Task 2 Performance Testing

To create a performance test plan, there are a large number of questions that come to mind. What would the goal of the test be? In the task description, “write a performance test to break/load the API”, in my opinion those are two different types of tests, and each would have different qualities that would depend on the service, application, and the customer. In an effort to keep this test plan clear and concise I want to focus on just the “break” load test. The following list of assumptions are made to help me think through the test plan.

The json placeholder api, is a lightweight social media platform which enables people to share experiences.

Assumptions:

- Json placeholder api has already been released. (Production data is available)
- Key performance indicators (KPI’s) have been defined for ‘typical’ user scenarios.
- Json placeholder api has been closely monitored using DynaTrace
- Based on production data, the team understands that the typical user logs in 6 times a day and is active for roughly 10 minutes a day.
- Based on production data, the team understands that 20% of the time a user logs in they are adding photos to their album and leaving comments on other peoples posts in one session.
- Based on production data the team understands that 50% of the time a user logs in they read other peoples posts and comment on a post.
- Based on production data the team sees that 30% of the time users simply read other peoples posts.
- The team has seen usage steadily going up as popularity of application increases, and recently have seen large spikes of traffic up to 100k requests per hour.

Performance Test Objectives

Verify the application can handle user spikes of 150k requests per hour, and response times meet current key performance indicators.

Scope

The scope of the test will be the Json placeholder api hosted in the test environment. The primary metrics that will be collected are response times, error rates and number of pods needed to handle the load.

Methodology

To test the system's ability to handle peak loads, the test will use the three primary scenarios, in a stepped pattern, where each step will last for 5 minutes, until the next step up is taken.

Test Scenarios

Based on the production data collected the top three user scenarios are, reading posts, commenting on a post, and creating an album with multiple pictures.

Scenario One:

Reading posts

Scenario Two:

Reading posts,
selecting a post,
leaving a comment on a post

Scenario Three:

Get all photo albums
Select photo album
Upload medium photo
Upload small photo
Upload large photo

Using a stepped approach, defining each step as 12,500 users added during a 2 minute ramp up time. That number of users will stay constant for 5 minutes in duration.

The test will last 84 minutes or until the application no longer meets KPI's for requests and failure rates.

Of the 12,500 users added,
20% will run Scenario Three
50% will run Scenario Two
The rest will run Scenario One

Each user will run the scenario in a loop for the entire duration.

What this will mean is that users created at the start of the list will run the scenario in a loop for the entire duration of the test, and users created at the last step will only run the scenario for the 5 minute duration.

Test Environment

Using AWS to host Json placeholder api in a kubernetes cluster set to scale out the number of pods.

Tools

To monitor the json placeholder api, dynatrace will be used during the test run to monitor the kubernetes host, and the individual instances of json placeholder api.

To create the load BlazeMeter will be used to run the JMeter tests.

Request metrics will be collected by BlazeMeter in the cloud.

(If the json placeholder api persisted data - A custom tool could be created to verify the correct number of photo albums where created.)