Big Data Coursework Report

Name: *Dimitrios Megkos*
Email: *dimitrios.megkos@city.ac.uk*
Student ID: *210034034*

Google Colab: https://colab.research.google.com/drive/1pgreMerjmTPdbKoiyA49s2v9hxVxCVju?usp=sharing

Task 1d) Optimisation, experiments, and discussion

i) Improve parallelisation

Two tests were run using the maximal cluster. The first test was run at 14:56 without using the suitable change in the parallelize code, the second test was run at 14:58 with the change. The difference in the utilisation of the cluster can be spotted right away, as Disk Operations and CPU Utilisation were higher during the second test, according to figure 1 below. These are the most important metrics since they have a direct impact on the processing time. The first test ended after 0.85 seconds, while the second test ended after 0.69 seconds. The difference is not that big, but this is mostly because of the very small size of the files that are read and written.

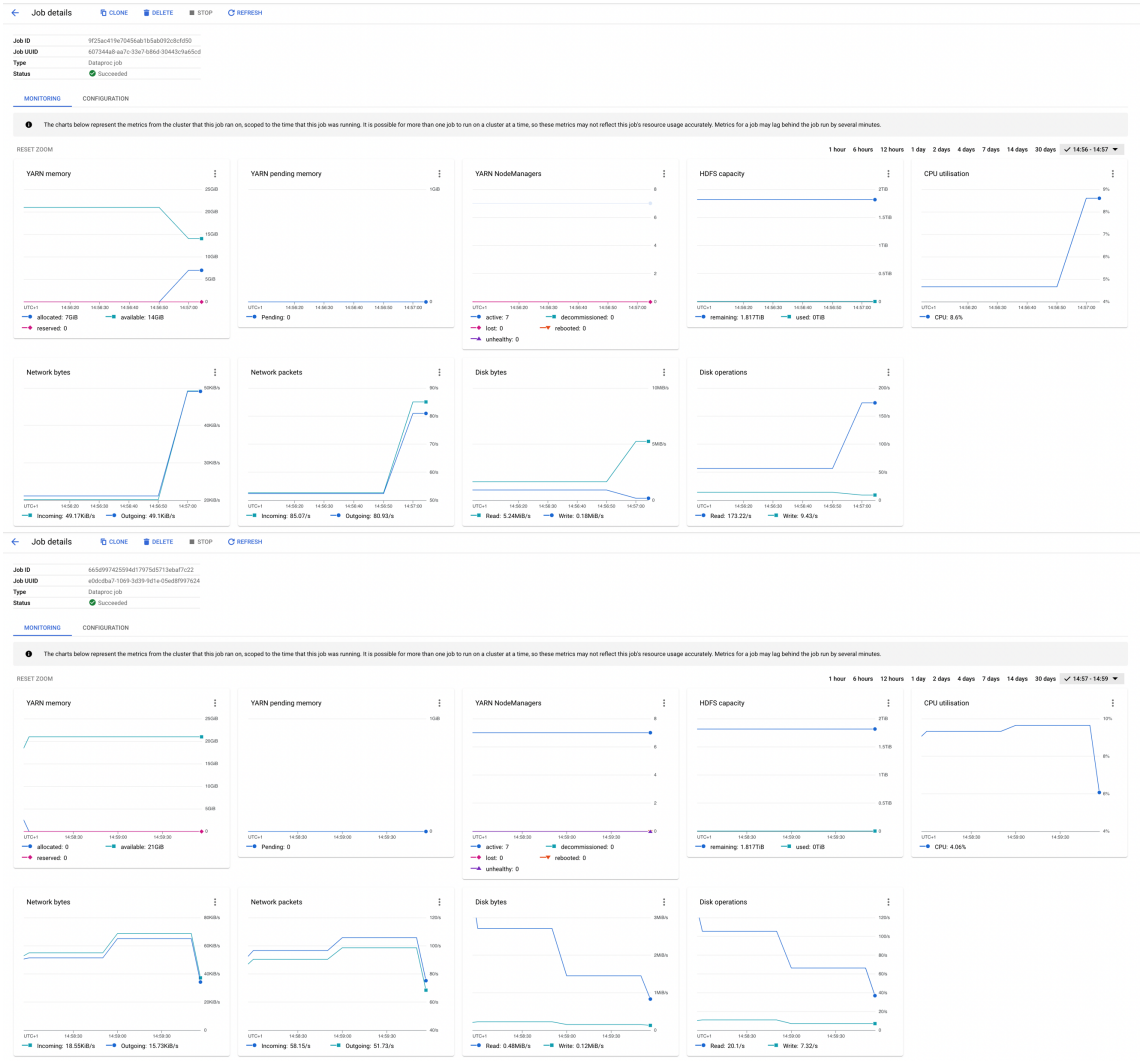


Figure 1: 2 Partitions (Top) vs 16 Partitions (Bottom), Google Cloud Metrics

Comparing the CPU usage of the two tests, the graphs in figure 2 below show that when there are more partitions, the CPU is not only used by the master, but is also used more by the available workers.

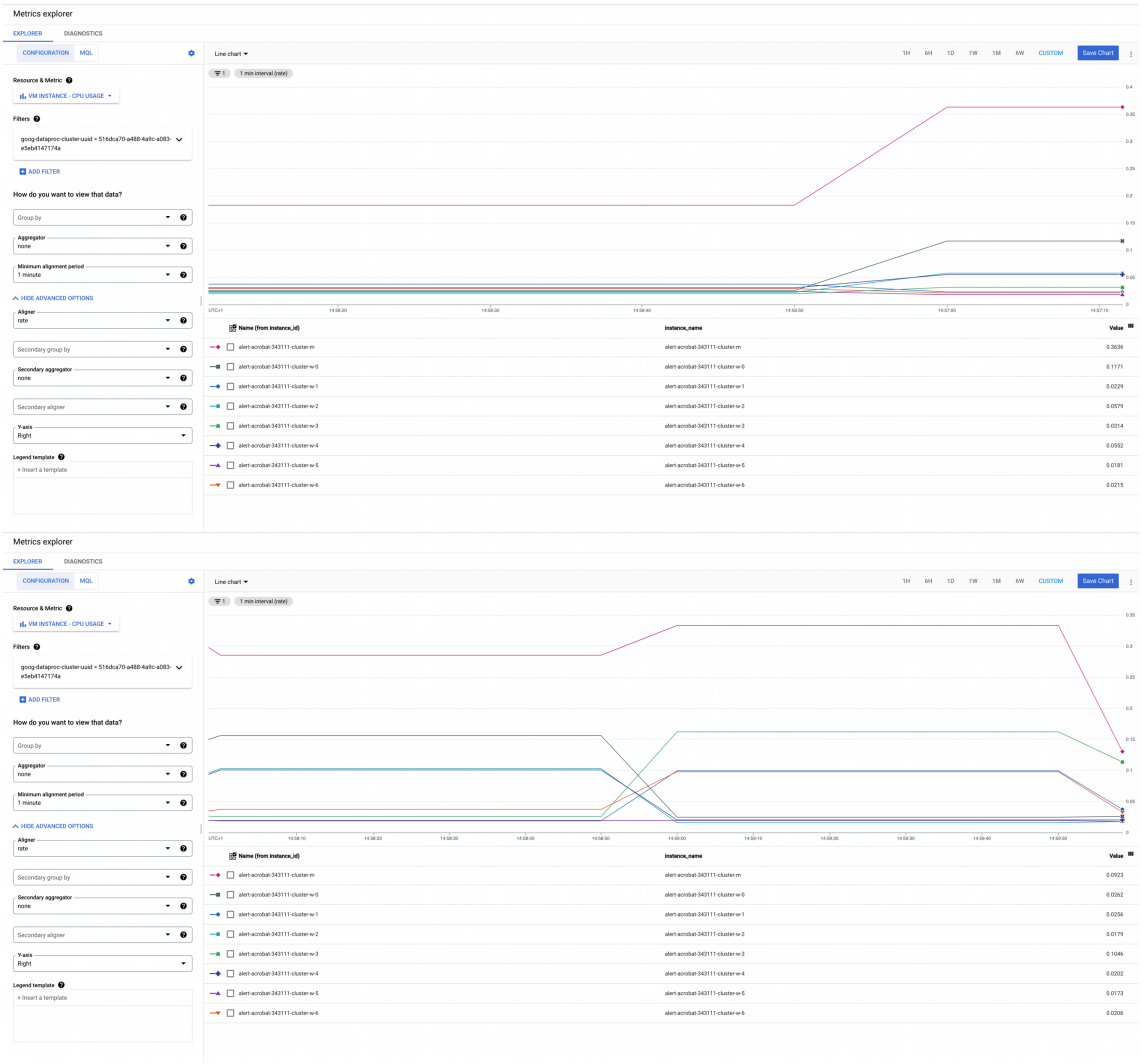


Figure 2: 2 Partitions (Top) vs 16 Partitions (Bottom), CPU Usage

ii) Experiment with cluster configurations

The maximal cluster executed the optimized script in 0.69 seconds while the initial single machine cluster executed the script in 0.60 seconds. A third cluster configuration was created with 4 machines with double the resources which executed the script in 0.67 seconds. Figure 3 below shows that CPU and Disk utilization on the 4-machine cluster is higher than the maximal cluster. Network bandwidth usage is also higher on the maximal cluster, due to the higher number of nodes. Based on the graphs, it is better to have more machines with fewer resources than fewer machines with more resources because of how parallelization and job distribution work. More machines utilize better the CPU and Disk resources.

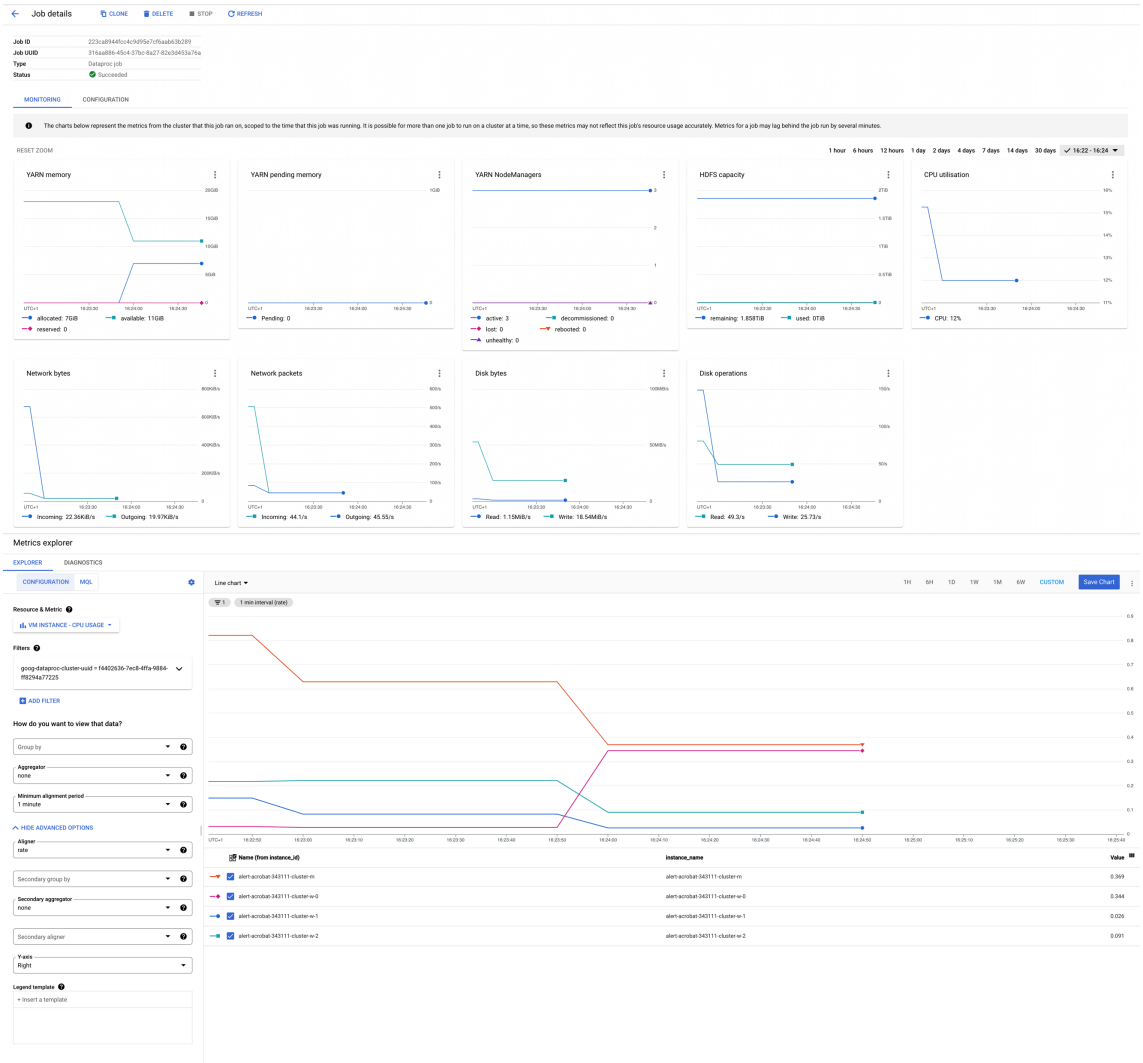


Figure 3: 4-Machine Cluster, Google Cloud Metric (Top) and CPU Usage (Bottom)

iii) Explain the difference between this use of Spark and most standard applications like e.g. in our labs in terms of where the data is stored. What kind of parallelisation approach is used here?

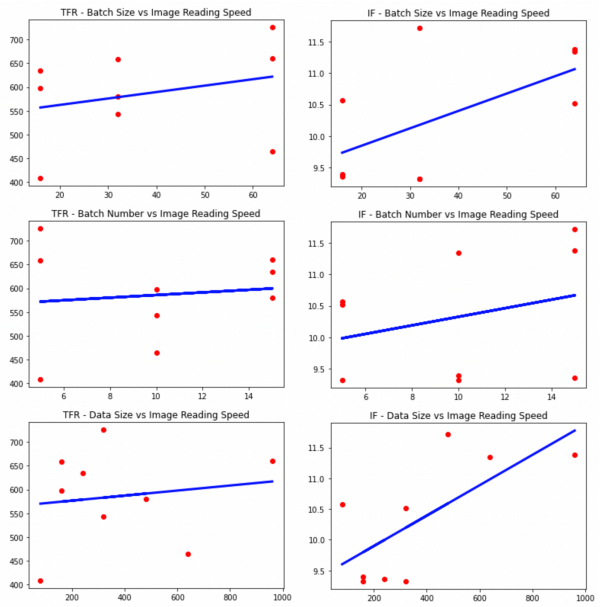
Spark is a distributed computing engine that enables the parallel processing of data. It is a powerful tool for processing big data. Spark's architecture is different compared to most standard applications, as it uses a "Master" that receives jobs and distributes them to multiple "Workers" for parallel execution. In the labs, we experimented with RDDs and the map() function, while in this coursework we used the mapPartitionsWithIndex() function. The difference is that in the map(), a function is applied to every element of the RDD and returns every other element of the new RDD. However, in the mapPartitions(), the function is applied to each partition of the RDD, instead of each element, and returns multiple elements of the new RDD. This leads to a performance improvement since the object creation is eliminated for every element as in map transformation.[1]

Task 2c) Improve efficiency

In PySpark, the RDD transformations run only when we execute the relevant actions. The files are read into the RDD every time an action is executed, which increases resource usage and execution times. By using RDD.cache() we are storing the initial RDD in the memory, making it available each time we execute an action, without having to create RDDs from files on every call, saving both time and resources. To see the impact of caching, I included some RDD action printing in my script, **take()** and **count()**, and ran the script in the cloud with and without caching. With caching disabled, the script ended after 59 seconds, while with caching enabled, it ended after 43 seconds. The tests were conducted reading TF Record files only, to save some time, but I expect similar findings with image files too.

Task 2d) Retrieve, analyse and discuss the output

Figure 4 below shows the results of all six linear regressions, one for each parameter for both TF Records and Image Files. The speed tests were done in the cloud. Although the training data are limited, the plots show that the Batch size is the most important parameter when it comes to Image reading speed. This means that applications like large-scale machine learning that implement batch learning will benefit more with higher numbers of batch sizes. However, reading more data in larger batch sizes does not always guarantee better speeds. If the data is stored in distant physical locations, there is an extra latency and cost of reading the data.



	TF Records	
	Intercept	**Slope**
Batch Size	535.45	1.35
Batch Number	558.24	2.75
Data Size	566.06	0.05

	Image Files	
	Intercept	**Slope**
Batch Size	9.30	0.027
Batch Number	9.65	0.068
Data Size	9.40	0.002

Figure 4: Linear Regression results for each parameter, for TF Records (Left) and Image Files (Right)

Using multi-worker machines to retrieve data leads to higher reading speeds, even if the data is being read from a distant physical location. However, this also leads to higher network bandwidth. Usually, cloud providers tie throughput to the capacity of disk resources, which is usually the bottleneck in these machines, to avoid wasting extra network bandwidth, which also leads to extra costs.

When we parallelize the speed test, we need to consider the type of cluster that is going to run the job and adjust the number of partitions. Fewer partitions than the number of available worker machines will result in a waste of available resources since some workers will not be utilized for the job. For this exercise, since the job was run by a four-machine cluster, the number of partitions was changed to eight, up from two.

Task 3c) Distributed learning

The strategy that was selected was MultiWorkerMirroredStrategy[2], which falls under the category of synchronous multi-worker training. This strategy creates copies of all variables in the model's layers on each device across all workers and aggregates gradients keeping the variables in sync. Two experiments were run, using a different number of batches, epochs, and workers, and were compared to the initial run that did not implement the multi-worker strategy. Table 1 below shows the parameters and results of the initial and the two experiment runs.

Initial Run				
Batch size: 64 *Epochs:* 5 *Workers:* 0	**Loss**	**Acc**	**Val_loss**	**Val_acc**
	1.53	0.28	1.45	0.34
First Experiment Run				
Batch size: 64 *Epochs:* 5 *Workers:* 2	**Loss**	**Acc**	**Val_loss**	**Val_acc**
	1.40	0.35	1.37	0.47
Second Experiment Run				
Batch size: 128 *Epochs:* 10 *Workers:* 4	**Loss**	**Acc**	**Val_loss**	**Val_acc**
	1.46	0.31	1.56	0.30

Table 1: Parameters and results for initial, first and second run

Figure 5 shows the plots of all three runs. Using the synchronous multi-worker training strategy improved the initial results. The first experiment run returned lower loss and validation loss, and both the training and validation accuracy improved. The second experiment run returned worse results than the first, probably due to having less batches per training epoch. Nevertheless, a multi-worker strategy should always be preferred when a multi-worker environment is available, as distributed learning gives better training results.

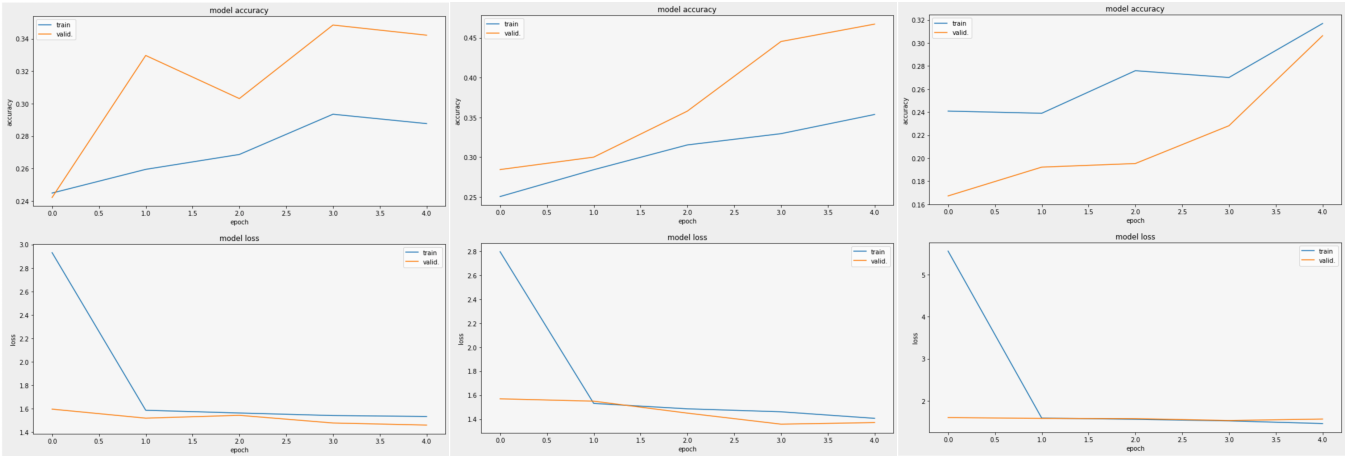


Figure 5: Initial run vs first experiment run vs second experiment run

Task 4 Discussion in context

a) Contextualise

In “Cherrypick: Adaptively unearthing the best cloud configurations for big data analytics”[3] Alipourfard et al introduce a system that uses Bayesian optimization to build models that find the best cloud configuration for recurring big data analytic jobs running in the cloud, saving extra searching costs. This is similar to what we did in section 1, where we tested our script with various cluster configurations so in this case, the CherryPick system would be useful. In the paper, the models try to find the best cloud configuration (e.g., number of VMs, CPU count, RAM). A good cloud configuration can significantly reduce the cost of analytic jobs and improve results, which was the case with the first experiment run in task 3c. CherryPick could be useful in this case also, selecting the right number of workers that will handle distributed learning. Furthermore, the paper suggests that diminishing returns must also be considered, as jobs may not benefit from extra resources beyond what they need. A bad configuration can give worse results and increase costs, which is something we saw with the second experiment run in task 3c.

In “An Oracle for Guiding Large-Scale Model/Hybrid Parallel Training of Convolutional Neural Networks”[4] Kahira et al talk about how distributed training enables faster time to convergence and alleviates memory capacity limitations when training large-scale models. They try to understand the trade-offs between different parallelism approaches on performance and scalability using model-driven analysis to detect the limitations and bottlenecks of different parallelism approaches. This is similar to what we did in task 3c where we compared a multi worker distributed learning strategy with a zero worker non-distributed one. The methods that are introduced in this paper could help choose the right distributed learning strategy for task 3c.

b) Strategize

To define strategies, we need to first identify what kind of data we have and the number of available computing resources. In batch processing, a large volume of data (e.g., millions of records), is processed all at once. This processing is very effective for data that are collected over a period of time and are not time-sensitive since the processing time is increased due to the increased size. The reading and writing of image files in task 1 fall under this category. Batch processing can take advantage of parallelization in the cloud however, cluster resources like storage size, need to be adjusted accordingly. CherryPick from the Alipourfard et al paper would be suitable for this task. Online and stream processing are somewhat similar since both do not require a large number of disk resources. Because in online processing the reaction to the data must be within milliseconds and stream processing analyses data in near real-time, we must make sure the cluster machines have enough CPU power and network bandwidth. CherryPick again can help us adjust clusters accordingly. Regarding the tasks above, stream processing could be used for the models in task 3, in case we were required to constantly feed our model with new data and train on the fly. The Oracle in the Kahira et al paper could help us choose the best model suited for this task.[5]

References

[1] Explaining MapPartitions and MapPartitionsWithIndex, data-flair.training
[2] Multi-worker training with Keras, TensorFlow Core, tensorflow
[3] Omid Alipourfard, Hongqiang Harry Liu, Jianshu Chen, Shivaram Venkataraman, Minlan Yu, and Ming Zhang. 2017. Cherrypick: adaptively unearthing the best cloud configurations for big data analytics. In Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation (NSDI'17). USENIX Association, USA, 469–482.
[4] Albert Njoroge Kahira, Truong Thao Nguyen, Leonardo Bautista Gomez, Ryousei Takano, Rosa M. Badia, and Mohamed Wahib. 2021. An Oracle for Guiding Large-Scale Model/Hybrid Parallel Training of Convolutional Neural Networks. In Proceedings of the 30th International Symposium on High-Performance Parallel and Distributed Computing (HPDC '21). Association for Computing Machinery, New York, NY, USA, 161–173. https://doi.org/10.1145/3431379.3460644
[5] Laura Shiff, Real Time vs Batch Processing vs Stream Processing, bmc

Word count: 1828