Big Data Report

Question 1: Cloud Speed Tests

Image Dataset

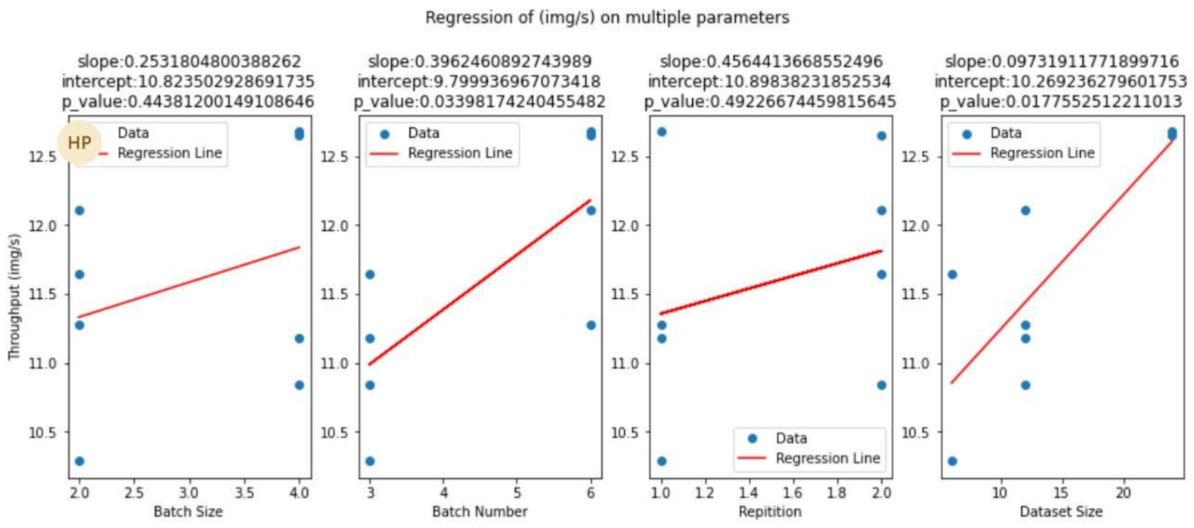


Fig 1. Image Dataset results for locally performed parameters

Decoded Dataset

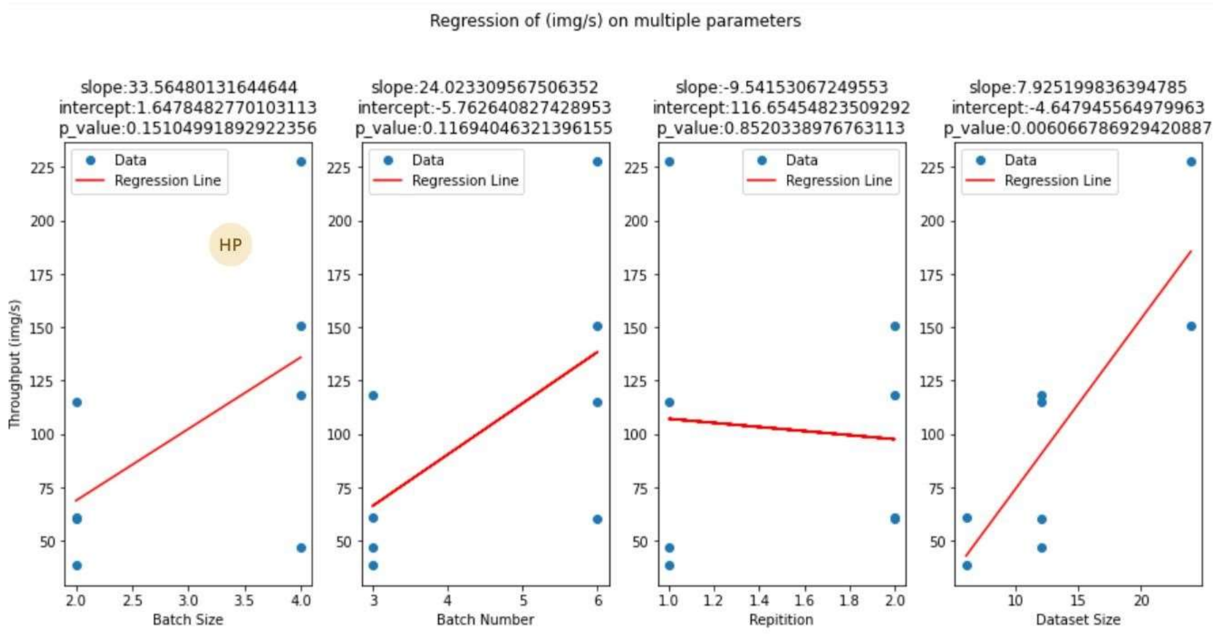


Fig 2. Decoded Dataset results for locally performed parameters

Reading speed tests were conducted in this task over pre-processed decoded file and individual image files. The x-axis corresponds to the chosen parameters, while the y-axis indicates throughput images per second. We noticed that the decoded dataset captures higher images per second in comparison to the image dataset due to it being pre-processed data and not decoded on the fly.

Question 2: Parallelizing the speed test with Spark in the cloud

To avoid limiting the capacity of our system by a single component we use a cache memory (temporary memory), which increases the efficiency of our CPU. We can access data faster as caching reduces the time needed for repeated access to the same RDD.

In this example we run efficiency tests by accessing the RDD with and without a cache memory and observe the performance based on CPU Utilization and Network Loads. The code in particular is rdd.cache() that we call to introduce greater speeds. sc.parallelise() method provides instructions for Spark to distribute the data across multiple nodes. Hence an increase in nodes will help us achieve greater speeds as it does not depend on a single node to process the data. We take advantage of additional virtual CPUs by setting to eight nodes whereas without it, we only computed on two nodes.

CPU utilization (Fig 3) shows the first peak as the cluster creation, second peak executing task 2 without caching, third peak is with caching. Utilization is high without caching (approx. 2.45AM) at around 60% and then significantly reduces once caching (2.55AM) is introduced at around 25%.

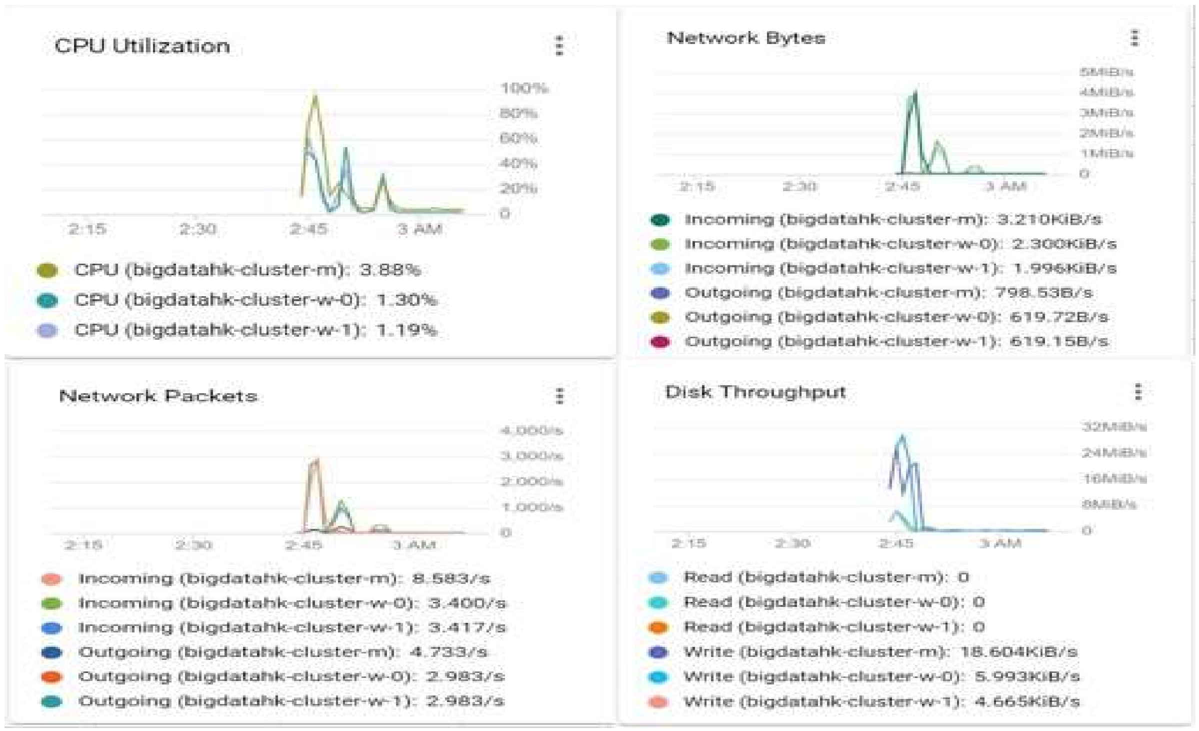


Fig 3. Usages of CPU, Network and Disk Throughput (1 master machine & 2 workers)

The use of a cache and parallelization (Fig 4) allows for higher disk throughput from the underlying resource. We see that caching/parallelization improves throughput by approx. 1.5x. Network Bytes shows less bytes are used once caching is introduced. Introducing workers reduces the burden and hence high throughput is achieved.

Higher latency results in lower throughput of image files. These effects are less of a problem for the decoded dataset. Higher values in all four parameters results in a greater throughput. This signals that large scale applications can have high computations in the cloud. We would expect that on a single machine may cause the system to crash or slow down performance due to its limited memory, but multiple machines will allow continuance as tasks can be distributed across other machines to complete the process on time. Caching will also not be effective due to much less local memory sizes.

Cloud storage offers an initial I/O capacity in its bucket. As request rates for the bucket increases, cloud storage automatically offers a higher disk I/O. If the request rate on the bucket increases faster than the cloud storage performing redistribution, then we run into temporary limits which gives rise to higher latency and error rates, which can be prevented by gradually increasing the request rates. In relation to cloud speed tests and the autoscaling method that providers offer disk I/O, the CPU might not be able to tolerate with the speed of inputs, causing a bottlenecking of operations. Hence cloud providers would tie throughput to capacity of disk resources to prevent this.

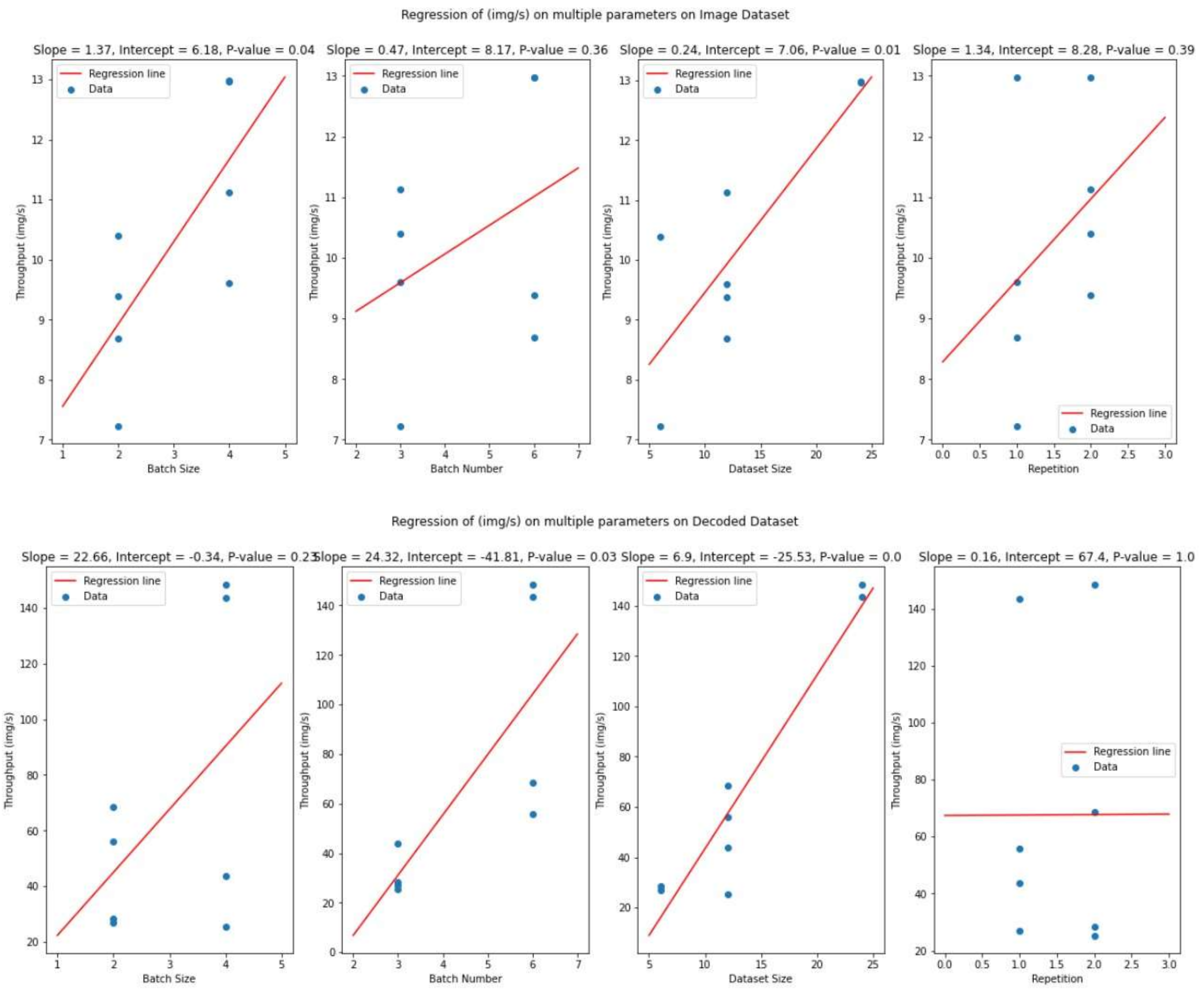
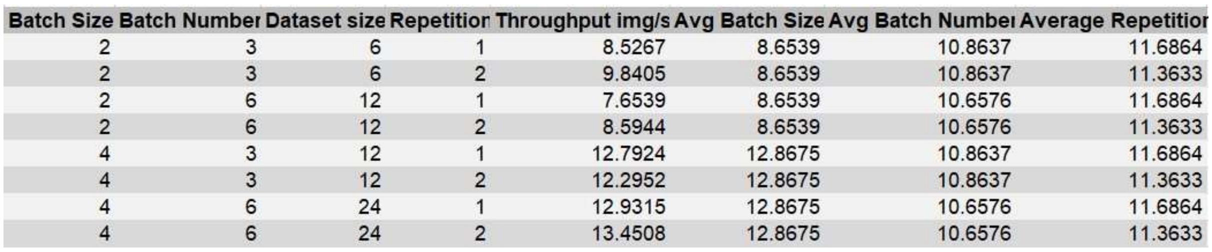


Fig 4. Parameter value results with caching and parallelization

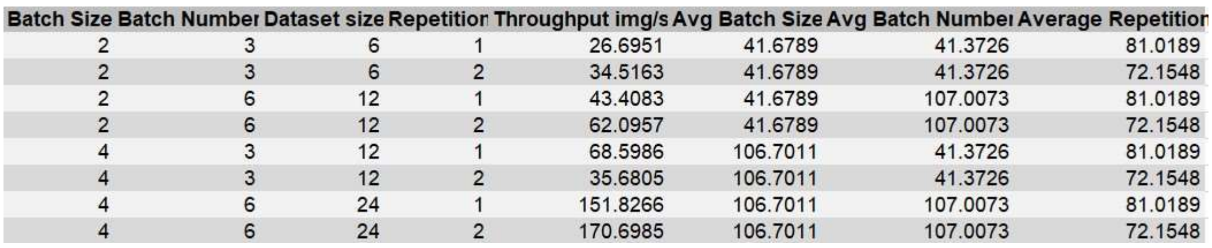
P-value < 0.05 represents good linear relationship between the two variables. Batch size and Dataset has low P-value and hence linear relationship can be concluded on the image dataset. Similarly, we note the trend as linear for batch number and dataset on the decoded dataset. Any parameters with p value > 0.05 show acceptance of the null hypothesis (No Linear relationship). Hence the parameters that follow a linear relationship can be said to increase throughput, improving latency and improving the overall performance of the system. However, for the parameters that have a non-linear relationship a different method should be used in practice. This trend conforms generally across the decoded datasets and across the average values given for each parameter in Fig 5.

Image Dataset



Batch Size	Batch Numbe	Dataset size	Repetitio	Throughput img/s	Avg Batch Size	Avg Batch Numbe	Average Repetitio
2	3	6	1	8.5267	8.6539	10.8637	11.6864
2	3	6	2	9.8405	8.6539	10.8637	11.3633
2	6	12	1	7.6539	8.6539	10.6576	11.6864
2	6	12	2	8.5944	8.6539	10.6576	11.3633
4	3	12	1	12.7924	12.8675	10.8637	11.6864
4	3	12	2	12.2952	12.8675	10.8637	11.3633
4	6	24	1	12.9315	12.8675	10.6576	11.6864
4	6	24	2	13.4508	12.8675	10.6576	11.3633

Decoded Dataset



Batch Size	Batch Numbe	Dataset size	Repetitio	Throughput img/s	Avg Batch Size	Avg Batch Numbe	Average Repetitio
2	3	6	1	26.6951	41.6789	41.3726	81.0189
2	3	6	2	34.5163	41.6789	41.3726	72.1548
2	6	12	1	43.4083	41.6789	107.0073	81.0189
2	6	12	2	62.0957	41.6789	107.0073	72.1548
4	3	12	1	68.5986	106.7011	41.3726	81.0189
4	3	12	2	35.6805	106.7011	41.3726	72.1548
4	6	24	1	151.8266	106.7011	107.0073	81.0189
4	6	24	2	170.6985	106.7011	107.0073	72.1548

Fig 5. Table of parameters and their averages.

Question 3: Write TFRecord files to the cloud with Spark

Classroom tasks focussed on text data whereas in this assignment we use image data. We used spark to distribute the workload across multiple machines by using the "RDD.mapPartitionsWithIndex" function for each partition.

Two tests were undertaken with varying cloud configurations (details below). Test 1 with Four machines, 2 cores each, workers and double the resources finished faster than test 2 with one machine with 8 cores and no workers. We conclude that more machines working in parallel can reduce computational times and are more scalable. If one machine fails, then we can rely on the support of the other machines to complete the tasks.

CPU Utilisation reached near 100% for Test 1 and approx. 30% to Test 2. Disk storage was read/written more frequently, and more operations were executed over the network in Test 1. Test 1 therefore outperformed Test 2.

<u>Test 1: 4x2 Cores, Master Memory: SSD, Disk Size: 100GB, Workers Size: 100GB</u>

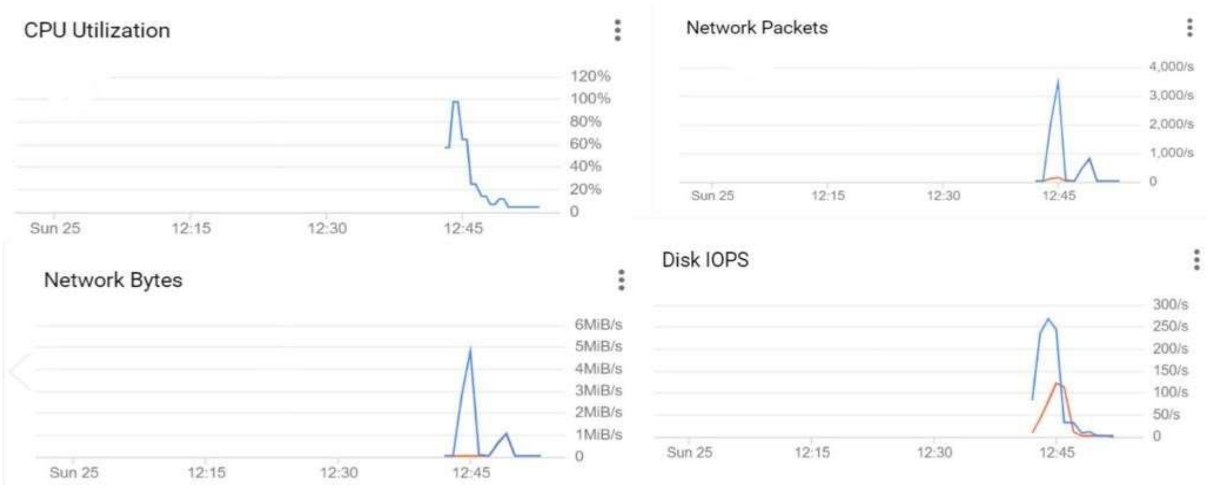


Fig 6. Usages of CPU, Network and Disk IOPS (note: these also approximately measure worker nodes)

<u>Test 2: 1x8 Cores, Master Memory: Standard, Disk Size: 50GB, Workers Size: N/A</u>

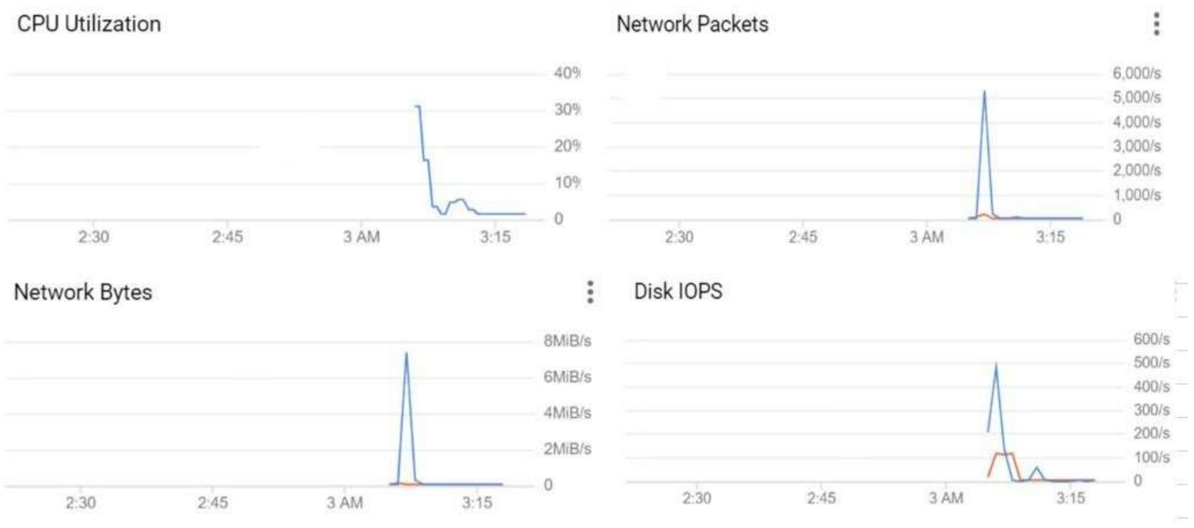


Fig 7. Usages of CPU, Network and Disk IOPS

Question 4: Machine learning in the cloud

We experiment the accuracy and training performance by applying the below distributed learning strategies and details.

Experimentation Details

Our strategies for this task were over three types:

- OneDevice
- Mirrored
- MultiWorkerMirrored Node types used:

- Standard (standard_gpu, 1 K80 GPU) – 8 CPUs/Node
- Complex (complex_model_1_GPU, 8 K80 GPUs) – 32 CPUs/Node Batch

sizes Used:

- 64
- 128

Epochs Used:

- 20

Assumptions

- The memory (Standard –30 GB, complex –120 GB) is unlikely to be a factor in our experimentation.

We ran our combination of parameters (strategy, cluster, batch size), to see if we get any improvement in our results. Increasing batch size across the ‘No strategy’ improved the wall clock time but changing the cluster to complex had negative effect.

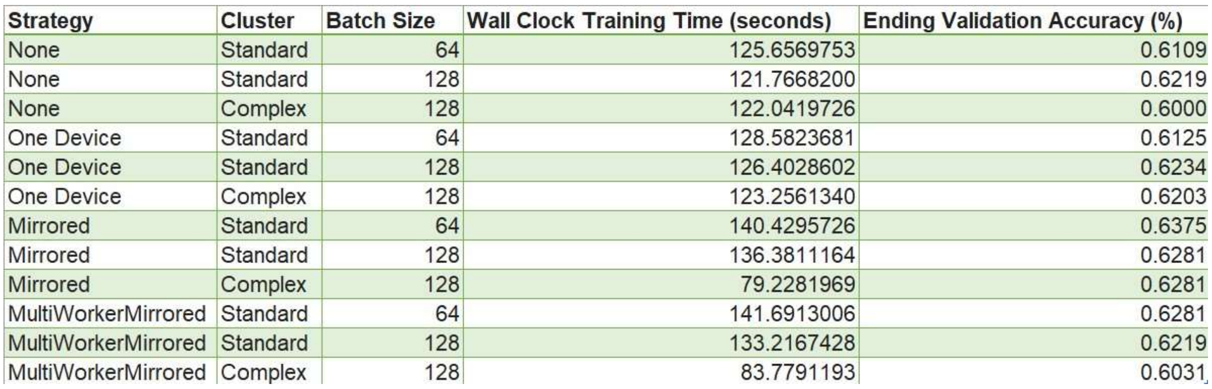
The ‘One Device’ strategy places all variables and computations on one GPU, so changing between standard and complex would offer little difference in the time considering the GPUs are the same (time: 126 V 123). Accuracy also remained consistent, however the increase in batch size seems to be slightly helping the model fit time.

Our mirrored strategy keeps its variables in sync by applying identical updates. This seems to benefit the most from a change in cluster to complex, however the accuracy remained unchanged (Time: 136 V 79). Parallelization distributed the workload across numerous machines whereas the higher batch size reduced the number of parameter updates hence improving training times.

MultiWorkerMirrored strategy is like mirrored strategy but also creates copies of all variables in the model on each device across all workers. This strategy run time reduced approx. similarly to the Mirrored strategy but not by as much (Time: 133 V 83). However, this strategy had an improvement in accuracy.

Our best accuracy was achieved on a standard mirrored strategy with a batch size of 64.

A limitation of increasing batch sizes and number of GPUs is that it can get very expensive. This could also give rise to higher errors and latency, higher network traffic etc. There should be a trade-off between cost and time such as using a lower number of batch sizes or using standard cluster types. Alternatively, another expensive option could be to use TPUs which are designed for powerful performance and flexibility using a TensorFlow Framework.



Strategy	Cluster	Batch Size	Wall Clock Training Time (seconds)	Ending Validation Accuracy (%)
None	Standard	64	125.6569753	0.6109
None	Standard	128	121.7668200	0.6219
None	Complex	128	122.0419726	0.6000
One Device	Standard	64	128.5823681	0.6125
One Device	Standard	128	126.4028602	0.6234
One Device	Complex	128	123.2561340	0.6203
Mirrored	Standard	64	140.4295726	0.6375
Mirrored	Standard	128	136.3811164	0.6281
Mirrored	Complex	128	79.2281969	0.6281
MultiWorkerMirrored	Standard	64	141.6913006	0.6281
MultiWorkerMirrored	Standard	128	133.2167428	0.6219
MultiWorkerMirrored	Complex	128	83.7791193	0.6031

Fig 8. Results of investigation into strategy, cluster, and batch size

Question 5: Theoretical Discussion

Contextualise

We will take a deeper look into our parameter results and local/cloud setups in relation to the two papers given in this section.

Batch size

We saw in our report (Task 1 and 2) that increasing the batch size resulted in a larger throughput for images per second in both local and cloud speed test results and an improvement in our fitting time for the model in later tasks.

Smith et al (2018) introduced the concept of varying the batch size during the training of neural networks as an important method of improving the model in comparison to other parameters such as learning rate. Once increasing batch size locally has reached its peak, there would be no benefit in additional increases. This gives us reason to move tasks to other big data technologies or to the cloud where we can parallelise the training across multiple machines in the cloud, hence reducing training time even further. Our results saw this in practice where we saw additional workers training models faster than less or no workers.

Alipourfard et al (2017) said that increasing batch size during training can reduce model training times without sacrificing ending validation accuracy. This is also evident in our results (Fig 8.), we see that increasing batch sizes improves the wall clock time whilst keeping the ending validation accuracy approx. in line with expectations. A larger batch size could negatively affect our metrics that we use to test however the cost benefit might outweigh due to the advantage of improving running time over costs. We saw in task 4 where we trained our models in different scenarios that a higher batch size can mean less parameter updates are required in comparison to other improvement methods that require higher computations.

Repetition

Alipourfard et al (2017) discusses the frequency with which tasks are repeated in big data analytics, with approx. 40% of all tasks considered to be recurring. We found in our report (Task 2 and 3) that caching our repetitive data by storing for quicker access improved the speed of our tasks by approx. 1.5x. Hence in similar scenarios as an example, high memory clusters give cloud configurations further importance.

Strategize

We provide different strategies for three scenarios described below:

Batch Processing processes large blocks of data in one go. This is not immediate, and the data is instead stored and is processed during a pre-defined batch time. Batch processing is ideal for automated queries and tasks. Task 3 and 4 showed us that parallelisation improves computational speeds by spreading the workload amongst workers. A cloud configuration would require a high throughput and therefore would also require CPUs and larger disk I/O. Disadvantage of batch processing in the cloud is that it cannot use caching due to the large amounts of data in each batch. Hence why a larger I/O bandwidth is beneficial as the data needs to be stored.

Real time processing is data that is processed continuously in real time instead of a predefined time period. This allows us to achieve instant results and is somewhat important for several scenarios such as banking (processing transactions). The data is not typically very large and hence would not require disk storage and instead it can take advantage of the memory of the computer and caching. As the data is continuous in nature, it will most likely require a higher network bandwidth to cope with the data being continuously input and output. Less data being processed means CPUs are usually not required. Real time processing would be best achieved using low disk storage, lower CPU count, high bandwidth and memory, caching, and less workers.

Stream processing is data that is analysed instantaneously where the data is split into small batches and shorter time intervals. This is similar to real-time processing, however more CPU use maybe required in comparison to real-time processing depending on the timing intervals for the data output. For this scenario to show success, we can focus on high memory, network bandwidth, CPU, Caching, low worker count and low disk.

Word count: 1979

Notebook Link:

https://colab.research.google.com/drive/1BO7hNMy2tKs5jMNuSxG6sb4pTuqwE_I7?usp=sharing

References

- Alipourfard, O., Liu, H. H., Chen, J., Venkataraman, S., Yu, M., & Zhang, M. (2017). Cherrypick: Adaptively unearthing the best cloud configurations for big data analytics.. In USENIX NSDI 17 (pp. 469-482).
- Smith, S. L., Kindermans, P. J., Ying, C., & Le, Q. V. (2018). Don't Decay the Learning Rate, Increase the Batch Size. In ICLR (no pagination).