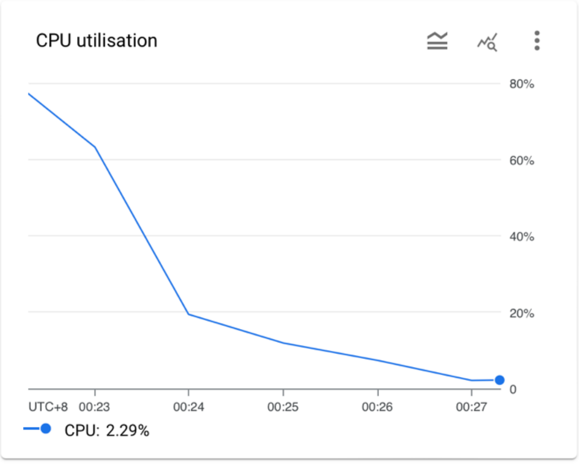
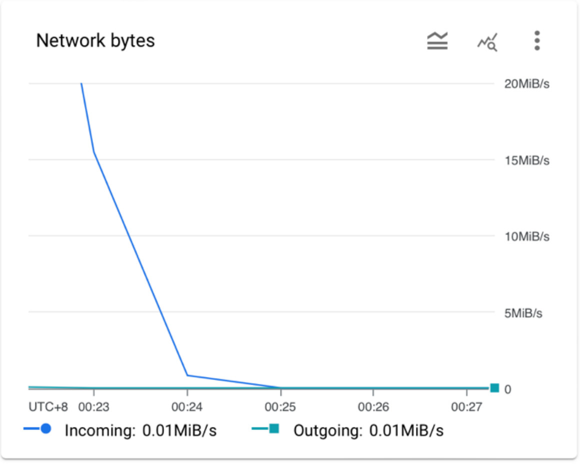
Big Data Coursework Submission

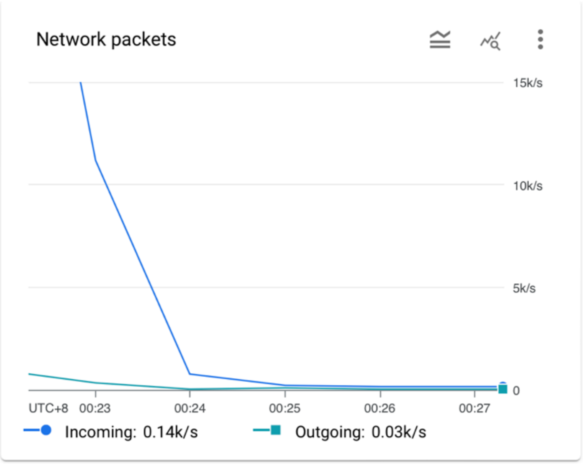
Nidhi Joshi

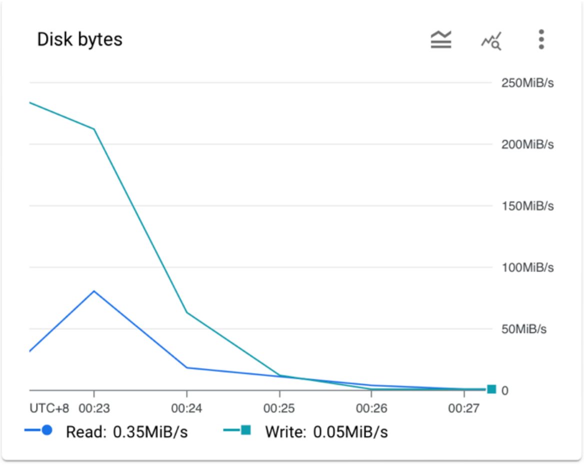
(Student ID: 230017936)
(Email: nidhi.joshi@city.ac.uk)
(Google Colab link: https://colab.research.google.com/drive/1degKcR7HMf8o-0picnRw6o-VG4qVbo1V?usp=sharing)

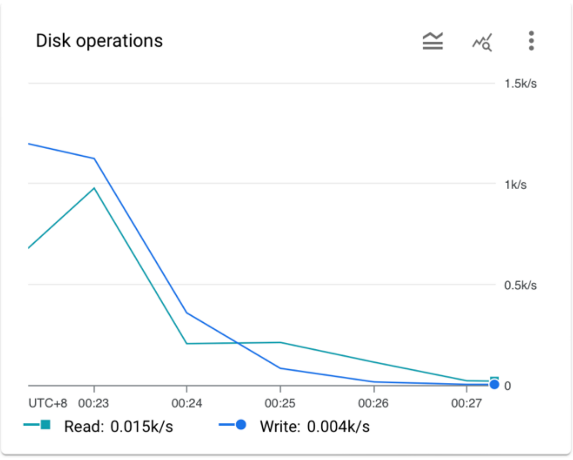
1(d) (i). Before improving parallelization (Fig. 1 Metrics of cluster with 1 master + 6 workers with 1 vCPU each (maximal) only on two nodes) (CPU times: user 3 μs, sys: 0 ns, total: 3 μs; Wall time: 6.44 μs)









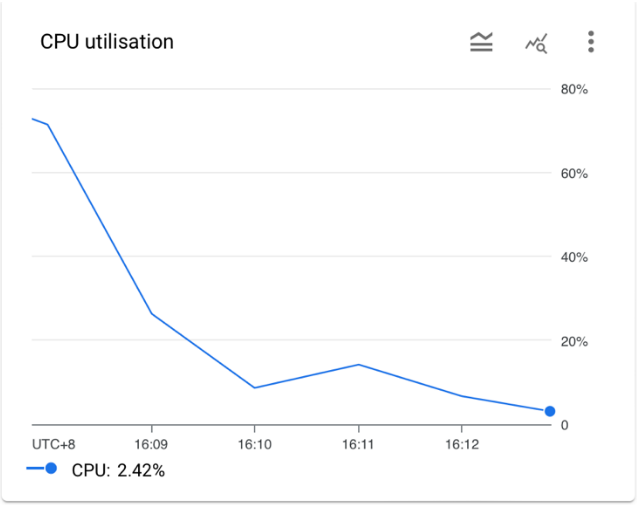


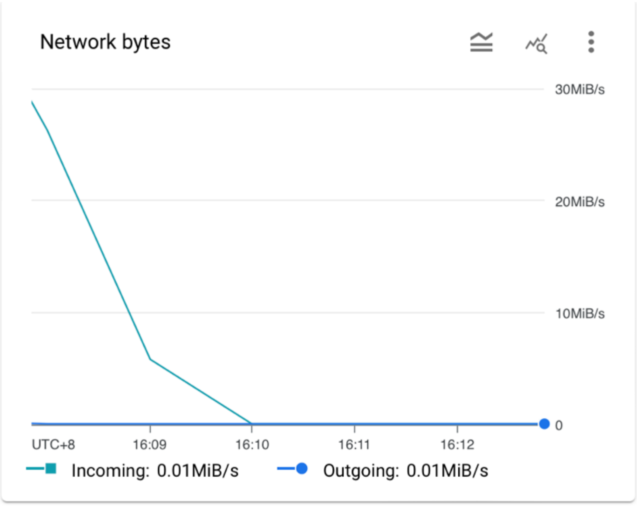
	Network Bytes	Network Packets
Max Incoming	15.49 MiB/s	11.17k/s
Max Outgoing	0.06 MiB/s	0.36k/s

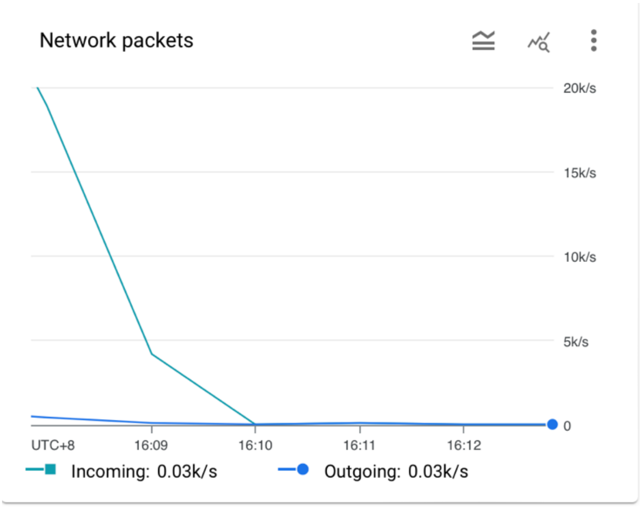
	Disk Bytes	Disk Operations
Max Read	80.56 MiB/s	0.975k/s
Max Write	212.23 MiB/s	1.123k/s

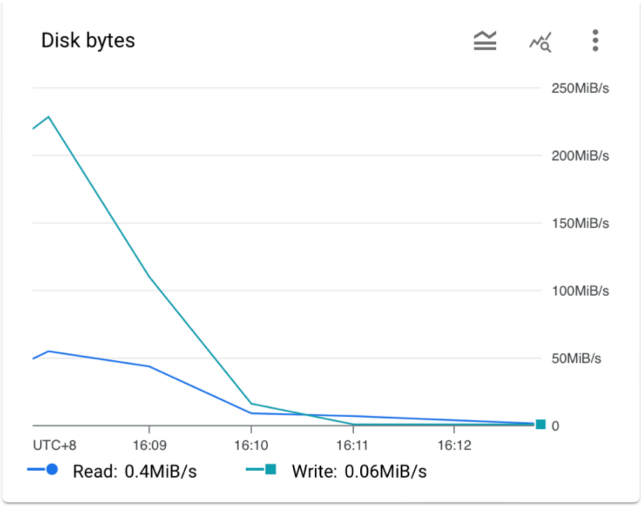
We also see that the initial CPU utilization was almost 80% but it fell drastically later.

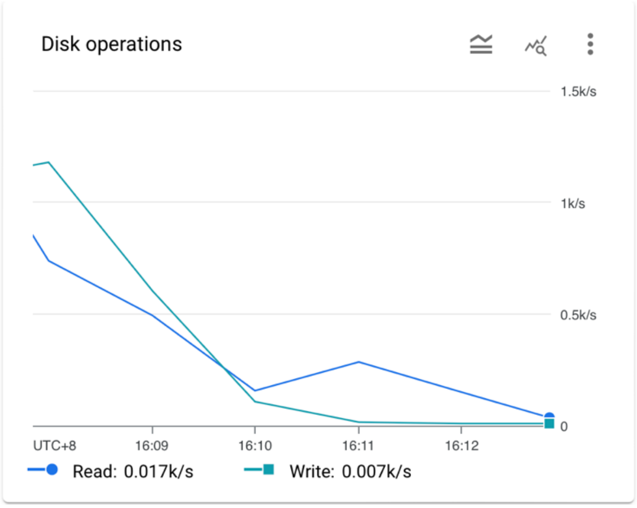
After improving parallelization (Fig. 2 Metrics of cluster with 1 master + 6 workers with 1 vCPU each (maximal) only on two nodes) (CPU times: user 4 µs, sys: 0 ns, total: 4 µs; Wall time: 8.82 µs)









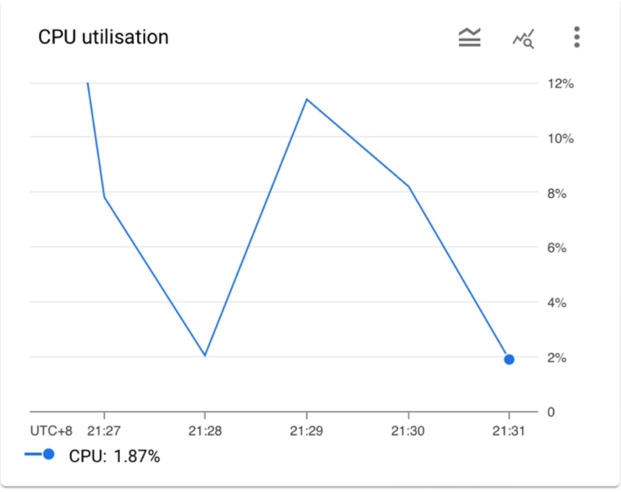


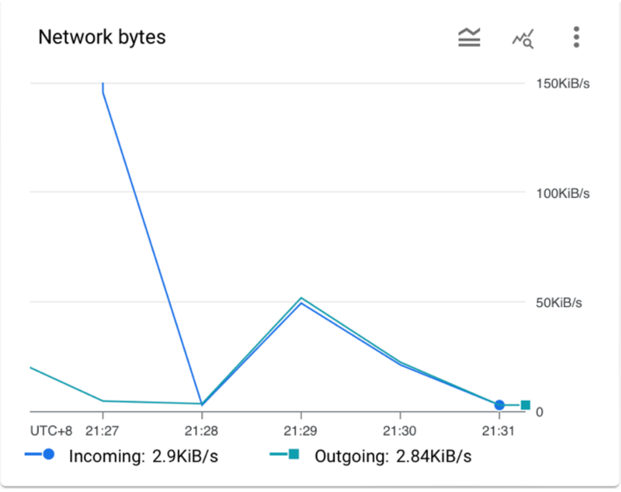
	Network Bytes	**Network Packets**
Max Incoming	26.26 MiB/s	18.93k/s
Max Outgoing	0.06 MiB/s	0.45k/s

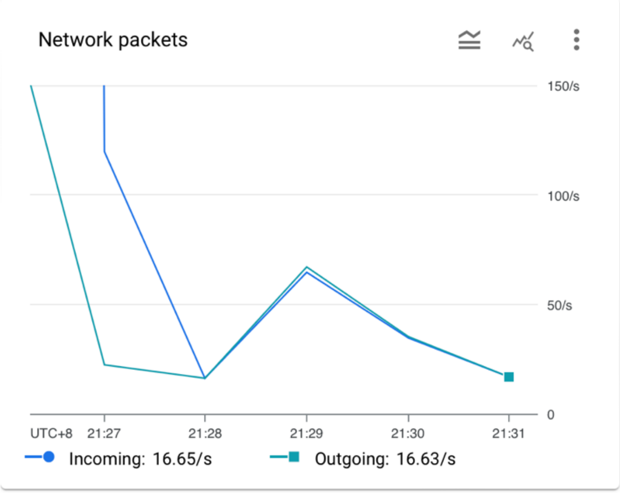
	Disk Bytes	**Disk Operations**
Max Read	54.25 MiB/s	0.741k/s
Max Write	228.46 MiB/s	1.179k/s

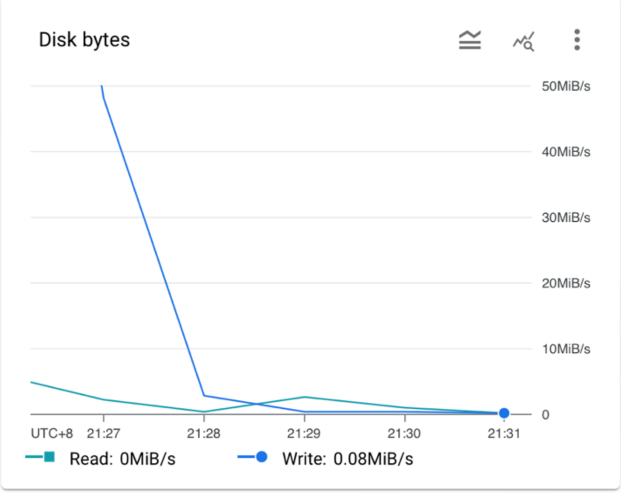
CPU was 77% utilized in the beginning falling to 2% with time.

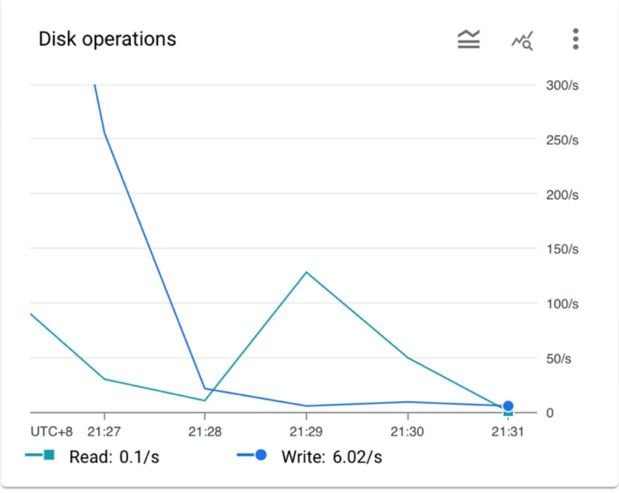
1(d) (ii). Experiment with cluster configurations (Fig. 3 Metrics of cluster with 1 master + 3 workers with 2 vCPU each with double the resources – memory, disk) (CPU times: user 3 μs, sys: 0 ns, total: 3 μs; Wall time: 5.72 μs)









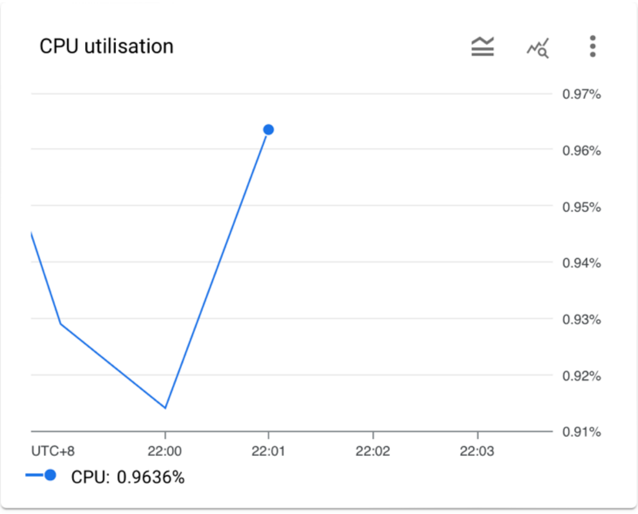


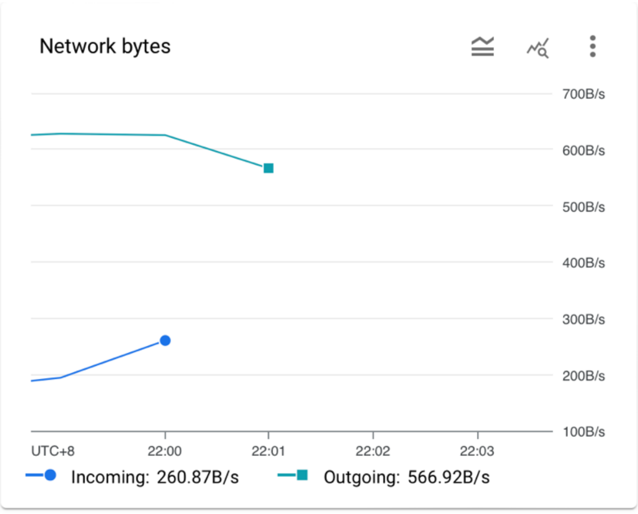
	Network Bytes	Network Packets
Max Incoming	145.29 KiB/s	119.72/s
Max Outgoing	51.61 KiB/s	66.82/s

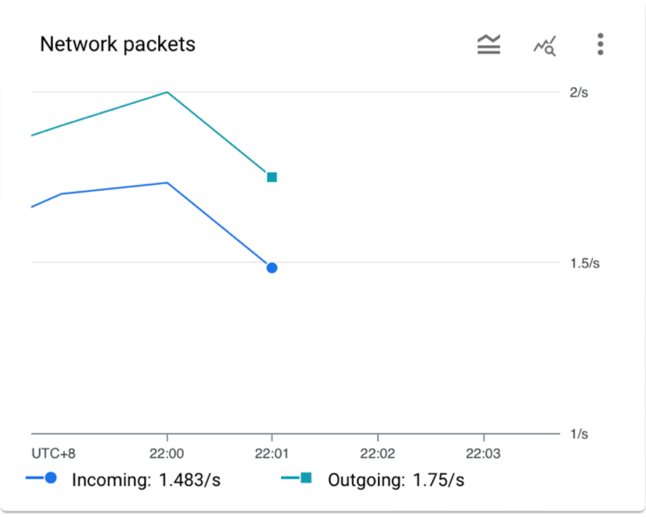
	Disk Bytes	Disk Operations
Max Read	2.15 MiB/s	127.55/s
Max Write	48.05 MiB/s	255.27/s

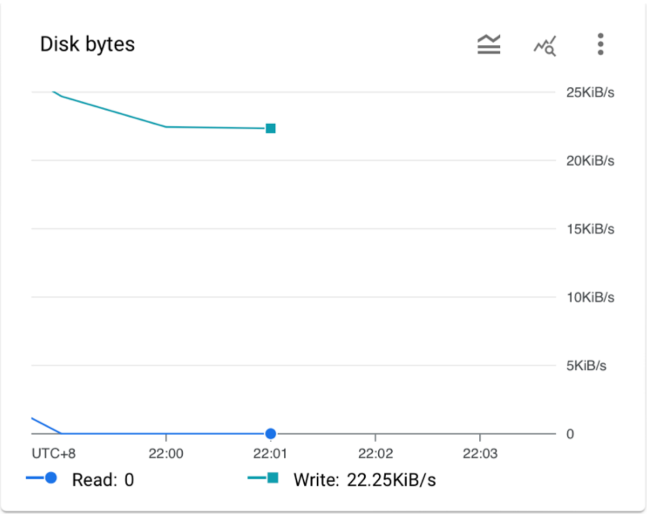
CPU utilization was very low overall with max utilization going up to 12%.

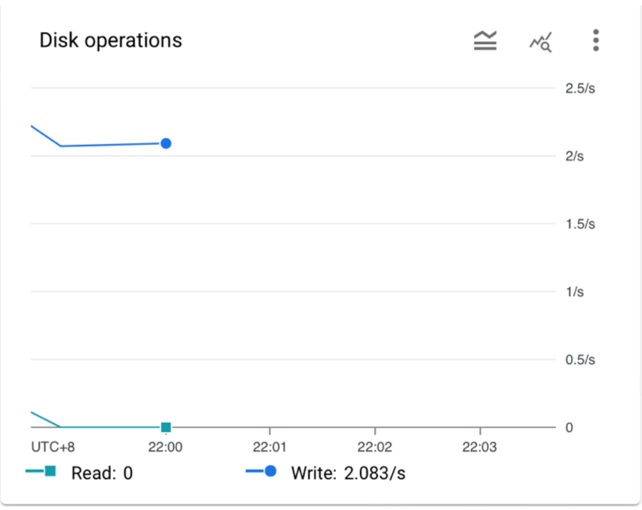
Experiment with cluster configurations (Fig. 4 Metrics of cluster with 1 machine with 8-fold resources) (CPU times: user 5 µs, sys: 0 ns, total: 5 µs; Wall time: 9.06 µs)









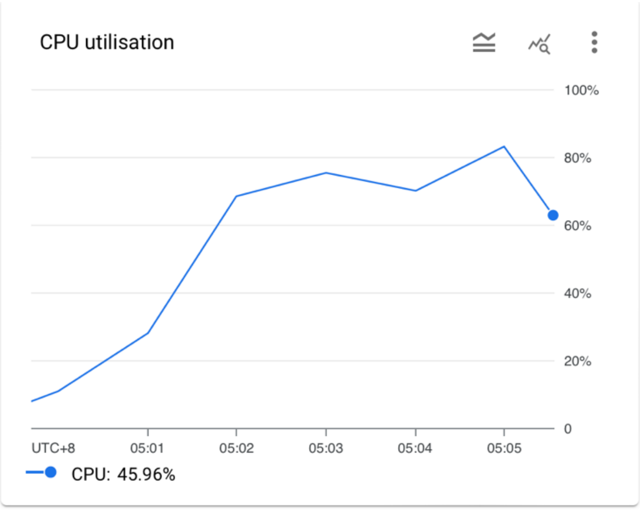


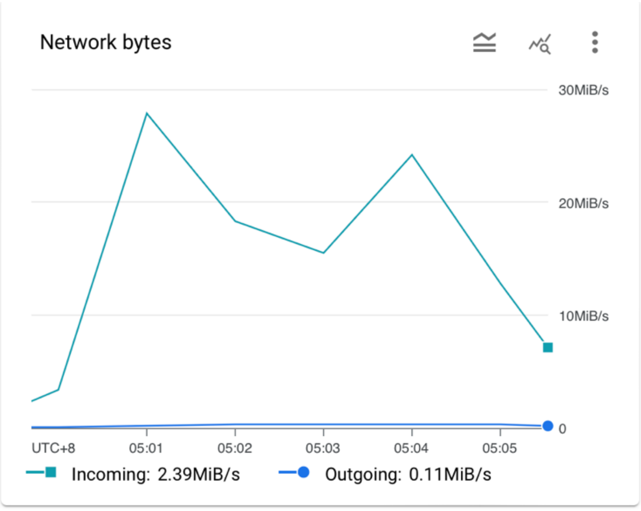
	Network Bytes	**Network Packets**
Max Incoming	260.87 B/s	1.733/s
Max Outgoing	624.72 B/s	2/s

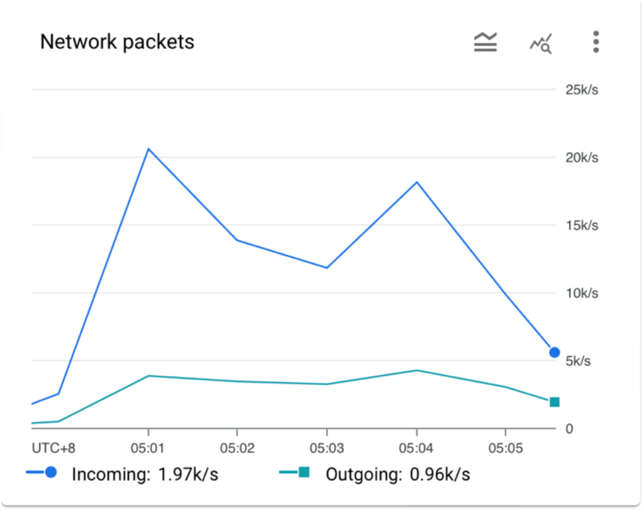
	Disk Bytes	**Disk Operations**
Max Read	2 KiB/s	0.2/s
Max Write	24.61 KiB/s	2.083/s

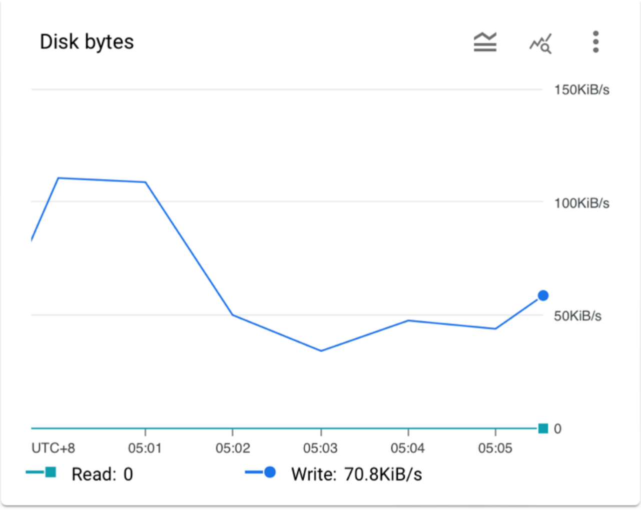
CPU utilization was really very low with max 0.965% utilised.

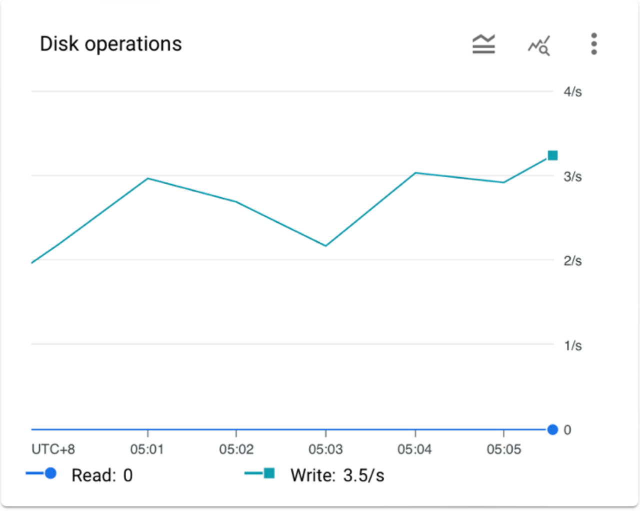
2(b) (ii). Before applying RDD.cache() (Fig. 5 cluster: 1 machine with 8-fold resources; Spark job) (CPU times: user 4 µs, sys: 0 ns, total: 5 µs; Wall time: 8.34 µs)









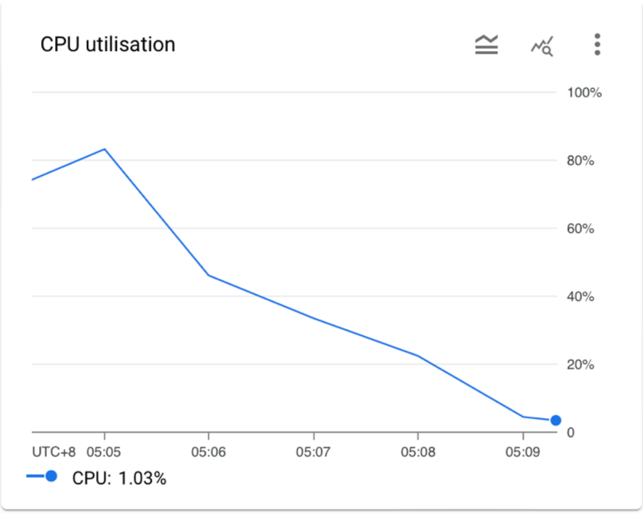


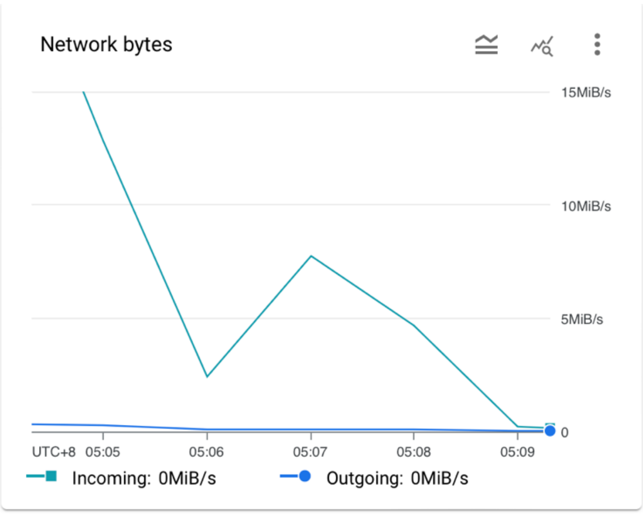
	Network Bytes	**Network Packets**
Max Incoming	27.85 MiB/s	20.6k/s
Max Outgoing	0.31 MiB/s	4.26k/s

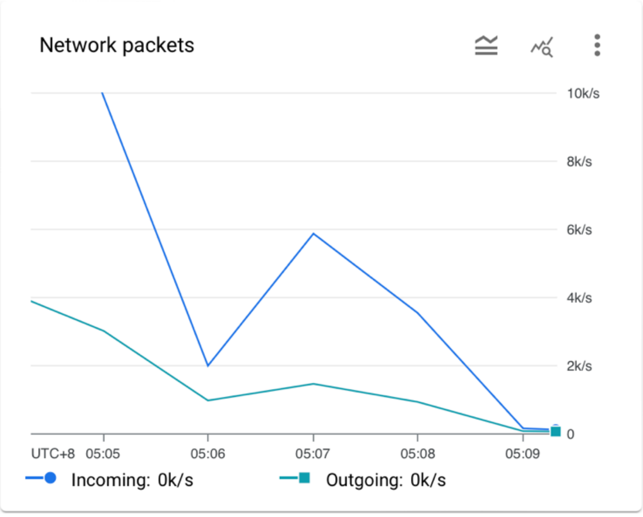
	Disk Bytes	**Disk Operations**
Max Read	0.02 KiB/s	0.03/s
Max Write	110.45 KiB/s	3.5/s

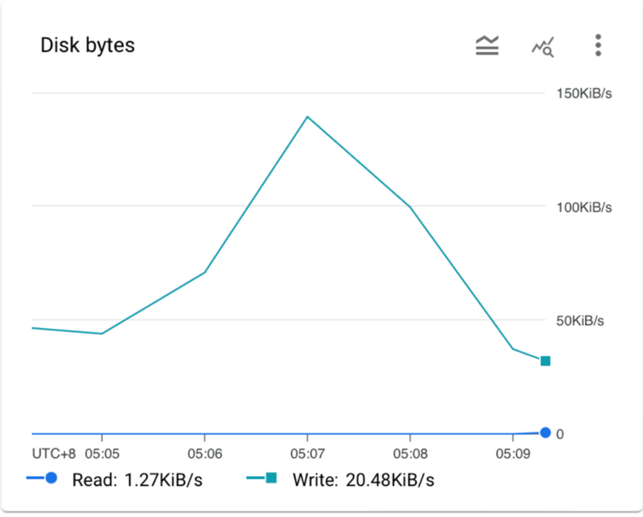
CPU utilization went up from 9% to 82% in 5 minutes. This was without caching.

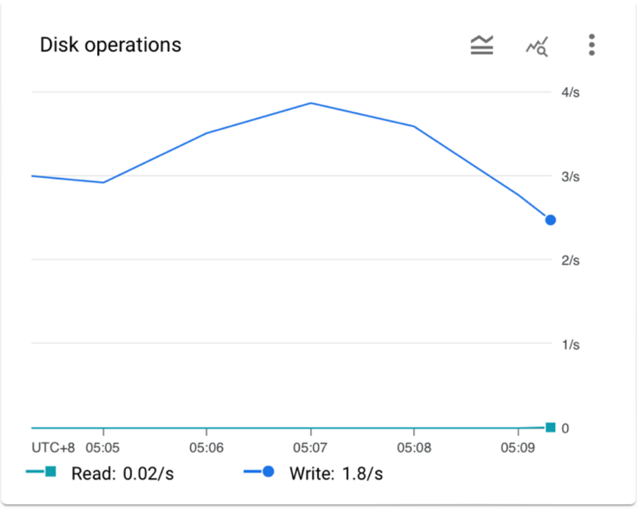
2(c). After applying RDD.cache() (Fig. 6 cluster: 1 machine with 8-fold resources; Spark job) (CPU times: user 5 μs, sys: 0 ns, total: 5 μs; Wall time: 11.7 μs)









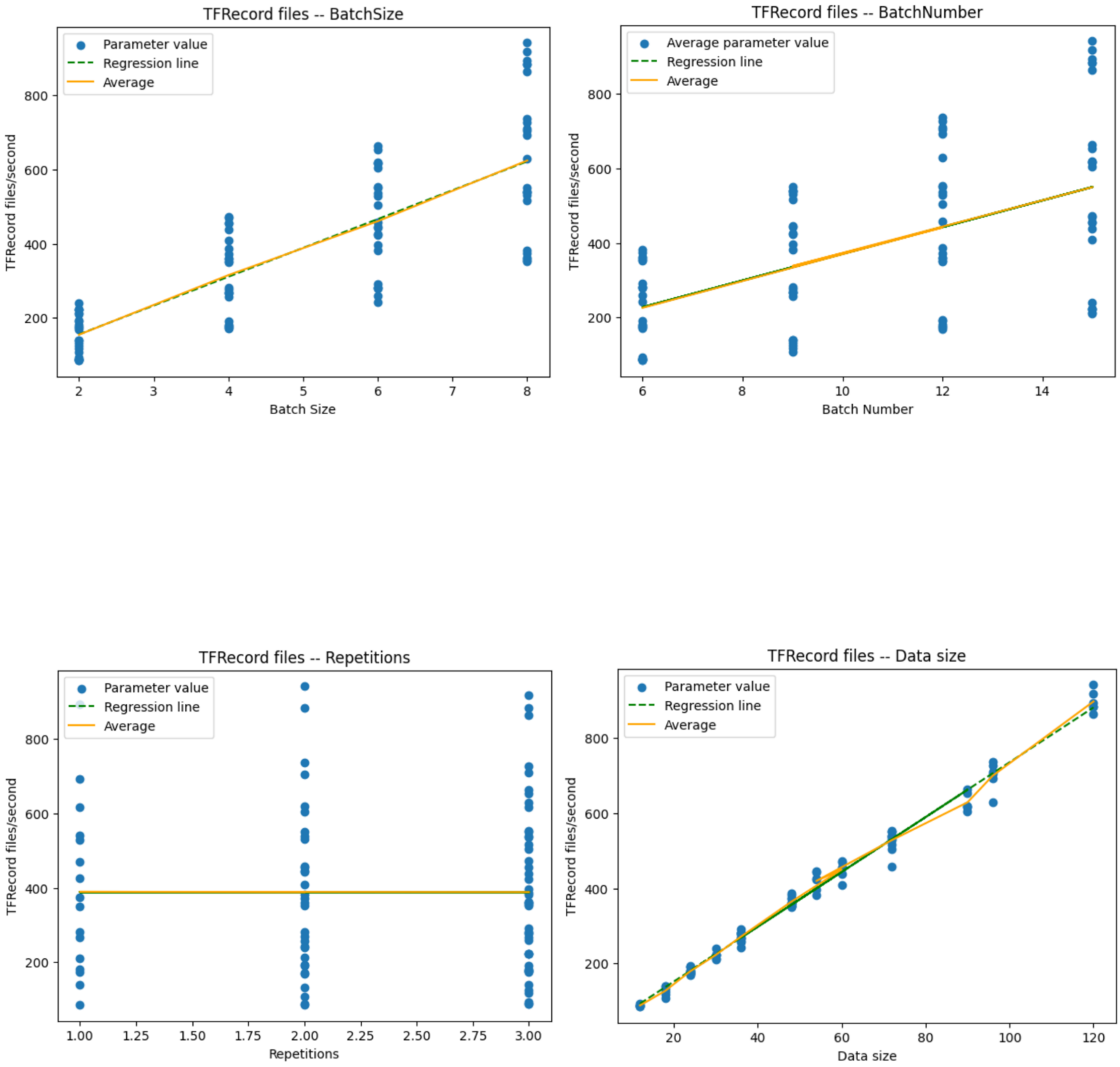


	Network Bytes	**Network Packets**
Max Incoming	12.83 MiB/s	5.84k/s
Max Outgoing	0.27 MiB/s	2.99k/s

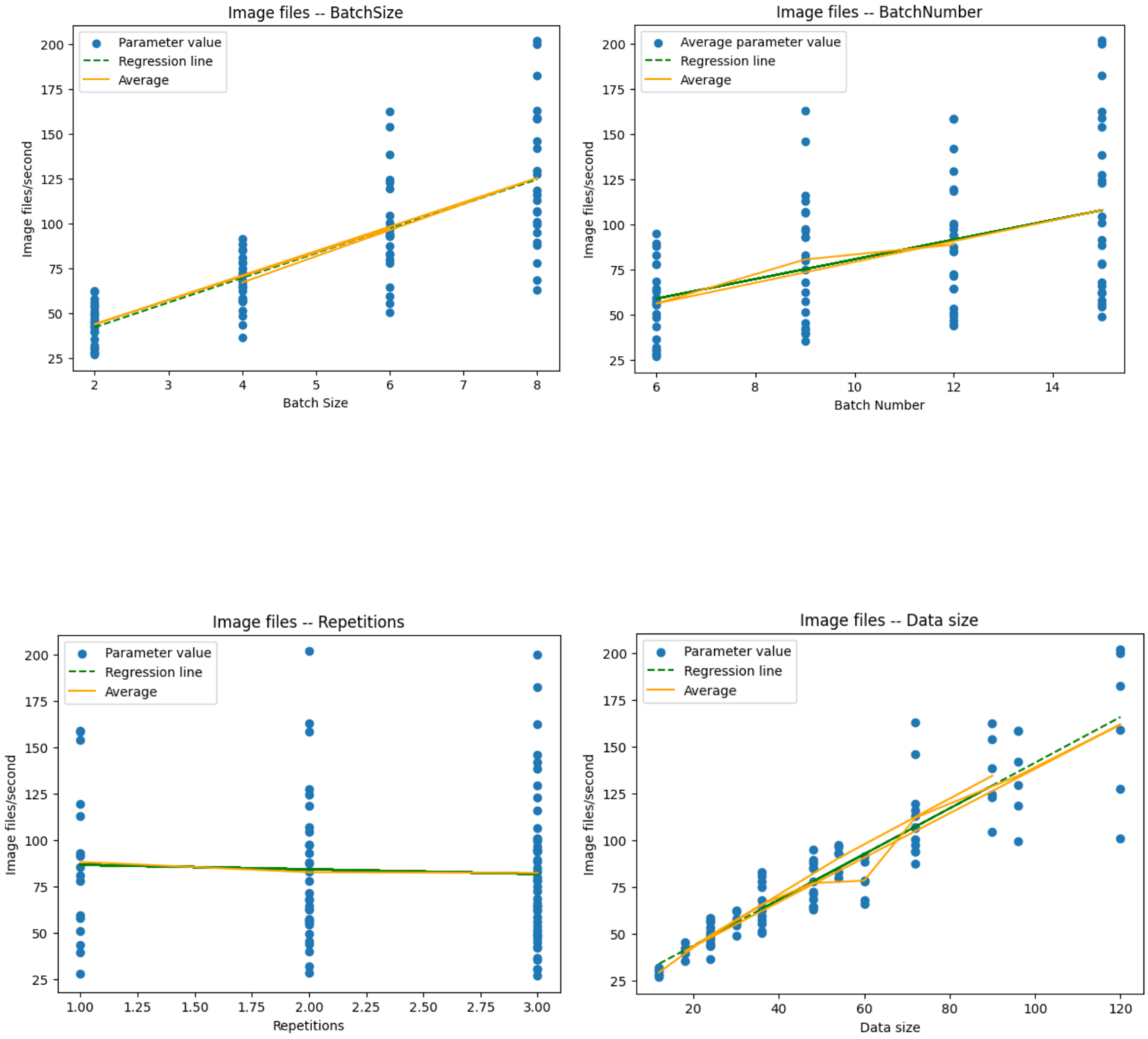
	Disk Bytes	**Disk Operations**
Max Read	0 KiB/s	0.02/s
Max Write	139.16 KiB/s	3.87/s

With caching, CPU utilization decreased from 84% to 2%. Lesser network bytes and packets used.

2(d). Reading TFRecord files (Fig. 7**)**



2(d). Reading Image files (Fig. 8)



File Type	Parameters	P-value	Slope	Intercept
TFRecord files	Batch Size	0.6312	77.6542	0.1702
TFRecord files	Batch number	0.3018	35.8001	12.5401
TFRecord files	Repetitions	1.2566	-0.3287	389.2086
TFRecord files	Data size	0.9898	7.2939	5.5077
Image files	Batch Size	0.5949	13.7521	14.4546
Image files	Batch number	0.2098	5.4457	26.0352
Image files	Repetitions	0.0021	-2.4676	88.9732
Image files	Data size	0.8346	1.2218	19.0707

In Figure 7, we observe that the significance level (P-value) for the batch size of TfRecord files is 0.6312, while for the batch number it is 0.3018, indicating a correlation between the two variables. By selecting better combinations, it is possible to further enhance the P-value when a substantial amount of data is available. Figure 8 demonstrates that the P-value for the batch size of Image files is 0.5949, and for the batch number it is 0.2098, suggesting a lower value.

When comparing the performance of the model running it locally versus in the cloud, we discovered that conducting local tests led to significant improvements. Connecting to cloud servers resulted in high latency, with a delay of 4535 milliseconds. Parallelisation helped save costs by eliminating the need for processors to wait for the completion of previous tests as they could proceed to the next test without any delay.

Typically for smaller tasks, systems optimized on speed are faster than those optimised on throughput. Therefore, reducing latency by utilizing storage space instead of solely relying on memory proves more advantageous in such cases. And so, cloud service providers can enhance efficiency by associating throughput with the storage capacity of the disk. However, it leads to increased latency and hence, lesser cost efficiency in case we perform other regressions apart from linear regressions on the cloud. Basically there’s a trade-off between accuracy and cost on the cloud as high accuracy requires larger datasets.

3a) Contextualise

Relate the previous tasks and the results to this concept. (It is not necessary to work through the full details of the paper, focus just on the main ideas). To what extent and under what conditions do the concepts and techniques in the paper apply to the task in this coursework? (12%)

During the course of our testing, we encountered instances where previously performed easier cloud tasks also took unexpectedly prolonged execution times. After thorough investigation, we attributed this issue to the presence of "cloud noise," a phenomenon characterized by frequent fluctuations within the cloud environment. To mitigate the impact of cloud noise on our system's performance, we employed adaptive algorithms that used a cost function which when multiplied with the algorithms, subdued the variability introduced by cloud noise. The resulting output helped us make informed decisions on different combinations of successful cloud configurations. This particular approach, aptly termed "cherry-picking," has demonstrated promising results within the context of our coursework. Nonetheless, it is important to acknowledge the challenges associated with adopting the cherry-picking strategy, which include the development of a sophisticated performance model, a comprehensive cost model, also addressing the inherent heterogeneity of applications among many others. Overcoming these obstacles is crucial to successfully implementing cherry-picking in practice.

Furthermore, the research paper emphasizes that in addition to cherry-picking, modeling and search techniques have proven to be effective methods for predicting most optimal cloud options.

3b) Strategise

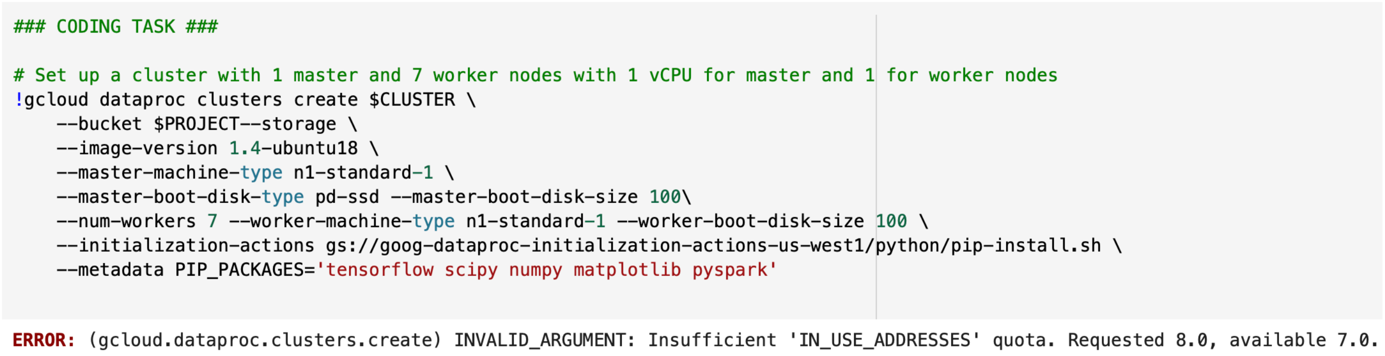
Define - as far as possible - concrete strategies for different application scenarios (batch, stream) and discuss the general relationship with the concepts above. (12%)

In the mentioned research paper, the data collection process involved conducting a batch processing speed test to gather relevant data. The CPU's broad range of configuration options enables users to customize it according to their specific requirements. The introduction of Bayesian optimization techniques has made cherry-picking a viable approach. By adopting cherry-picking, it becomes crucial to extract significant data, while simultaneously adjusting the pace of micro batch processing based on various factors, such as the specific context and user requirements.

Cherry-picking also proves to be an effective strategy for controlling the price movements as it focuses on extracting the most critical information. However, since this technique deals with multiple elements, there is a drawback of network delay involved. Furthermore, the probability of maintaining an optimal setup which is also consistent at all times is compounded by the availability of vast number of options in Google Cloud Platform (GCP). To address this challenge, systems like Cherry Pick are useful, which leverage extensive frameworks to identify the most appropriate configuration for a given task. It's important to note that the technique described above incurs a computational time cost, as it involves searching through a large number of configurations.

Word Count: 1241

Quota related errors encountered:

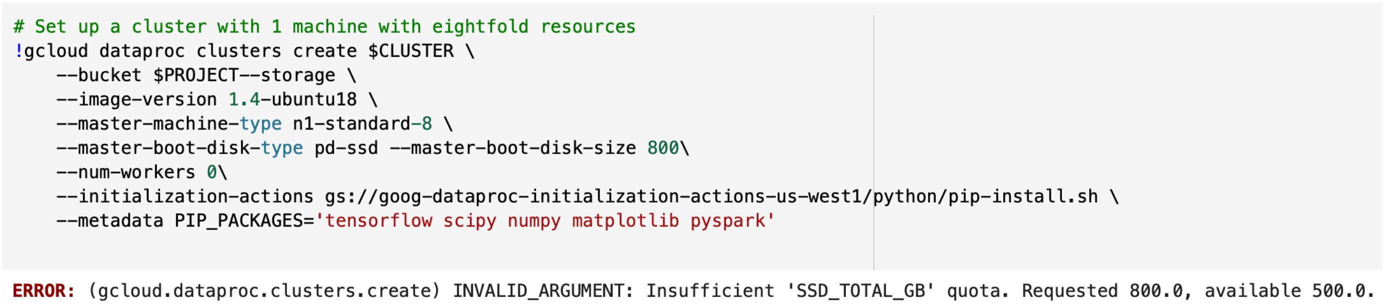


```
### CODING TASK ###

# Set up a cluster with 1 master and 7 worker nodes with 1 vCPU for master and 1 for worker nodes
!gcloud dataproc clusters create $CLUSTER \
    --bucket $PROJECT--storage \
    --image-version 1.4-ubuntu18 \
    --master-machine-type n1-standard-1 \
    --master-boot-disk-type pd-ssd --master-boot-disk-size 100\
    --num-workers 7 --worker-machine-type n1-standard-1 --worker-boot-disk-size 100 \
    --initialization-actions gs://goog-dataproc-initialization-actions-us-west1/python/pip-install.sh \
    --metadata PIP_PACKAGES='tensorflow scipy numpy matplotlib pyspark'
```

```
ERROR: (gcloud.dataproc.clusters.create) INVALID_ARGUMENT: Insufficient 'IN_USE_ADDRESSES' quota. Requested 8.0, available 7.0.
```

The above error occurred while trying to create a cluster with 1 master and 7 worker nodes. Hence, worker nodes were reduced to 6 later.



```
# Set up a cluster with 1 machine with eightfold resources
!gcloud dataproc clusters create $CLUSTER \
    --bucket $PROJECT--storage \
    --image-version 1.4-ubuntu18 \
    --master-machine-type n1-standard-8 \
    --master-boot-disk-type pd-ssd --master-boot-disk-size 800\
    --num-workers 0\
    --initialization-actions gs://goog-dataproc-initialization-actions-us-west1/python/pip-install.sh \
    --metadata PIP_PACKAGES='tensorflow scipy numpy matplotlib pyspark'
```

```
ERROR: (gcloud.dataproc.clusters.create) INVALID_ARGUMENT: Insufficient 'SSD_TOTAL_GB' quota. Requested 800.0, available 500.0.
```

The above error occurred while trying to create a cluster with 1 master machine with 8-fold resources. Hence, disk size was reduced accordingly.

References:

- *Where developers learn, share, & build careers* (no date) *Stack Overflow*. Available at: https://stackoverflow.com/ (Accessed: 02 May 2024).
- *Where good ideas find you.* (no date) *Medium*. Available at: https://medium.com/ (Accessed: 02 May 2024).