SCC.411 Building Big Data Systems Report

1. Team Members and Work Undertaken:

Different team members took on different challenges to distribute the work and ensure we did not repeat efforts. Where appropriate, multiple members collaborated to achieve more difficult tasks, notably the creation of the distributed MapReduce environment proved challenging and thus required efforts from all team members.

Harry Baines (35315878):

Harry primarily focused on the entire preprocessing section, completing all tasks (2.1 - 2.5), with assistance from Angelos for task 2.5 regarding the generation of Hive queries via Python, which included automation and seamless integration of the preprocessing and analytics phases of the pipeline. Harry also assisted setting up the distributed Hadoop and Hive environment amongst all other group members.

Charlie Jones (35059062):

Charlie was involved in the initial setup of Hadoop to work over multiple VMs and run MapReduce jobs across nodes, using Linux and Bash. He also created the first initial schema we used. Charlie also developed the initial Hive code to physically create the database on Hive, which covers section 1. He completed section 3.1 analytics questions, some as Hive queries and some as Python where appropriate. He also checked over the preprocessing code but was not heavily involved. Ensured actionable steps were confirmed amongst members after group meetings.

Angelos Malandrakis (35363431):

Angelos was responsible with Charlie for the distribution of the HDFS datanodes across the different virtual machines, using their Linux and Bash skills to complete the work. He implemented the automation process for the data preprocessing, ingestion to HDFS and insertion to Hive. He also did hiveserver's setup to create a direct connection between Hive and Python. For the analytics he was responsible for the calculation of machines' total downtime within the system (task 3.2) and the creative task (task 3.5), where the downtime of each machine platform was calculated.

Varush Simha (35365113):

Varush contributed by connecting a VM and running a few MapReduce jobs on the preprocessed data generated by Harry in the preprocessing phase, which was initially ingested into the database. Furthermore, he developed an Entity Relationship Diagram utilizing the database Schema provided by Charlie. For the creative task in section 1 he illustrated a high-level overview of our data pipeline.

Kyriakos Chiotis (35278597):

Kyriakos researched and explored different techniques regarding the minimization of computational complexities for join operations in our Hive queries. He also assisted in the development of the distributed MapReduce environment and he created the temporal visualization of the number of tasks executing on the busiest machine (task 4.2).

2. Executive Summary:

We began by constructing a distributed cluster of virtual machines capable of running MapReduce. For the setup the scc-411-30 VM was used as master and two more VM's as slaves (with scc-411-30 VM representing both a slave and master machine). Bucketing the Hive table 'task_event' was considered suitable to minimize the computational complexity of the queries with join operations. Specifically, machine_id, job_id and task_idx_inj were the most important. For further optimization, partitioning was applied on event_type and priority to improve Hive queries with filtering where clauses.

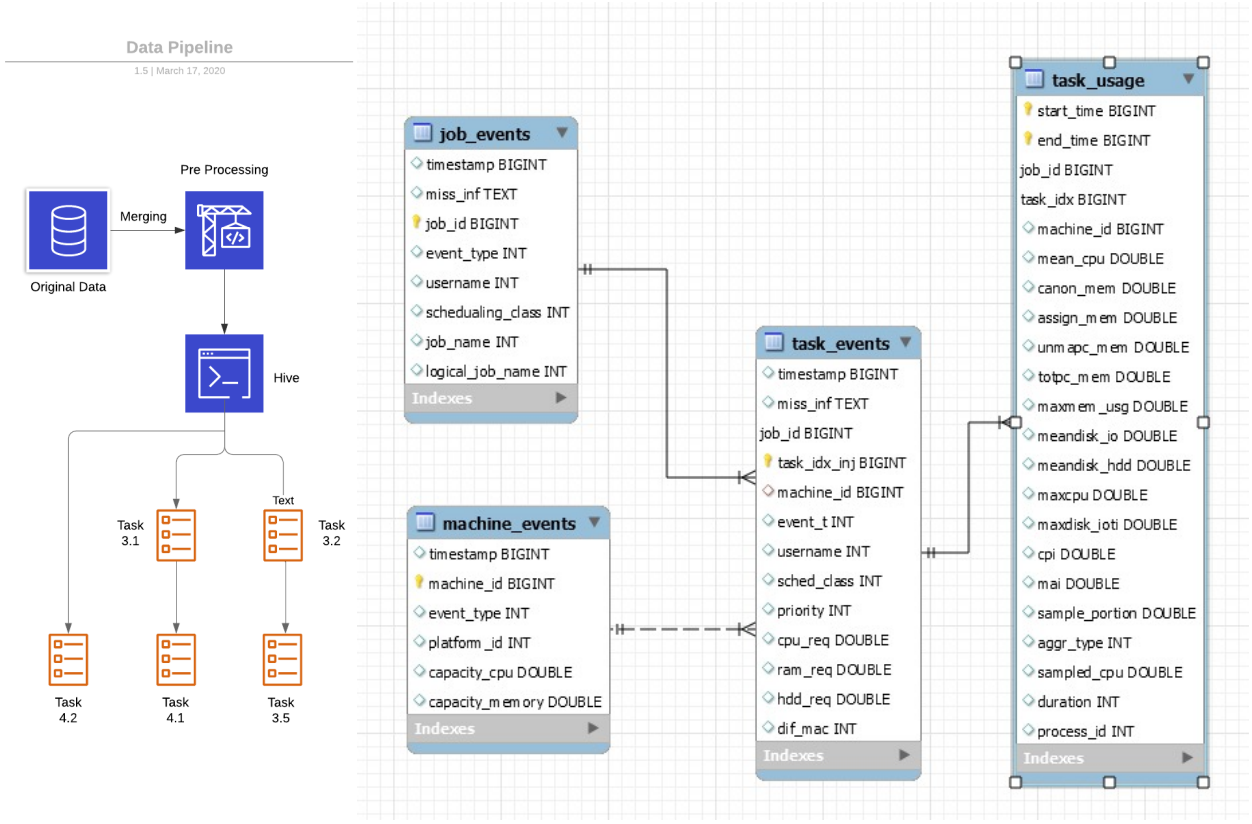
During development of the distributed environment, Harry focused on preprocessing. Data tables of the same type in multiple parts were merged into single CSV files for convenience. We implemented the preprocessing phase of the pipeline

via Python due to it's reliability, versatility and suitability for Data Science. The Pandas package facilitated data manipulation using data frames, which included detecting potentially anomalous values for numerical covariates with the aid of Chebyshev's inequality. Due to the large number of observations present in the dataset, we discarded observations containing anomalous values to prevent skewed distributions which would affect subsequent analyses. Attributes containing hash values and numbers expressed in scientific notation were converted to numerical representations. The jobID and taskID columns in the task_usage table were merged into a single column and expressed in numerical form to identify unique processes. A bash script was created to automate data preprocessing and loading into Hive by doing the following: downloading the data from a remote server as zip file, extracting it, running the Python preprocessing script to preprocess them, ingesting the extracted files to HDFS, creating the Hive database with its tables and finally loading the data. Three different approaches were explored for data processing. Firstly, one running queries from HiveQL files and exporting the results to CSV. Secondly, we developed Python files to load and process the data directly from the preprocessed files. Thirdly, Hive's hiveserver was set up and using Python's module PyHive, enabling us to send requests and execute Hive commands remotely using Python.

Following preprocessing we began analysing the preprocessed CSV files using Python and Hive queries where appropriate. Finally, we visualized the results obtained from the analytics section. Timestamps for tasks were divided into 5-minute intervals. Therefore, the number of tasks was calculated for each machine in the same interval. Combining subqueries led to the number of tasks of the busiest machines for each interval.

3. High Level Overview and ER Diagram:

The following diagram illustrates a high-level overview of our implemented pipeline, and an Entity Relationship (ER) diagram describing the entities, relationships and attributes in our relational database:



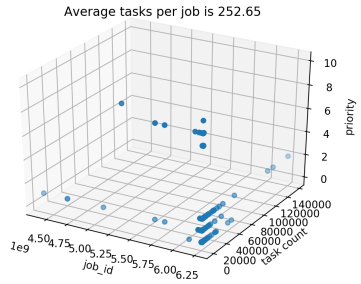
4. Design Path:

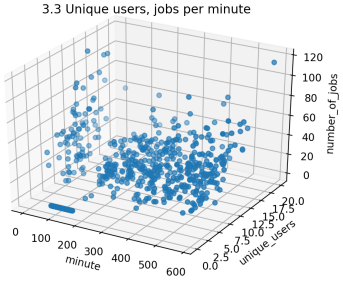
A list of issues you encountered, and how this affected your system design

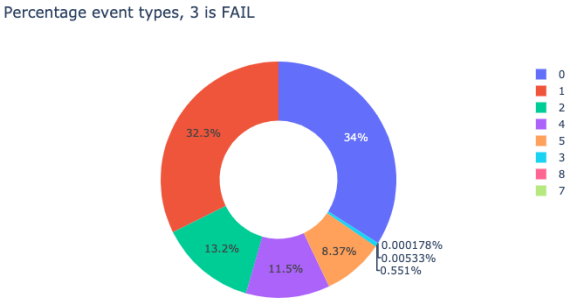
Due to timeout restrictions on the servers and VMs we are using, some programs and shell scripts would take too long to run and so we had to consider fast ways or sometimes localized solutions to problems, such as running some analytics on local machines in python.

1. The different nodes wouldn't be able to connect together as they were originally working with different cluster IDs. To solve the issue, all the datanode and namenode files where deleted from all of the virtual machines and the namenodes where reformatted.
2. Hive is not able to run multiple sessions simultaneously. Hence, only one person could work on Hive per time. To solve this issue, we setted up the hiveserver which is able to serve multiple requests per second.
3. The PyHive dependences were tricky to install for the new versions of python, for both Linux and Windows systems.
4. We weren't able to create an FTP connection with the master VM (there was a network issue with the port 21), so we could only transfer files from the VM and towards it only through http requests.
5. For the preprocessing part, the functions had to be applied elementwise to increase the algorithms efficiency compared to working with itterations

5. Main Outputs:





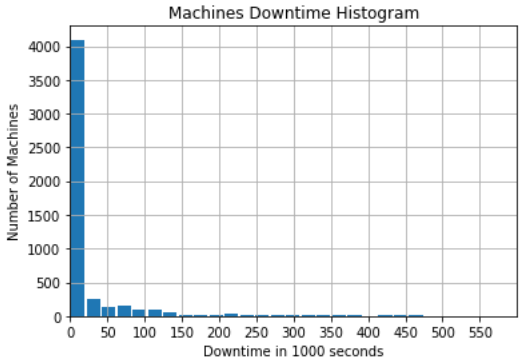


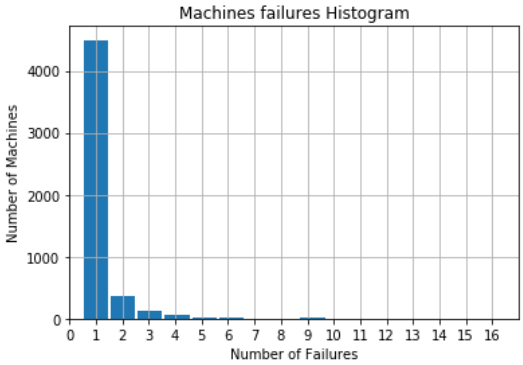
The first 3d diagram addresses 3.1 part 1 and has the statistic for the average number of tasks per job, and visualizes them in priority and job_id, which will increase over time.

The number of jobs per minute across users is given in the second 3d diagram, this addresses 3.1 part 2. It shows the times when there were peaks in demand from users.

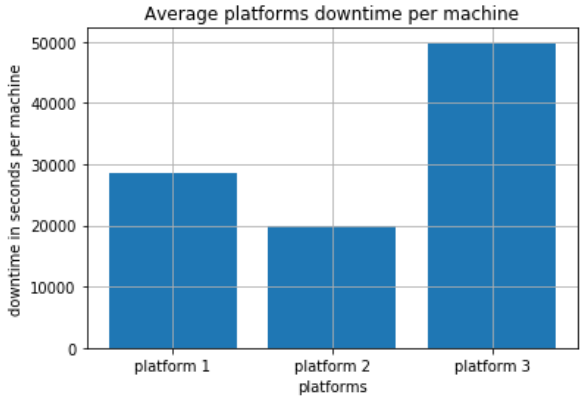
The pie diagram is in response to question 3.1 part 3, which asks for the number of failed tasks. That number is 6196 and can be answered in a single hive query. The failed tasks are given by the light blue number 3. other tasks percentages are given, showing failures are small compared to all other tasks.

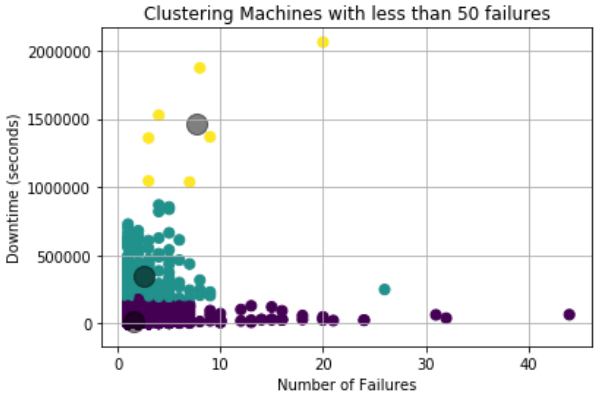
For the task 3.2, The two histograms bellow represent the machines total downtime and number of failures. We can see that most of the machines have a total downtime of less than 15000 seconds (approximately 42 minutes) and only one failure during the trace period. However, there were also a few machines that had downtime for almost the whole duration of the trace.





For the creative task, we have identified, that there are three different machine platforms. From the above histogram, which shows the downtime for each platform, it seems that the 3rd platform is the one with the highest downtime per machine. Also, a clustering between the machines according to their total downtime and the total number of failures has been implemented.





Finally, for the visualization task, the next figure, represents the number of tasks running on the machine for a five minutes interval. It shows, that even though the machine runs approximately 5,000 tasks per 5 minutes, sometimes it's overloaded and can run up to 30,000 tasks per 5 minutes. During the trace, there were 6 times that the busiest machine was overloaded with much more than its usual number of tasks.