

# **Big Data Coursework**

**INM432**

**Arkadiusz Nowacki**

**ID: 220055077**

## Task 1d\_i

Single cluster, SSD size (100), 8vCPUs, 2 partitions (default)

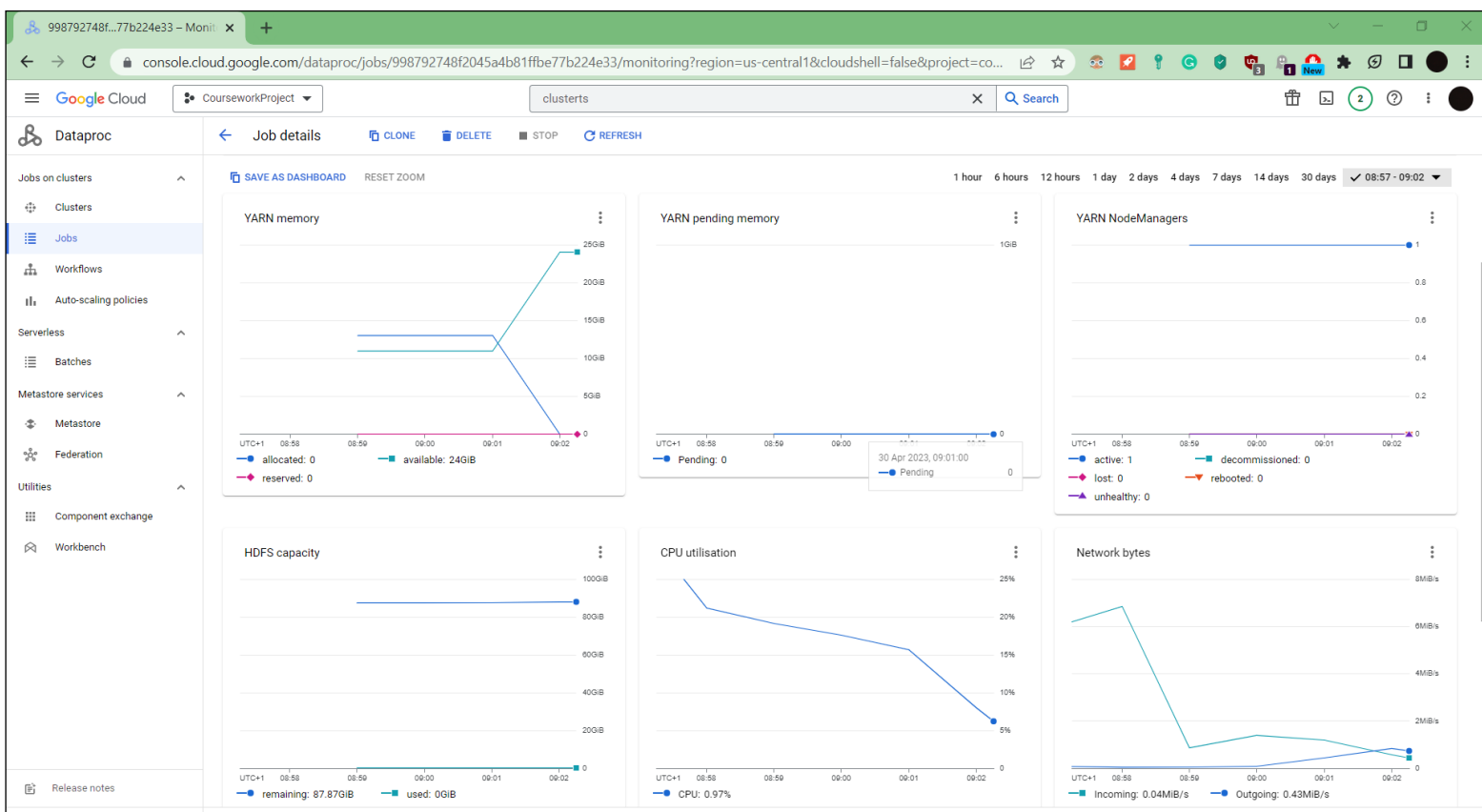
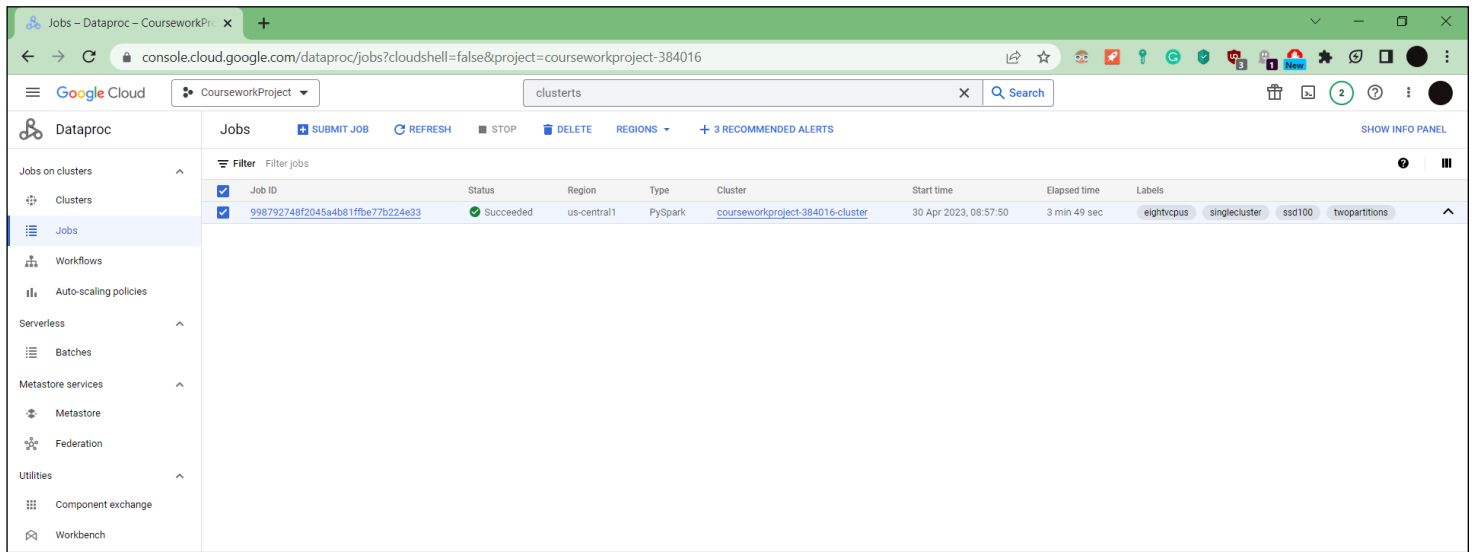


Figure 1 Single cluster (2 partitions) metrics

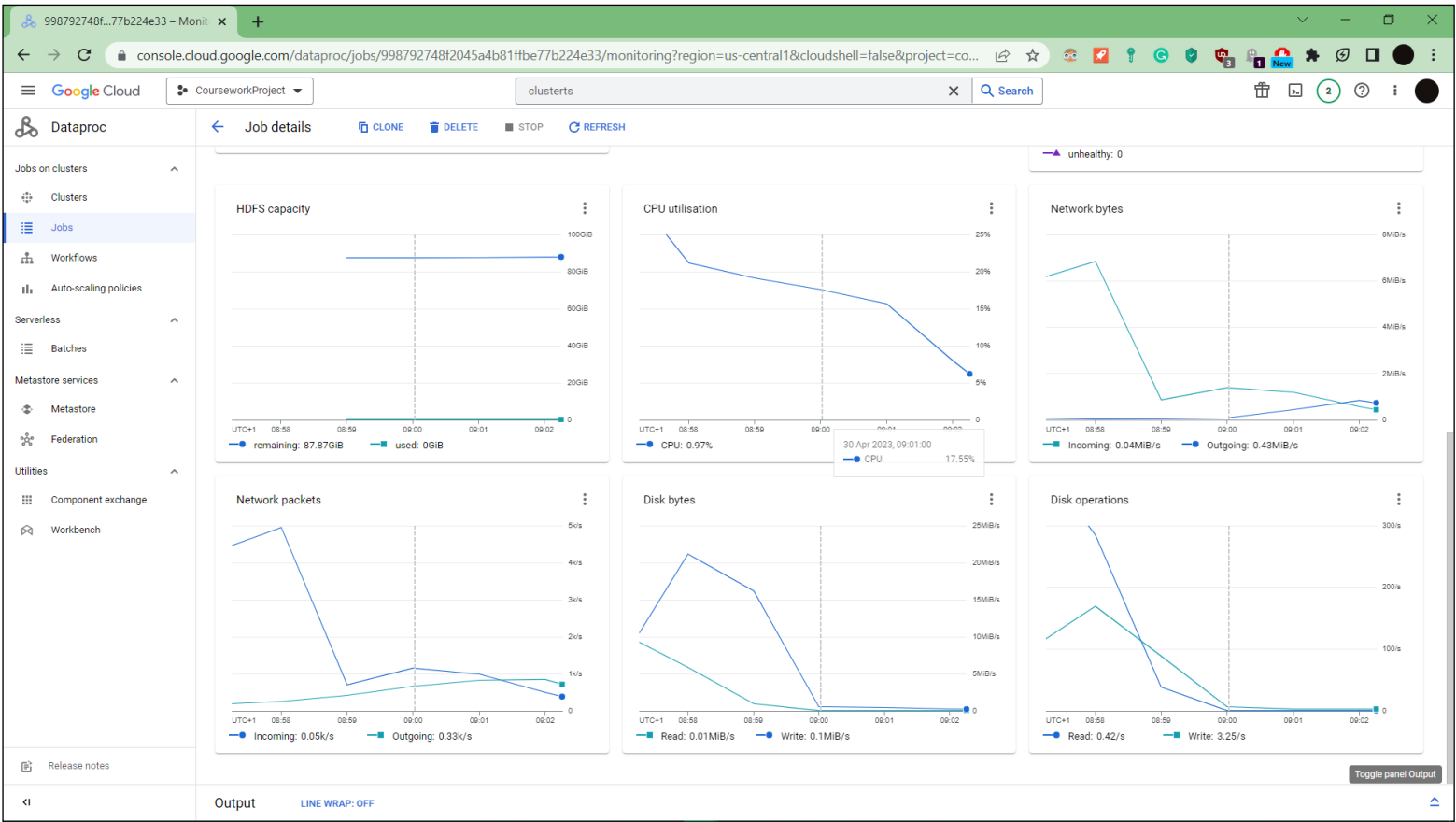
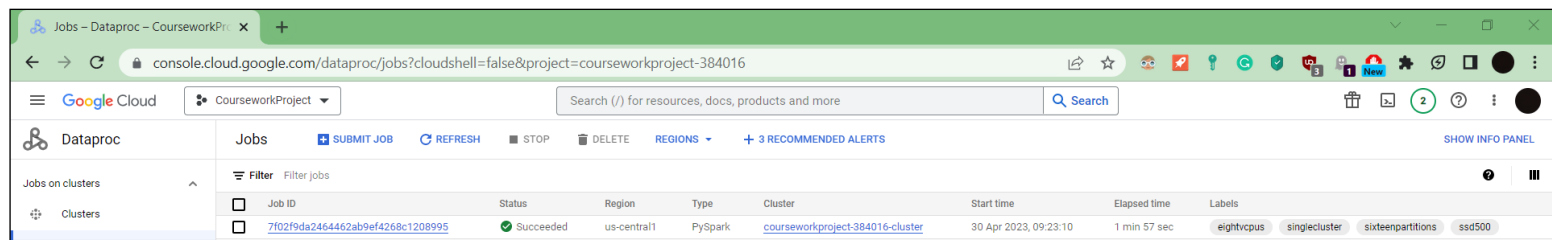


Figure. 2 Single cluster (2 partitions) metrics

Single cluster, SSD size (500), 8vCPUs, 16 partitions (after change)



The screenshot shows the Google Cloud Dataproc console. The 'Jobs' tab is selected, displaying a table of jobs. A single job is listed with ID 7f02f9da2464462ab9ef4268c1208995, status 'Succeeded', region 'us-central1', type 'PySpark', and cluster 'courseworkproject-384016-cluster'. The start time is 30 Apr 2023, 09:23:10, and the elapsed time is 1 min 57 sec. Labels include 'eightvcpus', 'singlecluster', 'sixteenpartitions', and 'ssd500'.

Job ID	Status	Region	Type	Cluster	Start time	Elapsed time	Labels
7f02f9da2464462ab9ef4268c1208995	Succeeded	us-central1	PySpark	courseworkproject-384016-cluster	30 Apr 2023, 09:23:10	1 min 57 sec	eightvcpus singlecluster sixteenpartitions ssd500

Figure 3 Single cluster (16 partitions) Elapsed time

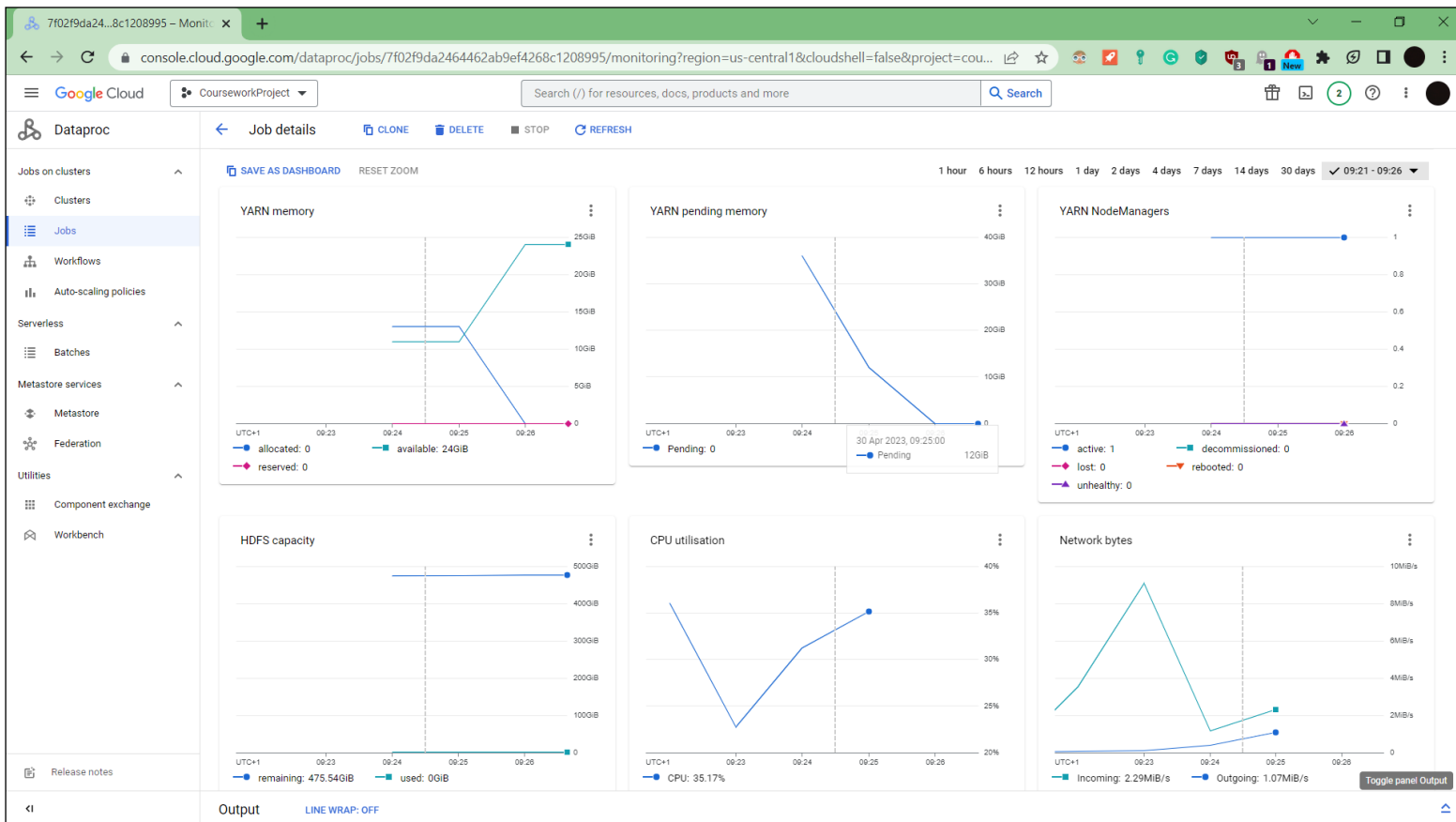


Figure 4 Single cluster (16 partitions) metrics

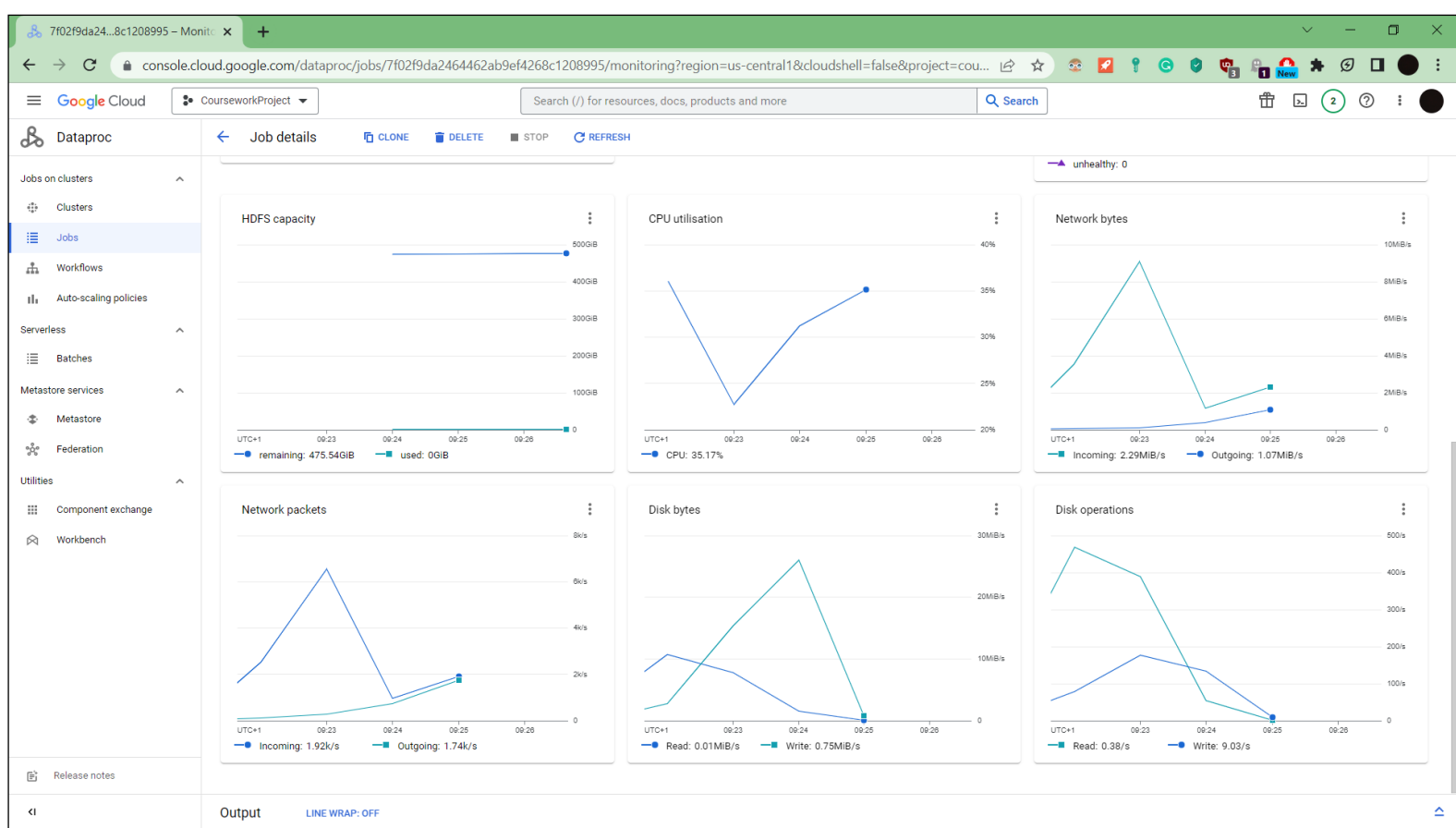


Figure 5 Single cluster (16 partitions) metrics

### Comparison after changes

Based on the Figure 2 we can see that the CPU utilisation was mostly between range of 16% to 25% where it would be approximately utilized 18% most of the time until it would drop. In addition, the elapsed time to fully submit and finish the job in the cluster took 3 minutes and 49 seconds as shown on Figure 1.

Based on the Figure 5 where second parameter is used we can see that the CPU utilisation is higher at the start, about 35% and then it drops to 24% after which it rapidly start to go back to initial value. In addition, the elapsed time to fully submit and finish the job in the cluster took 1 minutes and 57 seconds as shown on Figure 4.

	Network bytes	Network packets
Max incoming	6.84 Mib/s	4.95 k/s
Max outgoing	0.83 Mib/s	0.85 k/s

Table 1 Network bytes/packets of single cluster (2 partitions)

	Disk bytes	Disk operations
Max Read	5.85 Mib/s	284.43 /s
Max Write	16.14 Mib/s	169.58 /s

Table 2 Disk bytes/operations of single cluster (2 partitions)

	CPU utilisation
Max CPU used	21.12 %

Table 3 Disk bytes/operations of single cluster (2 partitions)

	Network bytes	Network packets
Max incoming	9.09 Mib/s	6.56 k/s
Max outgoing	1.07 Mib/s	1.74 k/s

Table 4 Network bytes/packets of single cluster (16 partitions)

	Disk bytes	Disk operations
Max Read	10.69 Mib/s	467.63 /s
Max Write	26.07 Mib/s	177.18 /s

Table 5 Disk bytes/operations of single cluster (16 partitions)

	CPU utilisation
Max CPU used	36.06 %

Table 6 Disk bytes/operations of single cluster (16 partitions)

The final conclusion is that after adding a second parameter that was about changing number of partitions the overall metrics such as network bytes, network packets, disk bytes, disk operations have increased. The significant change was that the overall CPU usage has increased drastically by 14.94% but it is also worth mentioning that the time to finish the job in the cluster dropped significantly by 112 seconds.

Maximal 8 node cluster, 1vCPU each, SSD size (500), evenly distributed worker memory (585), 2 partitions (default)

The screenshot shows the Google Cloud Dataproc Jobs page. The job ID is bbb03ad773c449479af685299d1a435f, which has a status of 'Succeeded'. The job was executed in the us-central1 region using a PySpark cluster named 'courseworkproject-384016-cluster'. The start time was 30 Apr 2023, 09:38:46, and it took 5 minutes and 1 second to complete. The job labels include 'eightnodecluster', 'onevcpueach', 'ssd500', 'twopartitions', and 'workers585'.

Job ID	Status	Region	Type	Cluster	Start time	Elapsed time	Labels
bbb03ad773c449479af685299d1a435f	Succeeded	us-central1	PySpark	courseworkproject-384016-cluster	30 Apr 2023, 09:38:46	5 min 1 sec	eightnodecluster, onevcpueach, ssd500, twopartitions, workers585

Figure 6 Maximal cluster (2 partitions) Elapsed time

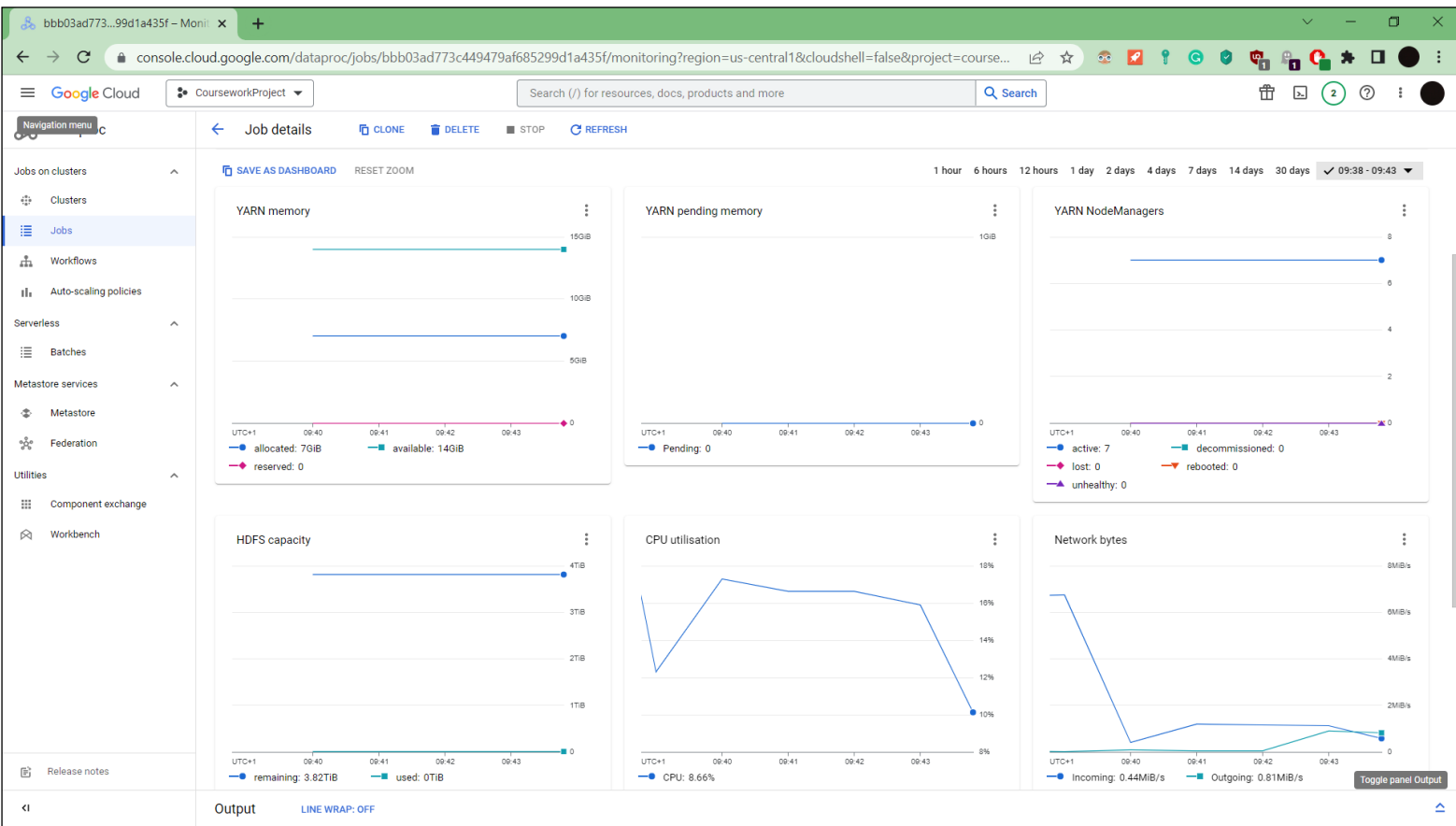


Figure 7 Maximal cluster (2 partitions) metrics

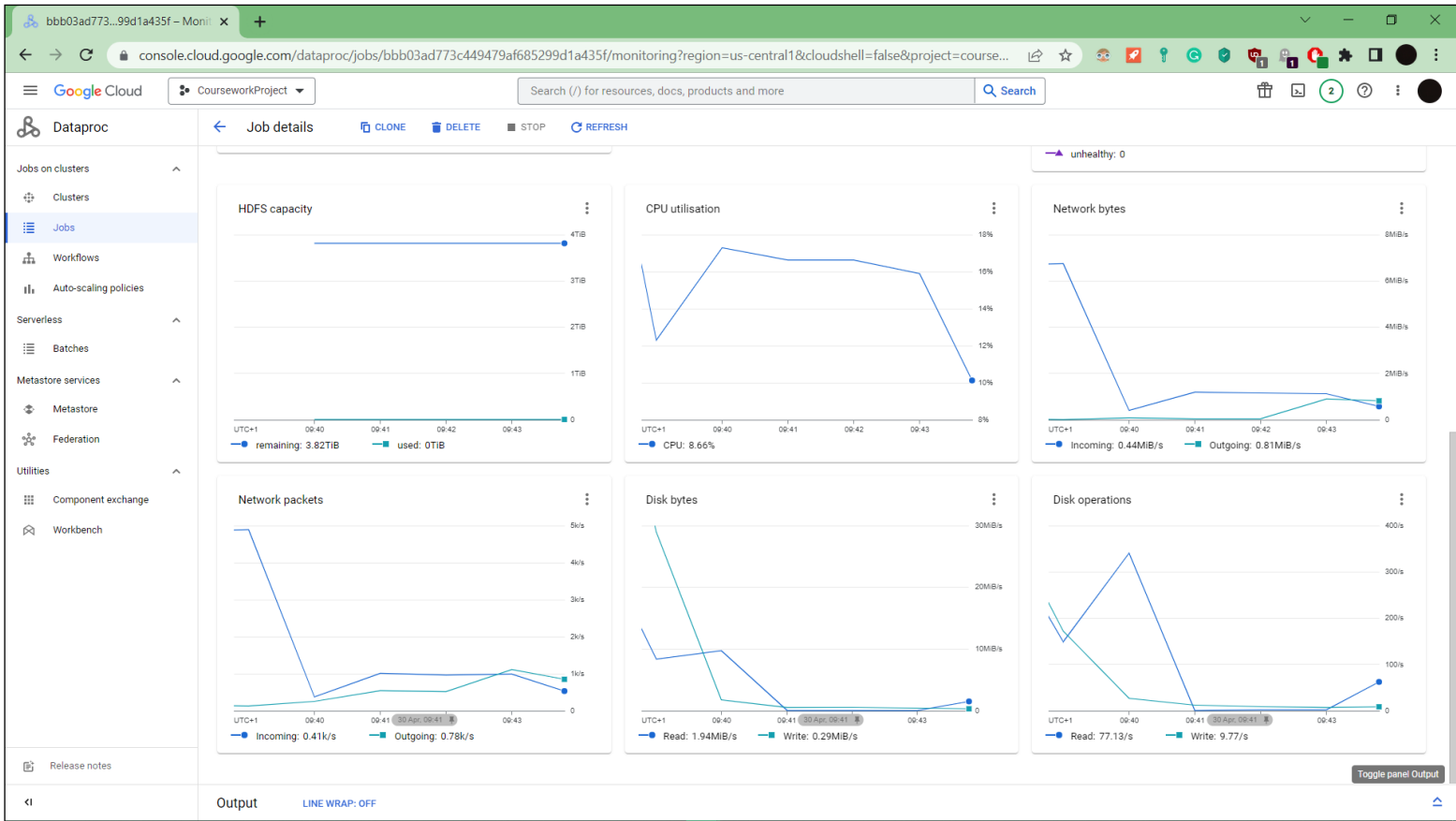


Figure 8 Maximal cluster (2 partitions) metrics



Maximal 8 node cluster, 1vCPU each, SSD size (500), evenly distributed worker memory (585), 16 partitions (after change)

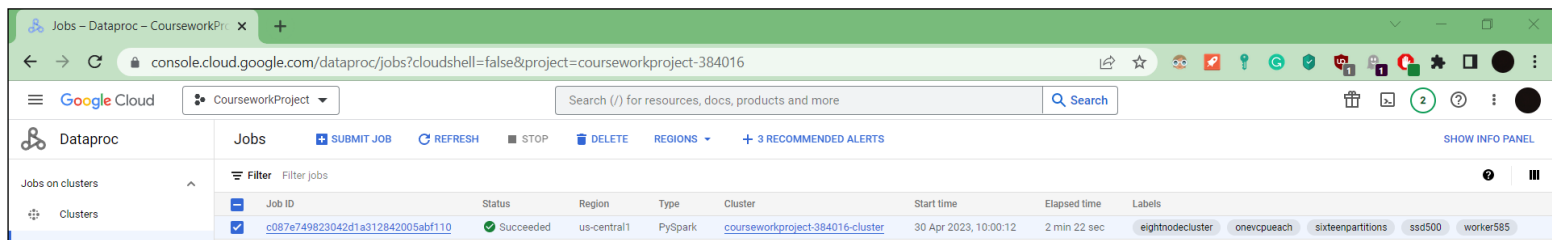


Figure 9 Maximal cluster (16 partitions) Elapsed time

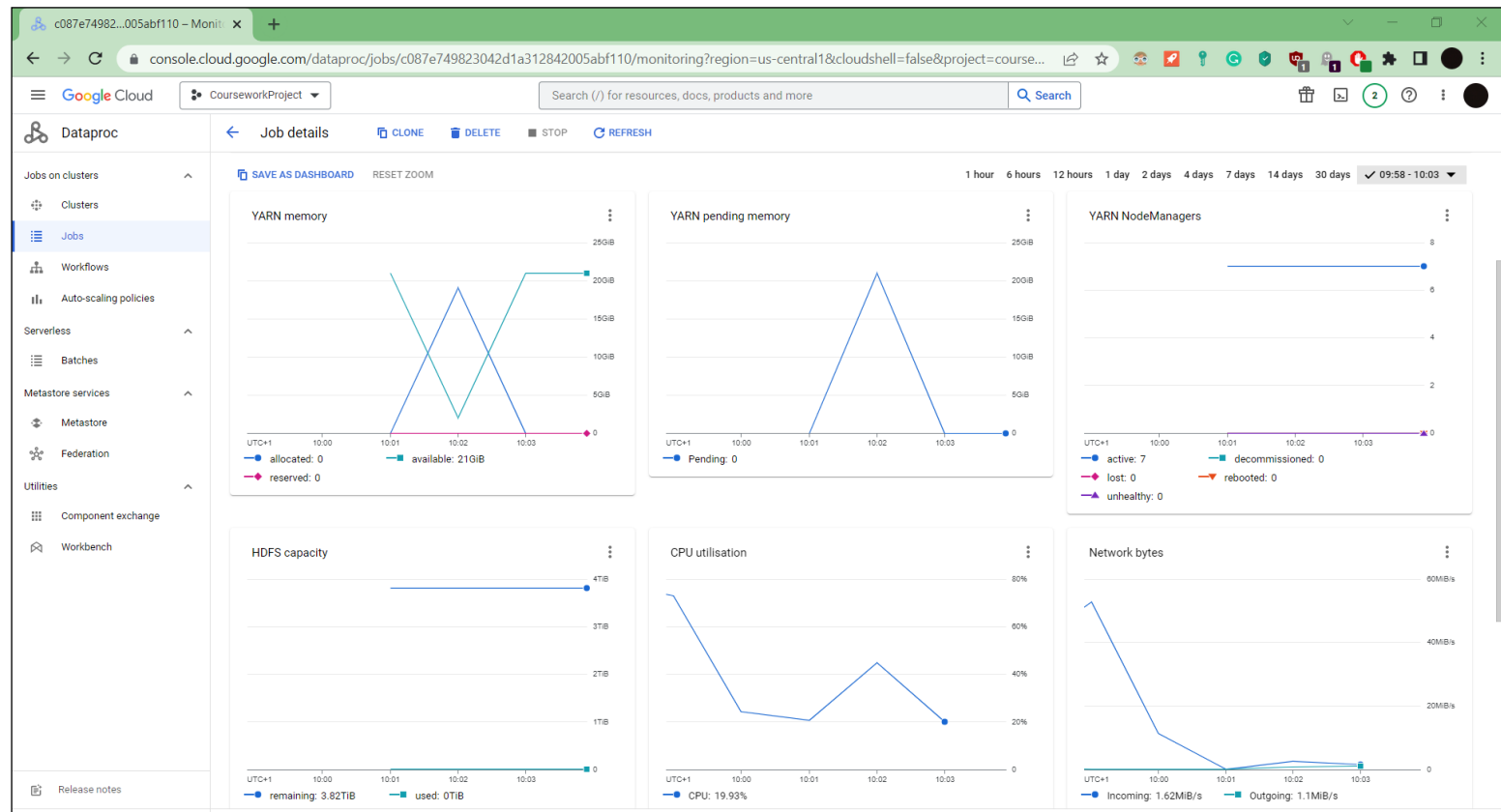


Figure 10 Maximal cluster (16 partitions) metrics

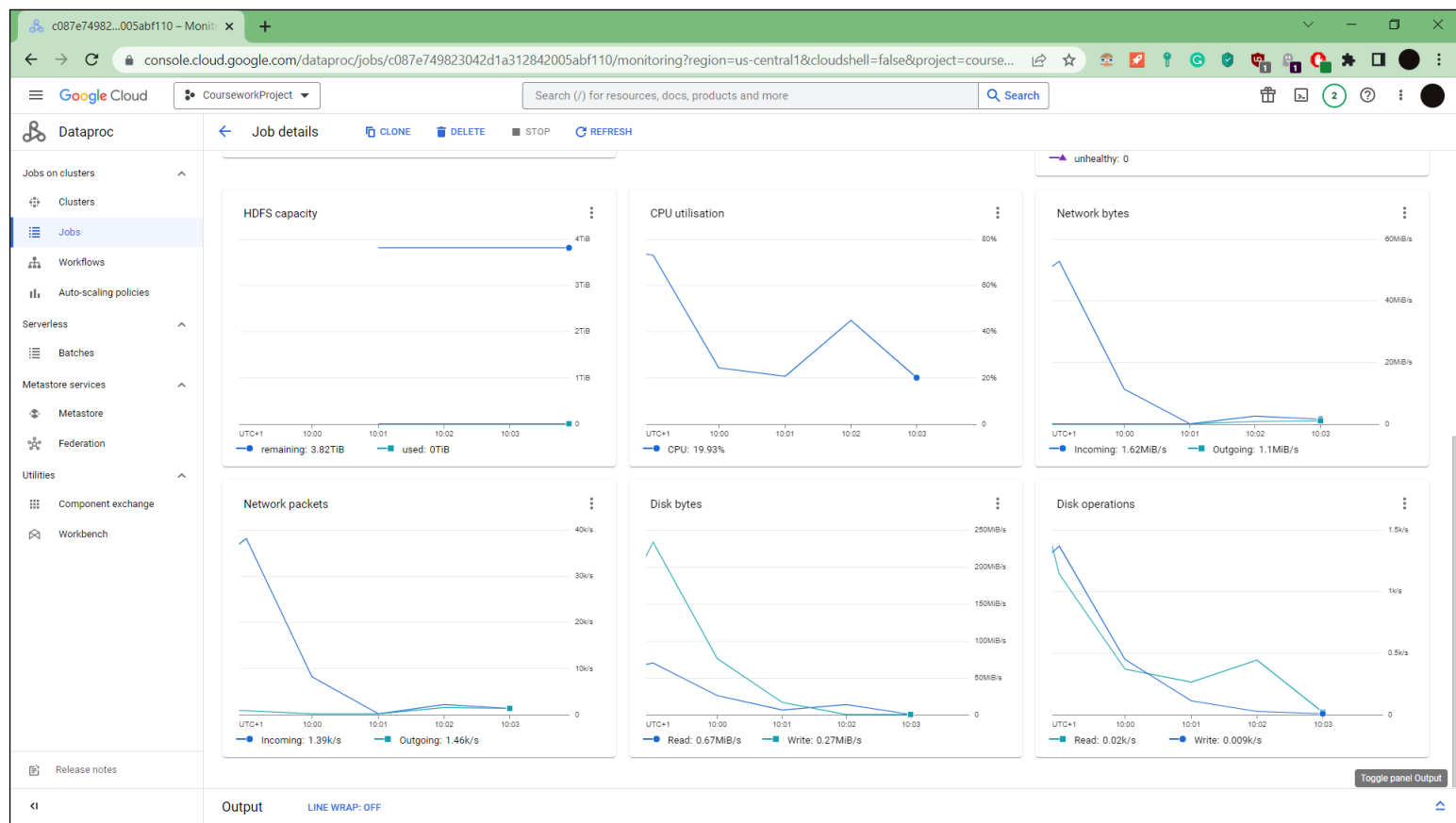


Figure 11 Maximal cluster (16 partitions) metrics

### Comparison after changes

Based on the Figure 7 we can see that the CPU utilisation was mostly between range of 16% to 18% where it would be approximately utilized 17% most of the time until it would smoothly drop to 10%. Moreover, the elapsed time to fully submit and finish the job in the cluster took 5 minutes and 1 second as shown on Figure 6.

Based on the Figure 10 we can see that the CPU utilisation was mostly between range of 21% to 42% where it would be approximately utilized 20-30% most of the time until it would climb to 40% and drop to 20%. Additionally, the elapsed time to fully submit and finish the job in the cluster took 2 minutes and 22 seconds as shown on Figure 9.

	Network bytes	Network packets
Max incoming	6.73 Mib/s	4.88 k/s
Max outgoing	0.88 Mib/s	1.1 k/s

Table 7 Network bytes/packets of maximal cluster (2 partitions)

	Disk bytes	Disk operations
Max Read	9.79 Mib/s	340.7 /s
Max Write	29 Mib/s	171.9 /s

Table 8 Disk bytes/operations of maximal cluster (2 partitions)

	CPU utilisation
Max CPU used	17.27 %

Table 9 Disk bytes/operations of maximal cluster (2 partitions)

	Network bytes	Network packets
Max incoming	52.72 Mib/s	38.06 k/s
Max outgoing	1.1 Mib/s	1.6 k/s

Table 10 Network bytes/packets of maximal cluster (16 partitions)

	Disk bytes	Disk operations
Max Read	69.85 Mib/s	1.14 k/s
Max Write	232.82 Mib/s	1.366 k/s

Table 11 Disk bytes/operations of maximal cluster (16 partitions)

	CPU utilisation
Max CPU used	72.82 %

Table 12 Disk bytes/operations of maximal cluster (16 partitions)

The final conclusion is that after adding a second parameter we can observe the high changes in the metrics, where some of them are doubled and a lot of them have multiplied by 10 or even higher than that in reading, writing, incoming traffic and outgoing traffic. The most harsh change is definitely the maximal usage of CPU that increased by 55.55% comparing it to the cluster where second parameter has not been used. When it comes to the elapsed time to finish the job in the cluster it has dropped by about 2.65 minutes in total.

## Task 1d\_ii

Four node cluster, SSD size (500), evenly distributed worker memory (1170), 2vCPUs each, 16 partitions

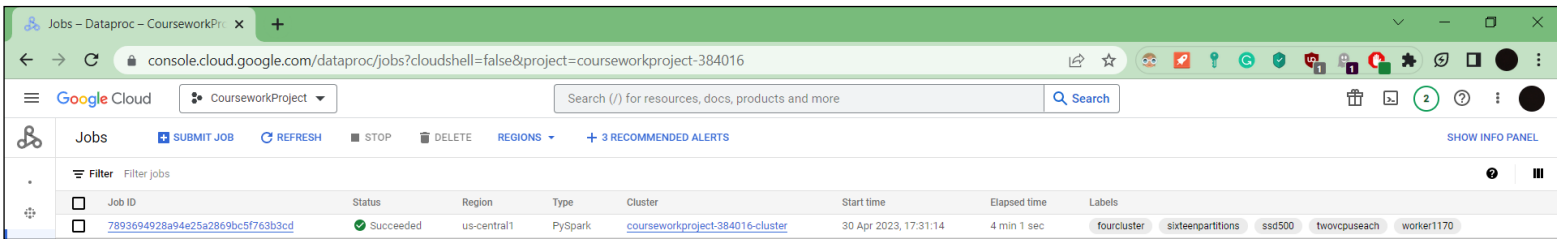


Figure 12 Four node cluster Elapsed time

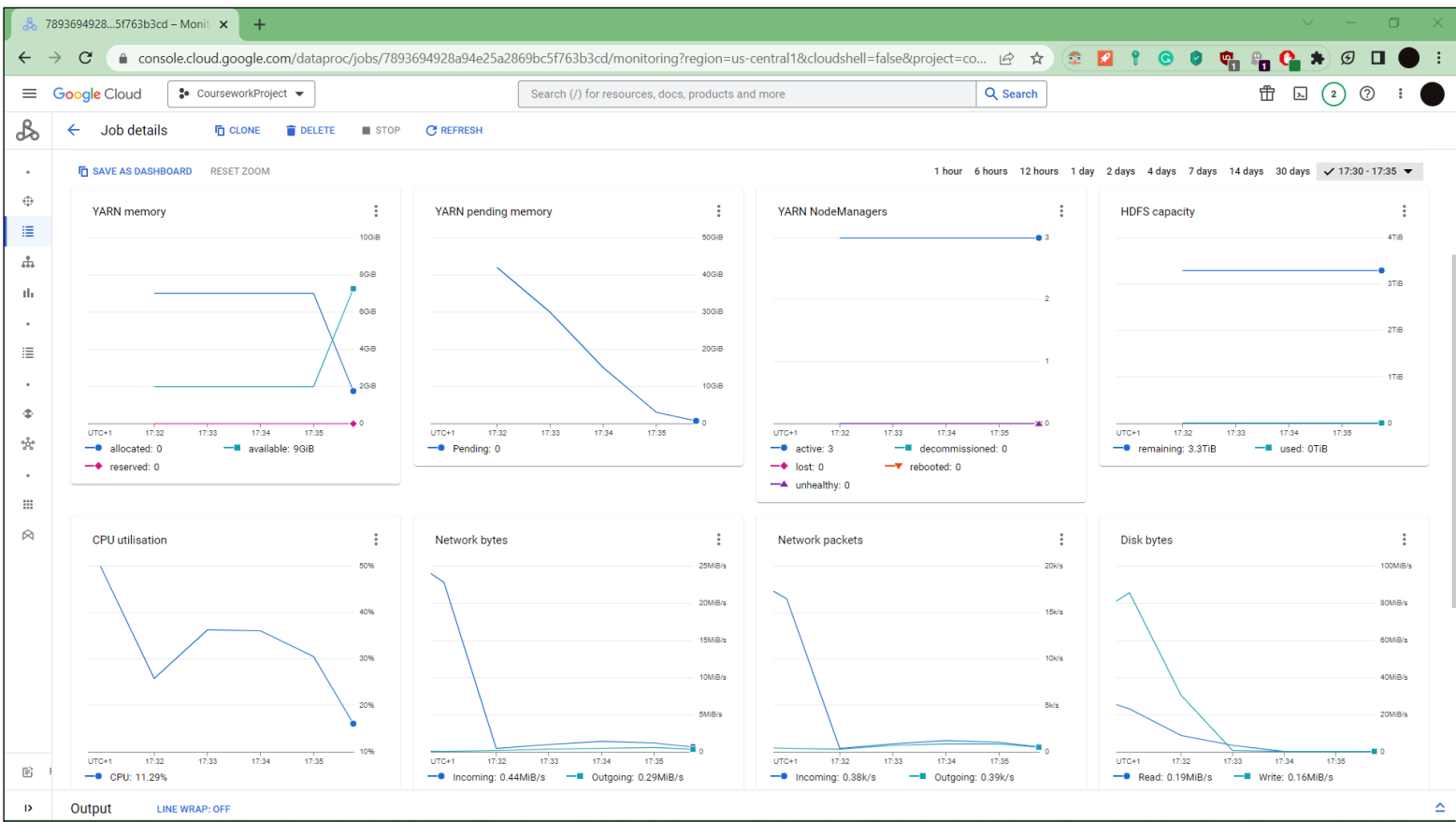


Figure 13 Four node cluster metrics

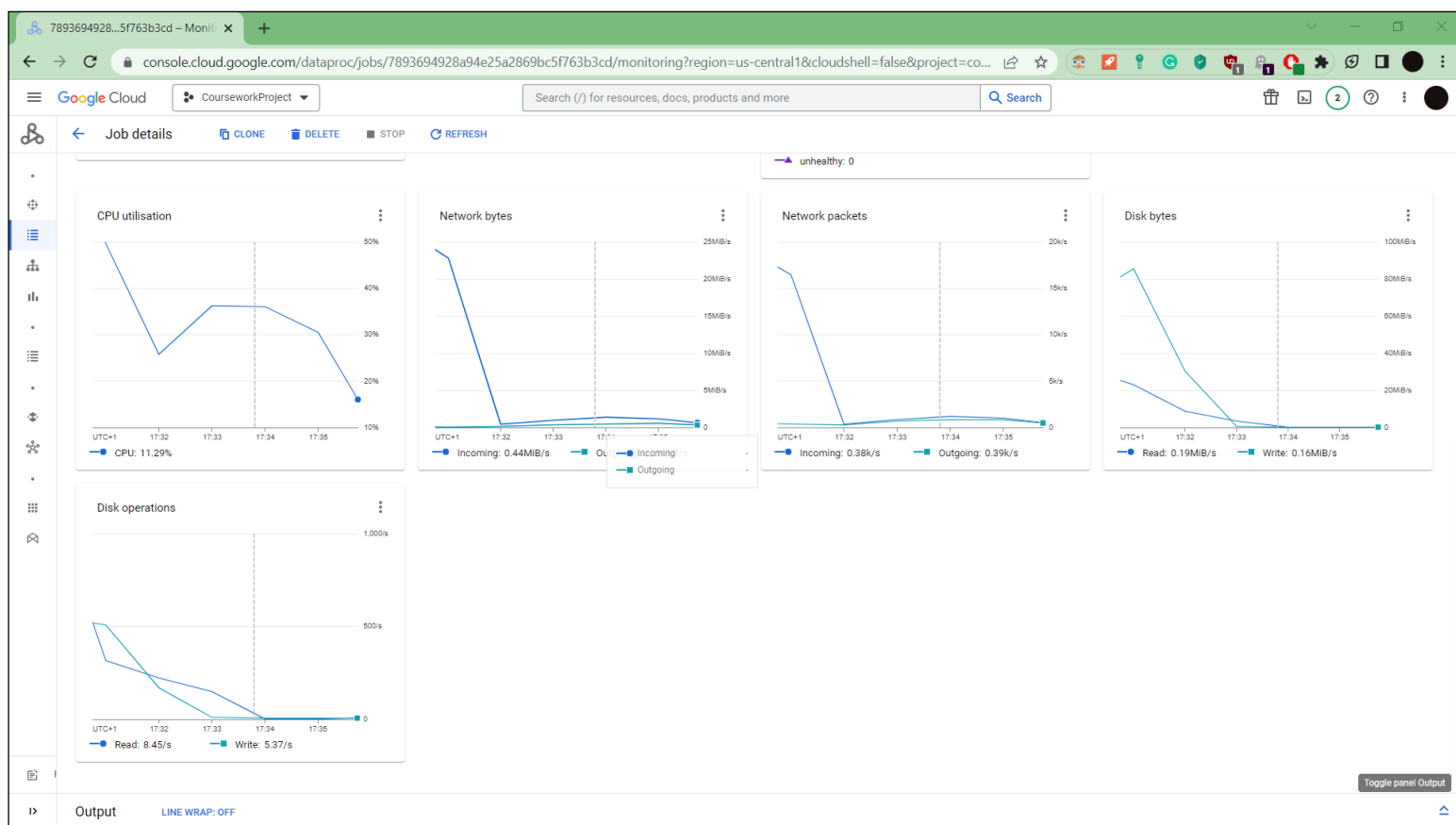
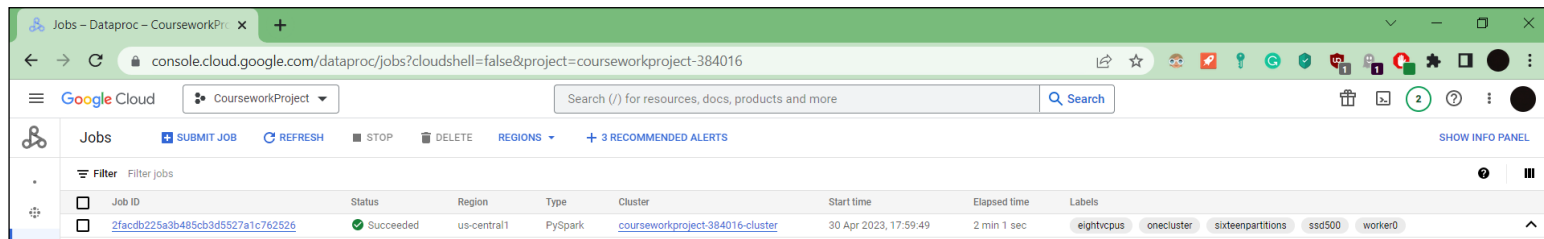


Figure 14 Four node cluster metrics

## Eightfold resource cluster, SSD size (500), 8vCPUs, 16 partitions



Job ID	Status	Region	Type	Cluster	Start time	Elapsed time	Labels
2facdb225a3b485cb3d5527a1c762526	Succeeded	us-central1	PySpark	courseworkproject-384016-cluster	30 Apr 2023, 17:59:49	2 min 1 sec	eightvcpus, onecluster, sixteenpartitions, ssd500, worker0

Figure 15 Eightfold resource cluster Elapsed time

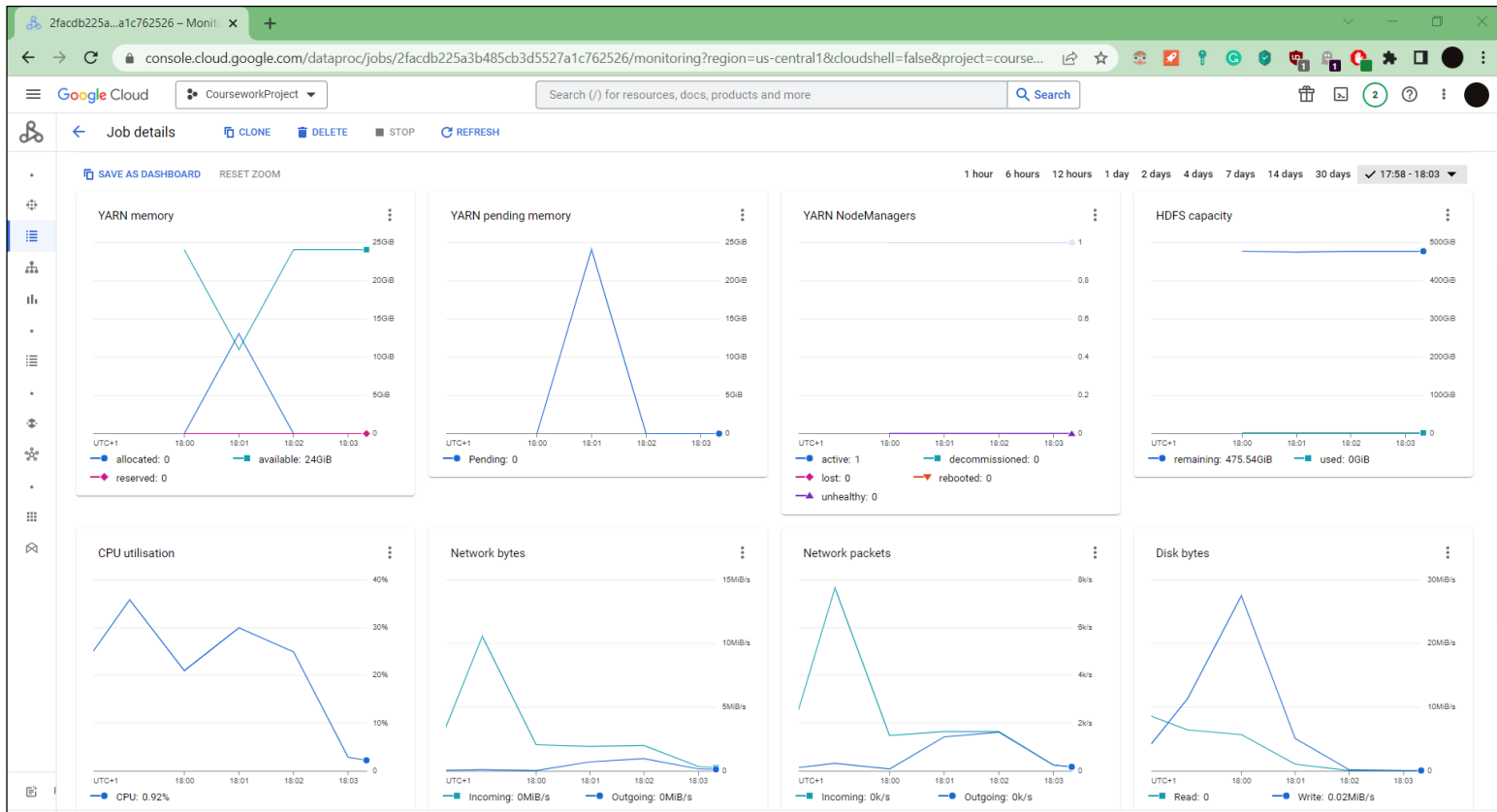


Figure 16 Eightfold resource cluster metrics

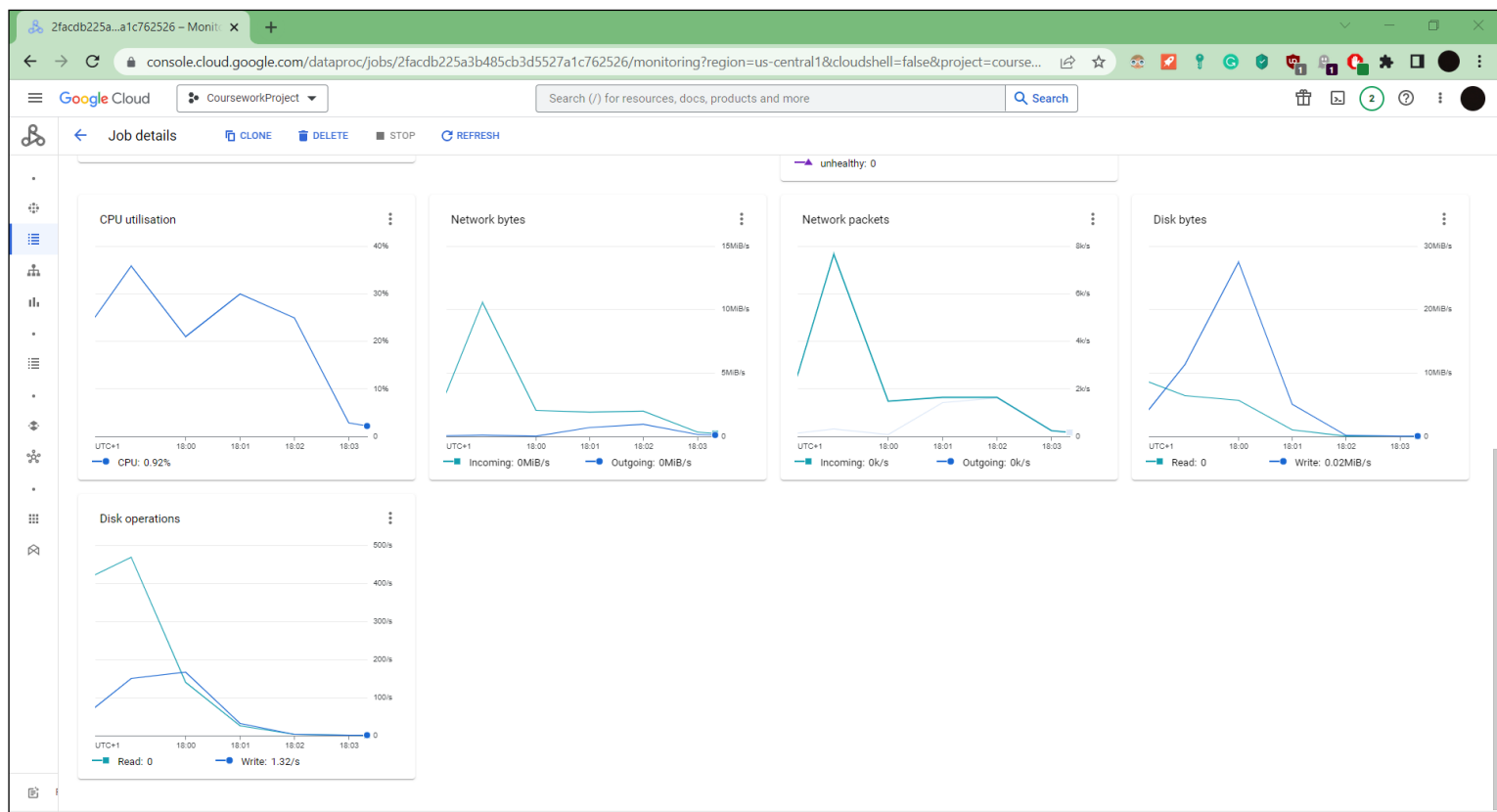


Figure 17 Eightfold resource cluster metrics

## Discussion about the results

Based on the Figure 13 we can see that the CPU utilisation was mostly between range of 18% to 40% where it would be approximately utilized 35% most of the time until it would gradually drop to 17-18%. Moreover, the elapsed time to fully submit and finish the job in the cluster took 4 minutes and 1 second as shown on Figure 12.

Based on the Figure 16 we can see that the CPU utilisation was mostly between range of 21% to 34% where it would be approximately utilized 27% most of the time. The next change is that it gets lower from the halfway of the graph to reach utilisation of 3% at last. Accordingly, the elapsed time to fully submit and finish the job in the cluster took 2 minutes and 1 second as shown on Figure 15.

	Network bytes	Network packets
Max incoming	22.76 Mib/s	16.43 k/s
Max outgoing	0.56 Mib/s	0.9 k/s

Table 13 Network bytes/packets of four node cluster

	Disk bytes	Disk operations
Max Read	23.12 Mib/s	314.67 /s
Max Write	85.41 Mib/s	506.9 /s

Table 14 Disk bytes/operations of four node cluster

	CPU utilisation
Max CPU used	49.55 %

Table 15 Disk bytes/operations of four node cluster

	Network bytes	Network packets
Max incoming	10.59 Mib/s	7.66 k/s
Max outgoing	0.96 Mib/s	1.6 k/s

Table 16 Network bytes/packets of eightfold resource cluster

	Disk bytes	Disk operations
Max Read	6.48 Mib/s	468.07 /s
Max Write	27.52 Mib/s	166.88 /s

Table 17 Disk bytes/operations of eightfold resource cluster

	CPU utilisation
Max CPU used	35.84 %

Table 18 Disk bytes/operations of eightfold resource cluster

Considering all details provided by the evidence we can come up with the conclusion such that the eightfold resource cluster has tendency to use less CPU than the four node cluster. On the other hand, reading and writing limits are higher for four node cluster but the time of getting the job done is quicker for the eightfold cluster of exactly 2 minutes.

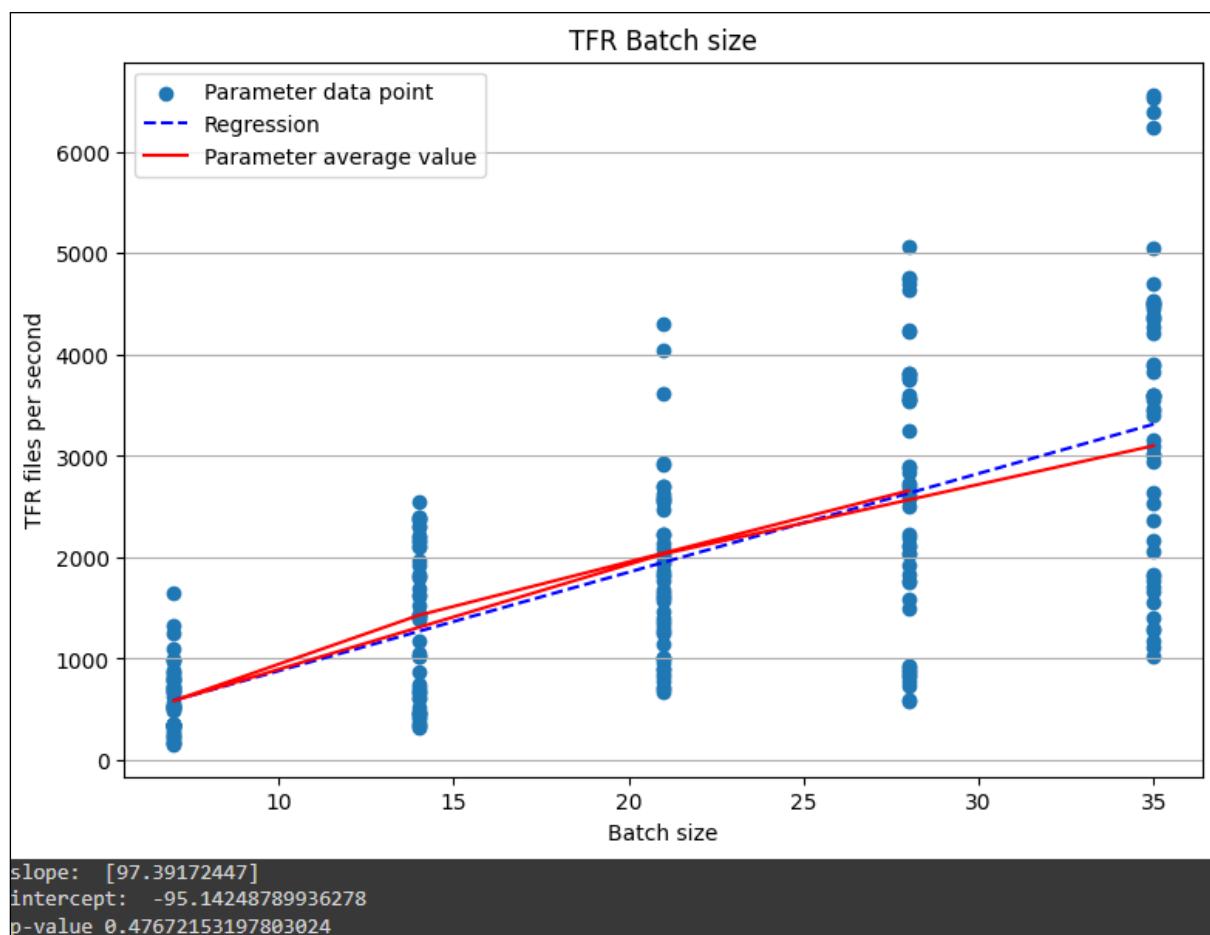


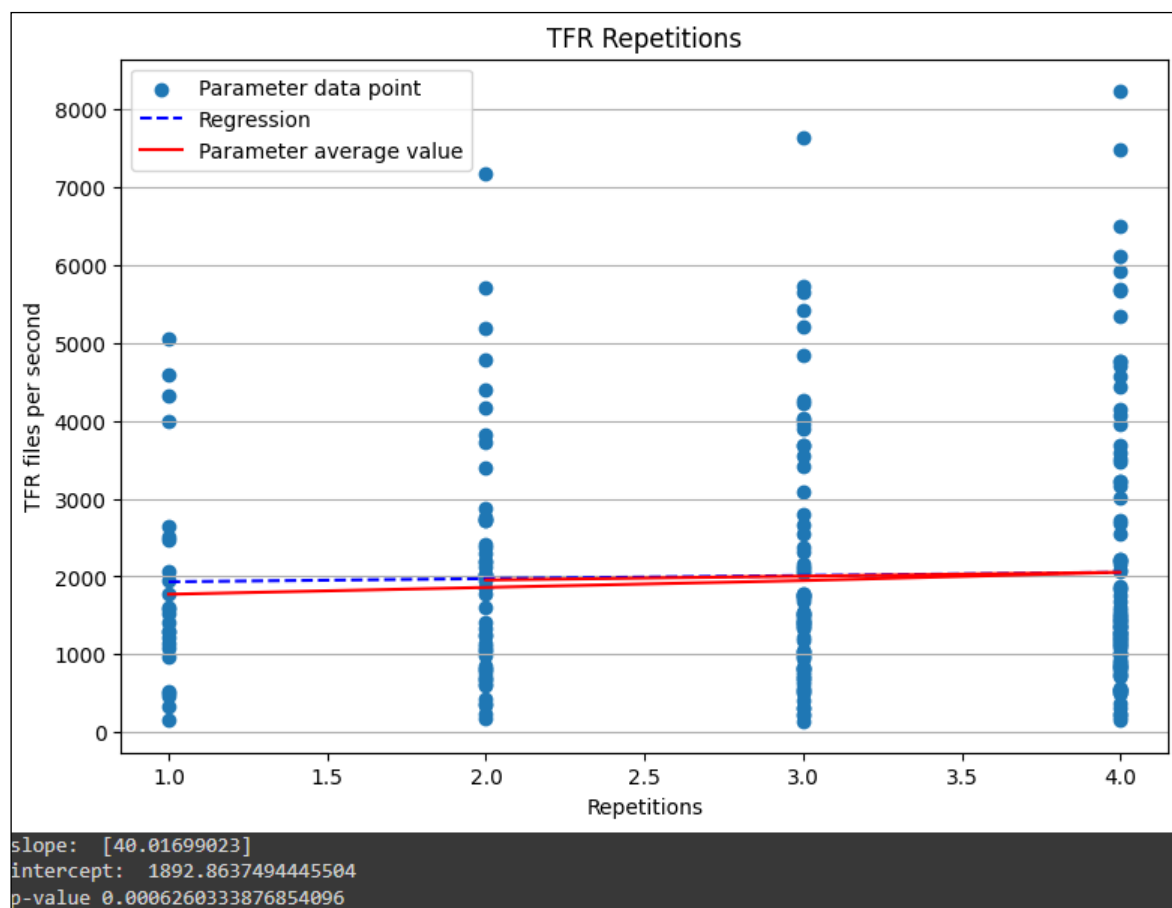
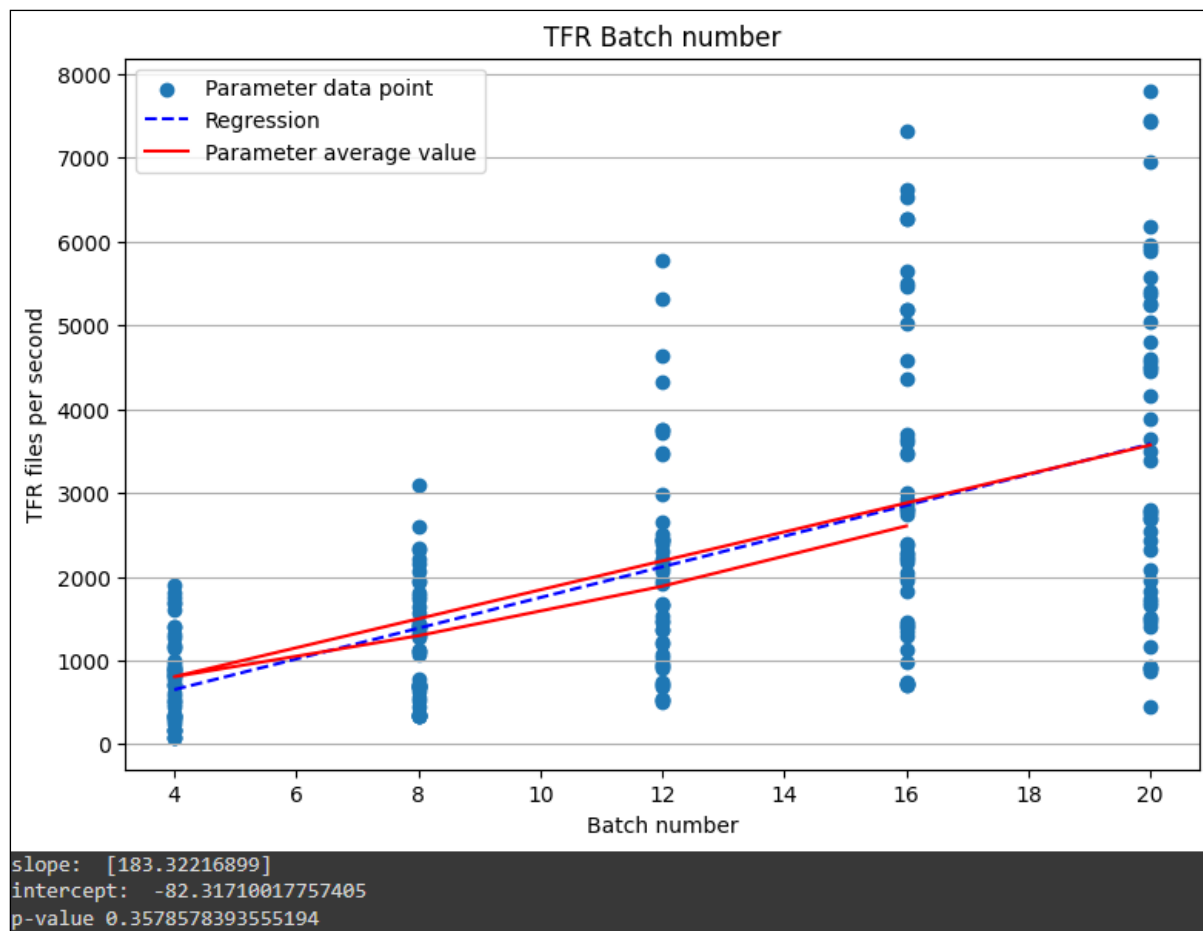
## Task 1d\_iii

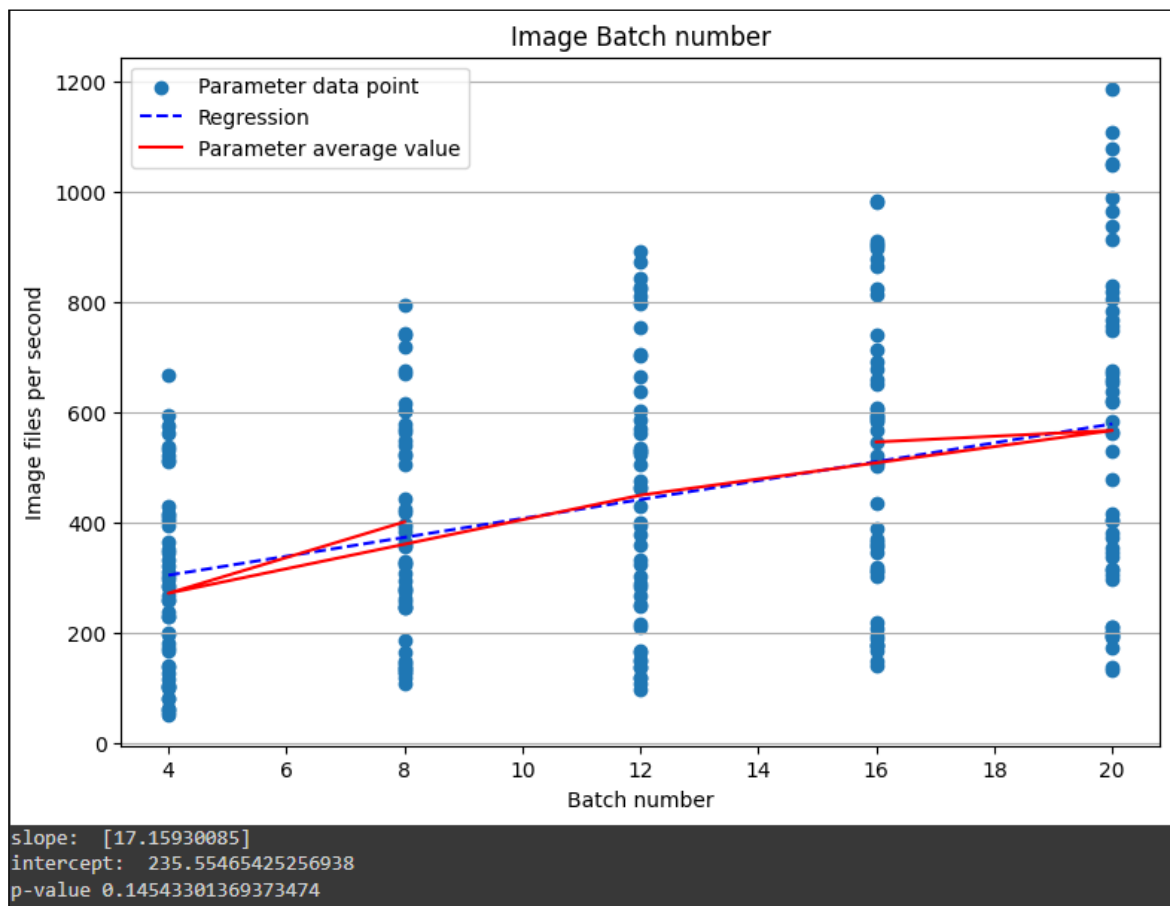
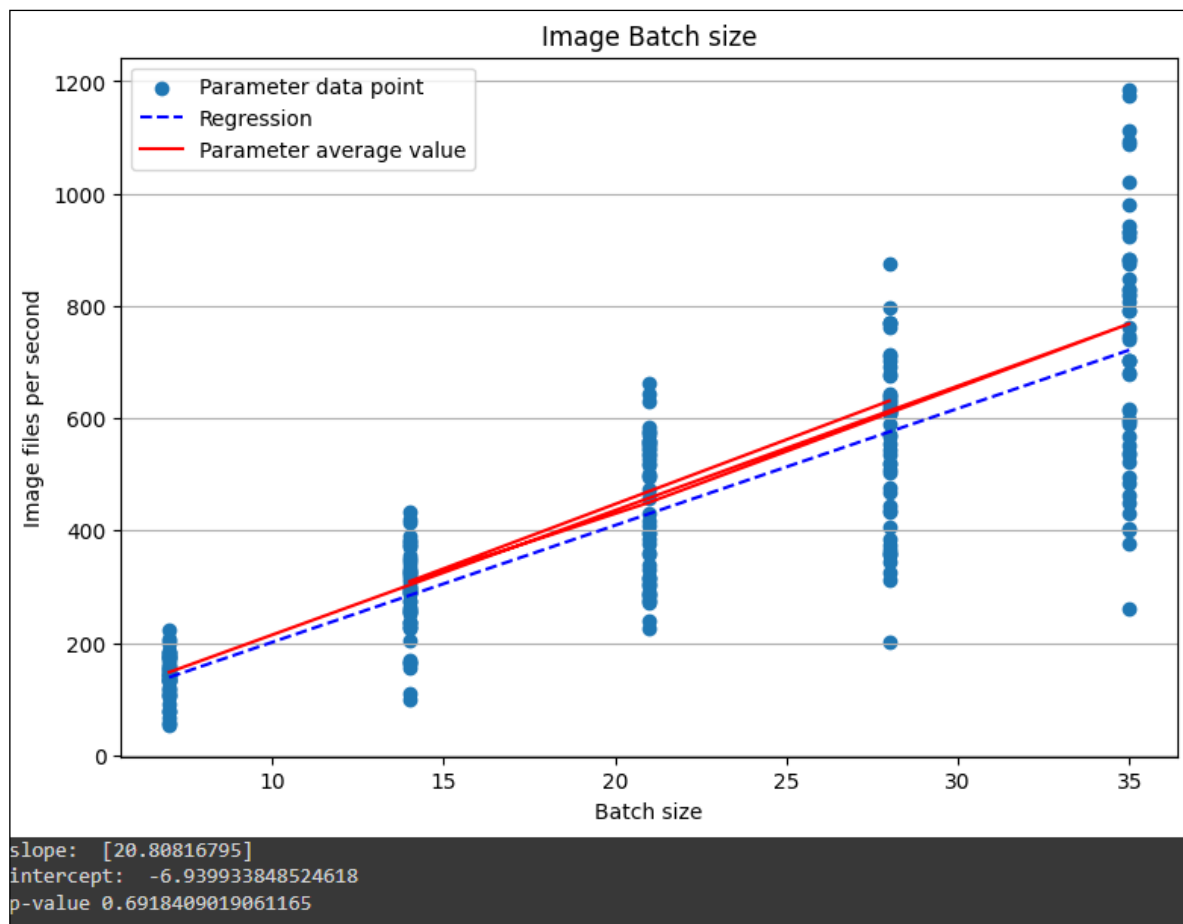
### Difference between Spark and standard applications

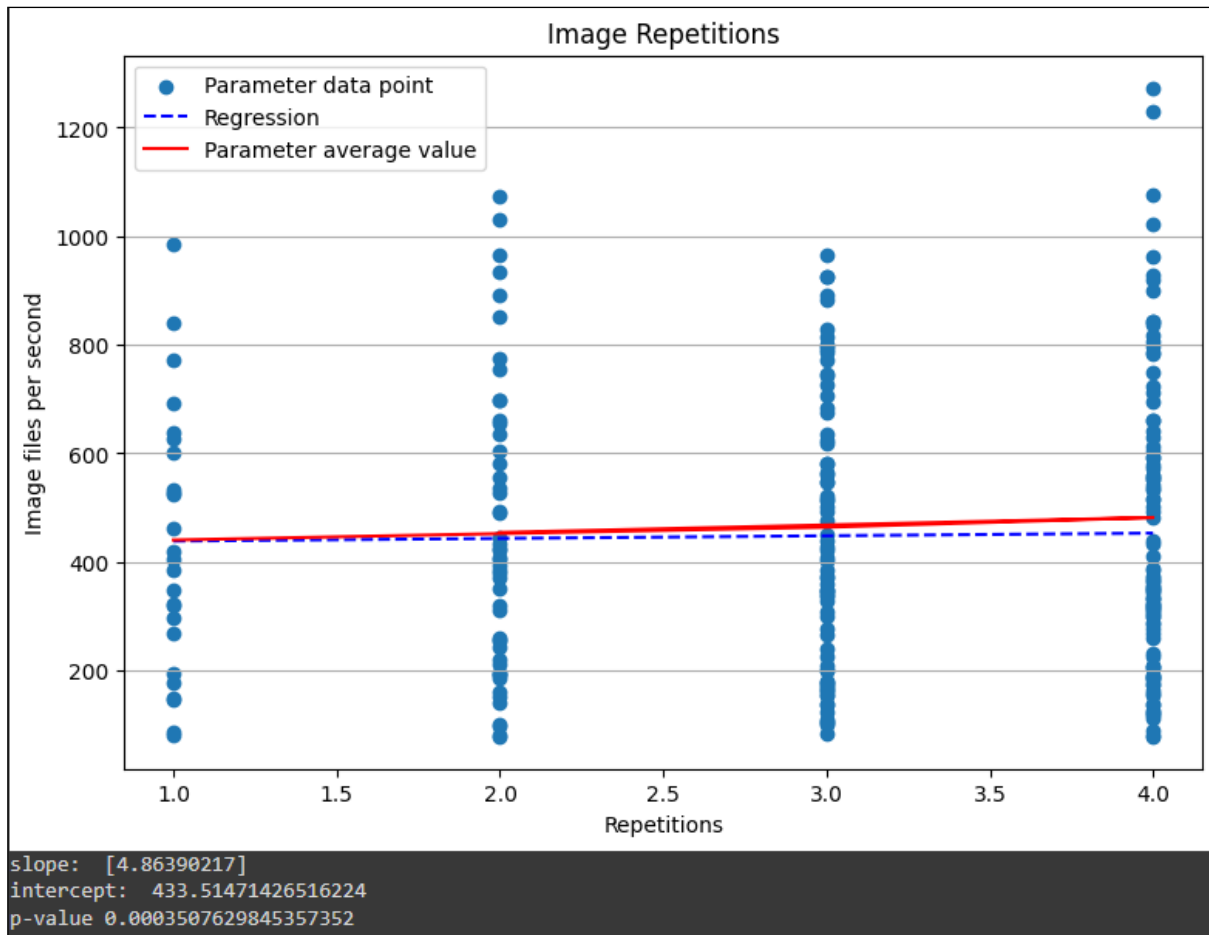
In our lab sessions, we would simply send a task that would be executed on one machine only. It would have been done locally. However, all of the current tasks and workers that would finish these tasks for us are located in the cloud (Google Cloud). The main differences between Spark and standard applications are the format of the data stored and the approach of parallelism, which is more about parallelising the tasks, whereas the application usually aims to parallelise data. The first one is about splitting of task into smaller subtasks that can be dealt with by multiple workers and completed simultaneously and the latter is just performing same operation concurrently on elements of data.

## Task 2d









	Slope	Intercept	P-value
TFR Batch Size	97.39	-95.14	0.47
TFR Batch Number	183.32	-82.31	0.35
TFR Repetitions	40.01	1892.86	0.00
Image Batch Size	20.80	-6.93	0.69
Image Batch Number	17.15	235.55	0.14
Image Repetitions	4.86	433.51	0.00

According to the value gathered, we can see that the P-values are generally decreasing going from top to bottom for both TFR files and Images. For image files, such a decrease is higher when compared to the TFR files, where the change was only 0.12. The values presented above are highly dependable on the parameters we have specified and could potentially be configured in a way that would benefit the results. On the other hand, the delay in executing such tasks in the cloud is much higher than it would take for execution in own machine in a parallel manner. Instead of sending them one after the other, they could be handled once to avoid the unnecessary time taken and the associated cost for the computational power required. In addition, it seems like the tendency of applications that have throughput as the objects do not compare to other solutions when it comes to slight in-size tasks. The tasks themselves should be done quickly, using available storage to decrease the latency. Thus it is a smart move by the cloud providers to tie throughput to the capacity of disk resources.

## Task 3d\_a

At the time when the tasks were sent out, there was the possibility of finding some inconsistency in the time they took (especially when repeated). Based on the paper, such inconsistency could be related to often reminded noise of various forms such as “observational” and “multiplicative”. In this case, we encounter common “cloud noise” that perhaps affects such timings and is described to be considered by the cherry-picking with a “black-box approach, it automatically learns the relation between cloud resources and the running time”. The concept of cherry-picking allows us to find optimal or at least near-optimal solutions of configuration making our work perform on a greater scale and increase the production of the outputs. Accordingly, based on the percentages in the paper, it will try to pick and find (static search for best cloud configuration) numerous options that will be suitable in our case (modelling of application performance) at a much better cost compared to alternatives. On the other hand, it comes with several challenges that we have to account for. The first one is the complex performance model that is non-deterministic in its nature. The second one is the cost model which has to be well-balanced otherwise the charges may be higher than it was anticipated. The last would be the heterogeneity of applications, which affects the best choice due to their configuration and is dependable on previous works.

## Task 3d\_b

Based on different analytic jobs a person can have a different configuration that would focus on the diverse resources, like CPU, memory disk, network might be chosen for the requirements of a person. The first strategy defined by the paper is the Bayesian Optimisation that the Cherry-picking leverages because it proposes the confidence interval that allows CherryPick to decide where is the optimal solution. Bayesian Optimisation gets improved as well as the decision based on it done by the CherryPick when more samples of data are available from the batch or stream. The issue lays in the amount of variables we need to consider ranging from resources, needs such as increasing or lowering the time taken to finish the job to lastly resources as mentioned before. If all of them were to be at least considered by the CherryPick perhaps a higher network delay would probably be encountered. Lastly, the time taken to find the solution would also increase significantly due to amount of variables to consider and usage of these parameters to find it.

## Colab link

Words: 1962

<https://colab.research.google.com/drive/1U6lLe9GLRF2O-uKNgVm7r6-wEoeGg-1k?usp=sharing>