**Testing Engagement: Performance Testing**

Performance testing is a type of non-functional software testing that is used to make sure software systems will operate properly under anticipated workloads. Its objective is to remove performance bottlenecks rather than identify bugs.

**Performance Tech Stack:**

1. Apache JMeter: java, Python
2. Gatling: **Java, Kotlin and Scala**
3. LoadRunner: java, c, JavaScript, and .net
4. k6: JavaScript
5. NeoLoad: JavaScript
6. BlazeMeter: jmx,json,yml,
7. Locust: Python
8. Artillery: Go, Python, Zig, C++

**Objective of Performance Testing.**

1. is to get rid of performance snags.

2. It reveals areas that should be enhanced prior to the product's release onto the market.

3. to quickly produce software.

4. to create dependable and stable software.

5. to assess a system or application's scalability and performance under varied loads and circumstances.

6. It aids in the identification of bottlenecks, performance measurement, and assurance that the system can manage the anticipated volume of users or transactions.

7. It also aids in making the system is robust, dependable, and able to manage the anticipated load in a production setting.

**Performance Testing Attributes:**

Speed: this defines how quickly a software product reacts.

Scalability: It establishes how much work a piece of software can do at once.

Stability: It assesses the software product's stability under various workload scenarios.

Reliability: The purpose of reliability is to assess the software product's security.

**Types of performance Testing:**

1. Load Testing
   * **What :** Is to apply increasing amounts of load on the system and reach a point that where degradation starts.
   * **Intent :** 
     + To baseline the performance parameters like response time, throughput, of the system under normal production working conditions.
     + To identify areas of performance bottlenecks
     + To proactively engage team when there is performance degradation build over build
2. Stress Testing
   * **What :** Is to keep increasing the load so as to break the system.
   * **Intent :** 
     + Is to measure the systems capability to recover after a failure.
     + Is to document the negative impact and risks to system (data corruption, unable to startup etc) post a system crash
     + Is to document the steps to get the system online post system crash
3. Spike Testing
   * **What :** Is to subject the system to sudden bursts of peak loads which a production system might encounter under special conditions like a black Friday sale opening
   * **Intent :** 
     + The goal is to see if all aspects of the system, including server and database, can handle sudden bursts in demand.
4. Endurance Testing
   * **What :** Is to keep the system under test on predetermined sustained loads for extended periods of test
   * **Intent** :
     + The goal is to check for performance degradation due to memory leaks, log file fill up, server errors, GC health etc
5. Scalability Testing
   * **What :** Is to increase or decrease certain parameters of the system Eg. Increasing the thread pool count in DB connection, Decreasing the number of server cluster, increasing cluster server and run load tests
   * **Intent :** 
     + The goal is to check if the system scales and continues optimal performance when resources are added or removed.
6. DB Volume Testing
   * **What :** Is to increase the size of data in DB and run load tests
   * **Intent :** 
     + The goal is to check performance of system and end user experience as the back end systems grow heavy
     + To check if DB CRUD operations take more time as DB size increases.

A diagram of a process

Description automatically generated

**Advantages of Performance Testing:**

1. The system's speed, load capacity, accuracy, and other performances are guaranteed by performance testing.

2. Should something go wrong, it recognizes, tracks, and fixes the problems.

3. It guarantees excellent program optimization and permits several people to utilize it simultaneously.

4. It guarantees the happiness of both the client and the final consumer.

5. Finding bottlenecks: Performance testing aids in locating system bottlenecks including sluggish database queries, memory shortages, or network congestion. By doing this, developers can make the system handle the anticipated volume of users or transactions and can optimize it.

6. Better scalability: Performance testing assists in determining the system's maximum capacity, which helps guarantee that the system can accommodate an increase in users or transactions over time.

**Pitfall of Performance Testing:**

Users may occasionally run across problems with performance in a real-time setting.   
Team members with extensive knowledge should be writing test cases or scripts in the automation tool.   
Team members should be very skilled at debugging scripts or test cases.   
A significant proportion of users may leave in the real world due to poor performance.

**When to Consider/Use?**

1. **Apache JMeter: Apache JMeter:**

Apache JMeter is a versatile performance testing tool suitable for various scenarios. Consider using JMeter as your performance testing tool in the following situations:

1. Web Application Testing:
2. Load Testing:
3. Stress Testing:
4. API Testing:
5. Database Testing:
6. Protocol Support:
7. Integration Testing:
8. Integration Testing:
9. Open-Source and Cost-Effective**:**
10. **Gatling:**

Renowned for its scalability, user-friendliness, and capacity to emulate thousands of users on low-end hardware, Gatling is a potent performance testing instrument.

Think about utilizing Gatling as your performance testing tool in the following scenarios:

1. Highly Scalable Testing:
2. Realistic Load Testing:
3. Continuous Integration and Delivery (CI/CD)
4. Protocol Support:
5. Real-Time Reporting and Analysis:
6. Script Reusability and Maintainability:
7. Open-Source and Community Support:
8. **Load Runner:**

Micro Focus created LoadRunner, a feature-rich performance testing tool renowned for its enterprise-level capabilities, support for several protocols, and sturdy features. Consider using LoadRunner as your performance testing tool in the following scenarios:

1. Complex and Heterogeneous Systems:
2. Large-Scale Load Testing:
3. End-to-End Performance Testing:
4. Integration with Enterprise Systems:
5. Rich Reporting and Analysis:
6. Protocol-Level Testing:
7. Regulatory Compliance:
8. **K6:**

K6 is a cutting-edge, open-source performance testing tool made with an emphasis on flexibility, scalability, and ease of use for developers..

Consider using K6 as your performance testing tool in the following scenarios:

1. Developer-Centric Performance Testing:
2. Continuous Performance Testing:
3. Scalable and Distributed Testing:
4. Realistic Load Scenarios:
5. Flexible and Extensible Architecture:
6. Real-Time Results and Insights:
7. Open-Source and Community Support:

NeoLoad:

Neotys created NeoLoad, a commercial performance assessment tool renowned for its sophisticated capabilities, scalability, and user-friendliness.

Consider using NeoLoad as your performance testing tool in the following scenarios:

1. Enterprise-Level Performance Testing:
2. Complex and Heterogeneous Systems:
3. Highly Scalable Testing:
4. Realistic User Behavior Simulation:
5. Integration with Enterprise Systems:
6. Advanced Reporting and Analysis:
7. Protocol-Level Testing:
8. Regulatory Compliance:
9. **BlazeMeter:**

Cloud-based load testing, performance testing, and continuous testing solutions are provided by the commercial performance testing platform BlazeMeter.

Consider using BlazeMeter as your performance testing tool in the following scenarios:

1. Cloud-Based Load Testing:
2. Scalability and Elasticity:
3. Realistic User Behavior Simulation:
4. Integration with CI/CD Pipelines:
5. Protocol Support:
6. Real-Time Reporting and Analysis:
7. Collaboration and Sharing:
8. Enterprise-Level Support and SLA:
9. **Locust**

Simple and adaptable, Locust is a distributed, scalable, open-source performance testing tool.

Consider using Locust as your performance testing tool in the following scenarios:

1. Developer-Centric Performance Testing:
2. Scalable and Distributed Testing:
3. Realistic User Behavior Simulation:
4. Continuous Performance Testing:
5. Protocol Support:
6. Real-Time Results and Insights:
7. Open-Source and Community Support:
8. Cost-Effective Solution:
9. **Artillery:**

A flexible and developer-friendly performance testing tool, Artillery can be used in a range of situations.

Consider using Artillery as your performance testing tool in the following situations:

1. Web Application Testing:
2. Scalable and Distributed Testing:
3. Developer-Centric Performance Testing:
4. Continuous Performance Testing:
5. Realistic User Behavior Simulation:
6. Protocol Support:
7. Cost-Effective Solution:
8. Real-Time Results and Insights:

**Cost: Yes/No**

1. Apache JMeter: No
2. Gatling: No
3. LoadRunner: No
4. k6: No
5. NeoLoad: Yes(cost need to get in touch with sales team from neo load)
6. BlazeMeter: Yes(cost need to get in touch with sales team)
7. Locust: No
8. Artillery: No

**Ease Of Integration:**

1. Apache JMeter: Yes
2. Gatling: Yes
3. LoadRunner: Yes
4. k6: Yes
5. NeoLoad: Yes
6. BlazeMeter: Yes
7. Locust: Yes
8. Artillery: Yes

**Open source: - Yes/No**

1. Apache JMeter: Yes
2. Gatling: Yes
3. LoadRunner: Yes
4. k6: Yes
5. NeoLoad: No
6. BlazeMeter: No
7. Locust: Yes
8. Artillery: Yes