Load Testing

DevOps Advanced

Agenda

- What is Load Testing
- Types of Load Testing
- Disposable Environments
- Best Practices
- Tooling

What is Load Testing

Integration Testing

Load Testing

- Many things can go wrong when a system is in use
- On an average day, the system must run numerous operations simultaneously and respond to different requests from a range of users
- But there could also be a sudden spike in users or a major event that pushes your system to its limits — or beyond
- To prepare for these performance risks, teams use load testing to see how an application or system will perform in a variety of use cases

Load Testing

- But a good load-testing strategy goes beyond just executing a single script
- Different patterns of traffic create different risk profiles for a given application
- For comprehensive preparation, teams must test the system against different types of load testing

Load testing vs. performance testing

- While load testing and performance testing are related, they are distinct types of testing
- Load testing simulates user activity to determine how well a system can handle increased traffic or load.
- Performance testing is an umbrella term for measuring how well a system or application performs overall
- This could include testing for speed, scalability, reliability, and resource utilization in order to identify areas of improvements
- Performance testing includes load testing but also encompasses other types of testing, such as browser performance testing and synthetic monitoring

What type of testing is load testing?

- Load testing is a subset of performance testing that generally looks for how a system responds to normal and peak usage
- You're looking for slow response times, errors, crashes, and other issues to determine how many users and transactions the system can accommodate before performance suffers
- Testers today typically rely on open source load testing tools to test the system with virtual users and simulated data volumes to see how the extra load impacts performance
- They monitor and measure response time, throughput, and resource utilization to detect potential bottlenecks or scaling issues that need to be addressed before the system is deployed in production

What type of testing is load testing?

- When you have a fast software development and delivery process, engineering teams need a robust and reliable testing suite that can keep up with the pace of continuous development and deployment
- Such a testing platform helps teams ensure the high quality of every release
- To make testing reliable, teams should perform different types of load testing across various environments, including development, canary, QA, pre-production, and production
- Teams should also automate testing in continuous delivery pipelines to prevent errors when shipping new features or experiments iteratively to the end-user

Types of Load Testing

Integration Testing

Types of Load Testing

- An application performs differently depending on the volume and duration of traffic it handles at any given moment
- You should never assume your app will perform the same when supporting 10 or 100 users vs. 1,000 or 5,000 users and beyond.
- There are six common types of load testing that you can execute on your applications to measure performance under different loads
  - Smoke test
  - Average-load test
  - Stress test
  - Spike test
  - Breakpoint test
  - Soak test

Smoke Test

- Smoke tests verify the system functions with minimal load, and they are used to gather baseline performance values
- This test type consists of running tests with a few Vus, typically only one
- Similarly, the test should be executed for a short period, either a low number of iterations or a duration from seconds to a few minutes maximum
- This test pretends to be a way to check if the system is working properly not if behaves well with load

Average-load test

- Average-load tests assess how the system performs under a typical load for your system or application
- Typical load might be a regular day in production or an average timeframe in your daily traffic
- Average-load tests simulate the number of concurrent users and requests per second that reflect average behaviours in the production environment
- This type of test typically increases the throughput or VUs gradually and maintains that average load for some time
- Depending on the system's characteristics, the test may stop suddenly or have a short ramp-down period

Stress test

- Stress testing is a popular type of load testing that assesses how the system performs when the workload is heavier than usual
- Stress tests verify the stability and reliability of the system under heavier than normal usage
- Systems may receive higher than usual workloads, such as process deadlines, paydays, rush hours, the end of the workweek, and many other occasions that might cause frequent higher-than-average traffic
- Load should be higher than what the system experiences on average

Spike test

- A spike test verifies whether the system survives and performs under sudden and massive rushes of utilization
- Spike tests are useful when the system may experience events with exceptional traffic volumes
- Examples of such events include ticket sales (Taylor Swift), product launches (PS5), process deadlines (tax declaration), and seasonal sales (Black Friday)
- Spike testing increases to extremely high loads in a very short or non-existent ramp-up time
- In the same way, the ramp-down is very fast or non-existent, letting the process iterate only once.

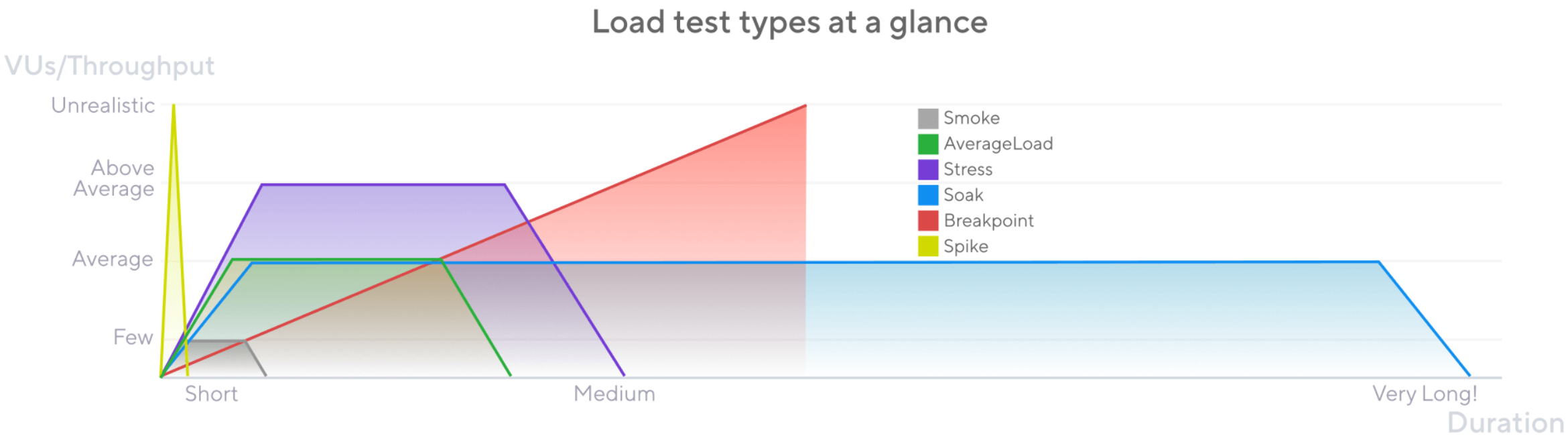
Breakpoint test

- Breakpoint tests discover your system's limits, also known as capacity, point load, and limit testing.
- The reasons you might want to conduct a breakpoint test include:
  - Tune or care for your system's weak spots to reallocate those higher limits at higher levels
  - Help plan remediation steps in those cases and prepare for when the system nears those limits
- It's not only about knowing at what point your system will fail. It's also a test to help determine where and how a system starts to fail and helps teams prepare for such limits.
- A breakpoint test ramps to unrealistically high numbers

Soak test

- Soak tests are a variation of the average-load test where the main difference is the test duration
- In a soak test, the peak load is usually an average load, but the peak load duration extends several hours or even days
- Though the duration is considerably longer, the ramp-up and ramp-down periods of a soak test are the same as an average-load test
- Soak tests focus on analysing the following:
  - The system's degradation of performance and resource consumption over extended periods
  - The system's availability and stability during extended periods

Load Testing Types



Load Testing Types Comparison

Type	VUs/Throughput	Duration	When?
Smoke	Low	Short (seconds or minutes)	When the relevant system or application code changes. It checks functional logic, baseline metrics, and deviations
Average-load	Average production	Mid (5-60 minutes)	Often to check system maintains performance with average use
Stress	High (above average)	Mid (5-60 minutes)	When system may receive above-average loads to check how it manages
Soak	Average	Long (hours)	After changes to check system under prolonged continuous use
Spike	Very high	Short (a few minutes)	When the system prepares for seasonal events or receives frequent traffic peaks
Breakpoint	Increases until break	As long as necessary	A few times to find the upper limits of the system

Disposable Environments

Integration Testing

Where to perform Load Testing?

- To deliver your solution, you rely on several environments
- When doing load testing you want to make them on an environment equal as much as possible with production environment
- To have that environment running 24x7 only to perform load testing you may have big costs
- Using all new tools related with DevOps culture, you may create an environment by request
- This technique is called Disposable Environments

Disposable Environments

- To put this in practice you need to have all your resources creation on a IaC approach
- Having this allow you to create an environment anytime, use it and after that destroy the environment
- This technique allow you to have environment similar with production to run your tests but being as much cost effective as possible
- This is a great example as all new tooling included on DevOps culture can give you great results being cost effective and providing a final solution with bigger quality on any release

Best Practices

Integration Testing

Start with a smoke test

- Before beginning larger tests, validate that your load testing scripts work as expected and that your system performs well with a few users
- After you know that the script works and the system responds correctly to minimal load, you can move on to average-load tests
- From there, you can progress to more complex load patterns.

The specifics depend on your use case

- Systems have different architectures and different user bases
- As a result, the correct load testing strategy is highly dependent on the risk profile for your organization
- What's more, no single test type eliminates all risk. To assess different failure modes of your system, incorporate multiple test types
- Some systems are more at risk of longer use, in which case soaks should be prioritized.
- Others are more at risk of intensive use, in which case stress tests should take precedence.
- A stress test for one application could be considered an average-load test for another

Aim for simple designs and reproducible results

- While the specifics are greatly context-dependent, what's constant is that you want to make results that you can compare and interpret
- Stick to simple load patterns
- For all test types, directions are the same: ramp-up, plateau, ramp-down
- Avoid "rollercoaster" series where load increases and decreases multiple times. These will waste resources and make it hard to isolate issues.

Tooling

Integration Testing

Tooling

- There are several tools available in the market to create Load Testing scripts
- Beside the tools to create scripts there are additional tooling to run the scripts
- The tools to execute the tests are particularly important because depending on the test type you may need to run during several hours
- Now the bigger cloud providers have some resources to allow load tests to run
- Azure Load Testing is one of that resources that allow you to run some defined tests or even Jmeter/Locust scripts

Tooling

- Open-source tools
  - Apache Jmeter
  - Locust
  - k6
- Paid Tools
  - LoadRunner
  - BlazeMeter
  - NeoLoad
  - NewRelic,…

Apache JMeter

- By far the most used open-source tool for load testing and performance measurement
- Supports various protocols such as HTTP, HTTPS, FTP, SOAP, and more
- Pros
  - Free to use
  - Highly extensible with plugins
  - Supports a wide range of protocols
  - Has a GUI for easy test plan creation
- Cons
  - Can be resource-intensive for large-scale tests
  - Steeper learning curve for complex scenarios

Locust

- An open-source load testing tool written in Python, allowing you to define user behaviour with Python code
- Pros
  - Highly flexible and programmable
  - Lightweight and scalable
  - Good for simulating complex user behaviour
  - Distributed testing support
- Cons
  - Requires Python programming knowledge
  - Limited built-in reporting features

k6

- A modern open-source load testing tool developed by Grafana Labs
- It's designed for testing the performance of APIs, microservices, and websites
- It's the "Jmeter replacer"
- Pros
  - Scriptable in JavaScript.
  - High performance and low resource usage.
  - Integrates well with CI/CD pipelines.
  - Good built-in reporting and visualization with Grafana
- Cons
  - Limited protocol support (mainly HTTP).
  - No GUI; requires scripting

Tools Comparison

Feature	JMeter	Locust	k6
Cost	Free	Free	Free
Ease of Use	Moderate	Moderate	Moderate
Protocol Support	Wide	HTTP/HTTPS	HTTP/HTTPS
Scalability	High	High	High
Extensibility	High	High	High
Integration	Moderate	Moderate	High
Reporting	Basic	Basic	Good
Resource Usage	High	Low	Low
Scripting Required	Optional (GUI)	Yes (Python)	Yes (JavaScript)

Automate Testing

Demo