COMP205P

Microsoft Capita Team 2 / Bi-Weekly Report 6

Date: 27/12/2016
Team: Lambros Zannettos, Nathan Liu, Junwen He

Sprint 1

Overview

These past weeks have been very productive both in coming up with new ideas and in prototyping those ideas and testing them out.

Meeting summary

Thu, 12 January 2017
This meeting with the client and Team 1 was a very productive one. James explained his thoughts on the project and his ideas regarding the solutions to the problems at hand. We discussed the overall design of the solution and the desired outcomes.

Wed, 18 January 2017
A short meeting with Dr Strange to catch up with both the Teams' progress and any possible issues.

Mon, 23 January 2017
A group work session. We worked on different things including zipping data streams before transferring to front end (to reduce transfer times), different data structures for storing sets in memory (Dictionary/HashSet/List etc.) amongst other things.

Tue, 24 January 2017
Syseng lab session. Shared our progress with our TA and worked on our individual tasks.

Thu, 26 January 2017
We discussed the current solutions we are each working on, and talked about possible improvements for the next sprint. We also run a few experiments (regarding speed of set operations) and looked at and improved our overall system design, to help us think about the problem better.

Fri, 27 January 2017
Another group work session. We worked on uploading data (for example sets or subsets created by users) from memory to a permanent Blob storage on Azure.

Tasks Completed

- REST Endpoint for dummy data for Team 1.
- REST Endpoint for list concatenation (i.e. Basic version of the Interpretation Engine).
- Experimented with Microsoft Enterprise Library (Data Access Application Block) to provide distribution-independent database access for the system.
- **Proof of concept**: Creating large sets in memory (around 300,000 elements each)

and performing set operations using the C# HashSet data structure is really fast, even when running in a VM.
- **Proof of concept**: Found a way to reduce size of data transfers by zipping the data before sending. This can also be done in-memory using MemoryStreams to improve speed of delivery. Made prototype of this.
- **Proof of concept**: Writing data from memory to Azure storage (blob storage), for more permanent storage.
- **Set Engine:** We created a basic set engine to take in any database and output a list of Sets in a structured way and we are now running this set engine on a scheduler every week.
- **Interpretation Engine:** We created a basic interpretation engine to parse GET requests and retrieve the correct sets accordingly. We linked this up with the REST service and data generated from the set engine to make a basic version capably of feeding Team 1 with more realistic data.

Problems to be resolved

- **Proof of concept:** A faster way of transmitting millions of datapoints from server side to client side is to use UDP instead of TCP. Although data may be lost over UDP such a high volume of data can guarantee enough data will get through without problem such that the data received is representative of what would have been sent through TCP.

- **Create trees of Set GUID and Set Operations**: This is an idea which aims to help the speed of delivery of sets, and the elimination of repetitively creating the same sets if they already exist, either in memory or in storage. The idea is that every time a new set operation is completed successfully, the GUID of the resulting set along with the set operation (which will include its parent sets) which was performed to create it will be saved in a tree-like data structure. Nodes are tuples of <Set, Set operations>.

- **Structure data on the Azure Storage Blob:** We have moved data outputted from the Set Engine from the Windows VM to an Azure Storage Blob. Currently data on the Azure storage blob is composed of several lists of the output produced from the Set Engine. We will need to structure the data in order for it to work with the Interpretation Engine.

Plan for next two weeks

- Create trees of Set GUIDs and Set operations and test the feasibility of this system as a way to make the system more efficient.
- Implement the use of UDP protocol to transfer the data back to the front-end, and compare speeds and reliability with current methods.
- Optimise existing Set Engine, Interpretation Engine and REST Service.

Individual reports

Lambros Zannettos:

Created REST API for Team 1 to get dummy data from in a form similar to that discussed with James. Started experimenting with MS Enterprise Library (Data Access Application Block) and ADO.NET Entity Framework, with the goal of abstracting the solution from specific DB providers. I also run set operation experiments with dummy data using C# HashSets, which provide a huge performance improvement in relation to other types of Lists. Lastly, I serialized those HashSets into Json and then stored them online in an Azure storage account, to test the feasibility of this as the storage solution for the system.

Nathan Liu:

I created a basic set engine that will take in any MySQL database and produce organised sets as output. I also made the Interpretation engine that parses GET requests from the REST service and retrieves the correct sets accordingly. I made further additions to the REST endpoint adding support for list concatenation. I wrote several general purpose libraries including a DB connector and a file navigation library that were used to help develop the Set Engine and Interpretation Engine.

Junwen He:

Since the data we are going to transfer is basically txt files, so to reduce the transfer time, we decided to use zip to compress the files before sending them. And I’ve looked into the C# Zip and MemoryStream library, so that I can compress files in memory. Besides that, I also looked into C# UDP services since we need to send these files, and UDP is faster than other protocols.