1d), i)

We use the same cluster for both reports, however we add the second parameter for the report of the second screenshot. More specifically, following the configuration from the previous task, we used the “maximum use of resources possible) with a configuration of 1 master with 1vCPU and 7 workers with 1 vCPU.

Before applying second parameter:

Screenshot 1. Metrics of the cluster without explicit parallelism.

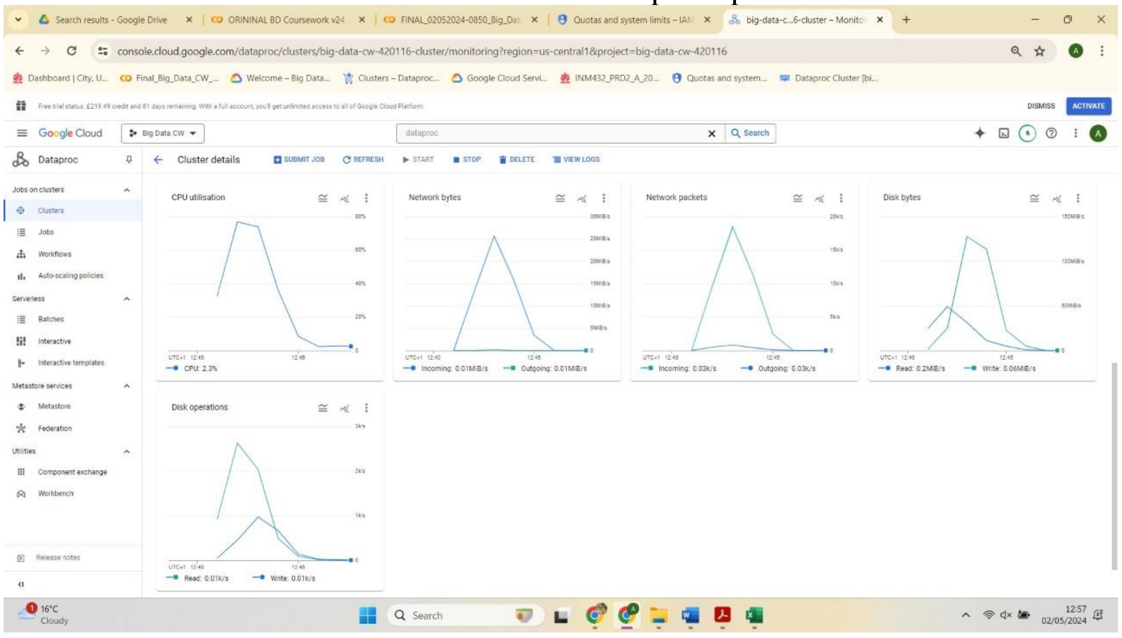


Table 1. Detailed metrics of the cluster without explicit parallelism

Without Second Parameter					
Metrics	CPU Utilisation	Network Bytes	Network Packets	Disk Bytes	Disk Operations
	%	MiB/s	Packets/s	MiB/s	Packets/s
Max	73.99	N/A	N/A	N/A	N/A
Min	N/A	N/A	N/A	N/A	N/A
Incoming (Max)	N/A	26.66	18.49	N/A	N/A
Outgoing (Max)	N/A	0.12	0.83	N/A	N/A
Read (Max)	N/A	N/A	N/A	31.47	2.63
Write (Max)	N/A	N/A	N/A	127.88	0.98

Execution time of the cluster's code: 261.37 seconds

After applying second parameter:

Screenshot 2. Metrics of the cluster with explicit parallelism.

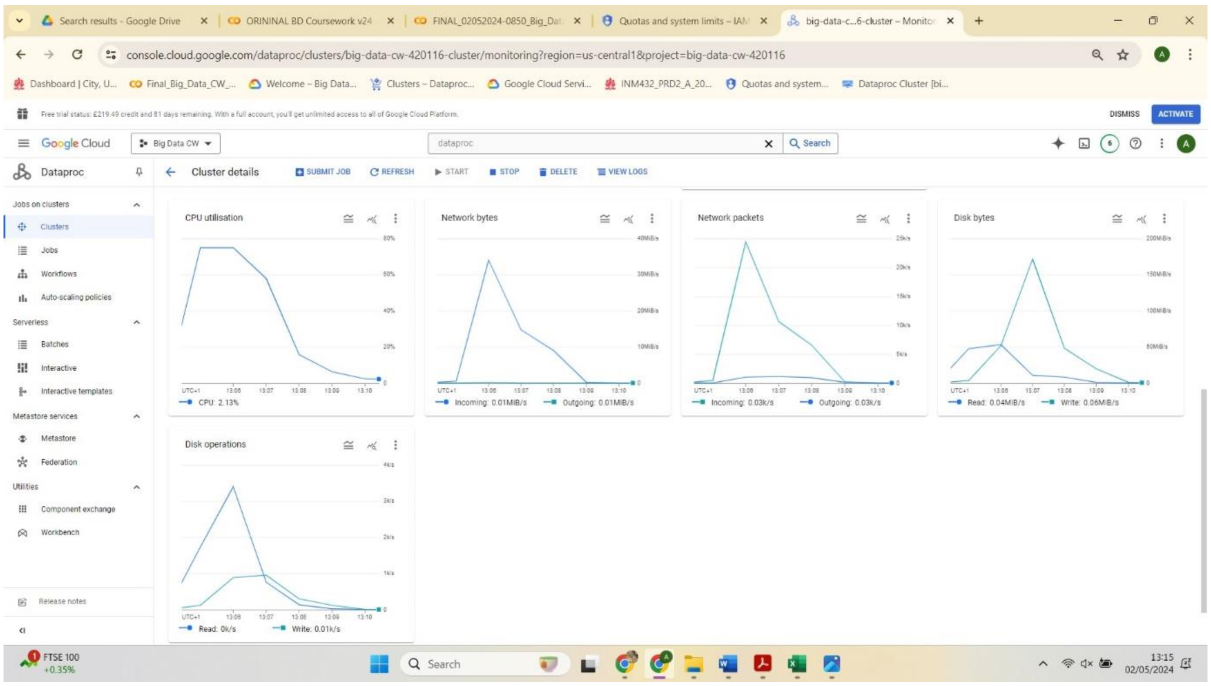


Table 2. Detailed metrics of the cluster with explicit parallelism.

With Second Parameter					
Metrics	CPU Utilisation	Network Bytes	Network Packets	Disk Bytes	Disk Operations
	%	MiB/s	Packets/s	MiB/s	Packets/s
Max	75.03	N/A	N/A	N/A	N/A
Min	N/A	N/A	N/A	N/A	N/A
Incoming (Max)	N/A	33.98	24.48	N/A	N/A
Outgoing (Max)	N/A	0.17	1.17	N/A	N/A
Read (Max)	N/A	N/A	N/A	52.21	3.41
Write (Max)	N/A	N/A	N/A	171.5	0.96

Execution time of the cluster's code: 282.25 seconds

Considering the presented metrics, here are my conclusions:

- The execution time is longer when we define the partition number (16), potentially due to the task itself of partitioning the data across the cluster into that specific number, 16.
- CPU. Reaches a higher rate when we specifically split data into 16 partitions.
- Network bytes: the incoming is higher in the case of using second parameter, this suggests that more data was moved between nodes, possibly due to increasing the shuffling and data distribution.
- Network Utilization: in the same line as prior, it seems that figures for these metrics are higher in the scenario of using second parameter. The reason lying behind is the same, if data is partitioned into more pieces the amount of data being transferred between nodes is larger.

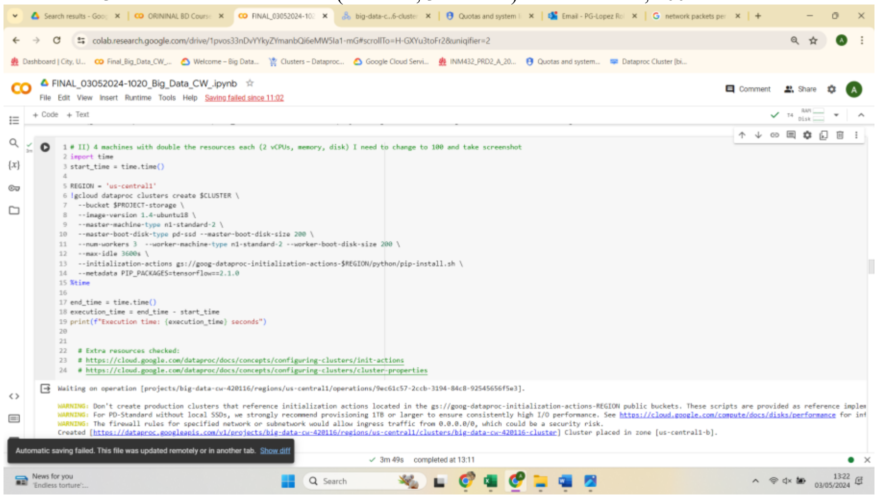
- Disk utilization: with second parameter, the max reached by “read” metric surpasses when second parameter is not used. Point out that the write max is higher for the script without second parameter.

---

1 d), ii)

A) 4 machines (1 master, 3 workers) 2 vCPUs each, 200 disk size

Screenshot 3. Cluster with 4 machines (1 master, 3 workers) 2 vCPUs each, 200 disk size



```
# II) 4 machines with double the resources each (2 vCPUs, memory, disk) I need to change to 100 and take screenshot
import time
start_time = time.time()

REGION = 'us-central1'
!gcloud dataproc clusters create $CLUSTER \
  --bucket $PROJECT-storage \
  --image-version 1.4-ubuntu18 \
  --master-machine-type n1-standard-2 \
  --master-boot-disk-type pd-ssd --master-boot-disk-size 200 \
  --num-workers 3  --worker-machine-type n1-standard-2 --worker-boot-disk-size 200 \
  --max-idle 3600s \
  --initialization-actions gs://goog-dataproc-initialization-actions-$REGION/python/pip-install.sh \
  --metadata PIP_PACKAGES=tensorflow==2.1.0
%time

end_time = time.time()
execution_time = end_time - start_time
print(f"Execution time: {execution_time} seconds")


  # Extra resources checked:
  # https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/init-actions
  # https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/cluster-properties
```

```
Waiting on operation [projects/big-data-cw-420116/regions/us-central1/operations/9ec61c57-2ccb-3194-84c8-92545656f5e3].

WARNING: Don't create production clusters that reference initialization actions located in the gs://goog-dataproc-initialization-actions-REGION public buckets. These scripts are provided as reference implem
WARNING: For PD-Standard without local SSDs, we strongly recommend provisioning 1TB or larger to ensure consistently high I/O performance. See https://cloud.google.com/compute/docs/disks/performance for inf
WARNING: The firewall rules for specified network or subnetwork would allow ingress traffic from 0.0.0.0/0, which could be a security risk.
Created [https://dataproc.googleapis.com/v1/projects/big-data-cw-420116/regions/us-central1/clusters/big-data-cw-420116-cluster] Cluster placed in zone [us-central1-b].
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

✓ 3m 49s completed at 13:11

Screenshot 4. Graphics for a cluster with 4 machines (1 master, 3 workers) 2 vCPUs each, 200 disk size.

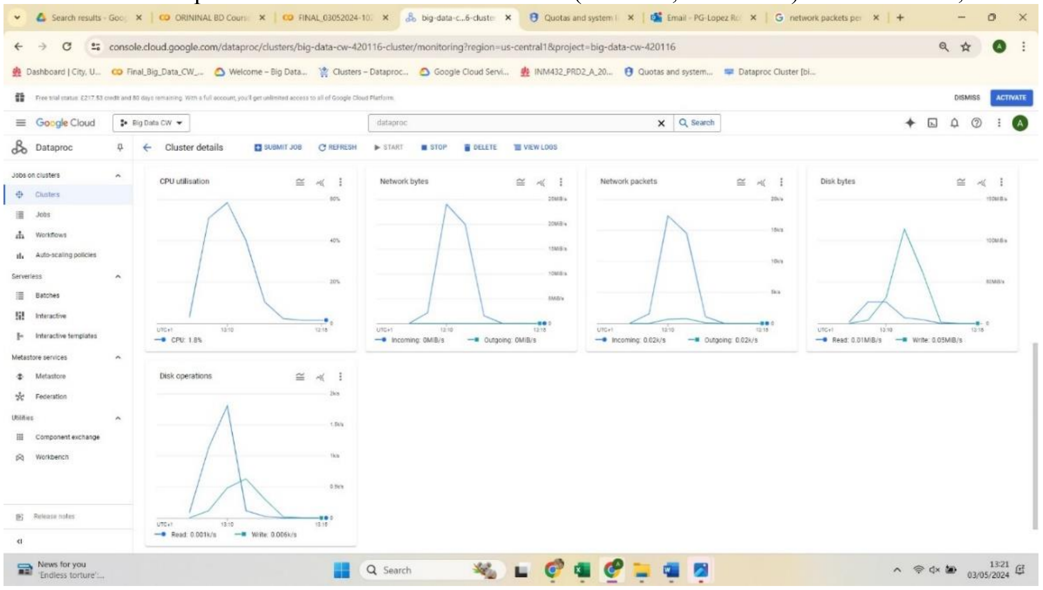


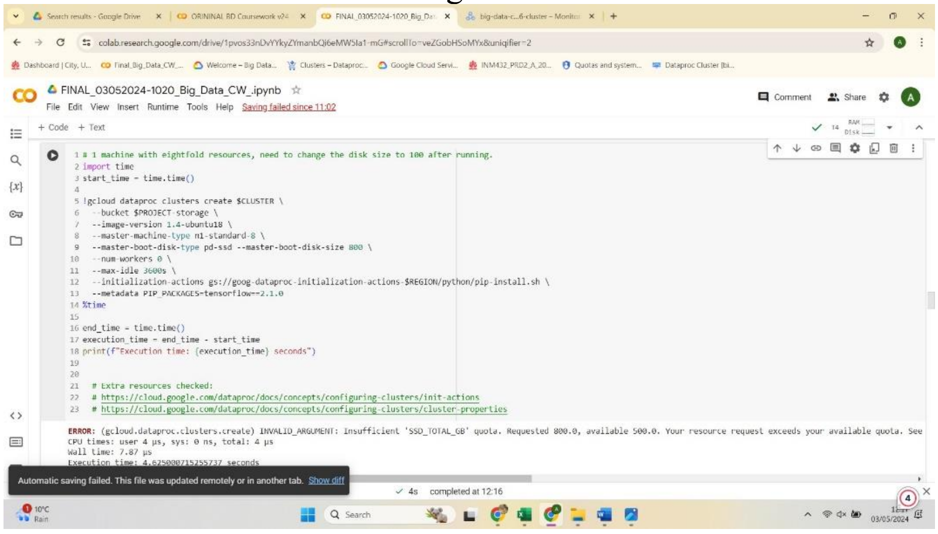
Table 3. Metrics for a cluster with 4 machines (1 master, 3 workers) 2 vCPUs each, 200 disk size.

4 machines: 1 master, 3 workers| 2 vCPUs each| 200 disk size					
Metrics	CPU Utilisation	Network Bytes	Network Packets	Disk Bytes	Disk Operations
	%	MiB/s	k/s	MiB/s	k/s
Max	58.63	N/A	N/A	N/A	N/A
Min	N/A	N/A	N/A	N/A	N/A
Incoming (Max)	N/A	24.05	17.35	N/A	N/A
Outgoing (Max)	N/A	0.09	0.74	N/A	N/A
Read (Max)	N/A	N/A	N/A	26.13	1.81
Write (Max)	N/A	N/A	N/A	114.38	0.63

Execution time of the cluster's code: 229seconds

B) 1 machine with eightfold resources. ERROR. I

Screenshot 5. 1 machine with eightfold resources. ERROR. I


colab.research.google.com/drive/1pvos33nDvYYkyZYmanbQi6eMW5Ia1-mG#scrollTo=veZGobHSoMYx&uniqifier=2

FINAL_03052024-1020_Big_Data_CW_.ipynb

Saving failed since 11:02

```
# 1 machine with eightfold resources, need to change the disk size to 100 after running.
import time
start_time = time.time()

!gcloud dataproc clusters create $CLUSTER \
  --bucket $PROJECT-storage \
  --image-version 1.4-ubuntu18 \
  --master-machine-type n1-standard-8 \
  --master-boot-disk-type pd-ssd --master-boot-disk-size 800 \
  --num-workers 0 \
  --max-idle 3600s \
  --initialization-actions gs://goog-dataproc-initialization-actions-$REGION/python/pip-install.sh \
  --metadata PIP_PACKAGES=tensorflow==2.1.0
%time

end_time = time.time()
execution_time = end_time - start_time
print(f"Execution time: {execution_time} seconds")


  # Extra resources checked:
  # https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/init-actions
  # https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/cluster-properties
```

```
ERROR: (gcloud.dataproc.clusters.create) INVALID_ARGUMENT: Insufficient 'SSD_TOTAL_GB' quota. Requested 800.0, available 500.0. Your resource request exceeds your available quota. See
CPU times: user 4 µs, sys: 0 ns, total: 4 µs
Wall time: 7.87 µs
Execution time: 4.625000715255737 seconds
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

4s completed at 12:16

Screenshot 6. Graphics for 1 machine with eightfold resources

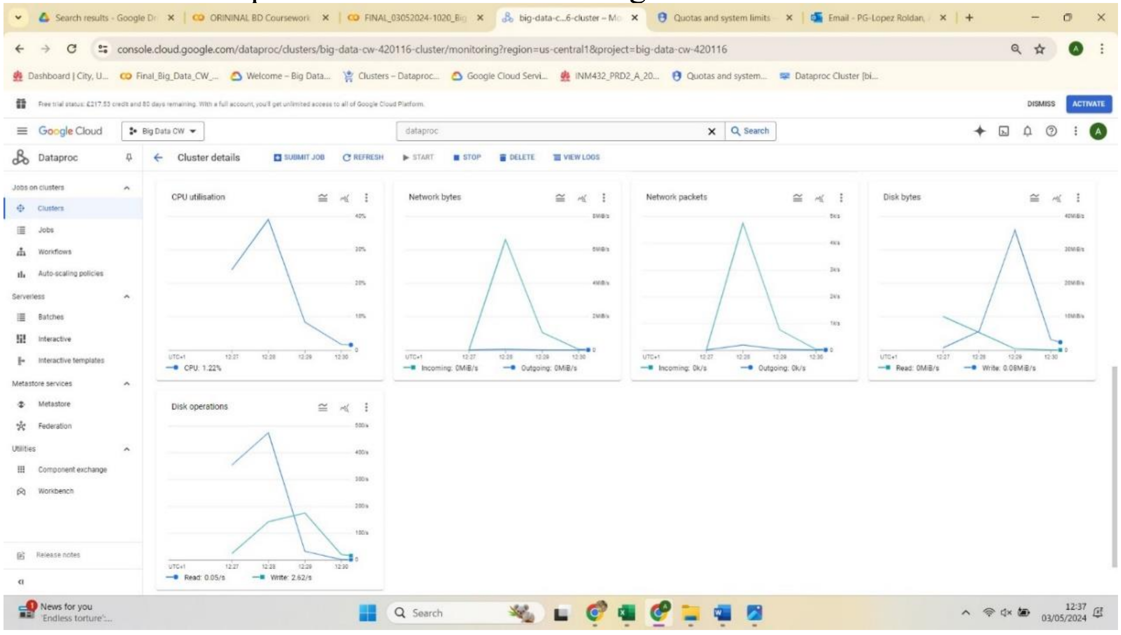


Table 4. Metrics for 1 machine with eightfold resources

1 machine with eightfold resources					
Metrics	CPU Utilisation	Network Bytes	Network Packets	Disk Bytes	Disk Operations
	%	MiB/s	Packets/s	MiB/s	Packets/s
Max	39.13	N/A	N/A	N/A	N/A
Min	N/A	N/A	N/A	N/A	N/A
Incoming (Max)	N/A	6.6	4.75	N/A	N/A
Outgoing (Max)	N/A	0.04	0.19	N/A	N/A
Read (Max)	N/A	N/A	N/A	5.38	474.07
Write (Max)	N/A	N/A	N/A	35.91	174.55

Execution time of the cluster's code: 165 seconds

a) Disk I/O allocation in the cloud. From my point of view, the most relevant is that 4 machines achieve higher disk read and write throughput, compared to the use of one single machine. On the other hand, a single machine has drastically higher disk operation counts, more specifically in reads. Therefore, it seems that when handling the workload, a single machine is more intensive in disk operation.

b) Network Bandwidth allocation in the cloud: The 4 machines cluster has an almost 4 times higher incoming network suggesting that distributing the workload across different machines improves the performance. In other words, the single machine can't keep up with the demands of the workload.

1d), iii)

Here Spark is used with a cloud-based storage, using buckets. On the other hand, in our labs we used Spark integrated with Hadoop Distributed File System (HDFS) or accessible from a file system. The implications of this in terms of where the data is stored are:

- When data is stored on HDFS or a file system and we use Spark to leverage data, Spark uses built-in optimizations. Data is usually preloaded and persisted in RDDs/data frames.
- Using Google Cloud storage, data is "dispersed" across the network rather than locally. This can potentially lead to higher latency and bandwidth constraints, although this way provides higher flexibility and scalability which is the strongest point in favour when dealing with large datasets that are not in the same geographical area.
- Also, for the case of Cloud Storage, having expertise in understanding and optimizing the environment as the different configurations is crucial because the built-in optimizations for the data that is stored locally (HDFS) are not inherently present.

Documentation consulted for this part:

[1]
M. M. Saeed, Z. Al Aghbari, and M. Alsharidah, ‘Big data clustering techniques based on Spark: a literature review’, *PeerJ Computer Science*, vol. 6, p. e321, Nov. 2020, doi: 10.7717/peerj-cs.321.

[2]
J. S. Damji, B. Wenig, T. Das, and D. Lee, ‘Learning Spark, Second Edition’.

2b); ii)

Screenshot 7. Job Without cache

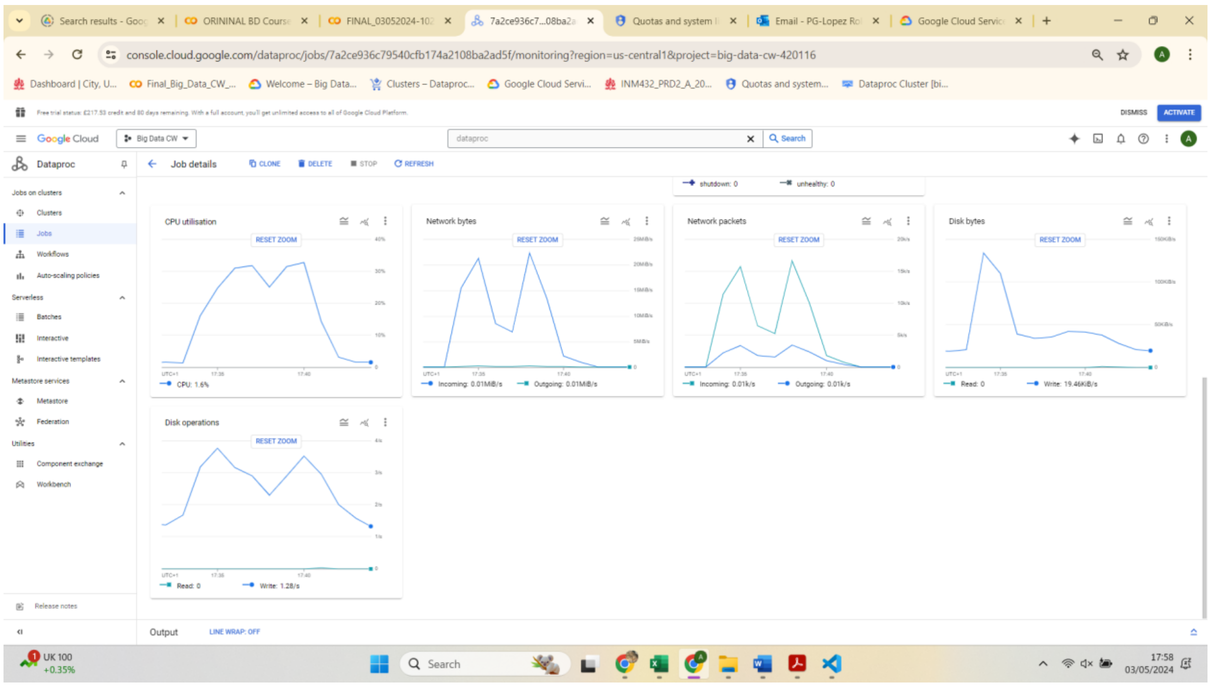


Table 5. Submitting job without “.cache()”

Submitting job without .cache()					
Metrics	CPU Utilisation	Network Bytes	Network Packets	Disk Bytes	Disk Operations
	%	MiB/s	k/s	KiB/s	Packets/s
Max	32.68	N/A	N/A	N/A	N/A
Min	N/A	N/A	N/A	N/A	N/A
Incoming (Max)	N/A	22.34	16.56	N/A	N/A
Outgoing (Max)	N/A	0.22	3.5	N/A	N/A
Read (Max)	N/A	N/A	N/A	0.88	0.02
Write (Max)	N/A	N/A	N/A	133.77	3.77

Execution time of the cluster's code: 7 Min 11 Sec

Screenshot 8. Job without “.cache()”

2c)

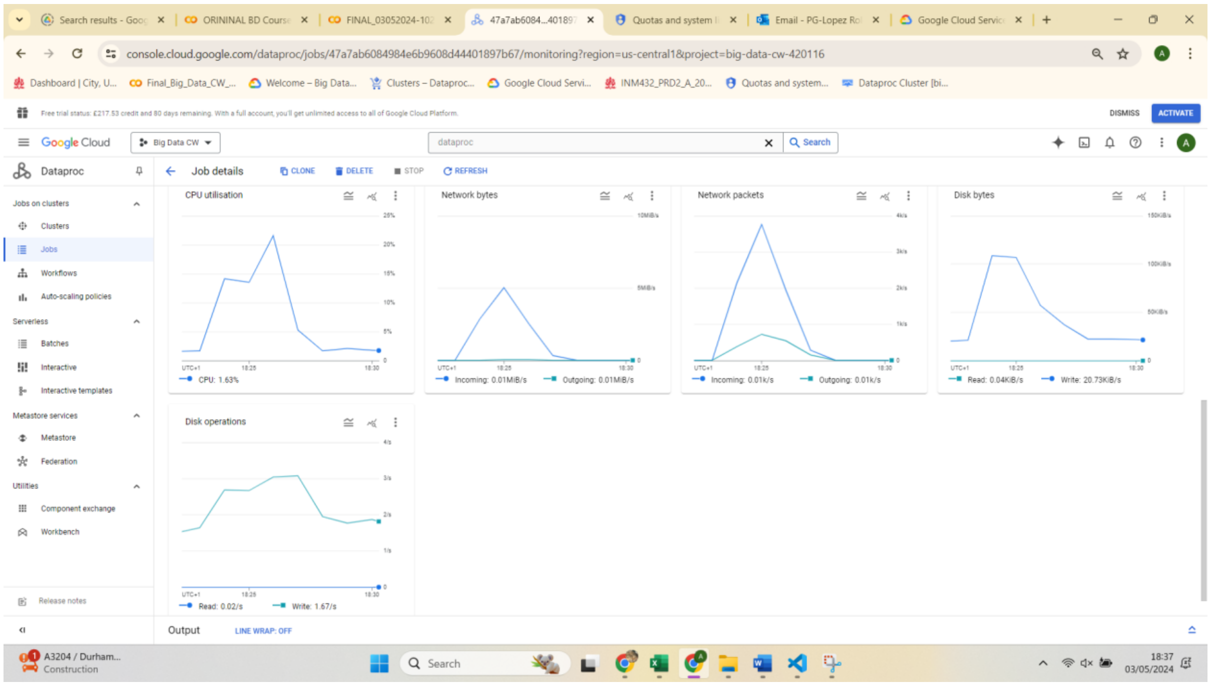
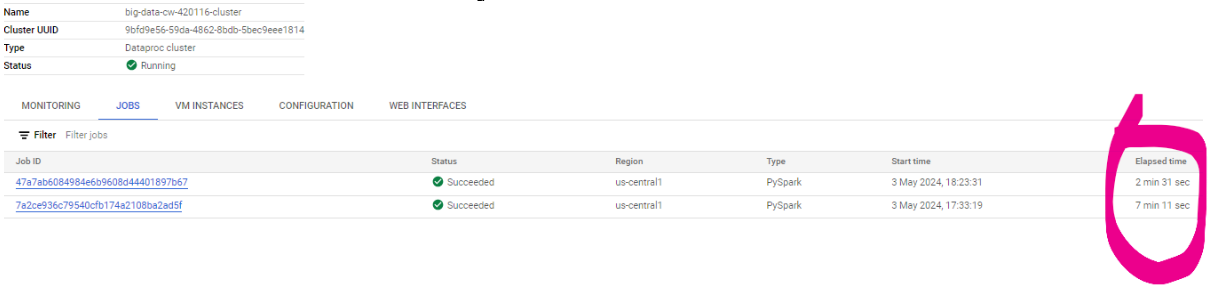


Table 6. Submitting job with ".cache()"

Submitting job with .cache()					
Metrics	CPU Utilisation	Network Bytes	Network Packets	Disk Bytes	Disk Operations
	%	MiB/s	k/s	KiB/s	Packets/s
Max	21.56	N/A	N/A	N/A	N/A
Min	N/A	N/A	N/A	N/A	N/A
Incoming (Max)	N/A	5.05	3.76	N/A	N/A
Outgoing (Max)	N/A	0.06	0.73	N/A	N/A
Read (Max)	N/A	N/A	N/A	0	0
Write (Max)	N/A	N/A	N/A	108.93	3.08

Execution time of the cluster's code: 2 Min 31 Sec

Screenshot 9. Difference between jobs in time execution



Name	big-data-cw-420116-cluster
Cluster UUID	9bfd9e56-59da-4862-8bdb-5bec9eee1814
Type	Dataproc cluster
Status	Running

MONITORING JOBS VM INSTANCES CONFIGURATION WEB INTERFACES

Filter Filter jobs

Job ID	Status	Region	Type	Start time	Elapsed time
47a7ab6084984e6b9608d44401897b67	Succeeded	us-central1	PySpark	3 May 2024, 18:23:31	2 min 31 sec
7a2ce936c79540cfb174a2108ba2ad5f	Succeeded	us-central1	PySpark	3 May 2024, 17:33:19	7 min 11 sec

The difference between using or not ".cache()" is huge in many metrics, starting with the execution time which is around 3 times less than without .cache(). The aplicaton of "cache" allows us to store certain computations temporarily so we don't need to re-calculate every time we need them during the performance of a task.

2 D) "Discuss to what extent linear modelling reflects the effects we are observing. Discuss what could be expected from a theoretical perspective and what can be useful in practice."

Table 7. Results for the different linear regressions.

Regressors\Metrics	Slope	Intercept	P-Value
TFR BatchSize	31.32	-16.68	0.39
TFR BatchNumber	22.83	25.43	0.46
TFR Repetitions	-1.02	267.59	4.56
TFR Datasetsize	2.7	9.64	0.88
Image BatchSize	8.52	2.2	0.96
Image BatchNumber	0.5	73.56	0.01
Image Repetitions	1.25	75.94	0.002
Image Datasetsize	0.33	47.65	0.44

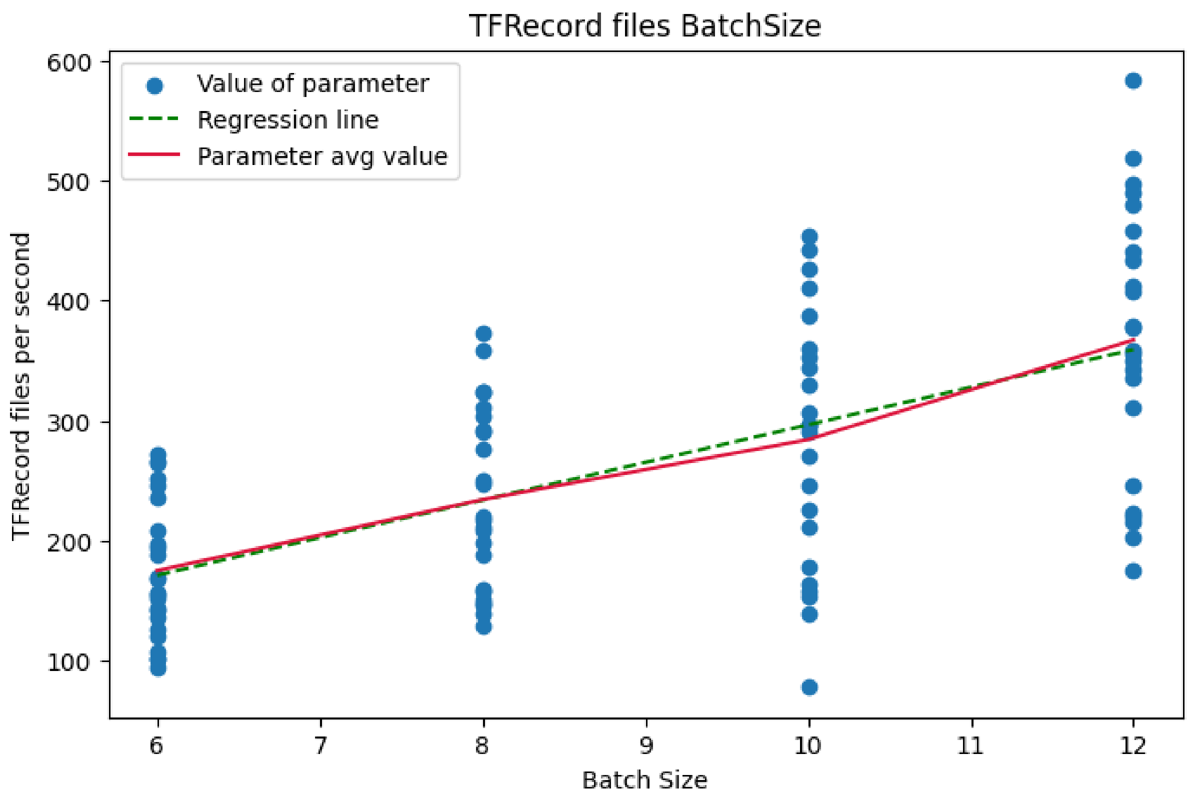
According to the results from the different linear regressions, this is my interpretation:

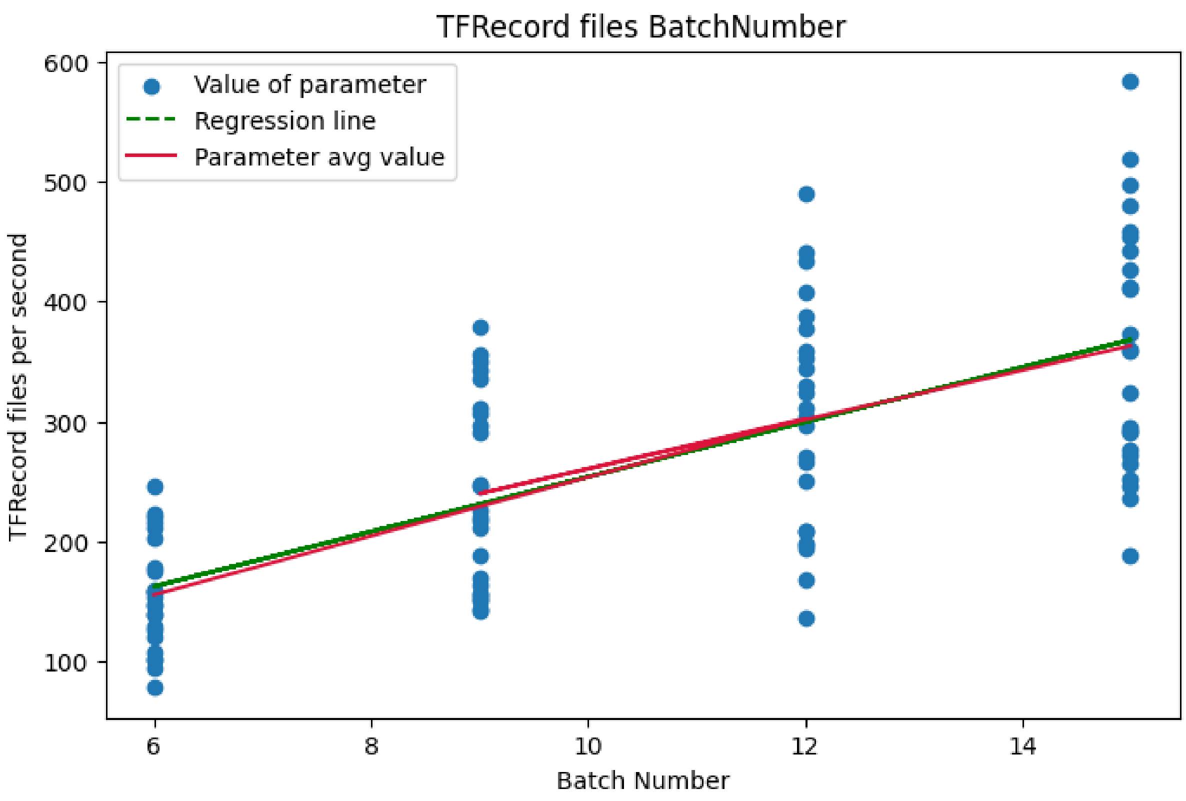
- **TFR BatchSize**
  - **Slope**: 31.32 indicates a strong positive relationship between the TFR BatchSize and the number of files (images) processed per second. As batch size increases, there is a significant increase in the number of files (images) processed per second.
  - **Intercept**: -16.68 suggests that when the TFR BatchSize is zero, the number of files (images) processed per second is negative "what doesn't make much sense".
  - **P-Value**: 0.39 Since is greater than 0.05, suggests that the relationship between TFR BatchSize and the number of files (images) processed per second is not statistically significant.
- **TFR BatchNumber**
  - **Slope**: 22.83 shows a positive relationship, indicating that increases in batch number correlate with increases in the number of files (images) processed per second.
  - **Intercept**: 25.43 suggests a positive start for the number of files (images) processed per second when batch number is zero.
  - **P-Value**: 0.46, not statistically significant.

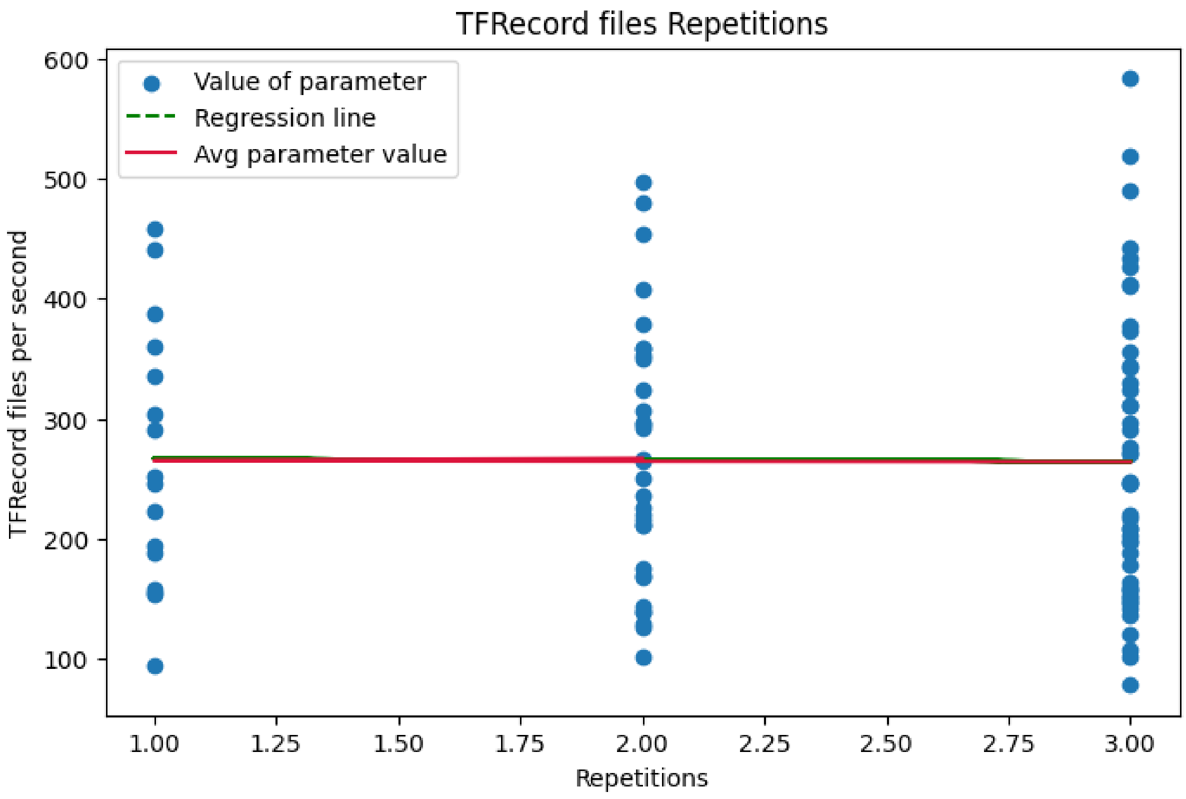
- **TFR Repetitions**
  - **Slope**: -1.02 This negative slope suggests that an increase in the number of repetitions is associated with a decrease in the number of files (images) processed per second.
  - **Intercept**: 267.59 indicates a high starting point for the number of files (images) processed per second when repetitions are zero.
  - **P-Value**: 4.56 indicating that this model is definitely not a good fit and the results are not statistically trustful.
- **TFR Datasize**
  - **Slope**: 2.7, it has a positive relationship with the number of files (images) processed per second.
  - **Intercept**: 9.64 indicating the starting point of the number of files (images) processed per second when datasize is zero.
  - **P-Value**: 0.88 indicating not statistsically significant.
- **Image BatchSize**
  - **Slope**: 8.52, positive relationship indicates that as the image batch size increases as does the number of files (images) processed per second.
  - **Intercept**: 2.2, suggesting a small starting value when the batch size is zero.
  - **P-Value**: 0.96, indicating that this relationship is not statistically significant.
- **Image BatchNumber**
  - **Slope**: 0.5 shows a minimal positive relationship between the batch number and the number of files (images) processed per second.
  - **Intercept**: 73.56, is the initial value of the number of files (images) processed per second when BatchNumber is 0.
  - **P-Value**: 0.01, indicating that this relationship is statistically significant.
- **Image Repetitions**
  - **Slope**: 1.25. This indicates a positive relationship, showing that an increase in the number of repetitions correlates with an increase in the number of files (images) processed per second.
  - **Intercept**: 75.94, this is the number of files (images) processed per second when there are no repetitions.
  - **P-Value**: 0.002, showing a strong significance.
- **Image Datasize**
  - **Slope**: 0.33, indicating a slight positive correlation between the size of the image data and the number of files (images) processed per second.
  - **Intercept**: 47.65 is the number of files (images) processed per second. when datasize is zero.
  - **P-Value**: 0.44, indicating that the relationship is not statistically significant.

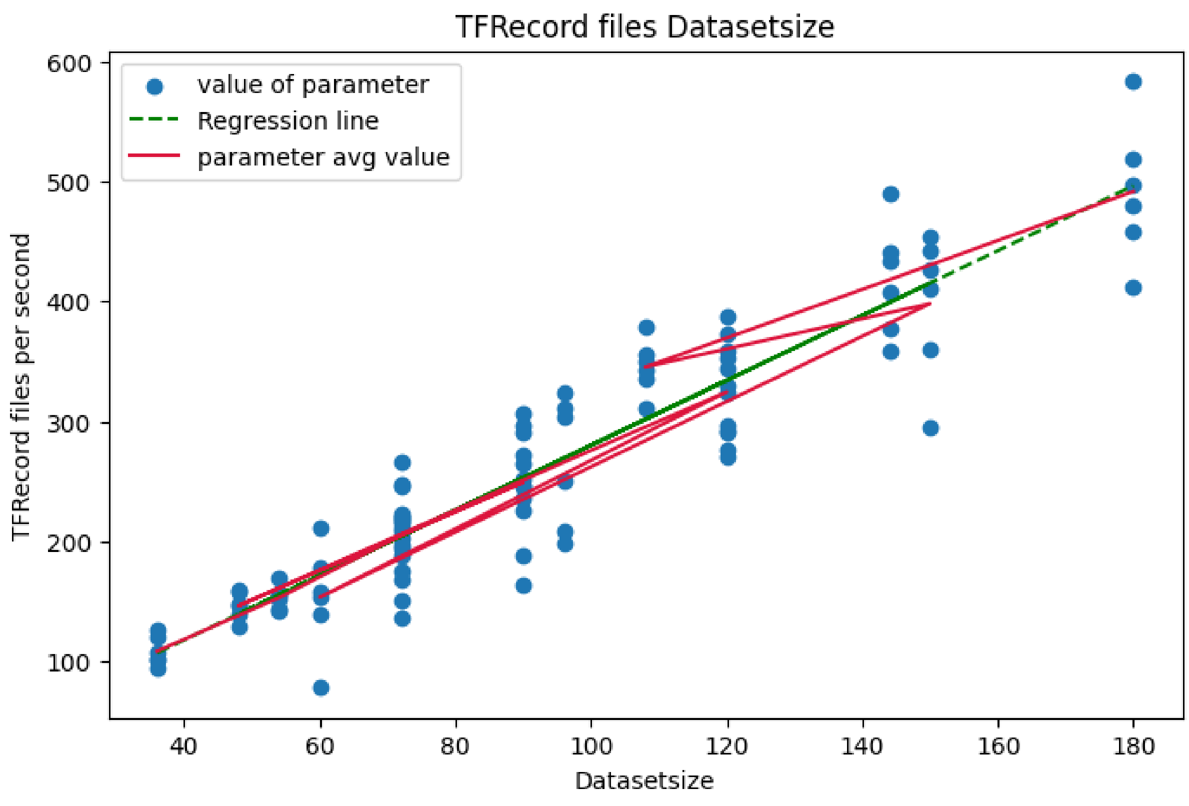
Regarding these interpretations would like to point out that some of the results don't make much sense (for me at least), I might be wrong in my interpretation and would be a matter of a deeper study, however, having a value of ~ "-16" for the **TFR BatchSize intercept**, is odd as mean that when the batch is 0, the number of files processed is negative, if there is no input how can exist a negative "job".

Next, I show the charts for the different linear regressions.




TFRecord files BatchNumber
Value of parameter
Regression line
Parameter avg value
TFRecord files per second
Batch Number
600
500
400
300
200
100
6
8
10
12
14


TFRecord files Repetitions
Value of parameter
Regression line
Avg parameter value
TFRecord files per second
Repetitions
600
500
400
300
200
100
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00


TFRecord files Datasetsize
value of parameter
Regression line
parameter avg value
TFRecord files per second
Datasetsize
600
500
400
300
200
100
40
60
80
100
120
140
160
180

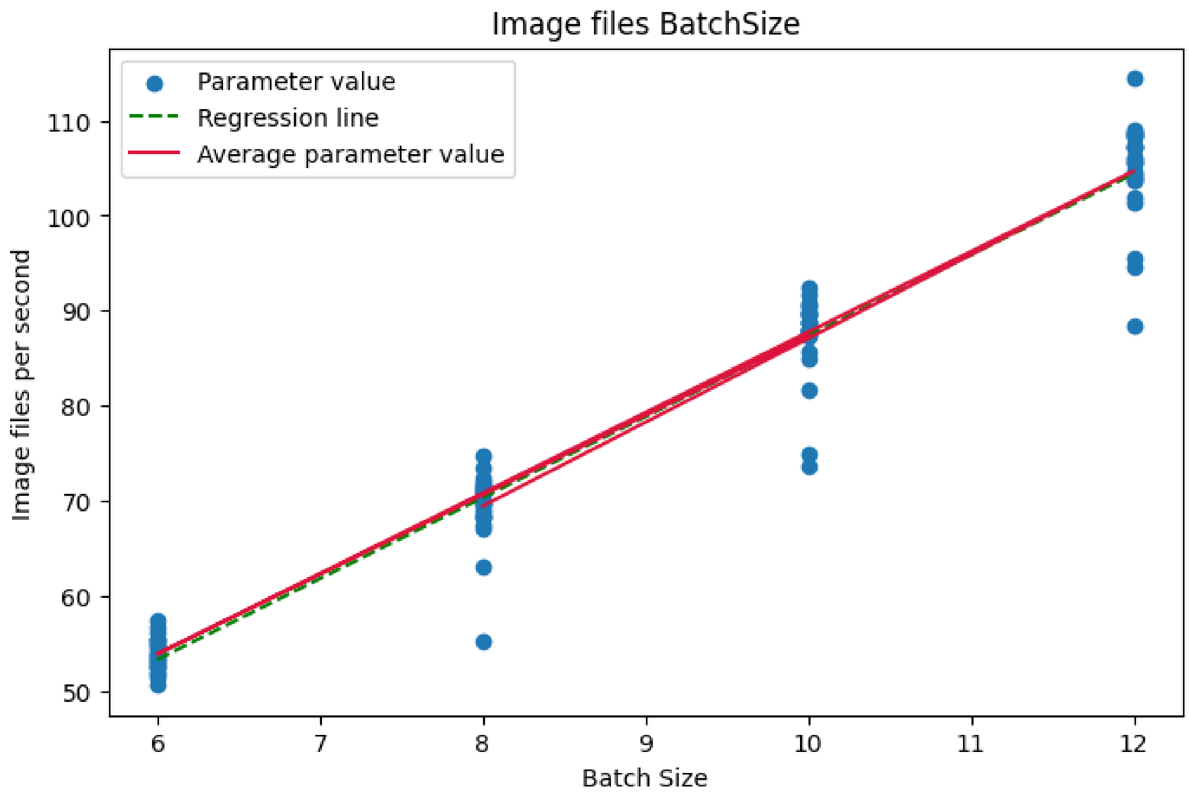


Image files BatchSize
Parameter value
Regression line
Average parameter value
Image files per second
Batch Size
110
100
90
80
70
60
50
6
7
8
9
10
11
12

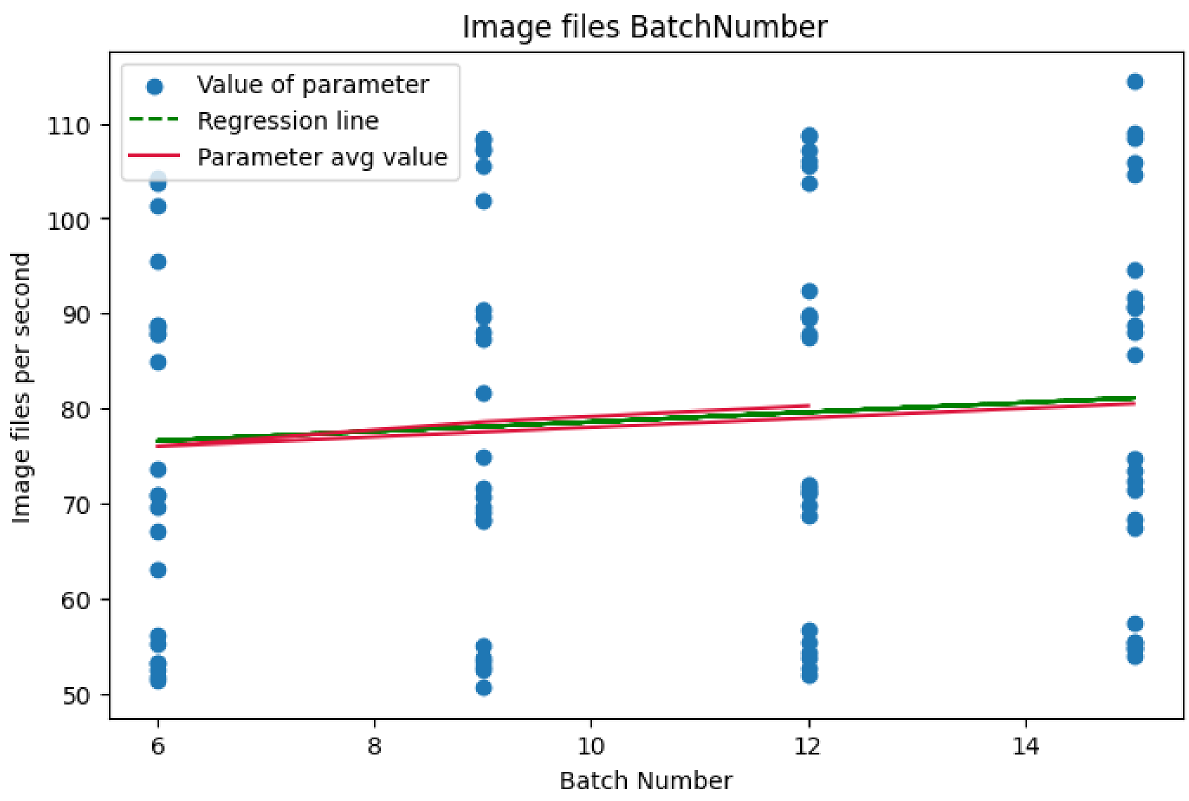


Image files BatchNumber
Value of parameter
Regression line
Parameter avg value
Image files per second
Batch Number
110
100
90
80
70
60
50
6
8
10
12
14

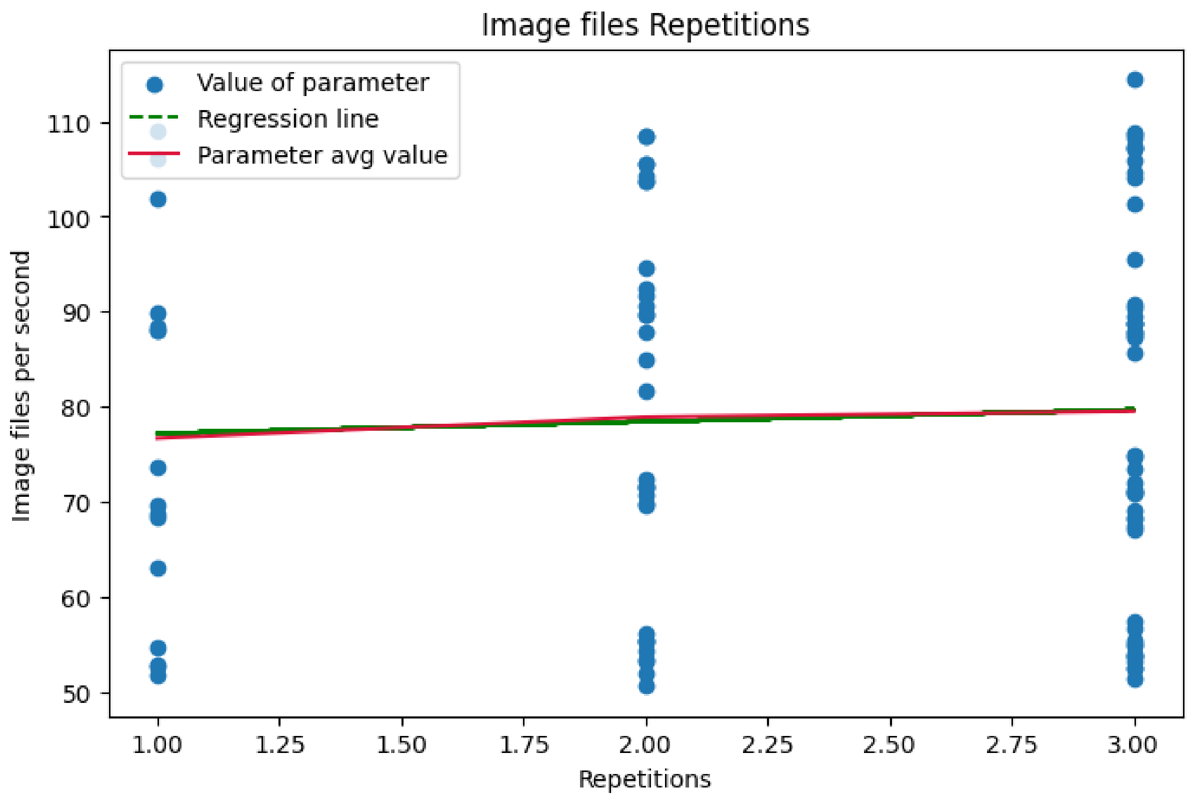
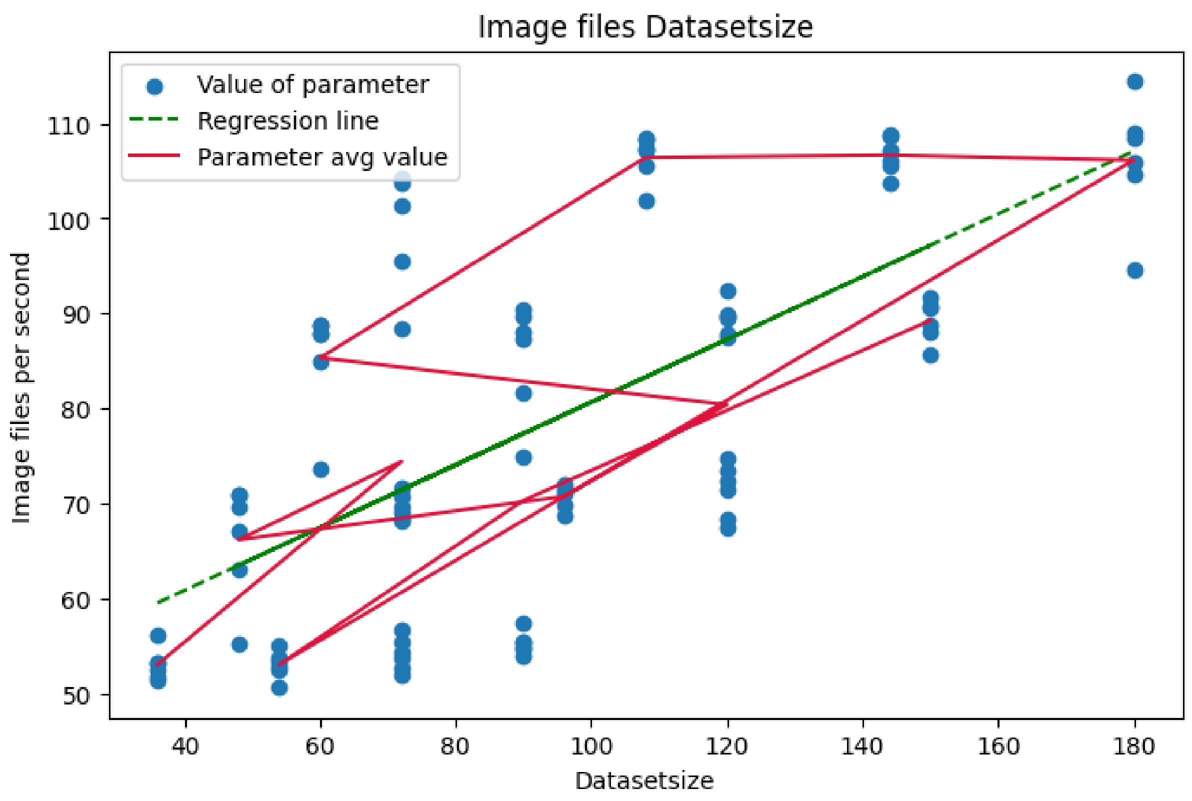


Image files Repetitions
Value of parameter
Regression line
Parameter avg value
Image files per second
Repetitions
110
100
90
80
70
60
50
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00



Task 3: Discussion in context. (24%)

3a) Contextualise.

The goal of the paper is to identify an optimal or near-optimal cloud configuration that minimizes the total execution cost while satisfying performance requirements.

In our case, about image processing tasks involving mapping functions, resizing, recompression, and writing to TFRecord files / Spark, we mainly:

- Tested different configurations for clusters changing master machines, worker machines, vCPU, disk (SSD).
- Defined specific parallelization with a second parameter, as well as, used " .cache()" to compare the efficiency before/after its application and how we can save a massive amount of time when saving jobs (for instance) that require repeatedly dealing with intermediate computations.

In this context, instead of randomly testing configurations, we could leverage CherryPick. I believe there are many possibilities and the following is an example of how we could improve the cloud environment (particularly in the case of a job):

1. First thing we need to do is to define an objective function, let’s say for example, to measure how long it takes to process images to one of our Spark Jobs for a certain configuration.
2. Secondly, we set up the Bayesian optimization defining the search space for cloud configurations specifying the values for each parameter we are considering (master machine, worker machine, etc). At this stage, we also set the algorithm with its configurations (initial samples, e.g.)
3. Iterative optimization:
    a. Run the Spark job with each sample configuration and record the performance.
    b. With the results, improve the Bayesian optimization algorithm.
    c. Then we run again and repeat iteratively.
4. It is important to consider the cloud noise:
    a. We could (as we already kind of did) run multiple trials and create an average to diminish the impact of the cloud noise.
5. Finally, since cloud environments are highly dynamic we should periodically monitor and improve the performance, paying attention to any potential degradation, for instance. Therefore, we should run periodically our Bayesian optimization process.

Besides that, according to the “CherryPick” paper, Bayesian Optimization also faces some challenges, here are some to mention:

- Complex performance model.
- Cost model.
- Heterogeneity of applications.

---

3 b)

Concrete strategies for batch:

1. The optimal size of batch processing can be optimized gathering many samples of batches, gathering information of their performance for a given configuration and the impact on the use of resources (disk I/O, Network Bandwidth, CPU), once we have this information we can apply a Bayesian Optimization (before we explained an example of how doing it)
2. Job speed can be optimized for throughput instead of latency, on the other hand, latency is critical for streaming. In this scenario we can take advantage of the optimization proposed by CherryPick.