Apica

Achieve Peak Web Performance

Load Testing 101

BEST PRACTICES

ApicaSystems.com

Stochholm, London, New York, Santa Monica

Interesting Fact

If an e-commerce site is making $100,000 per day, a 1 second page delay could potentially cost you $2.5 million in lost sales every year.

Did you know?

Google found that an extra .5 seconds of in-search page generation time drops site traffic by 20%

Who Should Load Test?

Any and every company with an online presence, especially those experiencing a higher load during Black Friday, Cyber Monday, product launches, marketing releases or other events with higher traffic than normal.

When Should You Load Test?

The simple answer here is whenever possible. If you plan to load test one week or even two weeks prior to a major influx of users, you're probably too late. At this point you don't know what type of problems you're going to find, nor the ability to fix these problems between test and launch.

Performance is a problem for applications on the internet. Surveys show about 75% of internet-facing applications are constantly in a state of degradation – most of which are found by users, not the organization.

Different Types of Bugs

Most people believe unit and functional testing work just fine for their sites. However there are a whole series of bugs that cannot be found with unit and functional tests that load testing can find.

Common Bugs Not Foud With Unit & Functional Tests

Concurrency Bugs: Bugs that only present when the same code is run multiple times at the same time
Compositional Bugs: Bugs caused by the interplay between separate pieces of your application
DB Indexes and Locks: Do the indexes and query locks on your database play nicely at load?
Application or Web Server Configuration: Do your application or web servers exceed memory, socket, or other configuration constraints?
Infrastructure Limitations: Bandwidth, Session Tables, Disk IO, Etc.

Three Most Common Types Of Load Tests

1.Stress Tests: This is basically throwing a very large number of users at your system and seeing if the site can handle it. The purpose of a stress test is to find the absolute theoretical bottlenecks and breaking points for your application. The typical methodology is to just create some basic scenarios and run them as fast as you can against your system and see what happens.
Who uses these tests: Any company trying to find bottlenecks or breaking points

2.Concurrency: A concurrency test has a slightly different aim. It's basically to have more realistic traffic. What you're looking at is the different common scenarios that your users would do on your site and create separate scripts for each of those common usages. Then what you want to do is have realistic pauses and breaks between each run of those scenarios. You want to balance the different scenarios.
Who uses these tests: Companies with various different user journies. i.e. retail companies

3.Disaster Recovery: Basically create sustained load. Typically it would be under the concurrency style with realistic traffic, but the idea here is to just have sustained load over a long period of time, and then test out what happens under different failure conditions.
Who uses these tests: Company with auto-scaling web servers

....Honorable Mentions

Scaling Tests: Using auto-scaling infrastructure pieces inside of your application, scaling tests can allow you to ramp up the number of concurrent users and see how your scaling works.
Who Uses these Tests: A company wanting to understand what the cost is going to be associated with an auto-scaling environment, and how much it will cost to handle certain numbers of users.

Stability Tests: Think of it as a flavor of disaster recovery test, where basically you're just running a large number of users for a very long amount of time, and seeing if your application has any failures over time.
Who uses these tests: Any company trying to determine if there's going to be any time-based failures that may appear in your application.

Load Test Targets

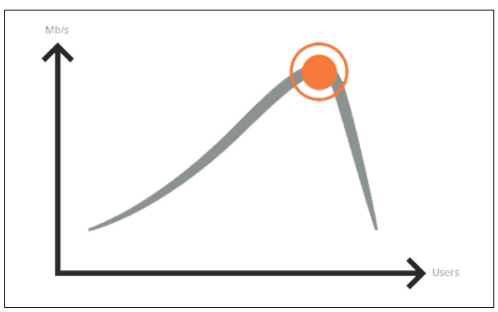
There's obviously many, many things that you could target. Below, you'll find three examples.

Full Site/Application: Most realistic, but difficult in some case to diagnose problems
Single Scenario: Useful for validating critical paths while simplifying troubleshooting
Isolated Function/API: Useful for optimizing individual functions or API's, and is the easiest to troubleshoot when problems are found

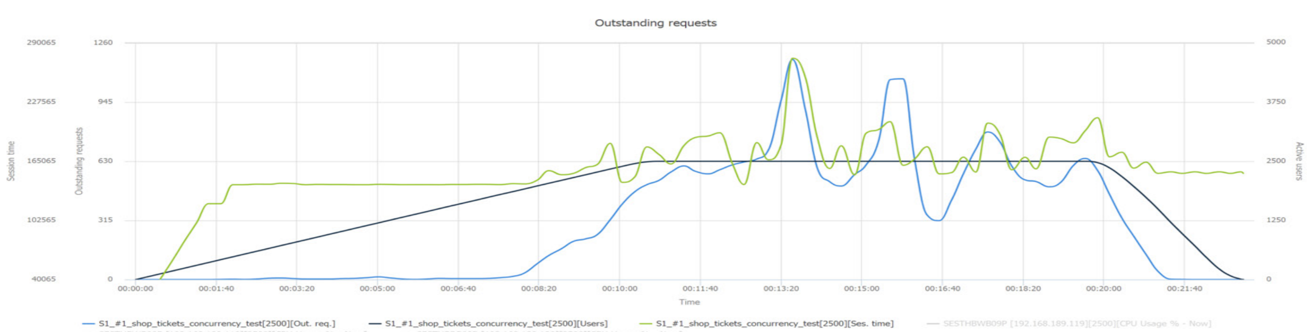
Interpreting Results

When you're running the load test, basically all you need to do to is build a script. The next thing is to run the load test. There's not much to it. You typically just put into either a SAS platform or into an onsite product to run the test.

Best Practice: Doubling - a term used when determining how much load to do. Pick a realistic number of users that's on the very low side of what you'd expect your infrastructure to be able to handle and start with that.

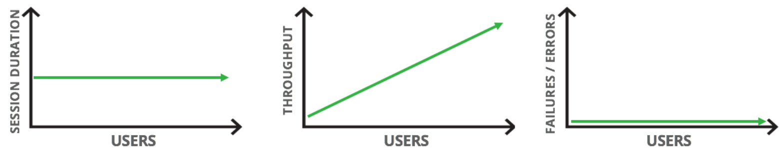


Concurrency Test Example Graph



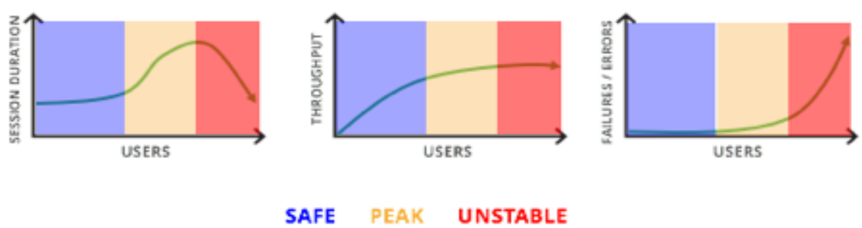
Three most important load curves: Session duration, Throughput, Failures and Errors In Application

Ideal Results



Session Duration: The session duration per user would stay flat because your application, or your infrastructure, would handle every request identically and would never increase in time, no matter how many users you threw at it.
Network Throughput: Goes up linearly and increases with no curve, perfectly linearly. This is because as you add users the amount of network traffic will increase
Application Errors: Has zero errors, so you'd have a flat line at zero on that graph.

Realistic Results



Session Duration: You'll see it's flat, then increases a bit before it falls. This is because when the application can no longer sufficiently handle the request. Average session duration goes down when you see failures come in.

Throughput: As the errors come in, error messages typically take less network traffic than actual responses, so you'll start to see the network throughput flatten out as you transition from valid requests to error requests.

Application Errors: You start to see failures at some point. You're going to get error messages going back. Typically, if you get an error, it's going to be faster than what the normal session would be.

Load Testing at Deployment Stages

Dev. Environments
Good for finding problems early, but unreliable for absolute performance metrics
Catch concurrency, composition, or DB problems

Staging/QA Environments
Catch problems caused by rolled up commits that weren't visible when testing singular changes
Compare to previous runs to get early warnings for reduced performance or peak capacity
May or may not be reliable indicators of production performance depending on the stability of the environment, and similarity to the production environment

Pre-Production
Should be as close to the production environment as possible
Best indication of production performance and load capacity before live deployment

Production
Useful for absolute verification of readiness for expected traffic spikes (e.g. product launces, sales, holidays, marketing events)
Can be used for periodic validations (DR, seamless code deployment, performance validation)

Load Testing - Key Takeaways

- Any and every company should load test - whenever possible
- The most common types of bugs are concurrency bugs & compositional bugs
- The most common types of load tests are: stress tests, concurrency tests, and disaster & recovery tests
- While interpriting information, make sure you pay attention to session duration, network throughput & application performance

To learn more about Apica, visit apicasystems.com

Contact Us

North America: +1 (310) 776-7540
Nordics: +46 (0)8-400 273 27
UK/EMEA: +44 20 8396 4909
info@apicasystems.com