Performance Test Plan for JSONPlaceholder API

General Information

Customer: JSONPlaceholder API

Prepared by: Vepa Orazov

Date: 06/23/2024

Overview

Purpose

Test the performance of the JSONPlaceholder API to ensure it can handle expected loads without issues.

Goal

Check how well the API performs under different loads and identify any weaknesses or breaking points.

Scope

What We Will Test

- API endpoints: `/posts`, `/comments`, `/albums`, `/photos`, `/todos`, `/users`

What We Will Not Test

- Third-party services
- Backend processes not exposed through the API

Key Success Factors

- API should perform well under expected load without errors.
- Measure response times, throughput (requests per second), and error rates.

Risks and Mitigation

Potential Risks

- Delays in getting information
- Changes in testing requirements
- Insufficient infrastructure monitoring

Mitigation Plan

- Plan meetings in advance
- Break testing into smaller parts
- Ensure all necessary details are obtained from the customer early
- Create server images from the production environment for testing. Terminate the servers after testing to save costs.

Resources

Team

- Program Manager
- QA Lead
- QA Team Members

Tools

- **Apache JMeter:** For creating and running performance tests
- **AWS EC2:** For setting up the test environment
- **New Relic/Dynatrace:** For monitoring performance
- **Grafana and Prometheus:** For collecting and visualizing performance data

Testing Strategy

Steps

1. Gather information and initial performance stats.
2. Prepare and confirm the test scenario.
3. Develop and record test scripts.
4. Run initial low-load tests and generate reports.
5. Update test scripts if needed.
6. Prepare the test environment.
7. Execute main performance tests.
8. Generate and deliver reports.

Acceptance Criteria

- Confirmed performance testing requirements
- Access to the application and test data
- Fully configured test environment
- Loaded test data

Test Execution

Main Test Run

- Start with 1 user for 1 minute.
- Increase load by 50 users every minute until reaching 1000 users.
- Maintain each load level for 10-20 minutes.
- Gradually reduce load over 3 minutes.

Test Scenario

Simulate typical usage patterns for the API:

1. **Create Data:** Create new posts, comments, albums, photos, todos, and users.
2. **Retrieve Data:** Retrieve existing posts, comments, albums, photos, todos, and users.
3. **Update Data:** Update existing records in each category.
4. **Delete Data:** Delete records in each category.

This ensures comprehensive coverage of the key API functionalities.

Load Infrastructure

- Use AWS instances to simulate the load.
- Include a load balancer and multiple virtual machines.

Pre-Testing Setup: Create server images from the production environment to mimic real-world conditions. Terminate servers after testing to save costs.

Deliverables

- Test Plan
- Performance testing goals
- Test scenarios and scripts
- Test results and analysis reports
- Final summary report