INM432 Big Data

Individual Coursework
Sahan Chowdhury
Sahan.Chowdhury@city.ac.uk

Colab notebook link: https://colab.research.google.com/drive/1_1u6RTMV-p0rVI-Rv5On6Vhui_kmmVEm?usp=sharing

Section 1:

1Di.

Initial cluster was created with single nodes and 8 virtual CPU's. The original spark script was run on the cluster. The metrics are displayed below figure (1)

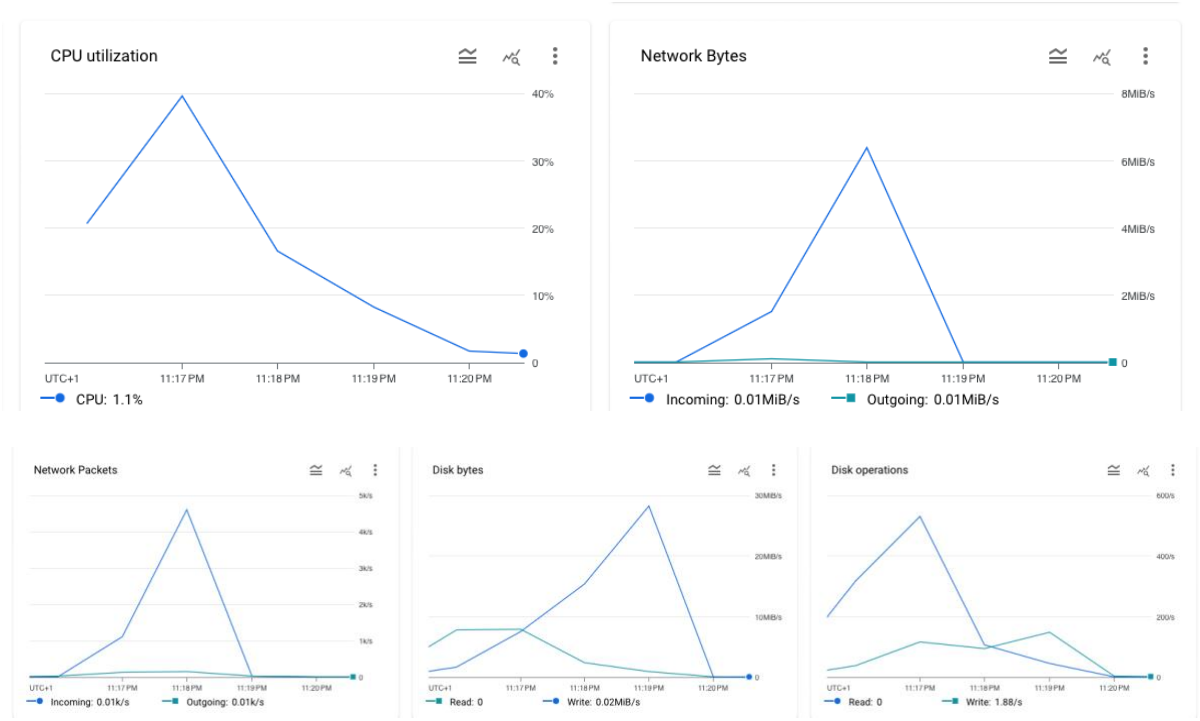


Figure (1) showing metrics of single node with 8 virtual CPU's.

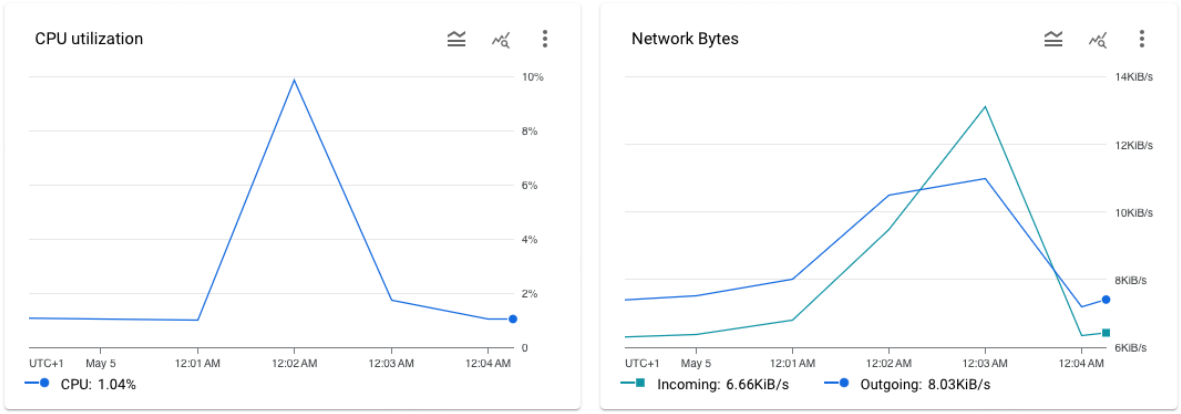
It is clear to see CPU utilization was 1.1%. Network bytes, network packets, Disk bytes and Disk operations were minimal. The time taken was 25 seconds.

Now the same spark script was run on a new modified cluster with maximum possible worker nodes available from resources, the configurations were 3 worker nodes and 1 virtual CPU the metrics are shown below.

Figure 2 metrics for maximum node cluster with the original spark script

From the metrics above its clear to see CPU utilization has increased to 3.58%, and all other metrics have remained minimal, except disk operations which had a significantly higher read of 1.97/s and a write of 4.52/s. The elapsed time in this instance was 53 seconds.

The metrics would be evaluated again, however with improved parallelisation, where a second parameter of 16 partitions is added. Observing the first cluster created with a single node and 8 virtual CPUs, with the new script the metrics are shown as below.



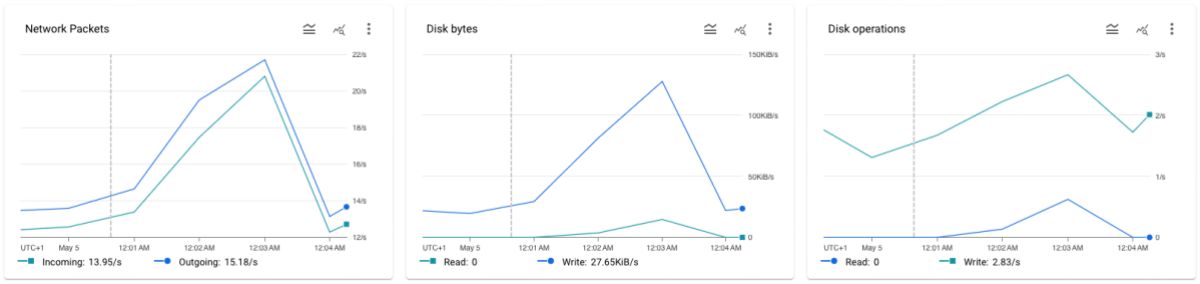


Figure 3: single node cluster with 8 virtual CPU run with the improved parallelization script.

From initial observations the CPU utilization seems to have increased very slightly, however the incoming and outgoing network bytes and packets has increased. It is also noteworthy the elapsed time was the same as with the original script.

Moving onto the other cluster with the maximum node cluster (3 nodes) and one virtual CPU, the metrics are as follows.

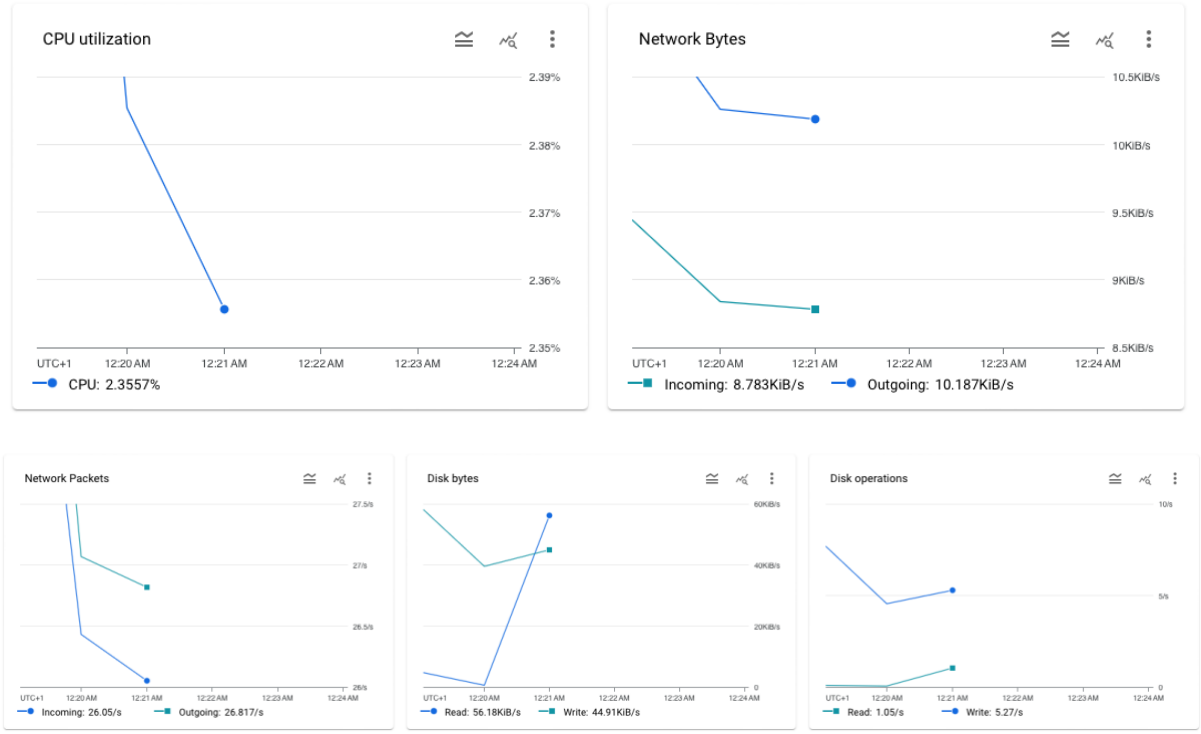


Figure (4) The metrics of the improved parallelization script run on the max cluster (3 nodes)

The elapsed time on this occasion was much less than previous (34 seconds)

1Dii.

A new cluster was created, with 4 machines (1 master and 3 worker) and 2 virtual CPUs used and another with 1 machine with eightfold resources. The Metrics are shown below for both clusters using the improved parallelization script.

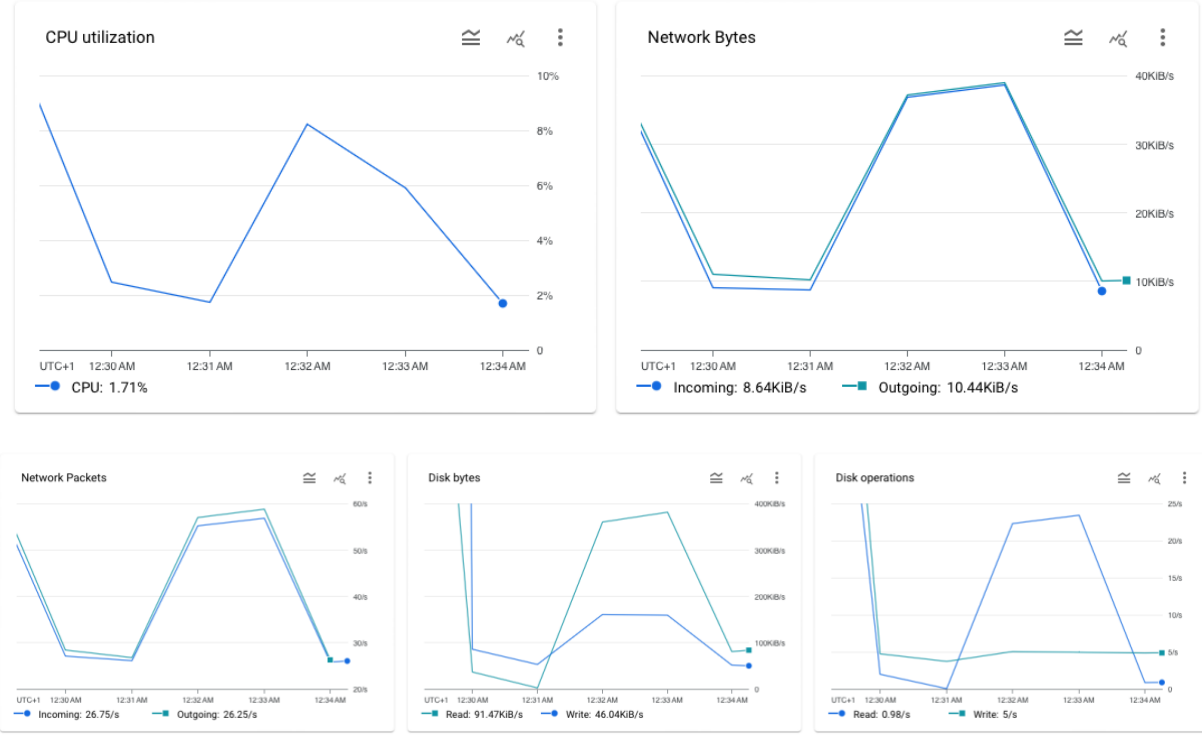


Figure (5) Metrics of clusters with 4 machines and 2vCPUs

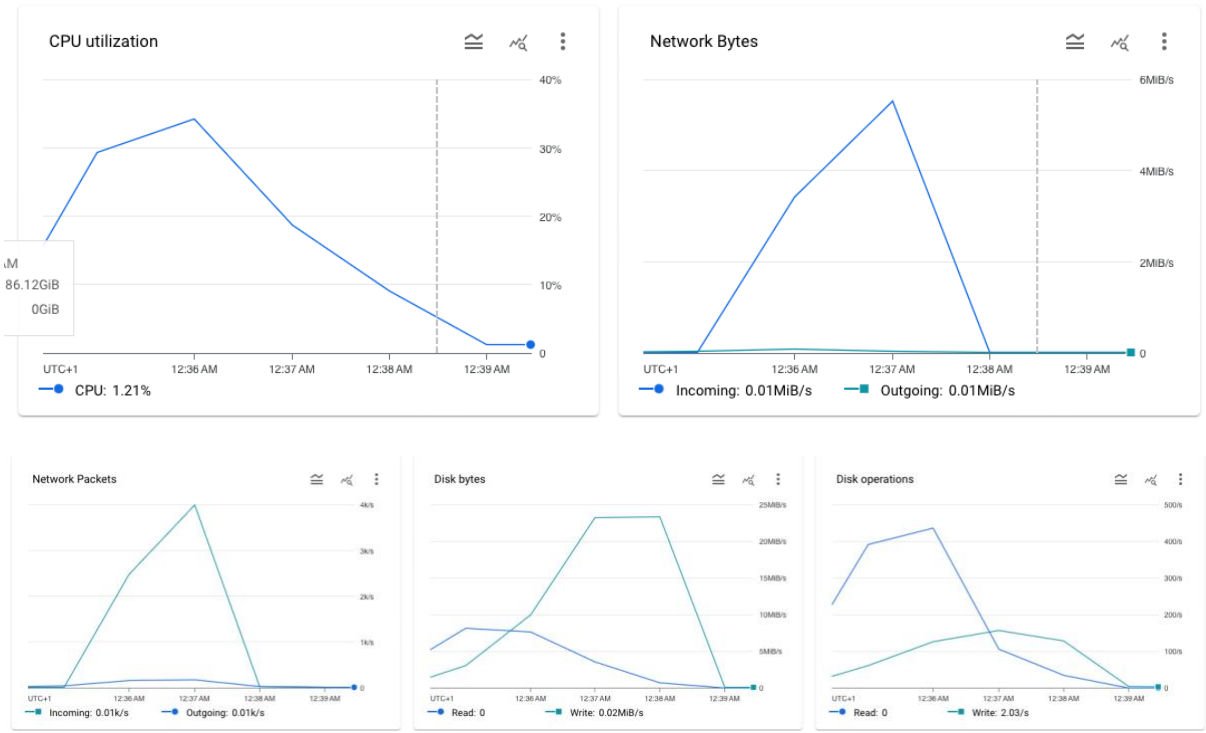


Figure (6) Metrics of cluster with eightfold resources and one machine

Metrics	Network Bytes	Network Packets	Disk Bytes	Disk Operations
Maximum incoming	38.66KiB/s	56.66/s	-	-
Maximum Outgoing	38.91KiB/s	58.72/s	-	-
Maximum Read	-	-	380.59KiB/s	23.47/s
Maximum Write	-	-	159.89KiB/s	4.98/s

Table (1) maximum metric value for clusters with 4 machines and 2vCPUs

Metrics	Network Bytes	Network Packets	Disk Bytes	Disk Operations
Maximum incoming	5.52KiB/s	3.99/s	-	-
Maximum Outgoing	0.04KiB/s	0.17/s	-	-
Maximum Read	-	-	23.23KiB/s	436.12/s
Maximum Write	-	-	3.57KiB/s	125.68/s

Table (2) maximum metric value for clusters with eightfold resources and one machine

The cluster with 4 machines has higher network byte and packets incoming and outgoing, as well as disk byte max read and write, suggesting more intensive communication between the nodes. The higher disk byte read and write operations indicate greater data access and storage activity.

However, the eightfold cluster with one machine has the higher max disk operation read and write, this could be due to concentrated processing and data storage activities within a single node.

In terms of network bandwidth allocation, the cluster with 4 machines, may experience higher throughput and longer latencies due to increased competition.

1Diii.

Lab sessions were performed locally. Data is typically stored locally, and this setup has limitations, especially when dealing with large amounts of data or complex tasks as everything relies on the resources of the machine. Spark spreads data across multiple clusters, hence as a result it is much faster.

Section 2

2C.

The ‘cache ()’ function retains the contents of an RDD in memory, so that it can be used later. In the code caching was used right after creating the RDD for parameter combinations, caching right after the RDD creation, ensures the data is in the memory before any actions or transformations, as a result reducing computation time, it also minimised usage of unnecessary computational resources.

Section 3

3A.

In the paper of O. Alipourfard et al, it talks and discusses the importance of picking the right cloud configurations to optimize and maximise performance metrics ‘Choosing poorly can significantly degrade performance and increase the cost to run a job by 2-3x on average.’ [1] This highlights the importance of choosing right cloud configurations, within this coursework this issue arose, when configuring clusters, it was difficult to create the ‘perfect’ cluster. The paper identifies “cherry picking’ as the optimal way to configure big data analytics job running in the cloud, for instance ‘CherryPick has a 45-90% chance to find optimal configurations, otherwise near optimal, saving up to 75% search cost compared to existing solutions. ‘’ This conveys how great Cherry Picking can be. Cherry picking leverages Bayesian optimisation modelling, where the model analyses various configurations and parameters such as disk count and CPU speed, it aims to optimize efficiency and reduce costs.

This coursework involved testing different cluster configurations and the impact on performance metrics. While Bayesian optimization or ‘Cherry Picking’ was not directly employed, the overarching goal to find the most suitable clusters and improvement techniques such as improved parallelization and caching, align with the objectives discussed in the paper. Therefore, we can say the concepts and techniques presented in the paper can apply to this coursework to a certain extent, particularly when there is a need to optimize cloud configurations for performance metrics.

3B.

For batch processing, if working with predictable workloads (small range of data). A strategy like traditional capacity planning can be deployed, where cloud configurations are dependent on historical data and expected workload patterns. However, a drawback is that historical data may not always be an accurate representation of the future demands. In times when the workload varies significantly, using techniques such as ‘cherry picking’ stated in the paper of O. Alipourfard et al, where different combinations of cloud configurations are analysed to obtain the best configuration, techniques such as partioning and use of multiple nodes enhances performance and scalability. Parallel processing frameworks such as spark ensure efficient processing of data by distributing workload across a cluster. Data replication and checkpointing are an example of fault tolerance mechanisms, this ensures there is reliability in batch processing systems. From the

coursework and as mentioned in the corresponding paper tuning parameters such as batch size and batch number can be differed to be more efficient

Online processing is useful where it is important to maintain up to date insights. It is also known as real time processing. This kind of processing has low latency to process data in real time, furthermore speed is very important and as a result low latency and less computation effort is required for maximum optimization. It also uses operations such as caching to speed up the process and partitions to split data into smaller chunks and utilize parallelization.

As for stream processing, also known as micro batching, works by groups of data incoming at regular intervals, they usually tend to have low latencies and high speed. However, this is usually requiring additional computational resources. Strategies for stream processing involve dynamic resource allocation and microservice architecture, whereby it enables scalability and flexibility to handle a varying workload efficiently. Stream processing frameworks come with features such as windowing, which analyses data over specified intervals and complex event processing queries which processes and assesses data prior to being stored [2].

The general relationship lies in the adaptative nature, where cloud configurations are continuously optimized based on real time, the underlying cost and efficiency.

References:

[1] O. Alipourfard et al., “This paper is included in the Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI ’17). CherryPick: Adaptively Unearthing the Best Cloud Configurations for Big Data Analytics CherryPick: Adaptively Unearthing the Best Cloud Configurations for Big Data Analytics.” Available: https://people.irisa.fr/Davide.Frey/wp-content/uploads/2018/02/cherrypick.pdf

[2] “What is Complex Event Processing?,” Databricks. https://www.databricks.com/glossary/complex-event-processing

Word Count: 1265