Project 1 Reflection – Eoghan McGlinchey 18300093

Q1: Give your opinion on the suitability of CLI (Bash) for some Big Data tasks

Bash, like most programming languages has its advantages and disadvantages. Its main advantages can be seen when it comes to manipulating data and file systems as it does this in an efficient and quick manner. As well as that, programs can be executed using a bash script, which enables the user to use many of the best features of Bash such as their functions 'cut' and 'sed'. The command line interface also allows for directories to be read, made and removed with ease. However, one disadvantage of the command line interface is that there is almost no use for the mouse – the user must use the arrow keys to navigate lines of code or text. Another disadvantage of Bash is that it is not object-oriented and does not have the built-in functionalities and data structures that other languages which are used for data collection and analysis such as Python have to offer. The command line interface also doesn't allow for the data visualisations that other interfaces such as Jupyter Notebook offer. CLI (Bash) is therefore most suitable for tasks that require data collection or analysis that is not complicated and tasks that do not require visualisations.

Q2: Compare relational and NoSQL database management models and in particular give your impressions on why one is better than the other and in which context

In order to compare relational and NoSQL database management models, we must take the task which is at hand into account. This is vital because it isn't fair to simply say that one database management model is better than the other – both types of database management models have their place and are useful in their own right.

Firstly, it is worth noting that the relational database management approach is the most popular. One of the main advantages of this approach is that the relational model ensures data integrity through constraints, that is, the accuracy of the data is ensured by preventing the deletion of a record if the primary record in the main table has not been deleted. As such, using relational models means that no records in a table are without a primary record in the main table. However, preserving the integrity of the data comes at a cost. This means that it is quite difficult to scale relational databases horizontally. In comparison, it is very easy to scale a NoSQL database horizontally which means that vertical scaling can be avoided which in turn means that the existing server or hardware will not have to be improved. The fact that NoSQL database management models aim for eventual consistency means that the availability of the data is improved. So in this context, it's fair to say that when dealing with normalised data that fits into a predefined schema, relational database models are better suited.

As mentioned above, NoSQL database management models aim for eventual consistency and as such it is easy to scale horizontally using this management model. This also allows for better flexibility than relational database management models. This makes NoSQL a better choice for storing data which is not normalised. Perhaps one disadvantage with regards to this is that it is up to the user to decide how best to organise and access the data. So in this context, when it comes to dealing with data that is not normalised, NoSQL database models are better suited.

In conclusion, when it comes to deciding which database management model is better, the user must keep in mind the type of data they wish to work with. The way in which the data integrity is preserved by both database management models is something that must be taken into account. If the user knows that the data is structured and if the data is small in size then perhaps it is better to use a relational model. On the other hand, if the data is not structured or normalised and the user is capable of organising and accessing themselves then NoSQL is probably the better option.

Q3: Write a short report on one of the research papers that are available on Brightspace

The research paper I have decided to report on is entitled "Cassandra – A Decentralized Structured Storage System". The challenge the paper addresses is that when running on top of infrastructures of hundreds of nodes, both small and large components of structured storage systems fail. The paper claims that Cassandra manages this in a way that "drives the reliability and scalability of the software systems relying on this service". The motivation for this research stemmed from the fact that because Facebook had so many users ("hundreds of millions … at peak times") and as such there was always a "small but significant" amount of components that would fail at any given time. Facebook also needed a system to handle "billions of writes per day". It was because of this that Facebook decided to develop Cassandra, which would "treat failures as the norm rather than the exception".

There were many solutions to which the authors compared themselves to. Among them were Ficus and Coda which could "replicate files for high availability" however, the limitation of these solutions was that they did this "at the expense of consistency". Another solution was Farsite, which could achieve "high availability and scalability using replication". The Google File System could split the data into chunks and store it in chunk servers and had recently "made fault tolerant using the Chubby abstraction". Bayou was a solution that allowed disconnected operations and could provide eventual consistency. Although all these solutions guaranteed eventual consistency, these solutions were "limited in scalability and availability". The solutions were not equipped to handle network partitions as a consequence of providing strong consistency guarantees.

The solution proposed by the authors involved a data model, API and system architecture. The main concern of the data model was the way in which the tables were designed. Tables in Cassandra were indexed using a key with an object value. It also allowed "every operation under a single row key" to be automatic per replica. It also separated Simple and Super column families. Columns could also be sorted by time or key name. The API consisted of three methods; insert, get and delete. Each method took a table and a key as arguments, while the insert method's third argument was different in it being a row mutation, while the other two method's third argument was the name of the column. The system architecture involved partitioning, replication, membership, bootstrapping, scaling the cluster, local persistence and implementation details.

The author's first experiment involved indexing 7TB of data for over 100 million users, then storing it in a MySQL infrastructure before loading it into the Cassandra system. This experiment involved running Map/Reduce and exposing background channels for the Map/Reduce process to aggregate the reverse index per user and send the serialised data over to the Cassandra instance. The outcome from performing this experiment was that the author found out that "the Cassandra instance is only bottlenecked by network bandwidth". Another experiment involved various implementations of Failure Detectors. It was found from this experiment that the time to detect failures increased beyond an acceptable limit as the size of the cluster grew. However, with the accrual failure detector with the PHI set to 5, the average time to detect failures was about 15 seconds. Then there was the Facebook Inbox Search experiment in which Cassandra provided hooks for intelligent caching of

data. The experiment found that when a user clicks into the search bar, an asynchronous message is sent to the Cassandra cluster to prime the buffer cache with that user's index. In this experiment the read performance was also measured in which it was found that for the maximum latency, the time for search interactions was 26.13 milliseconds and the term search was 44.41 milliseconds.

I found the idea behind this research paper to be very captivating. I really appreciated the overall structure of the paper as the sections were presented in a clear and well-defined manner. I also appreciated the clear and concise manner in which the authors identified the solutions to which they compared themselves to and their limitations, it was clear that a new approach was needed for them to tackle the issues they had within Facebook.