Big Data Coursework Report

Fasih Munir
fasih.munir@city.ac.uk

Answers to Individual Tasks

Question 1d.i

Figures 1 - 4 below demonstrate the following:

1. Is a single machine (1 master) with no second parameter for parallelization
    a. Total elapsed time - 27 seconds
2. Is a maximal machine (1 master 3 workers) with no second parameter for parallelization
    a. Total elapsed time - 37 seconds
3. Is a maximal machine (1 master 3 workers) with a second parameter for parallelization
    a. Total elapsed time - 26 seconds
4. Is a single machine (1 master) with a second parameter for parallelization
    a. Total elapsed time - 21 seconds

While I was unable to find the processing time specifically, I have given the total elapsed time as seen in Google Cloud Platform > Dataproc > Jobs.

Spark parallelization is a process in which a Resilient Distributed Dataset (RDD) is created when trying to distribute data across clusters. Distribution is useful as it increases efficiency and speed of tasks (optimizes processing performance). You can process tasks simultaneously, divide tasks amongst resources avoiding bottlenecks and scale by adding virtual machines.
The parallelize function takes a second parameter that specifies partitions (number of data splits). Specifying this parameter controls for optimization as using a number too low may not use resources optimally but setting it too high might increase costs.

The partitions chosen were 16 (notebook default). We can reason what will happen with other sizes as well. The maximal cluster improved total job time by 11 seconds and single machine cluster by 6 seconds. This shows that 16 partitions is good resource distribution as the time taken to process was reduced by a large margin. The single machine with 1 master node did not show as large of a change which likely means that another partition value would perform better. In both clusters partitioning helped reduce CPU utilization as well.

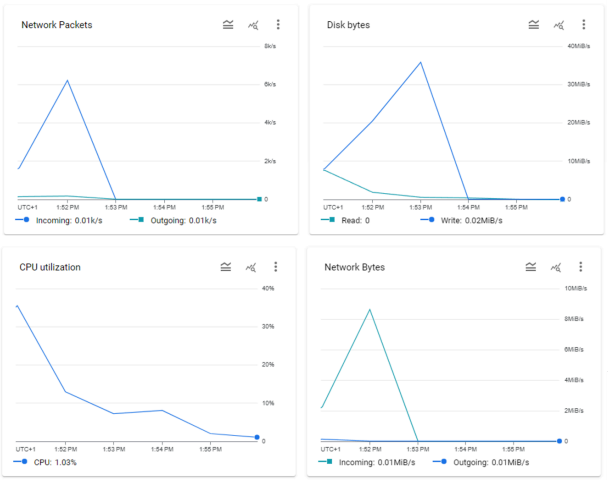


Figure 1

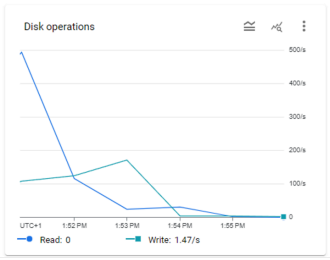


Figure 2

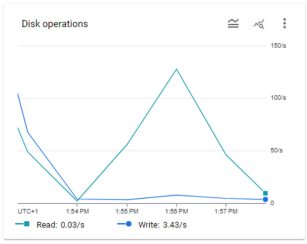


Figure 3

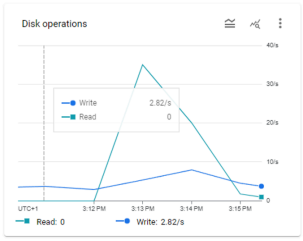


Figure 4

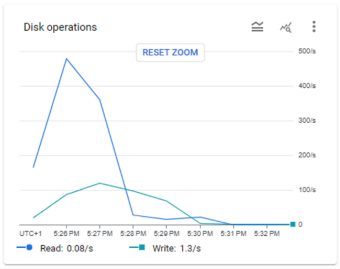
Question 1d.ii

Figures 5 and 6 below demonstrate the following:

5. Is a single machine (1 master) with 8 times the resources
6. Is a multiple machines (1 master and 3 workers) with double the resources each

Chosen partitions were 16. The Network Packets for Figure 5 were lower and saw a much sharper usage peak as compared to Figure 6, meaning that the single machine is more efficient at data packing (better at organizing the data). Network Bytes for Figure 5 are lower following a similar trend as Network Packets compared to Figure 6. This makes sense given that that single machine with 8 times the resources is better at packing the data before sending. Disk Bytes are also lower for Figure 5 with sharp usage peaks while Figure 6 sustains its peak for a few minutes before decreasing. The data being written in Figure 6 is sustained suggesting the cluster will need consistently higher resource allocation while the peak in Figure 5 suggests an ability to handle increasing loads. The plateaus will lead to higher costs on the cloud. Interestingly, even though Figure 5 has more resources than Figure 6, Figure 6 has higher Disk Operations capacity. This is likely because the cluster has multiple machines allowing operations to be split across nodes for simultaneous reading and writing rather than being limited to a single disk.

Figure 5



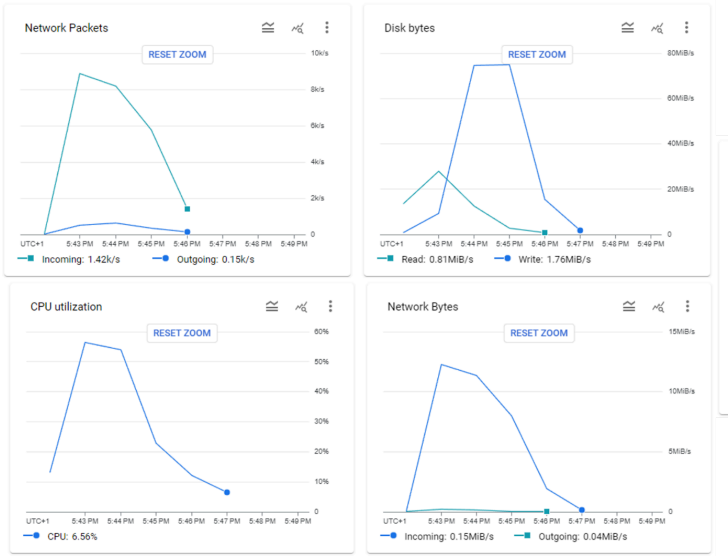
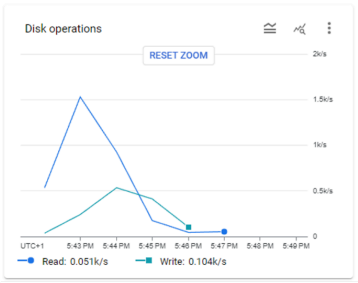


Figure 6



Question 1d.iii

Most lab sessions and standard applications like local file systems run locally.This means you are limited to the power of your machine, if it has 8GB of RAM then that is all you have. Spark can run on the cloud and can scale by allocating to virtual machines. This removes power bottlenecks however externalities such as zoning conflicts or unavailability of resources due to high demand can cause problems. Data is also stored on a virtual server in another country which can lead to privacy concerns especially for sensitive information. In this coursework we have parallelized our data by dividing it into smaller packets and distributing it across different nodes to speed up operations.

Question 2c

Caching stores results on memory across the job. Storing results allows reuse for other computations without recomputing. For example, collected RDDs can be cached and ,without recollecting again, used for other transformations. In this project, caching was added when data was mapped to the “time_config” function and when data was grouped by key before taking averages. The mapped data was further used to map the combination function and the grouped data was used to calculate averages for each group. These changes were made due to the number of computations being done on these instances post creation. These changes reduced total elapsed time of the job from approximately 50 minutes to 5 minutes. Our caching in the code was successful and gives room to experiment with different kinds of parameter combinations without the risk of increasing time spent waiting.

Question 2d

Figure 7 below shows the results of the linear regressions per parameter.

Figure 7

	Parameter	Slope	Intercept	R-value	P-value	Std Error
0	Processed Batch Size	0.669380	11.296686	0.562240	6.955858e-15	0.077836
1	Processed Batch Number	0.449395	10.828441	0.516866	1.905145e-12	0.058843
2	Processed Repetitions	-0.186191	14.346586	-0.078195	3.226310e-01	0.187667
3	Processed Batch Size x Batch Number	0.090555	11.438651	0.697114	6.796746e-25	0.007363
4	Unprocessed Batch Size	34.658317	16.811811	0.729943	3.108368e-28	2.565675
5	Unprocessed Batch Number	20.234375	13.804457	0.583542	3.665614e-16	2.226169
6	Unprocessed Repetitions	-1.672887	158.790853	-0.017616	8.239234e-01	7.506214
7	Unprocessed Batch Size x Batch Number	4.939494	17.139257	0.953463	3.221420e-85	0.123487

Note that “processed data” = dataset4/image files and “unprocessed data” = tfrecord files.
We observed high correlation (R value) of batch size and batch number with throughput suggesting a medium to strong linear relationship. Increasing batch size or batch number will increase throughput and is reasonably predictable. Further, the product of the 2 parameters shows an increasingly higher positive relationship suggesting that their joint use is more effective than their individual use. The processing done (resizing and cropping) affects the relationship between parameters and throughput by reducing the strength suggesting that the kind of processing will affect throughput. This is also reflected by the size of the slope where unprocessed data has a higher rate of change while processed data is stable meaning that in unprocessed data, batch size and batch number are better predictors. The repetitions show the least correlation and strength and all other p-values are significant so these results are not by chance.

Throughput is sensitive to batch size and number. In cloud environments where images may be stored across a network, fetching data can be slow compared to local machines (20,000 ns for 2k bytes over 1gbps network versus 100 ns on local memory). Storage type can affect performance with SSDs having faster read times than Disks (1mb sequentially, 1ms versus 20ms). Given it takes 150ms to send data between California and the Netherlands, having data centers close to data generation can affect latency.

This is similar to what would be expected from single/local machines where increasing physical resources like CPU power will improve throughput. They are different in that the cloud has external dependencies such as data travel distance or shared resources. Cloud providers might tie throughput to disk resources because it is predictable given our results. It becomes easier to know when to increase/decrease resources to handle load.

For cloud speed tests, we should consider storage, given that SSDs outperform regular disks. We must account for how far data travels before accessing results. We have seen caching significantly affect speed, so considering how systems and code cache results is also important. We could be bottlenecked by shared resources which can degrade performance or (as in the project) make the resources unavailable. The service type used comes with quotas which can limit how much you can do. The speed test in this project was done on a multiple machine cluster which (as observed) had higher disk operation capacity. Given the correlation of throughput to disk resources we might have seen slower times. Additionally we saw the improvement caching brought.

The use of linear models allows for simple interpretation of relationships and identification of trends. However this is only true if relationships are linear. Given externalities present on the cloud, if we scale resources these relationships may become non-linear which can make linear models difficult in practice. It would be wise to incorporate non-linear models to account for potential complex relationships and adapt them through monitoring via tools such as Dataproc.

Question 3a

We had 2 main tasks in this coursework, both of which were to be tested on the cloud. The paper in question talks about finding an optimal cloud configuration for your tasks through an algorithm called “Cherry Pick”.

One of the sub tasks in this coursework was experimenting with different kinds of cluster configurations. We were manually trying different combinations and seeing how they affect a parameter space. We found that certain configurations work better such as a multiple machine configuration having higher disk capacities. This manual process could be replaced with cherry picking to identify the best configuration saving us search time and cost.

We also find that linear models show a strong relationship with batch sizes and numbers to throughput. Having these models running and monitored can help us stay on top of resource needs. Incorporating Bayesian Optimization as the paper does could help in better adjusting batch sizes and numbers to improve allocation of resources.

During this project we have also observed the cost to run these cloud procedures. While we worked with the free tier of google cloud platform, the paid tiers (higher tiers) offered larger access to compute resources. Since we were running tasks in Spark and running them in parallel across machines we would need to be careful of the cost involved with each job. Outside the context of the coursework, and at scale in production settings, using the cherry pick approach can help balance cost to performance.

Another aspect of the coursework involved running speed tests using different parameter combinations. Varying the batch size and batch number has an effect on the workload being sent to the cloud. To help with the workloads we looked at techniques such as partitioning and caching. Finding “representative workloads” as suggested in the paper can help account for varying input workloads that could occur in production environments.

Question 3b

Batch processing is when jobs are only processed when a certain number of jobs are collected and then they are processed together while stream processing is just a continuous input where, as the jobs come, they are processed.

Both concepts are generally related to the cherry picking concept in that you are looking to find a cost-effective and optimal resource allocation for your chosen set up. While optimal allocation will look different for batch and stream the core idea of finding the optimal remains the same. The difference in strategies will likely boil down to taking a historic approach to batch processing versus a real time approach to stream processing.

For batch processing, we could use the bayesian framework used for cherry picking to find an optimal configuration based on previous batches sent. This would be similar to typical predictive models. Target variables could be configurations and input data could be the different kinds of jobs, the time spent on the jobs and the data in the job amongst others. We could augment this by further adding a costing element to output a predicted cost for the configuration. Maybe if you are processing audio files you might need higher memory which can be costly so finding how high is high enough will be important. Depending on the tolerance for how many jobs are accumulated before sent, we can also adapt the idea of workload variation from cherry picking to account for scale either horizontally if we need more compute power for more jobs (for example if processing 4k video data) or vertically if we need more raw power for jobs that require more complex work such as image processing (as we have seen in our coursework where even slight processing on the image affects the relationship to throughput).

For stream processing, many of the ideas for batch processing can carry over but we could also use the adaptive nature of cherry picking to adjust the configuration of partitions or compute in real time based on the current need of the jobs that are being streamed in. This can be important for applications that are live-streamed such as live-streaming service Twitch. Resource availability could also be increased or decreased as needed.

Coursework Notebook

The colab notebook can be found here

Total Report Word Count = 1993