Big Data Report

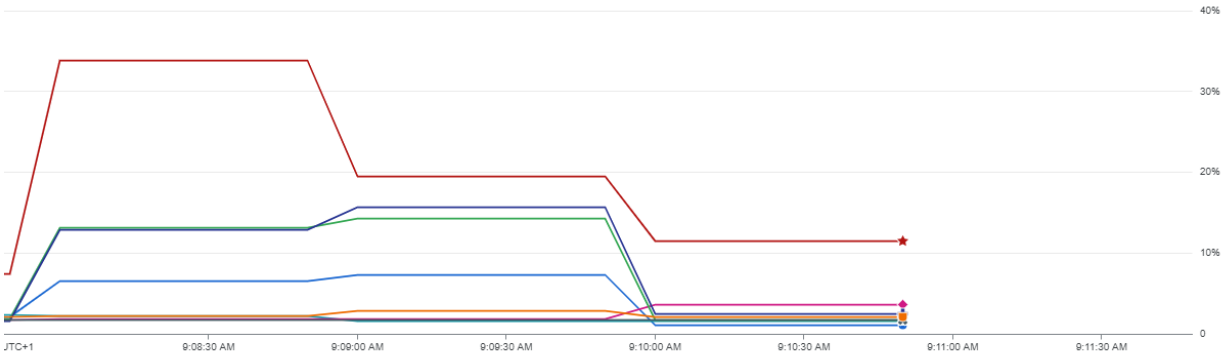
Benny Collins: 210036917

1d) i)

The processing time, without increasing the number of partitions used in parallelisation, was 0.85097 seconds (to 5 s.f.), whereas processing time was 0.74448 seconds (to 5 s.f.) once the number of partitions had been increased to 8 - equal to the number of machines in the cluster.

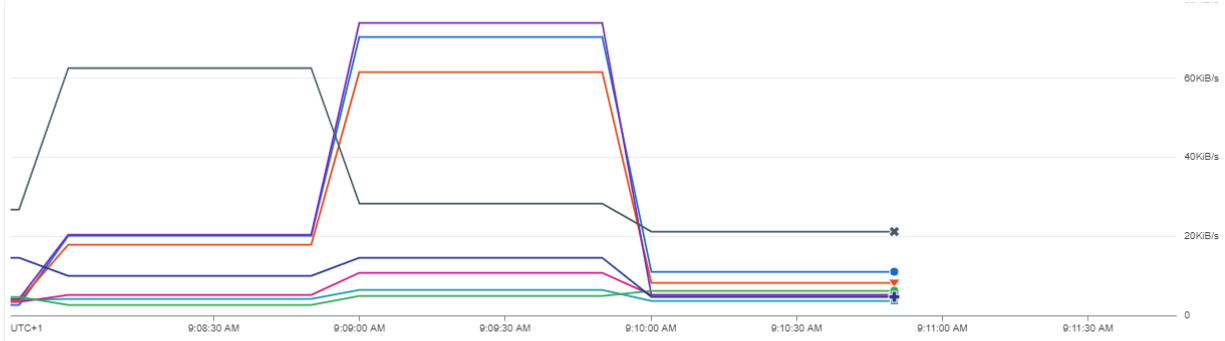
Number of partitions = 2:

CPU UTILISATION:



DISK READ BYTES:

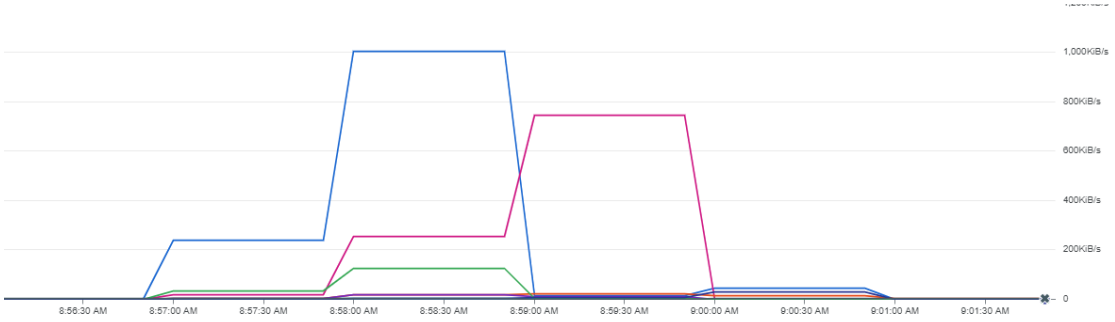
DISK WRITE BYTES:



Number of partitions = 8:

CPU UTILISATION:

DISK READ BYTES:



DISK WRITE BYTES:

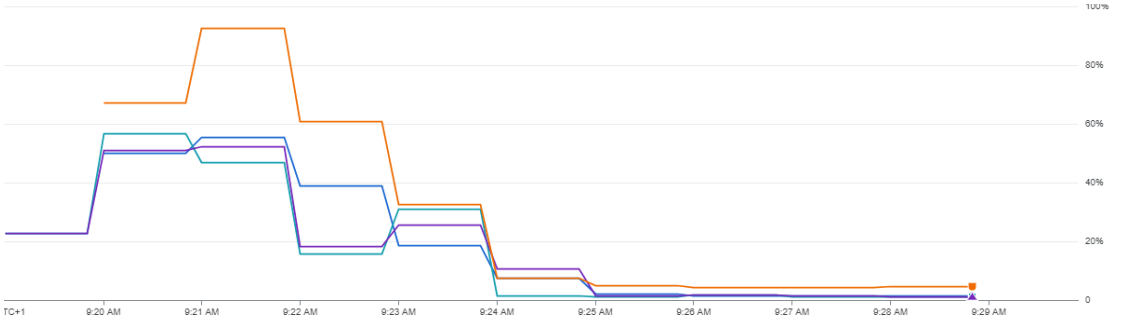
In the first set of graphs, we can see that about half of the machines have low values for CPU utilisation, disk read bytes and disk write bytes. This is contrasted with the corresponding graphs for the file run with 8 partitions, where CPU utilisation for all 8 machines peaks above 10%, compared to only 3 machines peaking above 10% when the number of partitions is equal to 2. Higher values of disk write bytes are also seen in the second set of graphs. The most insignificant changes are seen in the disk read bytes graphs, where levels are relatively similar for the file run with 2 partitions and the file run with 8. This is due to parallelisation being applied after reading the files and before writing the files.

1d) ii)

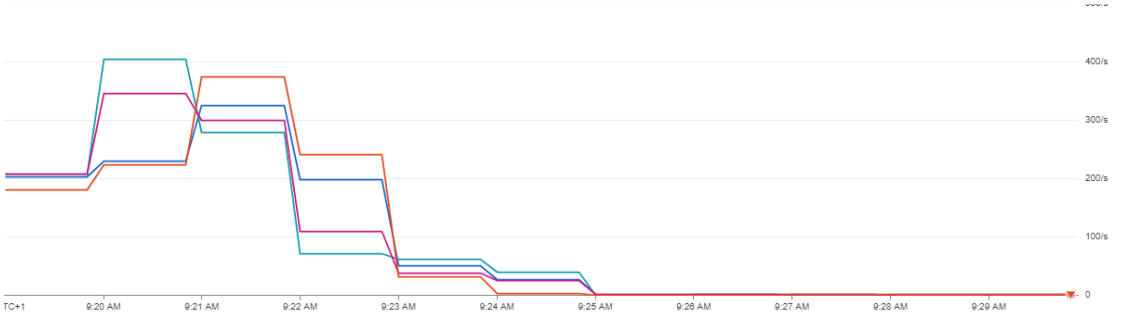
The graphs below show indications of the purpose of parallelisation. The machines in the 4-machine cluster, for disk read and write operations as well as received and sent packets, all have lower peaks than the single machine in the second cluster. This is because the work has been partitioned and distributed across multiple machines, allowing for parallel processing. This results in the machines sharing the work, whereas the single machine cluster has to deal with all of the processing, resulting in higher values for these metrics. The 8 CPUs used in the single machine, in comparison to the 2 CPUs in each machine in the 4-machine cluster, allows the single machine to achieve better individual performance in these metrics than the machines in the 4-machine cluster. This also explains the lower CPU utilisation value for the single machine, as it comparatively has more CPUs available for computation.

4 Machine Cluster:

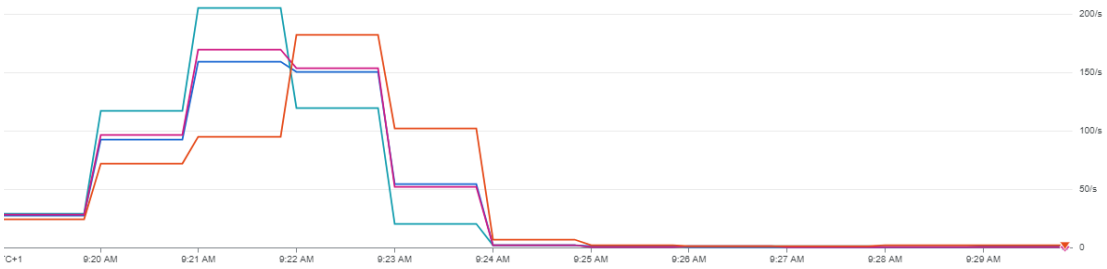
CPU UTILISATION:



DISK READ OPERATIONS:

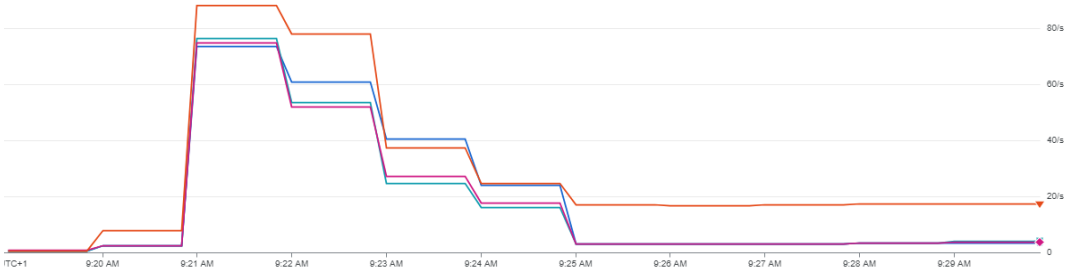


DISK WRITE OPERATIONS:



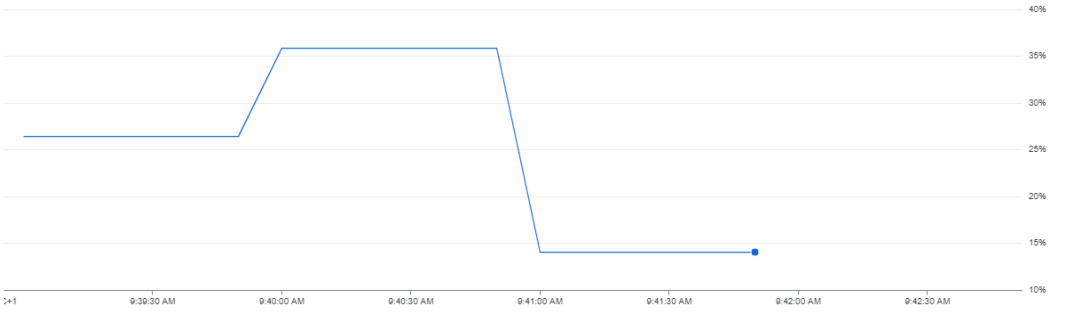
RECEIVED PACKETS:

SENT PACKETS:

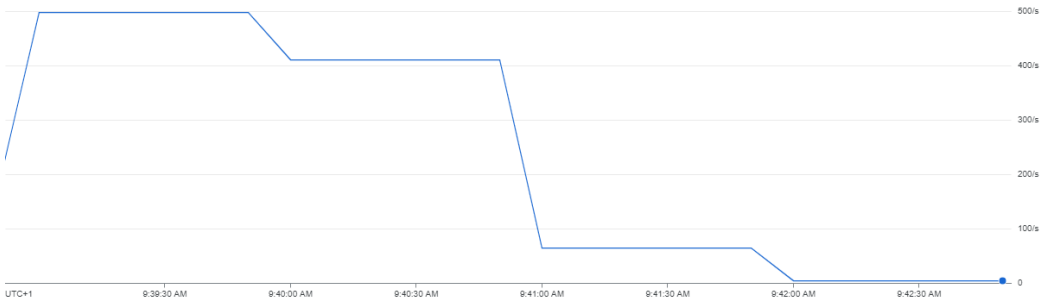


Single Machine Cluster:

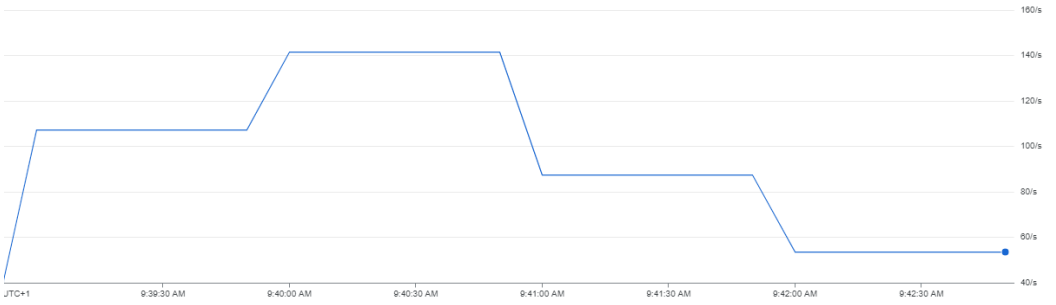
CPU UTILISATION:



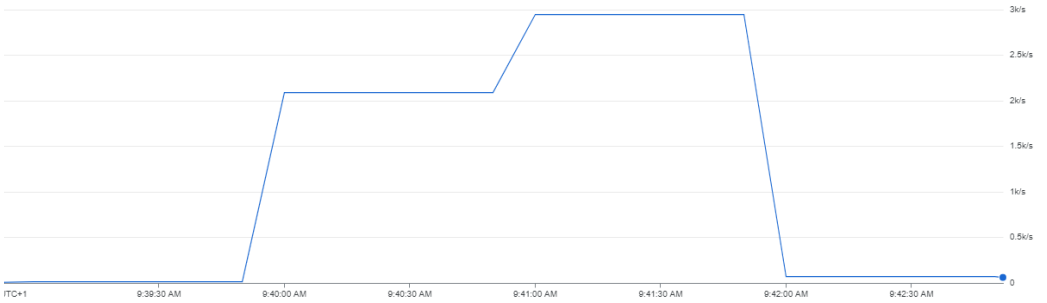
DISK READ OPERATIONS:



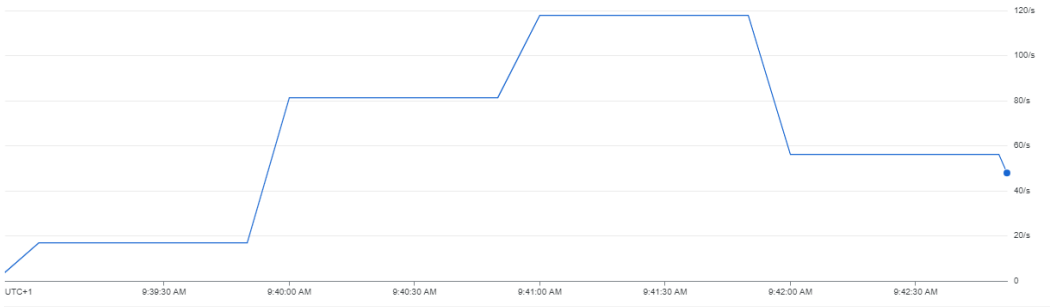
DISK WRITE OPERATIONS:



RECEIVED PACKETS:



SENT PACKETS:



1d) iii)

In our labs we were parallelising the data to create RDDs locally. In this context, however, the parallelisation happens in our .py file, which we are running in a remote cluster. The partitions are sent to each separate CPU, allowing for parallel computation. On top of this, the use of very large datasets provides the incentive to utilise parallelisation.

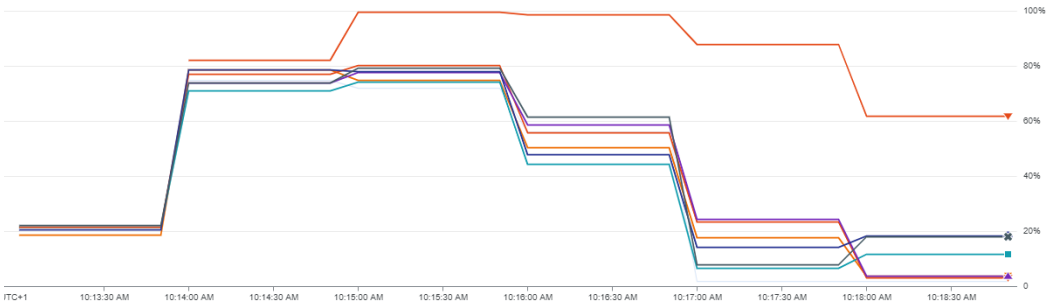
2c)

Caching involves storing data in high-speed, often expensive, memory to allow for extremely fast data retrieval when necessary. Due to the lazy evaluation of transformations, which are heavily utilised in this file, it is most appropriate to cache the initial results from which the final results are obtained, using collect.
To measure the effects of caching, we time the part of the file where the transformations are called and then collected:

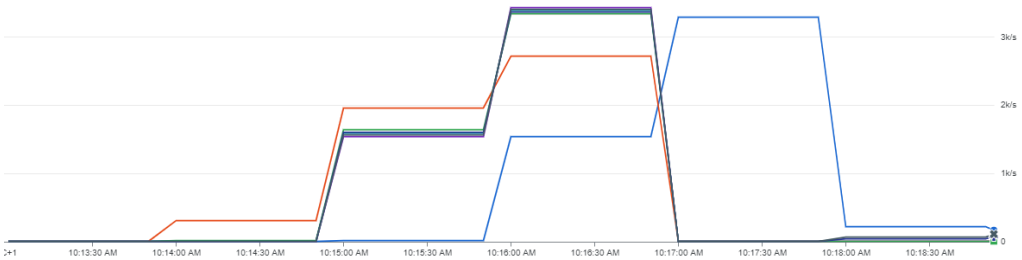
	Without Caching	**With Caching**
Transformation Time (secs)	34.87568211555481	1.6792850494384766

No Caching:

CPU Utilisation.

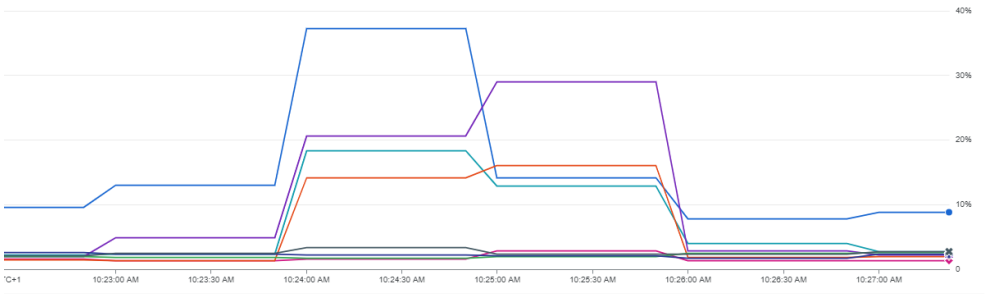


Received Packets

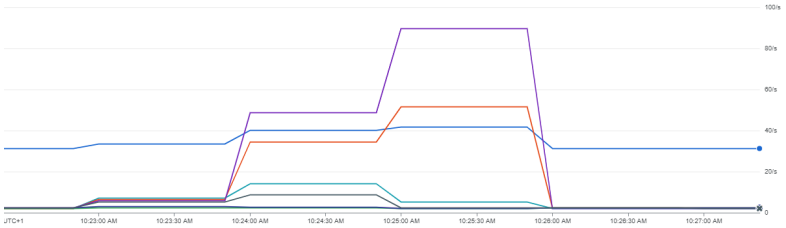


Caching:

CPU Utilisation.



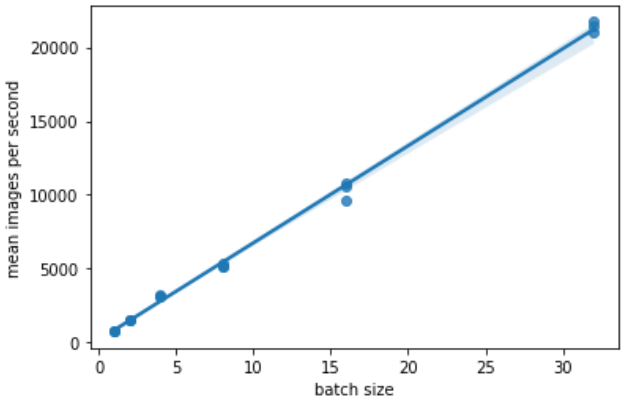
Received Packets

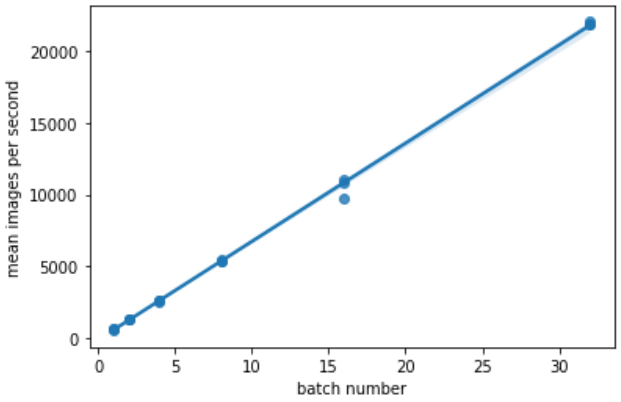


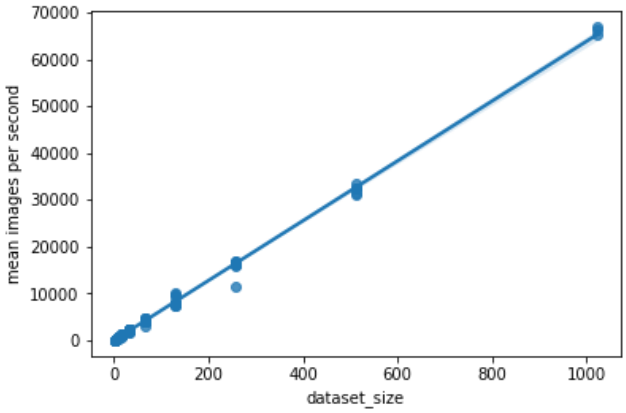
2d)

TF Records Dataset:

We can see, from the linear regression graphs, that a given parameter value almost has a positive linear relationship with images read per second, meaning the larger the data we are reading, the faster the images are read.





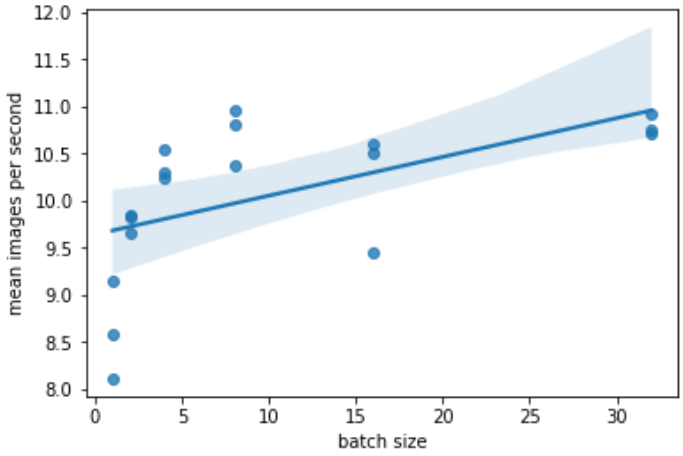


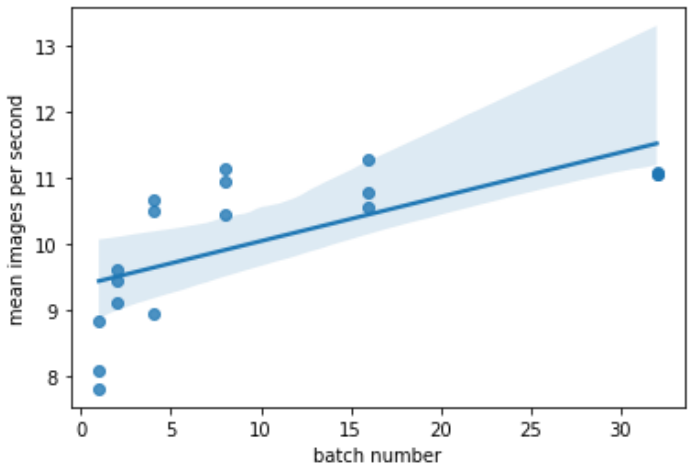
Dataset Size	Repetition 1	Repetition 2	Repetition 3
1	47.2	71.3	75.7
2	150.1	139.3	142.15
4	293.1	298.0	299.3
8	533.8	593.3	591
16	1106.6	1077.8	1089.5

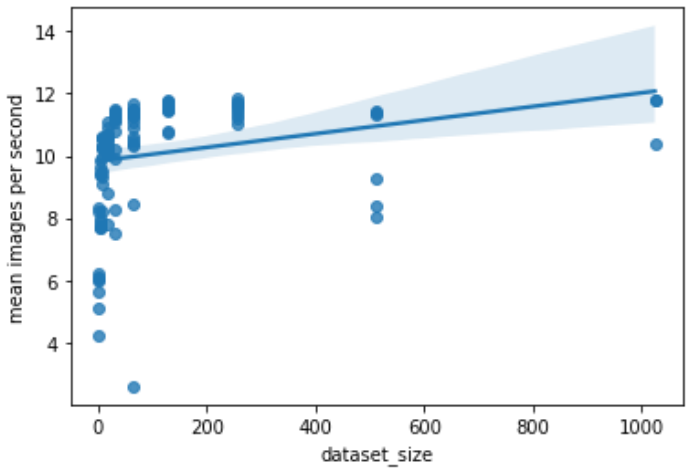
32	2147.4	2254.7	2216.3
64	4239.7	4032.8	4333.9
128	8358.9	8221.5	8412.0
256	16426.8	14877.8	16270.9
512	32922.7	31676.4	31987.6
1024	66882.1	65245.3	66184.4

Decoded Dataset:

We can see, from the linear regression graphs, that a given parameter value has a much less linear relationship with images read per second, when using this dataset. We also observe significantly lower values for images read per second.





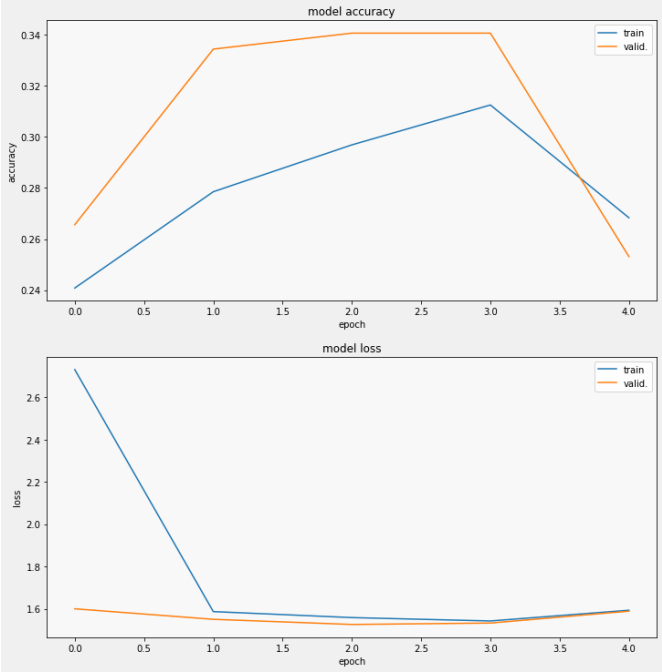


Dataset Size	Repetition 1	Repetition 2	Repetition 3
1	5.1	6.2	4.2
2	5.9	8.3	6.1
4	9.1	8.4	8.4
8	9.9	10.2	9.7
16	10.0	10.7	10.0
32	11.1	10.6	10.6
64	11.4	10.4	9.2
128	11.6	11.4	11.4
256	11.4	11.6	11.4
512	8.8	9.7	11.4
1024	11.8	11.8	10.4

3c)

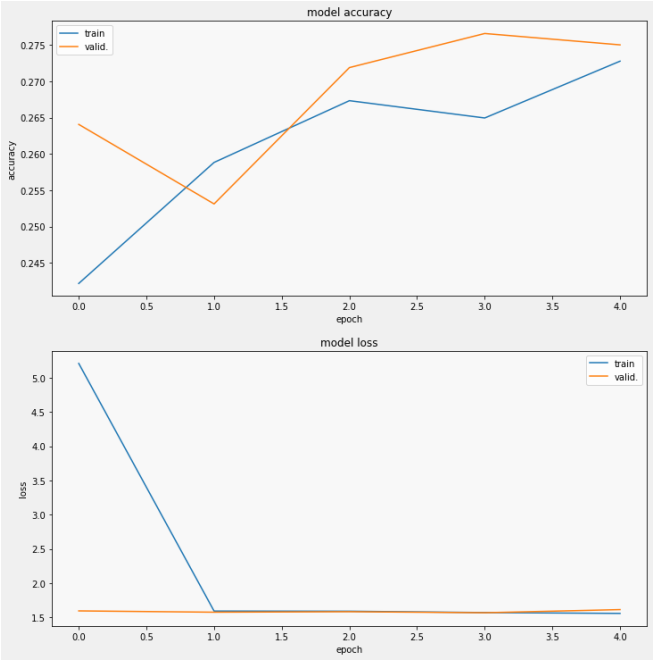
No Strategy:

We can see that the accuracy improves over the first 3 epochs and then decreases over the final epoch, which could imply overfitting.



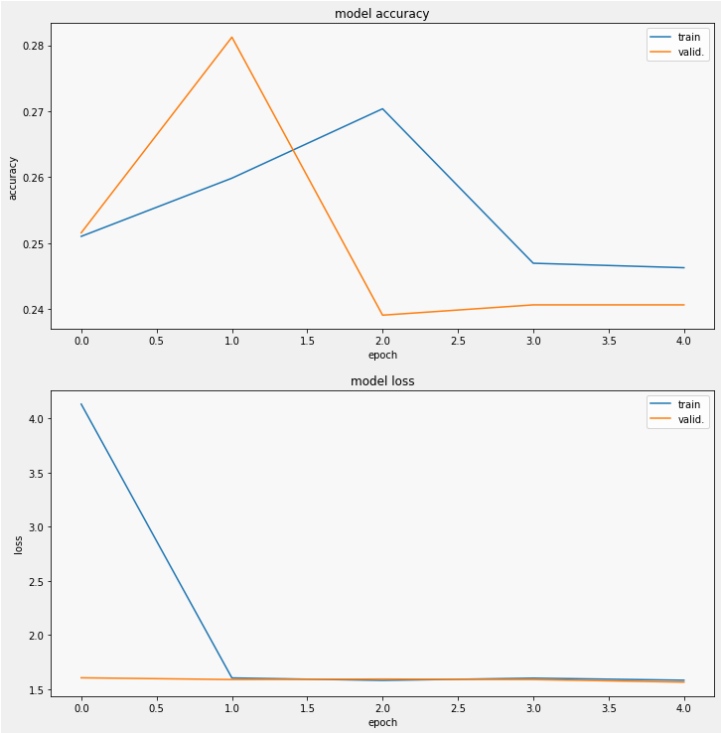
Mirrored Strategy:

The mirrored strategy, run with batch size 128, results in an initial decrease in validation accuracy, with increases occurring from the end of epoch 1 and onwards. The validation accuracy does not reach the same levels as the accuracy for the model run with no strategy, but it finishes the epochs on a higher value. The validation loss remains minimal throughout.



Central Storage Strategy:

The central storage strategy, run with batch size 128, results in an initial increase in validation accuracy. However, from the end of epoch 1 to the end of epoch 2, the validation accuracy decreases massively, stagnating around this level until the end of the epochs. Again, the validation loss remains minimal and relatively unvarying throughout.



4)

The article by Alipourfard, in Figure 13, confirms the importance of the number of nodes in a cluster and the extent to which it affects computation time. The paper also explores the effects of RAM and CPU on computation time, often referring to a CPU/RAM ratio, deeming the importance of CPU to be higher than that of RAM. These cluster parameter importance evaluations are particularly relevant to the first two tasks in this coursework. This information could be used to tune the parameters of the clusters, utilised in task 1 and task 2, in order to improve computation time as well as avoiding allocating unnecessary resources to a particular parameter. The paper confirms our understanding of the effects of increasing nodes or CPU on computation time, as well as establishing a methodology for efficiently tuning cluster parameters, which would prove useful in future work with clusters.

The second article identifies limitations of parallel strategies as well as bottlenecking that might be encountered in distributed systems that are using gradient descent. This is particularly relevant to the mirrored strategy used in question 3's neural network training, which involves distributed training across multiple replicas of a GPU on a single machine. This is also applicable to the central storage strategy that does computations in a single local GPU, which communicates with the CPU that is storing the model.

Colab notebook link:
https://colab.research.google.com/drive/1ywU4796D_vm0X0SBLK9KqAaSFKAZu9e2?usp=sharing