLoad

Free Tools to Provide a Cost-Effective Solution

Commercial Performance Testing tools are rich in features but can be very expensive. On the other hand, free tools lack a number of features available in commercial tools. Disha has lots of experience working with free Performance Testing tools and has enhanced their usability by creating methodologies and scripts that bridge the gaps in these tools and make them more effective. To the customers this means great value addition at no additional cost.

Data Analysis Skills

Using a tool is relatively easy but the real challenge lies in analyzing the generated test data. We have expertise and in-house tools that lets us do this job efficiently.

Disha's Performance Testing Methodology

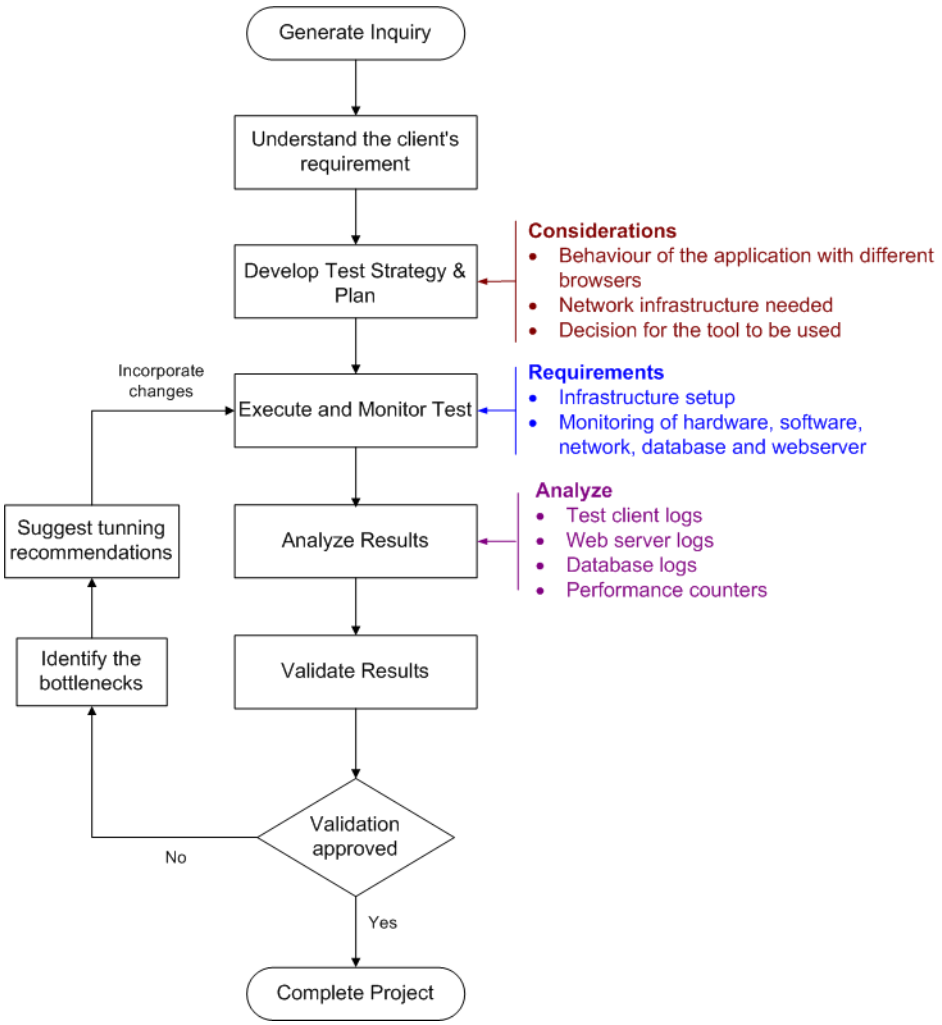


Figure 1: Disha's Performance Testing Methodology

Disha's Experience in Performance Testing

Over the years Disha has gained rich experience in performance testing. It includes a wide range of software applications starting from utilities to enterprise applications. Few projects in brief are listed below.

1. Performance testing of a feature called P4 promo

Performance testing of an Instant Messaging application developed and sold by one of the largest product development company.

'Promo' was a feature designed to direct users (of our customer's application) into the Launch Site to promote specific applications/games.

Project involved following significant tasks.

- Test case development.
- Verification of the user interface.
- Test the functionality of the P4 promo feature, across various platforms and browser matrices.
- Automating the possible tests.

Testing involved the following.

- Sending 1000 invitations to one or multiple users to check
  - CPU usage
  - Senders system's hard drive status
  - System memory status
- Executing tests at different times in a day to evaluate peak load.
- Launching of application from Promo over a local area network, dial up connection (with different speeds like 28.8, 56 KBPS and so on.)
- Launching of automated scripts to generate about 1000 users (with variation in time.)

We verified that the response times are not alarming differently than the expected times.

2. Performance testing of an Instant Messaging (IM) gateway

Objective was to make sure if all IM clients worked as desired when used via another Gateway simultaneously.

Project involved the following significant tasks.

- Defining test strategy.
- Defining test coverage based on components.
- Developing test cases based on the product specification/engineering specification and test coverage defined.
- Designing Test Framework to facilitate future expansion.

Challenges faced:

- Application had memory leaks.
- Insufficient time to test.
- Convincing developers that the breaks/crashes are not because of dependencies but actually in their code.
- Isolating problem and identifying where it occurred. This was challenging when it involved multiple IM clients using different protocols.
- Analyzing test results from custom test tool provided by the client.
- Synchronizing events logged in the log file versus performance counters.
- Reporting results of test execution and filing bugs in client's database.
- Regression testing of bugs on a daily basis.
- Automation of test cases or using/modifying existing harness to run tests.
- Finding workarounds in the area where test automation support was missing.

Benefits & achievements:

- Disha identified most of the core bugs by executing defined test cases/scenarios.
- Logged around 50 bugs in client's database (memory leaks and functionality bugs).

3. Performance testing of User Interface Automation

Objective was to test the performance of a product used for performance monitoring console. The product used Sysmon.ocx, an active-x control. Project involved development and automation of test cases on customer's upcoming OS.

Project involved following tasks.

- Defining test strategy.
- Defining test coverage based on components.
- Developing test cases based on product specification/engineering specification and defined test coverage.
- Defining test passes (Regression, Sanity, etc).
- Designing Test Framework to facilitate future expansion and dynamic component building.
- Automation of test cases.
- Finding workarounds in the area where test automation support was missing.

Challenges faced in the project:

- Development of Test Framework for automation.
- Use of custom automation tool which was under development.
- Handling COM Interoperability issues.
- Finding workarounds for issues where even customer provided tool did not support automation.
- Making the same code to work on three different OSs, handling each OS's issues.
- Making code language neutral to make it work on any other supported language.

Some of the benefits & achievements of the project were:

- Disha developed Test Framework capable of handling regression tests.
- Disha developed code for Sysmon API BVT testing and UI test Automation.

4. Performance testing of an application used in manufacturing industry

This application was used to parse input data from XML files and update records in an Oracle database. The input XMLs contained data in form of batches. Objective of the test was to find out the performance (Time required for inserting data) of an Oracle database by,

- increasing number of batches with single instance of the application.
- increasing number of batches with two instances of the application.

Database schema that was used by the application needed testing under above-mentioned conditions. The threat involved was, after certain number of batches if locks on certain tables are not released, the system may get into a deadlock and it may affect the further data updates.

5. Performance testing of XML Firewall

Objective was to monitor HTTP response from a sample web service with respect to HTTP requests of various message sizes, message types and with multiple clients passed through the XML firewall. The project involved development of the test harness using OpenSTA (Automation Tool)

To develop this harness we,

- created a sample web service (to echo the HTTP request) and deployed it to a Tomcat web server.
- deployed the test policies to the XML Firewall.

6. CRM performance testing

This testing was conducted for CRM Sales (Standard version) running on one of our customer's server. Performance test did not include CRM Sales for Mail client (Outlook client) offline scenarios; however, cases applied in these tests validate both, the CRM Web client and the Outlook client, running in online mode.

Specific scenarios tested demonstrate the ability of CRM application to scale to 1,000 users.

While performance test represents a fixed workload in a static test environment the test cases that were developed took into account the practical implementations of CRM. The test methodology created for this performance test was used to examine varying deployment workloads for CRM.

Test scenarios and workloads were designed using a sample of CRM customer deployments and standard sales force automation usage. Hardware configuration was selected based on the type of infrastructure likely to be selected within an IT organization of 1,000 sales force automation users.

Objectives of the performance test were to,

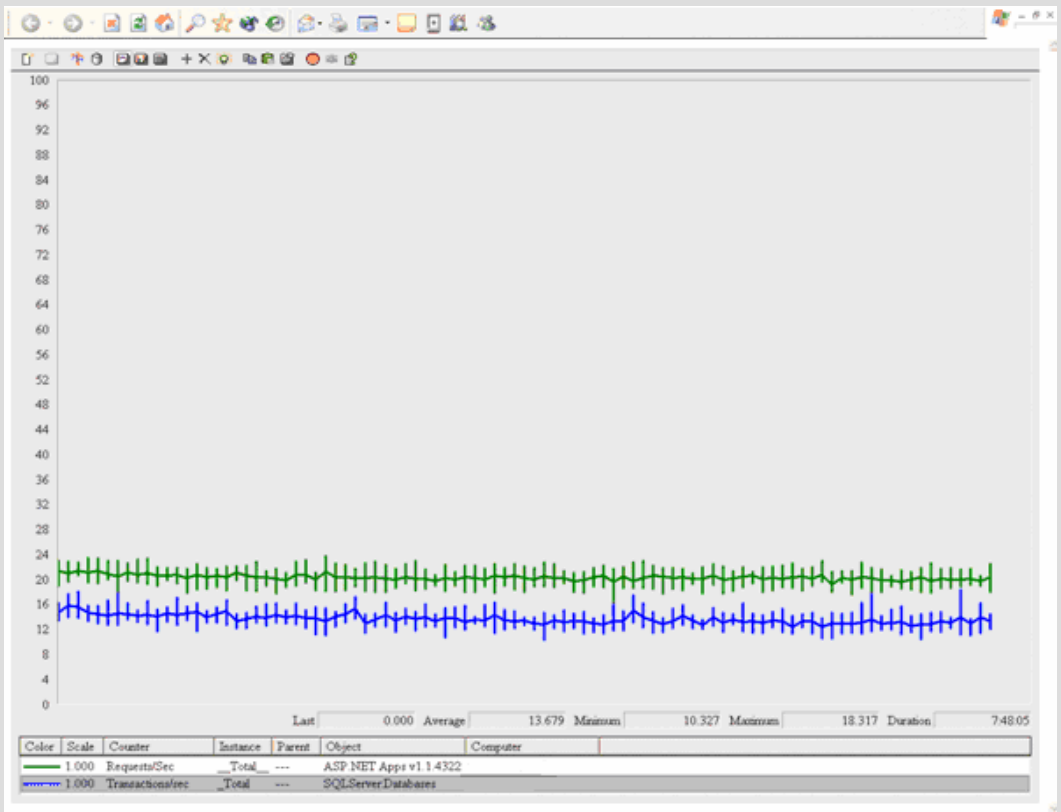
- measure the percentage of utilization (availability) and accuracy (reliability) of CRM application under test throughout the eight-hour test interval.
- record CRM application response under workload scenarios.
- monitor CRM database transactions and CPU utilization on SQL Server database.
- measure CRM web site traffic and CPU utilization on web server.

Test Results

CRM application tested had 1,000 virtual users performing medium-heavy sales force automation workload, over an 8-hour period.

Following data was collected using System Performance Monitor and Microsoft Application Center Test (ACT).

- CRM Web application server had an average CPU utilization of around 9.66 percent leaving roughly 90 percent of CPU available to manage additional processing overhead.
- Average processor utilization of the CRM database server was 60.71 percent, leaving approximately 40 percent of CPU available to manage additional overhead.
- CRM was 99.998 percent accurate within the 8-hour test period.



Test results depict a fixed representation of the hardware environment, workload and network traffic experienced during actual test. Changes to these variables may generate different results with subsequent performance tests.

7. BizTalk MSMQ adapter performance testing

This project was carried out for one of the world's largest software product development companies. The company had developed an enterprise application product. The product connected company's trading partners integrated all the systems and automated business management. Our customer had also developed adapters to facilitate integration, and enhance interoperability with a wide variety of applications.

This project involved testing the performance of the adapter under most basic run-time scenarios, in comparison with the fastest adapter FILE, and protocol MSMQ. Performance determination involved testing the product for its throughput and latency.

The project involved following significant tasks.

- Defining test strategy, test plans, and test objectives.
- Developing test cases based on the product specification/engineering specification.
- Developing stress and performance tools (Load generator, Load controller, Monitor systems CPU and Memory usage).
- Developing and maintaining the batch size for the performance tests.
- Executing the five day iterations of long-haul stress testing.
- Performance benchmarking against the adapters that were already released.

Some of the benefits & achievements of the project were:

- Disha automated about 80% of the test cases and saved 70% of the test-cycle time.
- Disha created exhaustive reports using test logs.
- Disha helped customer to deliver a quality product within the stipulated time.

COPYRIGHT

Contact Information:

For more details visit: www.disha.com

USA Office and Test Lab

Disha Technologies
Bellevue, WA
Phone: +1 (425) 867-3900
Fax: +1 (425) 861-8151

Santa Clara, CA
Phone:
Fax:

India Offices and Test Labs

Disha Technologies
Pune
Phone: +91 (20) 5604-6300
Fax: +91 (20) 2588-7305

Disha Technologies
Hyderabad – 500 081
Phone: +91 (40) 2311-3090/2311-3091
Fax: +91 (40) 5566-2322

Disha Technologies
Bangalore - 560 068
Phone: +91 (80) 2573-8005
Fax: +91 (80) 5110-0587

UK

Vizuri Ltd.
London
Phone: +44 (20) 7014-8907
Fax: +44 (20) 7014-1398
website: http://www.vizuri.ws