Network Distributed Computing

Assignment 2 - Capacity Testing a Cloud App & Report

Deadline: 9th April @ 11.55pm
Marks: 25% of Module
Group Project (max. 3)

1 Description

As a member of a quality assurance team, you've been given the task of analysing the performance of a soon to be released application called Cloud Survey. Cloud Survey has passed a number of white-box tests, ensuring functional programme correctness. However, as a black-box software testing engineer, you need to asses the performance quality of Cloud Survey.

2 Application Overview

The Cloud Survey sample demonstrates how to build modern web applications using the latest tools such as ASP.NET MVC 4, Ember.js, and SignalR. This sample consists of a survey designer page, a survey page, and a survey summary page. Each of the pages uses Ember.js to update the UI in real time and perform data binding. The survey summary page also uses SignalR to update the survey results in real-time as surveys are completed.

This sample is designed to be deployed to Windows Azure Web Sites and uses a Windows Azure SQL Database for persistence. You can run this site in either shared or reserved mode on Windows Azure Web Sites with any number of instances. Please see lab 4 for more details about how to deploy and demo the application.

3 Assignment Description

Your task is to conduct a full set of capacity tests on the Cloud Survey .NET application previously deployed in Microsoft Azure. Your capacity testing report should be aimed at developers, and thus, should contain lots of details and analysis. Given that your application is deployed in a multi-tenant PaaS environment, your tests should consider performance at different times (on-peak and off-peak). Please refer to the paper presented in the lecture on the 10th March (available on moodle) for some idea of what is expected for this assignment.

3.1 Deliverable

You must submit a report documenting your tests (e.g.,Introduction, system overview, methodology, test cases, test parameters, etc.), results, and analysis. Guidelines for your report:

3.1.1 Load testing: characterise performance trends for a range of average user loads (e.g., 50, 100, 150, 200, 250, etc).

1. Design realistic test scripts in Jmeter to test 2-3 user interactions, run each test multiple times in order to get the average results across multiple runs (for statistical soundness).
2. Client-side profiling: response time, throughput, errors, etc.
3. Application profiling ***: Use Application Insights to monitor application performance.

3.1.2 Stress testing: Characterising upper performance bound

Determine the threshold when the throughput drops sharply while the response time increases exponentially.

1. Client-side profiling: response time, throughput, errors, etc.
2. Application profiling***: Use Application Insights to monitor application performance.

3.2 Notes

1. *** Given that some usage limitations have been found when using Application Insights to monitor server's health, it is difficult to perform application Profil-

ing. Instead, you are required to perform a deeper performance analysis of the web application focusing only on the metrics that are available in Application Insights. That is, looking at the trends of the number of requests, number of failures, server response time, and the other options that Application Insights offer under ”live metrics stream” (incoming and outgoing requests). Therefore, take into account that different perspectives can be used to report these metrics (e.g., aggregation functions, time ranges, filters, etc.).

2. The Imagine Azure has a strict quota limitation, therefore 3 minutes is an acceptable duration for the individual tests.
3. Testing the app's admin functionality should be a part of your profiling of the app. There could be multiple users using the admin panel to make surveys - think along the lines of SurveyMonkey where a user can either create a survey or take a survey (or both!). Accordingly, you should test this scenario of multiple users using the admin panel when profiling the app.
4. You may get a credentials error in JMeter, to resolve this you will need to add the following two additional elements to your test script:

   - HTTP Cookie Manager
   - HTTP Cache Manager

   They will be in charge of handling the cookies (e.g., for identifying user sessions including login information) and browser cache, respectively. To add these elements, click on TestPlan $\rightarrow$ Add $\rightarrow$ Config Elements.
5. If you receive 404 errors when testing:
   /easy-survey OK
   /Scripts/jquery-ui-1.8.11.js FAIL 404 resource not found
   /api/surveys/ OK

   The following workaround can be used to resolve these errors:

   - In the web app's associated git repo (i.e. on GitHub), create the file ”jquery-ui-1.8.11.js” in / source/CloudSurvey/Scripts/
   - Copy the contents of the following link and paste them into the newly created file: https:// ajax.aspnetcdn.com/ajax/jquery.ui/1.8.11/jquery-ui.js
   - Save your file (and commit changes).
   - Modify /source/CloudSurvey/CloudSurvey.csproj to include: ”⟨Content Include=”Scripts-ui-1.8.11.js”/⟩” after line 204.
   - Save your file and commit changes. You may need to sync your web app after this process however when I was testing this solution, Azure had synced with the repo almost immediately after the commit.

4 Load Testing Tool

Before going into the details of JMeter, let us first understand the jargon associated with the testing of any application.

Performance Test This test sets the best possible performance expectation under a given configuration of infrastructure. It also highlights early in the testing process if any changes need to be made before the application goes into production.

Load Test This test is basically used for testing the system under the top load it was designed to operate under.

Stress Test This test is an attempt to break the system by overwhelming its resources.

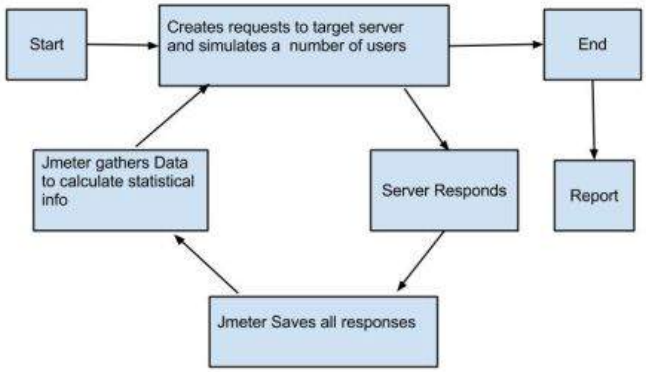

Fig. 1 JMeter Process

4.1 What is JMeter?

JMeter is a software that can perform load test, performance-oriented business (functional) test, regression test, etc., on different protocols or technologies. JMeter is a Java desktop application with a graphical interface that uses the Swing graphical API. It can therefore run on any environment / workstation that accepts a Java virtual machine, for example Windows, Linux, Mac, etc.

4.2 How JMeter Works?

JMeter simulates a group of users sending requests to a target server, and returns statistics that show the performance/functionality of the target server/application via tables, graphs, etc. See Figure 1, which depicts how JMeter works.

4.3 JMeter Video Tutorials

4.3.1 Getting Started with Jmeter

Please copy and paste the urls into a browser window; for some reason the formatting into pdf messes with the underscore in the url. Please make sure that the underscore is in the url you paste into the browser address bar.

https://www.youtube.com/watch?v=dJw8sBk_wSo

4.3.2 Jmeter Browser Recording

https://www.youtube.com/watch?v=zYAiBwJK1v8

4.3.3 Jmeter Errors and Assertions

https://www.youtube.com/watch?v=SVxB3Tk4O4A

4.3.4 JMeter Login to a Web App

https://www.youtube.com/watch?v=hGkrSFKcj10