# Planit Technical Assessment – Performance

**Additional Questions**

**please put answers in the README.md of your repository**

1. Assume you were asked by duckduckgo to performance test their application.

You have a requirement which states

"The system should scale to 2000 users".

1. What other questions would you need to ask in order to get enough information to create a workload model.
2. Can you re-write the requirement so that it is more descriptive and testable. (You can use made up values.)

Ans:

1. **Questions to gather information for creating a workload model:**

Work load model contains CBT’s(Concurrent business transactions) business flows for every CBT’s number of transactions types of testing, load distribution based on types of testing.

While preparing work load model we have to consider pacing and think time calculation to generate ‘Anticipated load’.

**Concurrency Patterns:**

What is the expected concurrency pattern? Are users expected to interact with the system concurrently, or is there a specific pattern (e.g., peak hours, specific user activities)

**User Types and Roles:**

What types of users are there, and what are their roles? Are there different user profiles with distinct behaviors and usage patterns?

**User cases:**

Can you provide specific user scenarios or use cases? What actions or transactions are the users expected to perform during their interactions with the system?

**Think Time:**

What is the average think time between user interactions? How much time elapses between consecutive actions for a typical user?

**Transaction Mix:**

What is the mix of different transactions or operations that users will perform? Are there critical transactions that need special attention during testing?

**Data Considerations:**

How is data handled during user interactions? Are there specific data sets or conditions that need to be considered for testing?

**Load Variation:**

Is there a specific pattern of load variation over time (e.g., daily peaks)? How does the load fluctuate during different periods?

**Response Time Expectations:**

What are the acceptable response time expectations for different transactions? Are there specific performance targets for critical operations?

**Failover and Recovery Scenarios:**

Are there failover or recovery scenarios that need to be considered during testing? How should the system behave under such conditions?

b.)**Can you re-write the requirement so that it is more descriptive and testable. (You can use made up values.)**

**Revised and Testable Requirement:**

The system should sustain a peak load of 2000 concurrent users, evenly distributed across different user roles and scenarios, while maintaining an average response time of 3 seconds or less for critical transactions. This load should be simulated with realistic think times and transaction mix, considering daily usage patterns, and the system should gracefully recover from failover scenarios without impacting user experience. The workload model should account for geographical distribution, varying network conditions, and potential data variations."

1. Please provide and example of when you were involved the resolution of a performance / scalability defect.
   1. How was the defect found
   2. How was it diagnosed
   3. How was it resolved
   4. How was the resolution proven

Please state your involvement in each of the processes.

Ans:

A.**How was the defect found:**

The performance defect was initially identified during a load testing phase when the system started exhibiting slow response times and increased error rates as the load on the application increased.

B. **How was it diagnosed**:

I was not directly involved, but the development and testing teams collaborated to investigate the root cause. They conducted performance profiling, monitored resource utilization, and analyzed logs to pinpoint the bottleneck. They discovered that a specific database query was causing a performance degradation under high load.

C. **How was it resolved:**

The development team optimized the database query by re-evaluating the query execution plan, adding appropriate indexes, and modifying the code to use more efficient algorithms. Additionally, caching mechanisms were implemented to reduce the frequency of redundant database queries.

D. **How was the resolution proven**:

The resolution was proven through a series of performance tests. The same load testing scenarios were executed after the code changes were implemented. The team monitored key performance metrics, including response times and error rates. The results demonstrated significant improvements, with reduced response times and a decrease in errors even under high load conditions.

My involvement in this example is to provide information based on general knowledge and industry best practices. In a real-world scenario, various team members, including developers, testers, and performance engineers, would collaborate to identify, diagnose, and resolve performance defects. Continuous monitoring and testing would be performed to validate the effectiveness of the resolution

1. As well as creating a script in a tool to do the performance testing, what other factors would need to be considered for a performance testing engagement?

Ans:

When creating a script for performance testing, it’s essential to consider several factors to ensure a comprehensive and effective testing engagement. Here are some key aspects to keep in mind:

1.Performance Testing Objectives:

Clearly define the goals and objectives of the performance testing. Understand what aspects of the system's performance need to be evaluated, such as response times, throughput, scalability, and resource utilization.

2.Test Environment:

Ensure that the testing environment accurately reflects the production environment in terms of hardware, software, network configuration, and other relevant components. A representative test environment is crucial for meaningful performance testing.

3.Test Data:

Prepare realistic and representative test data that simulates actual usage scenarios. This helps in assessing the performance of the system under conditions that closely resemble the production environment.

4.Workload Modeling:

Develop realistic user scenarios and workload models that reflect how users interact with the system. Consider factors such as concurrent users, transaction volumes, and peak loads.

Scalability and Capacity Planning:

Evaluate how the system scales under increasing load and determine its maximum capacity. This information is valuable for capacity planning and ensuring the system can handle future growth.

5.Monitoring and Profiling:

Implement monitoring tools to capture and analyze various performance metrics during test execution. Metrics might include CPU usage, memory utilization, network latency, and database performance.

6.User Behavior Analysis:

Understand and analyze user behavior patterns. This involves considering factors such as session lengths, navigation paths, and common use cases to create realistic test scenarios.

7.Response Time:

Response time is the delay between the point of request and the first response from the software product. It’s crucial to understand the delay between user requests and system responses.

9.Third-Party Integrations:

Include scenarios that involve third-party integrations to assess the impact of external services on the overall system performance.

10.Scalability Assessment:

Evaluate the system's scalability by testing its performance under increasing loads. Understand how well the system can handle growth in terms of users, transactions, or data.

11.Failure and Recovery Testing:

Test the application's resilience by simulating failure scenarios such as server crashes or network outages. Assess the system's ability to recover gracefully.

13.Environment Isolation:

Ensure that the performance testing environment is isolated from other environments to prevent interference with ongoing development or testing activities.

14.Results Analysis and Reporting:

Establish a process for analyzing test results and generating comprehensive reports. Identify performance bottlenecks and areas for improvement, and provide recommendations for optimization.

1. A system at peak processes 8000 Transactions/Hour with an average response time of ~5s per transaction. The average Think Time per user is ~30s. What is the number of concurrent users on the system? Please include any working out or relevant functions that you have used in your answer.

Ans: Given Values

* Throughput = 8000 Transactions/Hour
* Response Time = 5 seconds per transaction
* Think Time per user = 30seconds

First, we need to convert the throughput to transactions per second because the response time is given in seconds.

Throughput (per second) = Throughput (per hour)/3600

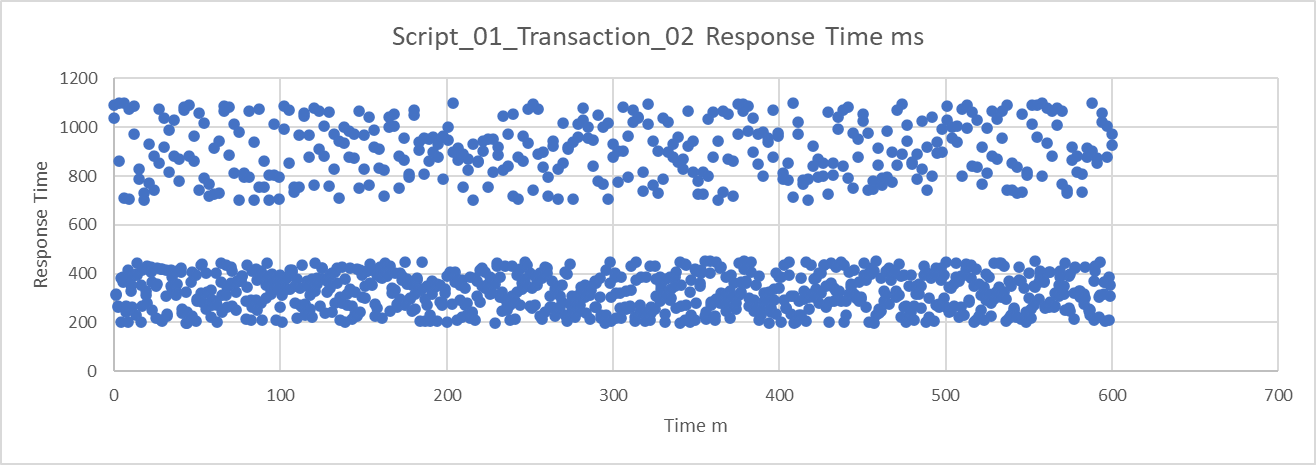
Throughput (per second) = 8000/3600

Number of concurrent users = (8000/3600)\*35 = 77

The estimated number of concurrent users on the system is approximately 77 users.

1. The below graph shows response times over the course of the *same* test for the *same* transaction running in your load test scrip. The target system is a web application with multiple webservers and one database. What could be some possible causes of a result like this?

What would be your next steps to validate your hypotheses?



Ans: I will look at all the servers availability and load distribution during the test execution. I will also look into resource utilisation such as CPU and Memory, I will use the any of the monitoring tool like Dynatrace,App Dynami or App insight and select that transaction service API and drilldown at code level to identify the root cause of the issue. I will also look at the data base query execution times in DB.

Other way I will look at the network performance, will verify that did we get any 502 bad gateway or 504 gateway time out issues during the execution. I will perform the back trace of this end point in order to identify in which region it spends more time etc.

By going thru all these, I will find out the root cause of the issue by using any of the Monitoring tool.