Application Performance Testing

TOC

Software Performance Engineering

- Performance Requirements Analysis
- Design Engineering for Performance
- Performance Inspection
- **Performance Testing**
- Performance Profiling
- Infrastructure Performance Monitoring
- Application Performance Monitoring (APM)
- Performance Tuning
- Performance Scaling-up and out

Performance Testing Terminology

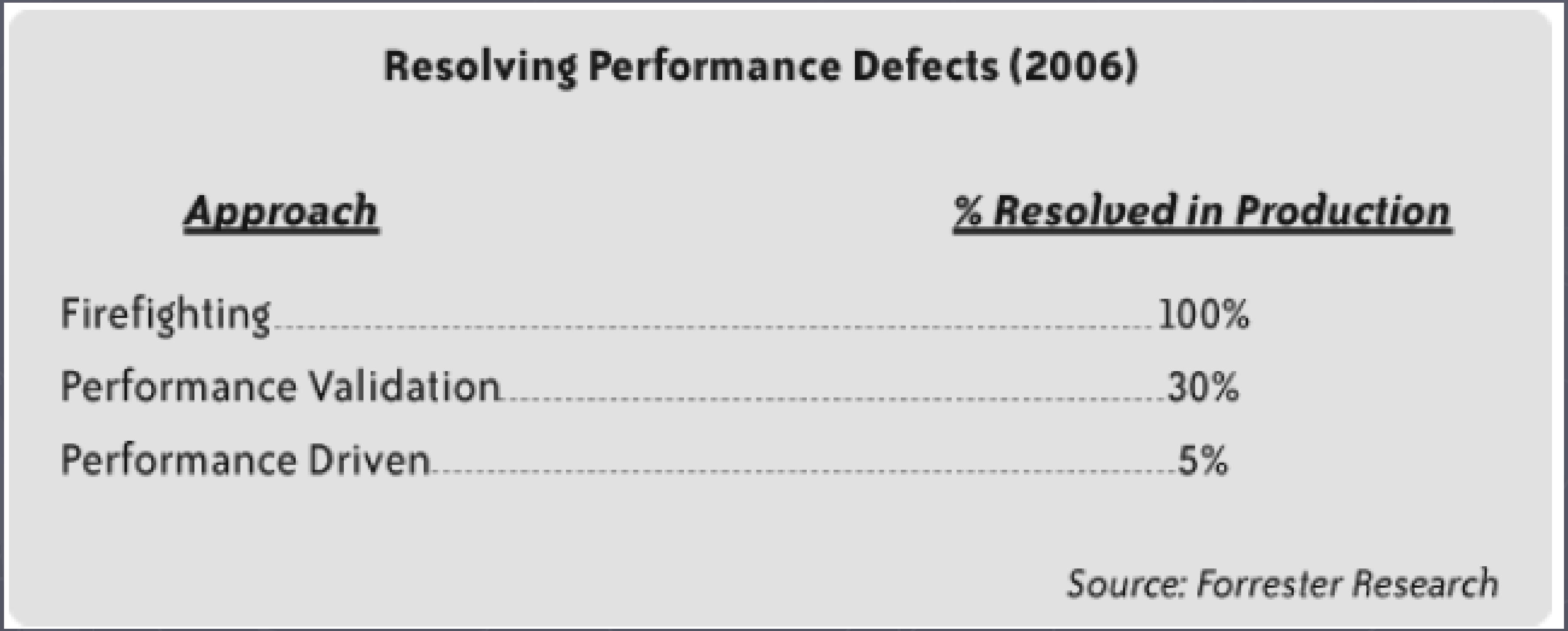
- What is performance testing?
- Performance testing maturity
- Performance requirements
- Performance metrics
- Performance workload

What is Performance Testing?

- Performance testing is in general testing performed to determine how a system performs in terms of **responsiveness** and **stability** under a particular **workload**.
  - It can also serve to measure or verify other quality attributes of the system, such as **scalability**, **reliability** and **resource usage**.

- Performance testing is a subset of performance engineering, which strives to build performance into the implementation, design and architecture of a system.

Performance Testing Maturity



Resolving Performance Defects (2006)

Approach	***% Resolved in Production***
Firefighting	100%
Performance Validation	30%
Performance Driven	5%

Source: Forrester Research

Performance Requirements

- Service-oriented
  - Availability
  - Response time

→ *they measure how well (or not) an application is providing a service to the end users.*

- Efficiency-oriented
  - Throughput
  - Utilization

→ *they measure how well (or not) an application makes use of the application landscape*

Performance Testing Metrics

- Availability
- Repose Time
- Throughput
- Concurrency
- Resource Utilization

Availability

- The amount of time an application is available to the end user.
- Lack of availability is significant because many applications will have a substantial business cost for even a small outage.
- In performance testing terms, this would mean the complete inability of an end user to make effective use of the application.

Response Time

- *The amount of time it takes for the application to respond to a user request.*

- *For performance testing, one normally measures system response time, which is the time between the user's requesting a response from the application and a complete reply arriving at the user's workstation.*

- *Other definitions:*
  - Response Time is the time interval between the last byte of an operation request sent to the SUT and the first byte of the response.
  - Response time = transmission time + processing time + storage access time + wait time + rendering time.

Response Time (Cont.)

- Response Time metrics are usually specified in terms of *average* (mean) and *90th percentile* requirements.

- Although, the average response time is a common measure, it may be insufficient in many cases. This is because there can be wide fluctuations in response times (ranging from 100 ms to 5000 ms, say).

- By requiring that 90% of operations complete within a specified interval, we are assured of a more stable system.

Throughput

- A throughput metric measures how many operations can be processed by the SUT during a unit of time.
- For a single workload driving all operations, throughput (X), can be calculated by keeping a count (N) of all the operations executed during a certain amount of time (T) as X = N / T.
- Use a single operation type that contributes to the throughput metric. This method should be used when one particular operation is dominant.

Throughput (Cont.)

- Use a geometric mean of all different operation types. The geometric mean weights all operations equally. This is the recommended method if there are multiple workloads that may not directly relate to each other and a single metric is desired.
- If multiple workloads are involved, an arithmetic mean of the throughput of all the workloads can be used if the throughputs of the different workloads are related in some manner.

Workload

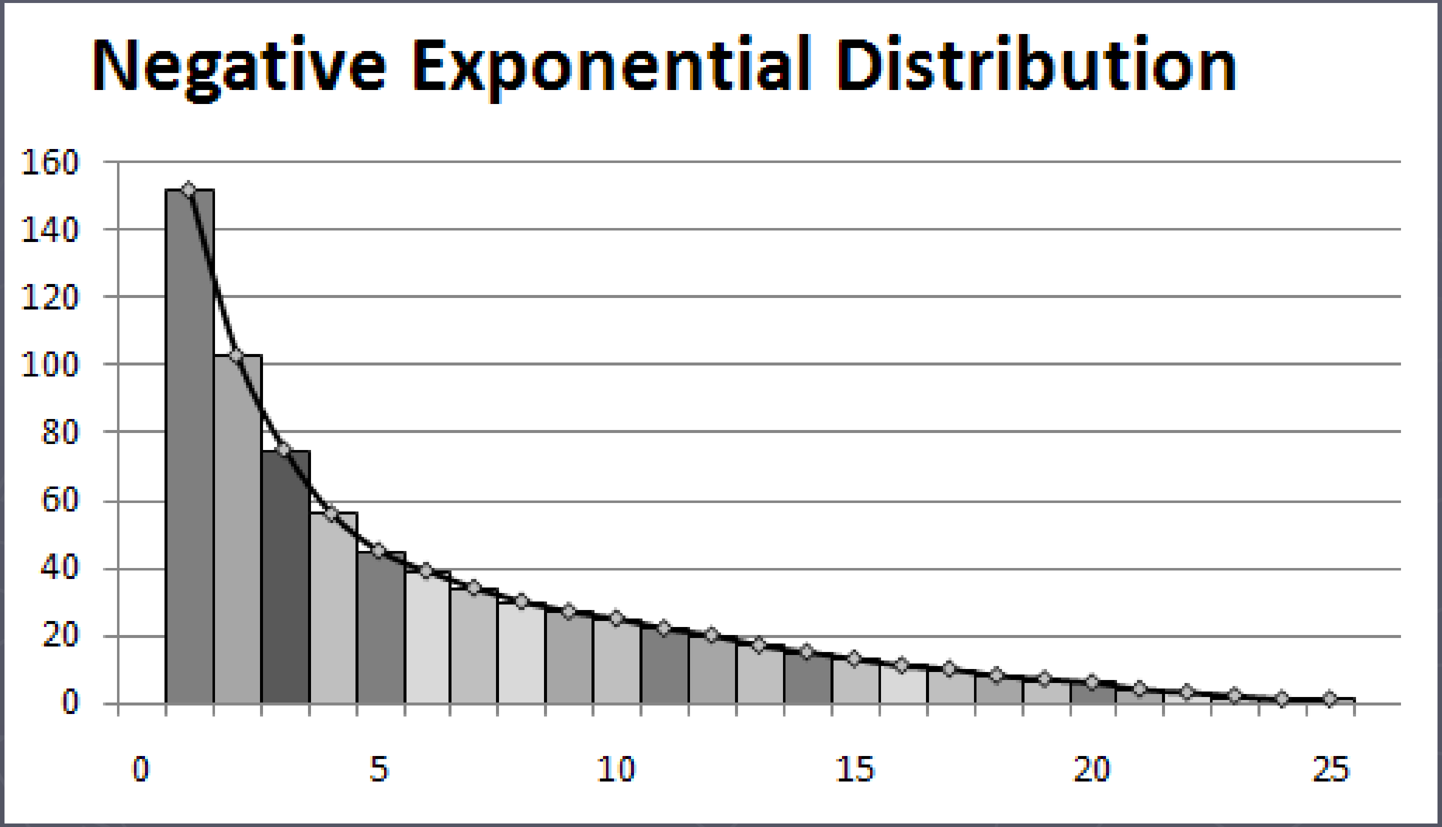
- The relevance of the workload will depend on how closely it emulates the application load in production.

- Workload Parameters:
  - Arrival Rates
  - Operation Mix
  - Think Times
  - Operation Data
  - Others (Initial Data Store, Bandwidth, etc)

Arrival Rate

- The rate at which requests are submitted to the SUT is called the arrival rate and the time between requests is known as the inter-arrival time.
- Arrival rates are used in open systems where the user population is large and unknown, as for example, in web applications.
- Arrival rates are typically modeled using a negative exponential distribution.

Arrival Rate (Cont.)



Think Time

- In benchmarks that model applications involving direct user interaction, it may make sense to model think times.
- The idea is that the user takes some time to respond to what is being presented to him. Think times can vary based on the type of operation.
- If an operation presents lots of data, it might take the user more time to respond.
- Think times are also typically modeled using a negative exponential distribution

Operation Mix

- We need to decide on the frequency with which each of the operations will be exercised.
- This is often expressed as a operation mix% for each type of operation, the total adding up to 100%.
- The mix for any type of operation should not be very small (say < 1%).
- Three different ways for mixing operations:
  - Flat Mix
  - Sequence Mix
  - Matrix Mix

Operation Data

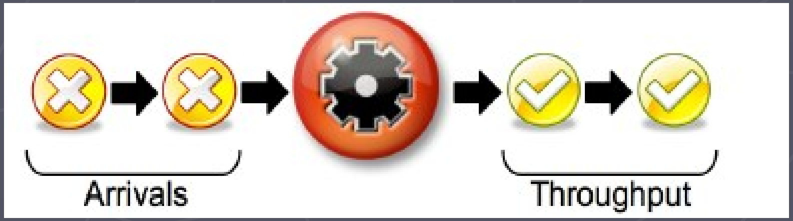
- Depending on the operation, various pieces of data may have to be generated as input to the request.
- It is essential to vary the data used in order to test realistic scenarios.
- For example, if an operation accesses a varying number of items ranging from [1..100], it is a good idea to test the entire range instead of always selecting a fixed number of items.
- It is also good practice to inject a small number of errors by using invalid data as input to the operations. This may catch performance problems in error handling.

Operation Data (Cont.)

- When using a copy of the production data:
  - the workload developer must know the possible values for all of the input fields.
- When using synthetic data:
  - by using algorithms to generate the initial data in the database as well as data generated by the workload, it is easier to keep things in sync.
  - The most common method of generating synthetic data is by the use of random number generators.

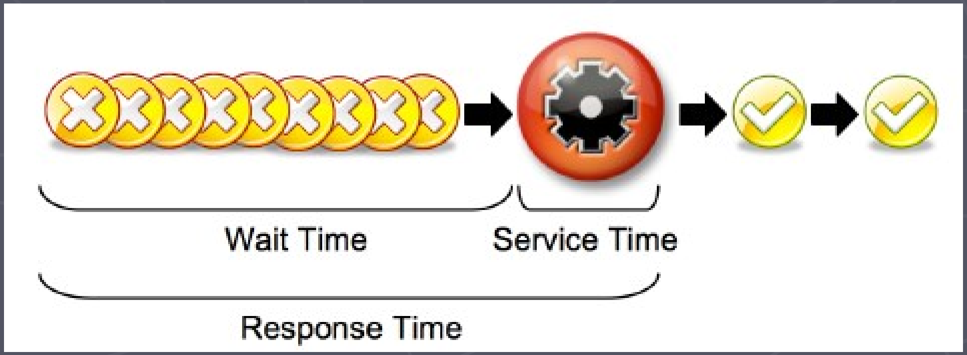
Workload vs. Throughput

- On an idle system, when work shows up, it gets processed right away and exits.
- So early on, as the workload starts to build for the day, the arrival rate of work equals the throughput.



Workload vs. Throughput (Cont.)

- This happy circumstance continues until some part of the transaction path can no longer keep up with the arriving work.
- Now the following things start happening:
  - Wait time starts to build
  - Response Time starts to build (it includes Wait time)
  - The throughput stops matching the arrival rate



Server Response Time vs. Throughput

- Example: move people 10 miles
  - Car: capacity = 5, speed = 60 miles/hour
  - Bus: capacity = 60, speed = 20 miles/hour

- Q: Which is faster?

- <u>Response Time</u>: car = 10 min, bus = 30 min
- <u>Throughput</u>: car = 15 PPH (count return trip), bus = 60 PPH

Server Response Time vs. Throughput (Cont.)

- A is X times faster than B if:
  - Latency(A) = Latency(B) / X

    or
  - Throughput(A) = Throughput(B) * X
- Car/bus example:
  - Response time: Car is 3 times faster than bus.
  - Throughput: Bus is 4 times faster than car.

Performance Testing Process

- Performance Test Planning
- Performance Test Designing
- Performance Test Implementation
- Performance Test Execution

Performance Test Planning

- Identifying business-critical transactions
- Setting realistic and appropriate performance targets
- Making sure your application is stable for performance testing
- Obtaining a code freeze
- Choosing an appropriate tool
- Performance test scheduling and time-allocation

Identifying business-critical transactions

- For each transaction, you should complete the following actions:
  - Define and document each execution step so that there is no ambiguity in the navigation path
  - Identify all input data requirements
  - Determine the type of user that the transaction involves
  - Will this be an active or passive transaction?

Setting realistic and appropriate performance targets

- Availability
- Concurrency, scalability, and throughput
- Response time
- Network utilization
  - Data volume
  - Data throughput
  - Data error rate
- Server utilization
  - CPU utilization
  - Memory utilization
  - Disk input/output
  - Disk space

Making sure the application is stable enough for performance testing

- Undetected application errors
- Poorly performing SQL
- Large number of application network round trips

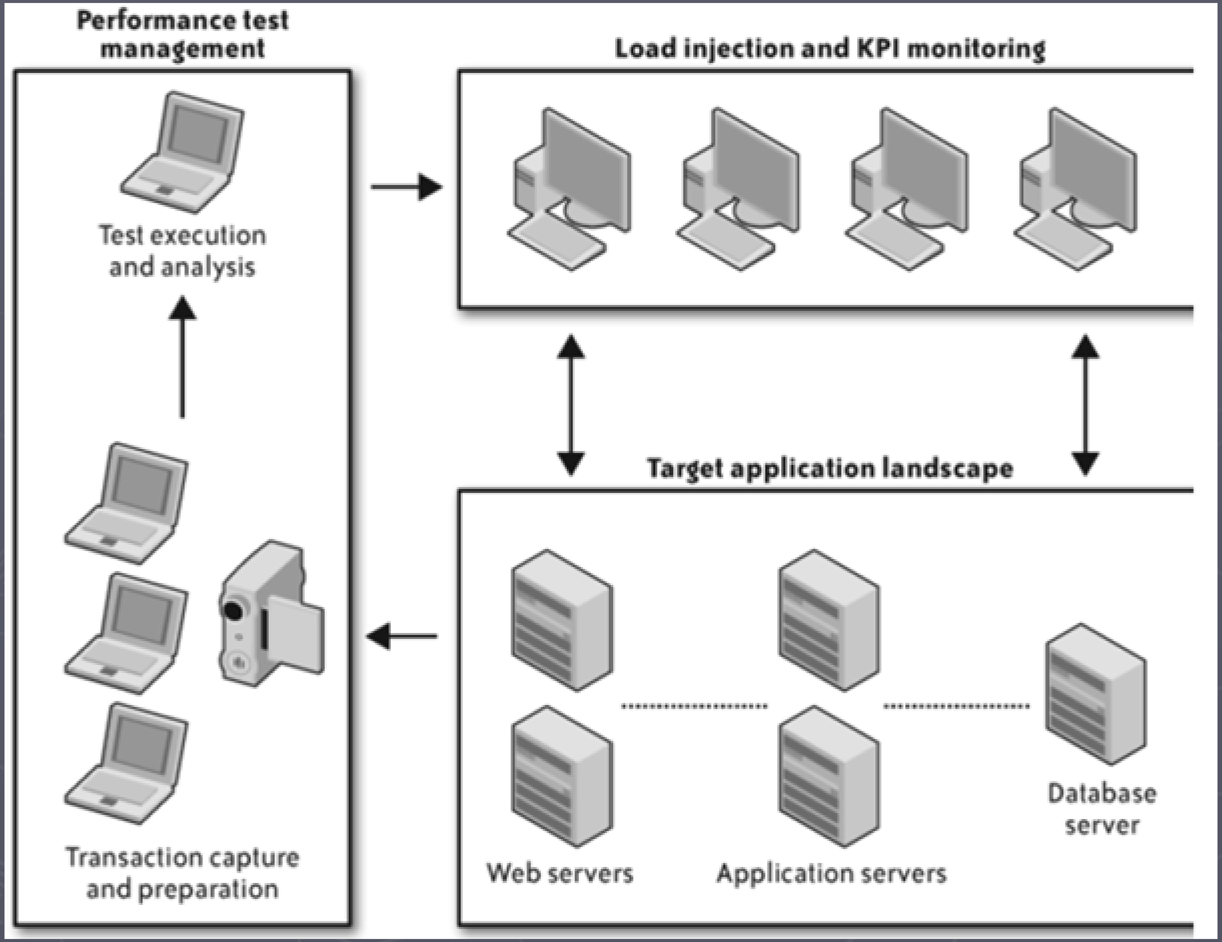
Obtaining a code freeze

- It is absolutely vital to carry out performance testing against a consistent release of code.
- automated performance testing relies on scripted transactions. These scripts are version dependent.
- An unanticipated new release of code may partially or completely invalidate these scripts.

Choosing an Appropriate Performance Testing Tool

- Automated performance test tools typically have the following components.
  - Capturing (scripting) module
  - Controller (management) module
  - Load injector(s)
  - Analysis module
  - Monitoring module

Typical Performance Tool Deployment


Performance test management
Test execution and analysis
Transaction capture and preparation
Load injection and KPI monitoring
Target application landscape
Web servers
Application servers
Database server

Evaluation of a performance testing tool

- Evaluation of a performance testing tool should include the following steps:
  - Proof of concept (POC)
  - Scripting effort
  - Tool vendor support
  - Tool capabilities
    - automated data creation and management
    - response-time prediction and capacity modeling
    - application server analysis to component level
    - integration with end-user experience (EUE) monitoring after deployment.
  - In-house or outsource?

Performance Test Scheduling

- Time to prepare test environment, including load injectors
- Time to identify and script business transactions
- Time to identify and create enough test data
- Time to prepare and execute performance test runs
- Time to deal with any problems identified

Performance Test Designing

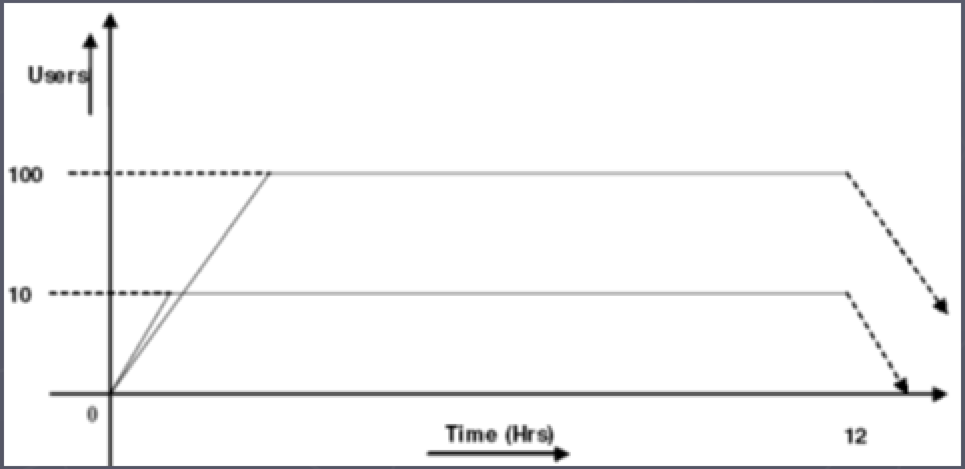
- Types of Performance Testing
  - Baseline Test
  - Load Test
  - Stress Test
  - Stability Test
- Types of Injection Profile
  - Big-bang
  - Ramp-up
  - Ramp-up with step

Baseline Test

- This provides an indication of the best performance you're likely to get from whatever activity the transaction represents.
- You should run this sort of test for a set period of time or a number of iterations.
- There should be no other application activity going on at the same time

Load Test

- up to the target concurrency but usually no further

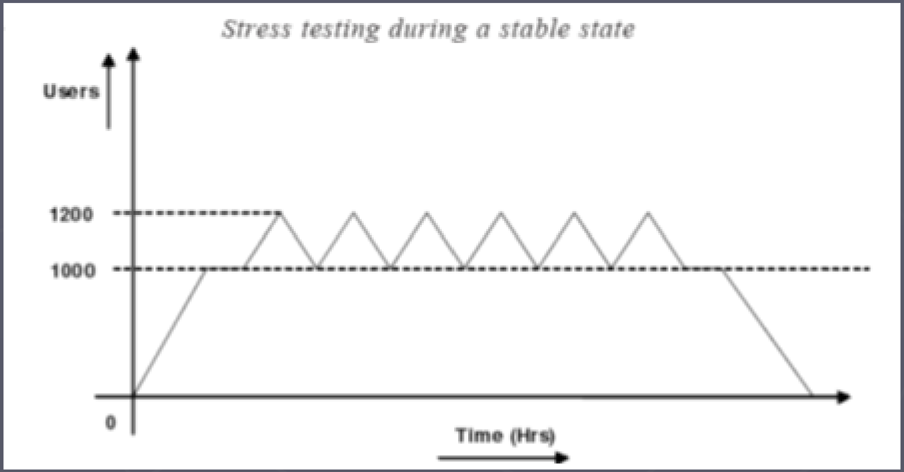


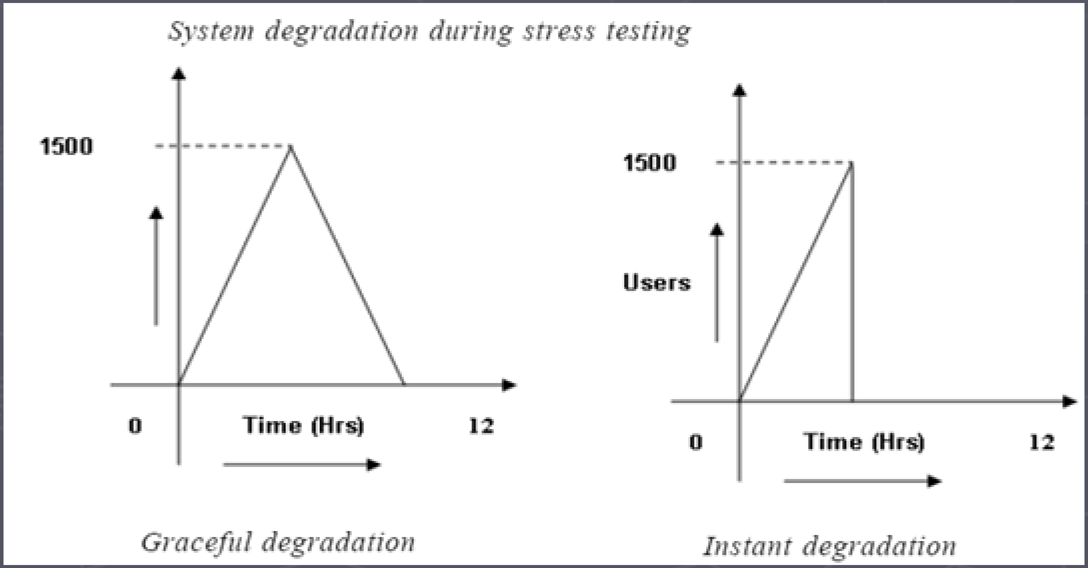
- performance targets
  - Availability
  - concurrency and/or throughput
  - response time
- closest approximation of real application use

Stress Test

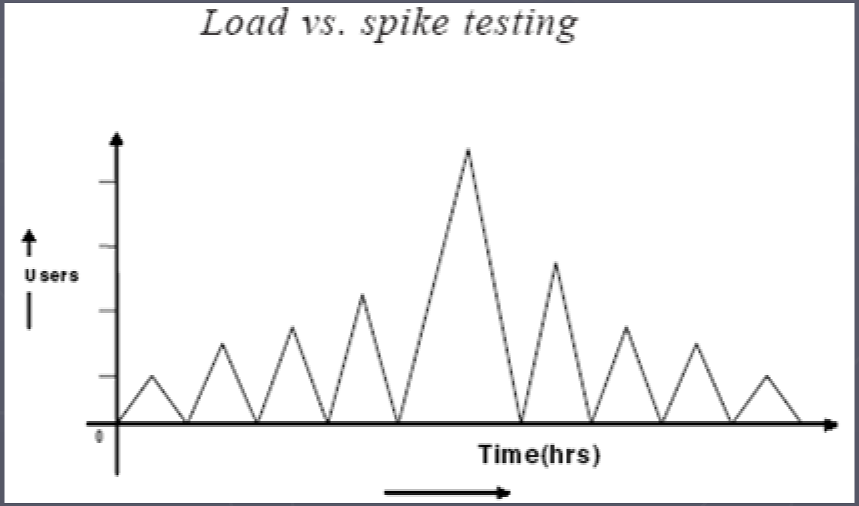
- determine the upper limits or sizing of the infrastructure.
- a stress test continues until something breaks
  - no more users can log in
  - response time exceeds the value you defined as Acceptable
  - the application becomes unavailable
- to know your upper limits

Stress Test (Cont.)



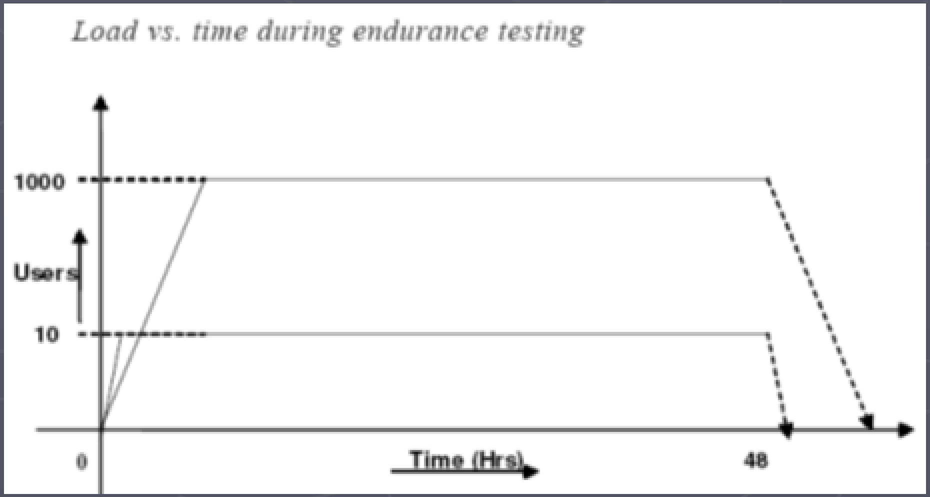


Stress (Spike) test

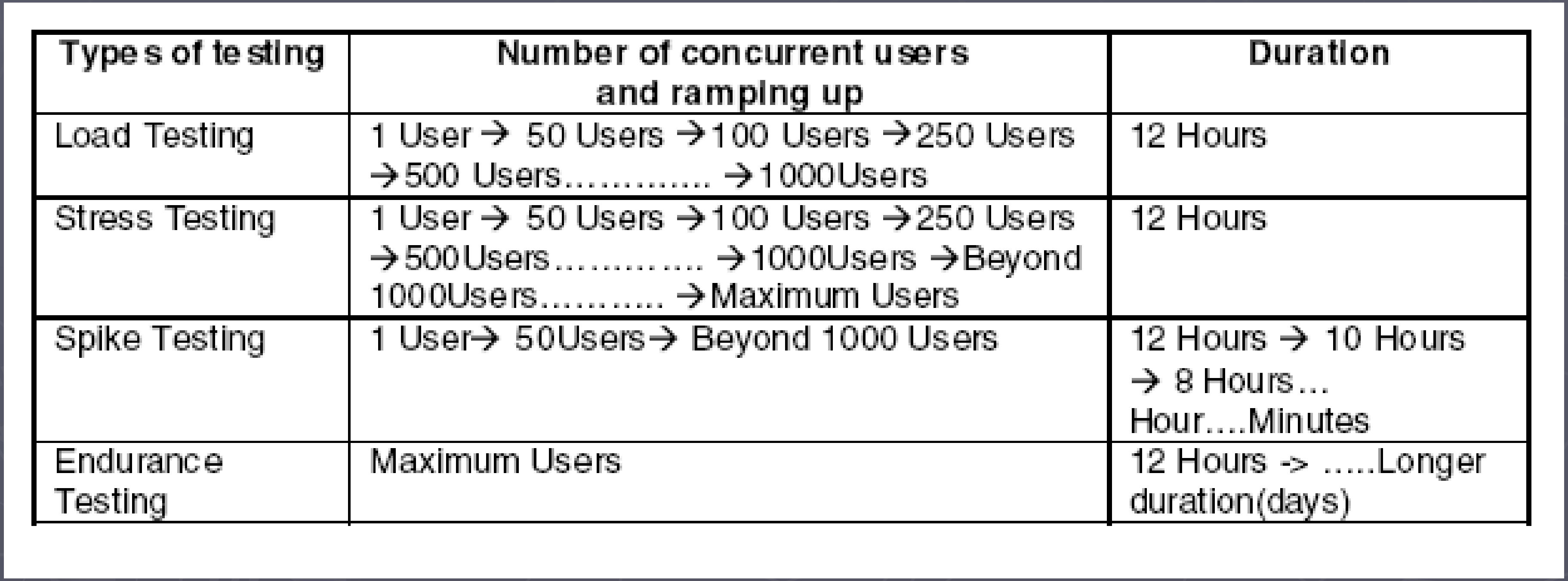


Stability or Soak or Endurance test

- This sort of test cannot be carried out effectively unless appropriate server monitoring is in place.



Performance Test Types Summary



Types of testing	Number of concurrent users and ramping up	Duration
Load Testing	1 User → 50 Users →100 Users →250 Users →500 Users............. →1000Users	12 Hours
Stress Testing	1 User → 50 Users →100 Users →250 Users →500Users............. →1000Users →Beyond 1000Users........... →Maximum Users	12 Hours
Spike Testing	1 User→ 50Users→ Beyond 1000 Users	12 Hours → 10 Hours → 8 Hours... Hour....Minutes
Endurance Testing	Maximum Users	12 Hours -> .....Longer duration(days)

Type of injection profile

- Big Bang
  - Stress test
  - For Big Bang load injection, caution is advised because large numbers of virtual users starting together can create a fearsome load, particularly on web servers. This may lead to system failure

- Ramp-up
  - This mode begins with a set number of virtual users (typically 0 or 1) and then adds more users at specified time intervals until a target number is reached.
  - This is the usual way of testing to see if an application can support a certain number of concurrent users.

Type of injection profile (Cont.)

- Ramp-up (with step)
  - In this variation of straight ramp-up, there is a final target concurrency or throughput, but the intention is to pause injection at set points during test execution.
  - For example, the target concurrency may be 1,000 users, but there is a need to observe the steady-state response time at 250, 500, and 750 concurrent users
- Ramp up (with step), Ramp down (with step)

Performance Test Implementation

- Scripting the business-critical transactions
- Designing an appropriate environment
- Providing sufficient test data of high quality
- Identifying the server and network KPI’s

Scripting business-critical transactions

- Verify single user replay
- Verify multi-user replay
- Decide which parts of the transaction to measure in terms of response time
- Decide which parts of a transaction are repeated during test execution
- Consider whether your application will exist in isolation or have to share resource with other applications.

Designing an Appropriate Performance Test Environment

- Differences between performance test environment and operational environment
  - ✓ The number and specification of servers (probably the most common reason)
  - ✓ Bandwidth and connectivity of network infrastructure
  - ✓ Number of application tiers
  - ✓ Sizing of application databases

3 Levels of preferences when it comes to designing a test environment

- An exact or very close copy of the live environment
- A subset of live environment with fewer servers but specifications and tire development matches to that of the live environment
- A subset of the live environment with fewer servers or lower specification

Some issues about test environment

- When using virtualization
  - Bus Versus LAN-WAN
  - Physical versus virtual NIC
- How application reacts to IP address of each incoming virtual user.
- From where user access the application:
  - Available bandwidth
  - Network latency

A number of ways to simulate WAN-based users

- Modify transaction replay: slowing down the rate of execution
- Load injection from a WAN location: load injector machines at the end of real WAN links
- Network simulation: simulate WAN conditions from a network perspective, including bandwidth reduction

Providing sufficient test data of high quality

- Three types of test data
  - input data
    - User credentials
    - Search criteria
    - *Associated documents*
  - target database
  - runtime data
- Challenges of creation test database
  - Sizing
  - Variety

Identifying the server and network Key Performance Indicators (KPI's)

- It is important to approach server KPI monitoring in a logical fashion—ideally, using a number of layers.

- The ideal approach is to build separate templates of performance metrics for each layer of monitoring. Once created, these templates can form part of a reusable resource for future performance tests

Key Performance Indicators (KPI’s)

- depending on the application architecture, any or all of the following models or templates may be required:
  - ✓ Generic templates
    - Processor utilization percentage
    - Top 10 processes
    - Available memory in bytes
    - Memory pages/second
    - Processor queue length
    - Context switches per second
    - Physical disk: average disk queue length
    - Physical disk: % Disk Time
    - Network interface: Packets Received Errors
    - Network interface: Packets Outbound Errors
  - ✓ Web and application server tier
  - ✓ Database server tier

Key Performance Indicators (KPI's)

- Network monitoring focuses
  - Packet round-trip time
  - Detection of any errors that may occur as a result of high data volumes
  - For the Windows and Unix/Linux operating systems there are performance counters that monitor the amount of data being handled by each NIC card as well as the number of errors (both incoming and outgoing) detected during a performance test execution.

Performance Test Execution

- Things to look out for during execution
- Monitoring infrastructure
- Monitoring application

Things to look out for during execution

- The sudden appearance of errors
  - Some limit has been reached within the application landscape
  - Sudden errors can also indicate a problem with the operating system's default settings
- A sudden drop in transaction throughput
- An ongoing reduction in available server memory
- Panic phone calls from infrastructure staff

Mechanisms to monitor server and network performance

- Capabilities of the performance testing solution
  - **Remote monitoring**
    - monitor many servers from a single location
    - usually, no need to install any software onto the servers
  - **Installed agent**

Remote monitoring technologies

- Windows Registry
  - Microsoft's Performance Monitor (Perfmon) application.
  - Windows operating systems
- Simple Network Monitoring Protocol (SNMP)
- Java Monitoring Interface (JMX)
  - JMX is useful mainly when monitoring Java application servers such as IBM WebSphere, ORACLE WebLogic, and JBOSS
- Rstatd
  - It provides basic kernel-level performance information

Installed agent

- When it isn't possible to use remote monitoring—perhaps because of network firewall constraints or security policies—your performance testing solution may provide an agent component that can be installed directly onto the servers you wish to monitor

Application server monitoring

- Simple generic monitoring of application servers won't tell you much if there are problems in the application.
- In the case of a Java-based application server in a Windows environment, all you'll see is one or more java.exe processes consuming a lot of memory or CPU.
- You may also run into the phenomenon of the stalled thread, where an application server component is waiting for a response from another internal component or from another server such as the database host.
- Java application server's worst-performing SQL calls

References

- The art of application performance testing
- Performance Testing Guidance for Web Applications
- http://en.wikipedia.org/wiki/Software_performance_testing
- The Every Computer Performance Book (http://www.treewhimsy.com/TECPB/Book.html)
- Performance Workload Design (http://faban.org/docs/WorkloadDesign.pdf)
- CIS 501, Computer Architecture, Unit 2: Performance (https://www.cis.upenn.edu/~milom/cis501-Fall09/lectures/02_performance.pdf)

Please Join us at:

Persian Software Testing & Quality Assurance Group in Linked-In

Thanks For Your Attendance

Any Questions?