Top 10 Factors

Of Successful Performance Testing

Presented By: Chris Lawson

Outline

Top 10 Factors of Successful Performance Testing

1. Understanding terminology
2. Architecture for system under test
3. Usage patterns
4. Test scenarios
5. System capacity
6. Tool selection
7. What to measure
8. How long to run
9. Expecting common problems
10. Analysis and reporting results

Scott Barber

“Only conducting performance testing at the conclusion of system or functional testing is like conducting a diagnostic blood test on a patient who is already dead.”

1 Terminology

Understanding the Basics

Think Time

Comparing results against baselines

Protocol

Technology used to communicate across topology

Baselines

Metric data for comparisons

Benchmarking

Comparing results against baselines

Workload

Simulating a usage pattern

Transaction

Grouping of requests for a page or form

1 Terminology

Understanding the Basics

Performance Testing

- Load
- Endurance
- Stress
- Volume
- Spike
- Scalability

A — Normal or expected workload

B — Long running soak test

C — Huge data volume impact

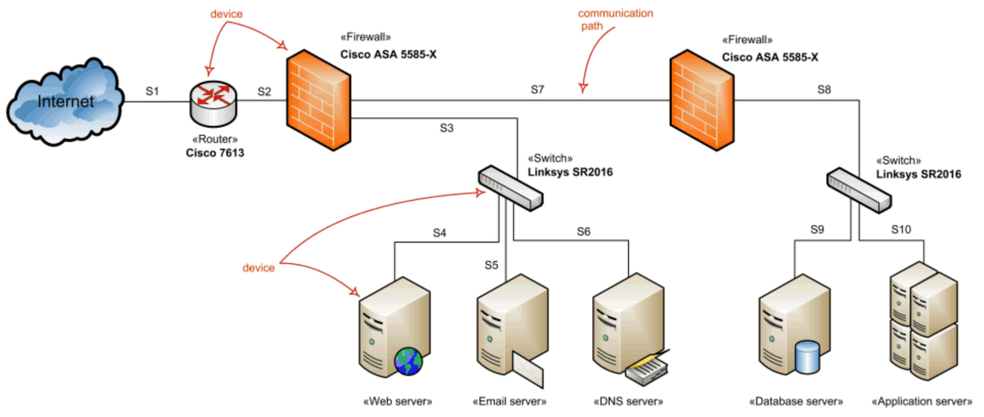
D — Impact of adding resources

E — Sudden traffic burst

F — Exceeding limits to determine breaking point

2 Architecture

System Under Test Topology

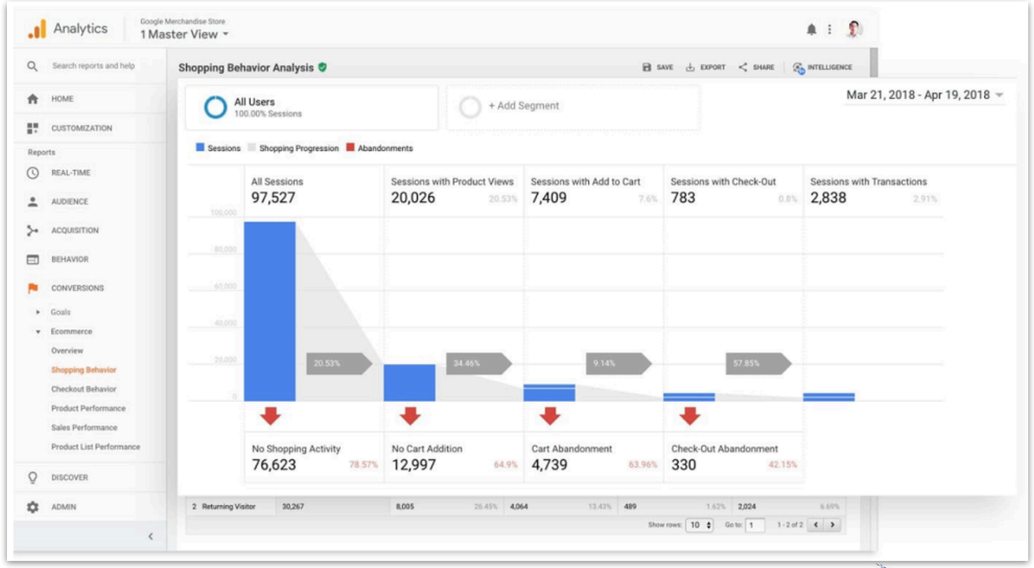


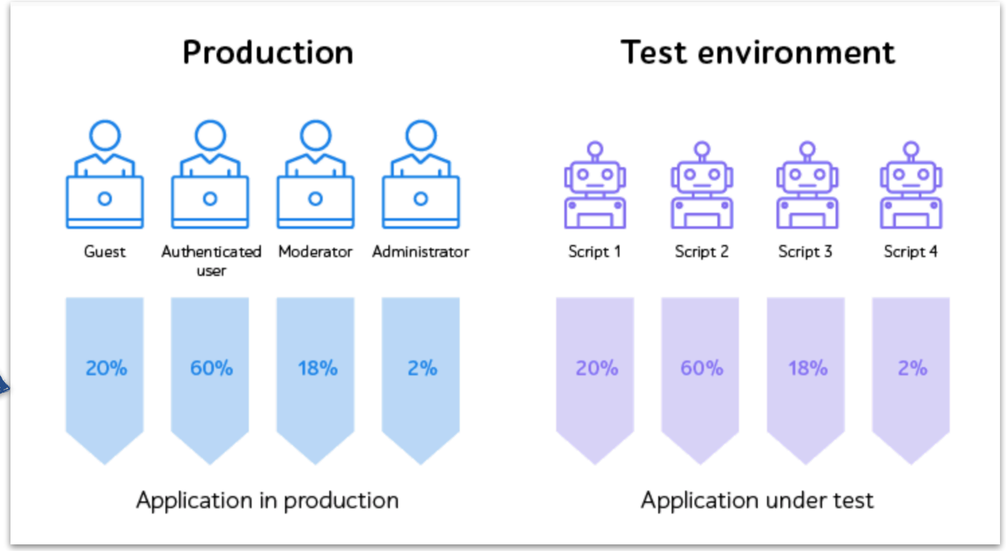
Hardware and Software

Understanding the basics of hardware infrastructure or where software resides, aids in developing scripts and scenarios to fully exercise the system under test relative to its hardware components.

3 User Modeling

Load Profile





3 User Modeling

Load Profile

Approach for Modeling Application Usage

- Identify the objectives.
- Identify the key usage scenarios.
- Determine navigation paths for key scenarios.
- Determine individual user data and variances.
- Determine the relative distribution of scenarios.
- Identify the target load levels.
- Prepare to implement the model.

4 Test Scenarios

Transactions and Workload

Values derived using the following calculations

- Transactions per hour = Number of users*Transactions made by a single user in one hour.
- The number of users = 1600.
- The total number of transaction in the Browse scenario = 17.
- Response Time for each transaction = 3.
- Total time for a single user to complete 17 transactions = 17*3 = 51 rounded to 60 sec (1 min).
- Transactions per hour = 1600*60 = 96000 Transactions.

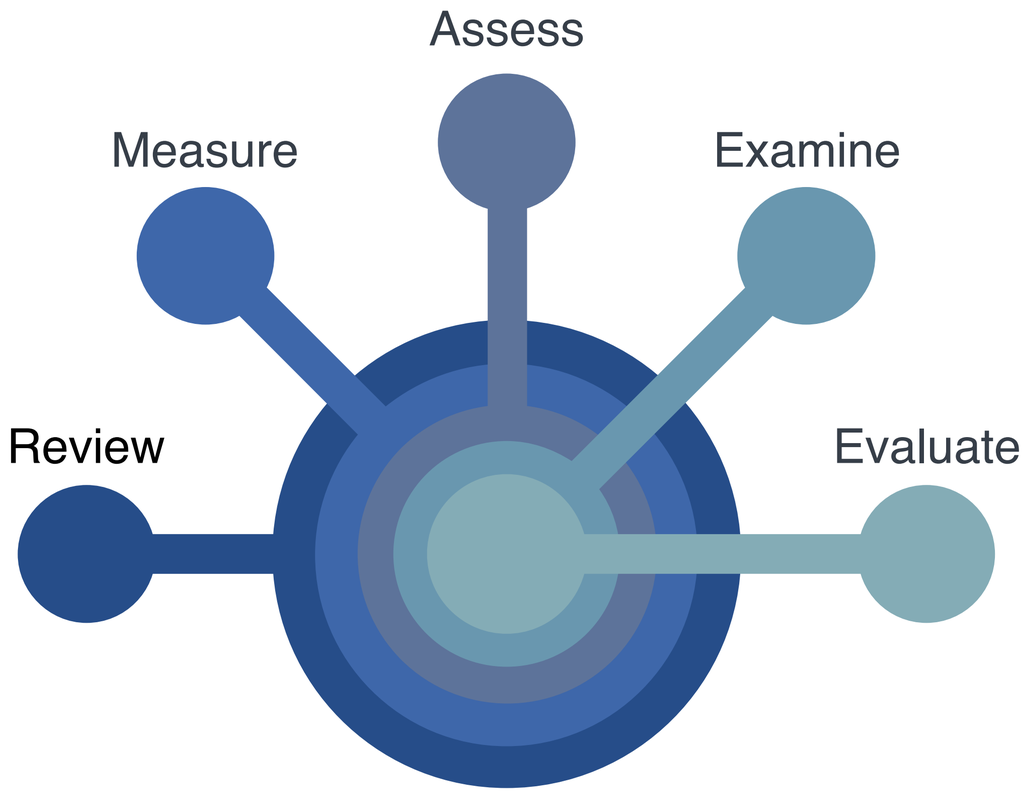
S.No	Business Flow	Number of Transactions	Virtual User Load	Response Time (sec)	% Failure rate allowed	Transactions per hour
1	Browse	17	1600	3	Less than 2%	96000
2	Browse, Product View, Add to Cart	17	200	3	Less than 2%	12000
3	Browse, Product View, Add to Cart and Check out	18	120	3	Less than 2%	7200
4	Browse, Product view, Add to cart Check out and Makes Payment	20	80	3	Less than 2%	4800

5 System Capacity

What the system can handle

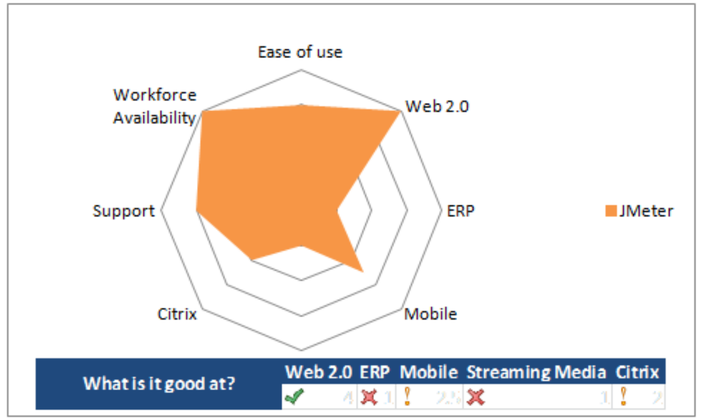
Start small before you go big!

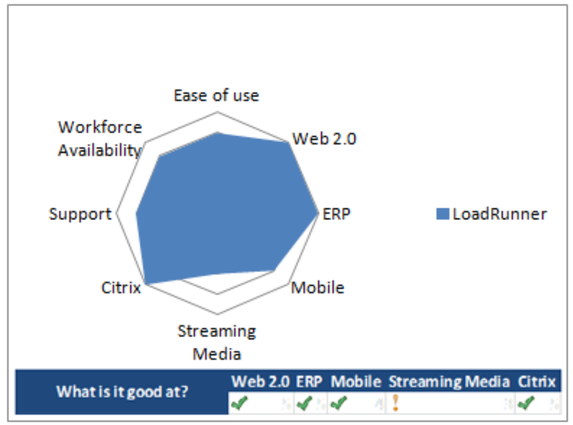
1. Start with running a test with just one virtual user, single threaded for a short period.
2. Next run with multiple concurrent virtual users (5 or more) to determine if users will collide.
3. Continue ramping up until you feel confident that your scenario will run within the defined failure rate.

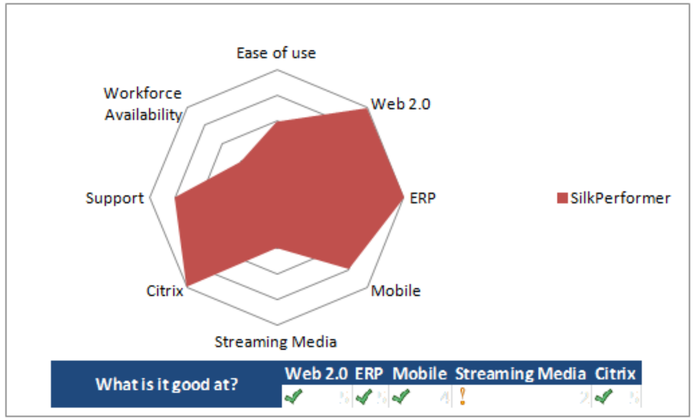


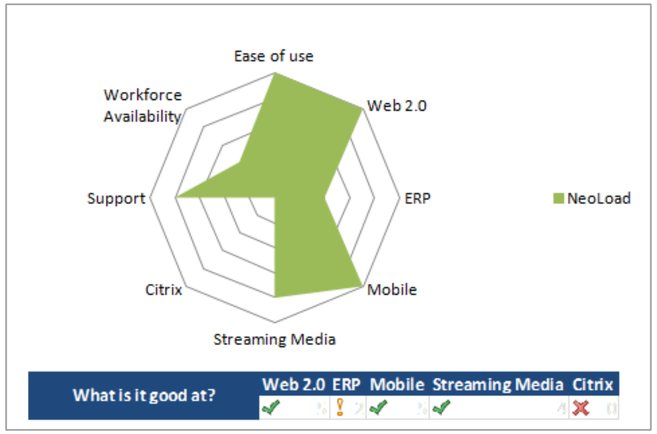
6 Tool Selection

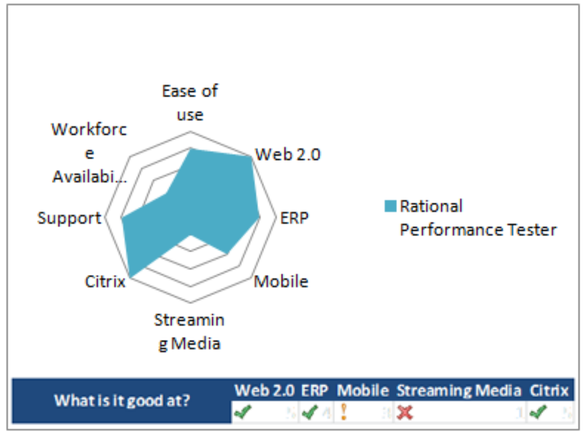
Identify the Right Tool









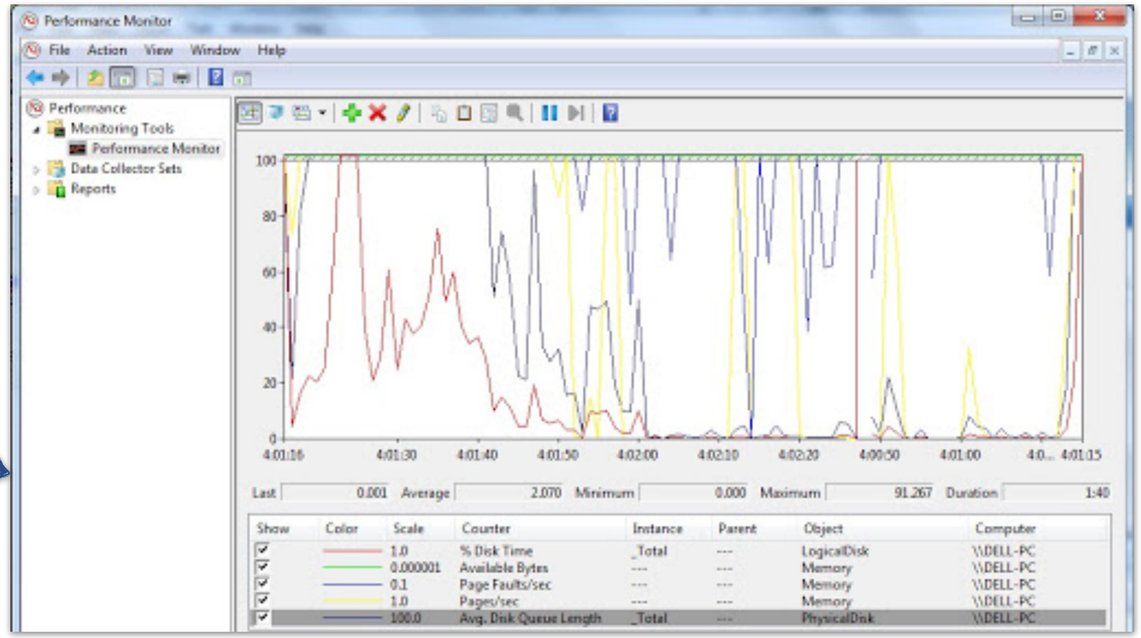


7 Measurements

Metrics to Consider

Performance Testing Metrics Checklist

- Transaction Response Time
- Error Rate
- Concurrent Users
- Throughput
- Requests per Second
- CPU Utilization
- Memory Utilization



8 Duration

How Long to Run

Normal Load Testing Isn't Enough

- Run shorter durations initially to determine if system can handle a normal load
- Once normal load is achieved, consider stress and long soak options

Normal Load Test

Simulates typical user behavior for a short period of time, such as one hour.

Long Soak

Checks general application behavior under a typical load over a significant period of time.

Stress

Attempts to identify the point of failure in a specific system component likely to create a bottleneck or failure by placing an unusually high load on the system.

9 Common Problems

Most Common Observed

During performance testing we are looking for performance symptoms and degradation patterns

- Bottlenecking — This occurs when data flow is interrupted or halted because there is not enough capacity to handle the workload.
- Poor scalability — If software cannot handle the desired number of concurrent tasks, results could be delayed, errors could increase, or other unexpected behavior could happen that affects:
  - Disk usage
  - CPU usage
  - Memory leaks
  - Operating system limitations
  - Poor network configuration
- Software configuration issues — Often settings are not set at a sufficient level to handle the workload.
- Insufficient hardware resources — Performance testing may reveal physical memory constraints or low-performing CPUs.

10 Analysis & Reporting

Reporting Results

Analyze, report, retest.

- Analyze the data and share the findings.
- Run the performance tests again using the same parameters and different parameters.
- Compare to baseline for improvement and/or degradation.

Q & A

THANK YOU!