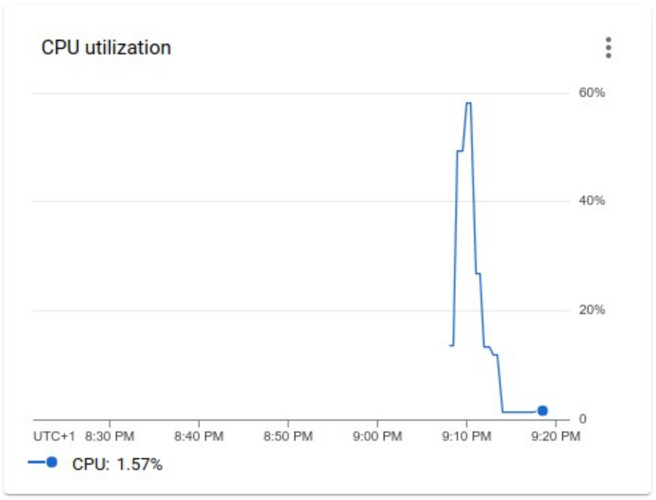
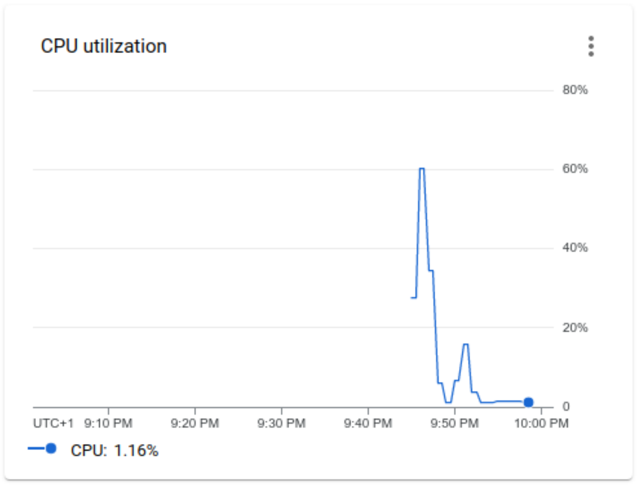
Report – 140000067
https://colab.research.google.com/drive/1isRAOiWAglBKqRYiqkHzQiFccj_eux3g?usp=sharing
1c – Part ii
My google platform account has 4 IP addresses with 8 CPUs. Therefore, the maximal cluster configuration I can have is 1 master machine with two CPUs and 3 workers with 2 CPUs each. The standard disk capacity for worker nodes to maximize throughput is 716/3= 238. The code and output are provided in the notebook.

1d-Part 1
The different in cluster utilization before and after the change based on different parameter values – with Maximal Cluster and Single Machine Cluster

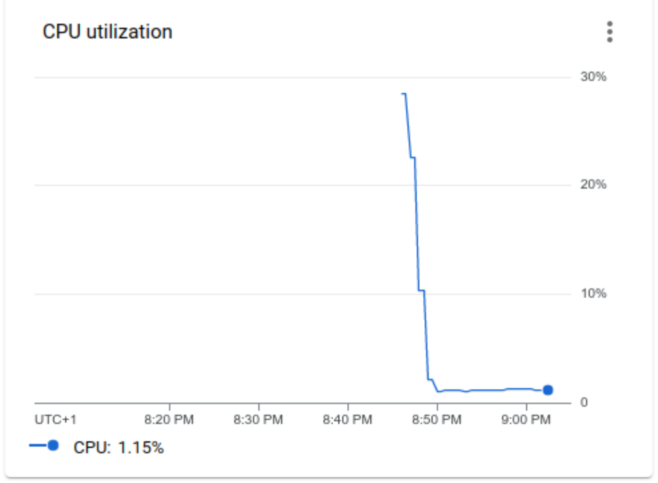
Before – Max Cluster



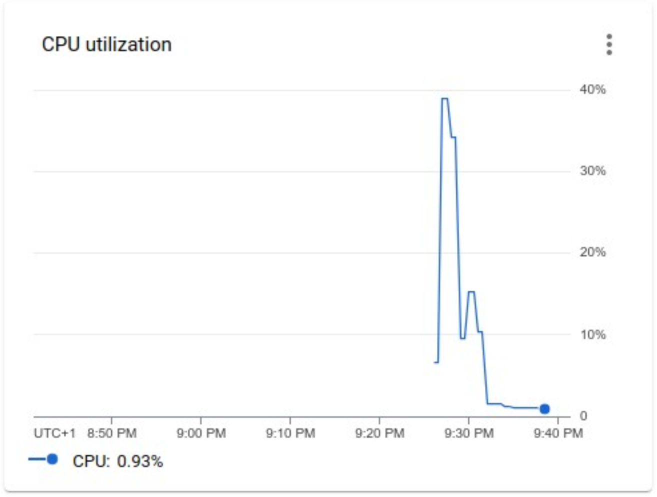
After – Max Cluster



Before – Single Machine Cluster



After – Single Machine Cluster



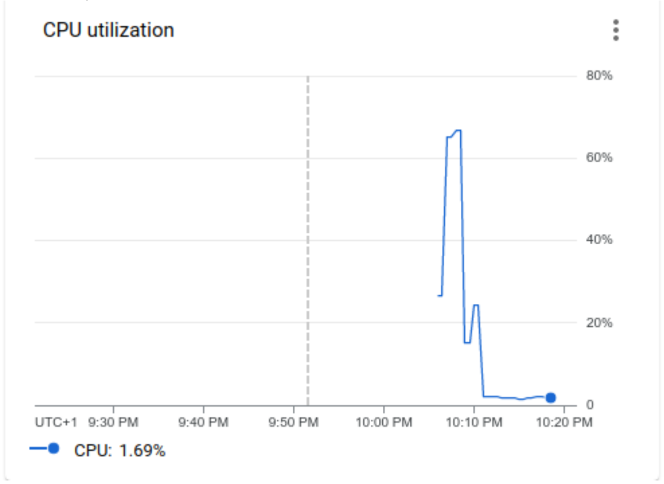
From the CPU utilization graphs, we can see the effect of using the second parameter in the initial call to parallelize. The CPU utilization for both single machine cluster and maximal machine cluster is higher than before the change, with larger change for single machine cluster. By using the second parameter, we have reduce the running time by half for maximal cluster and not much change for single machine cluster

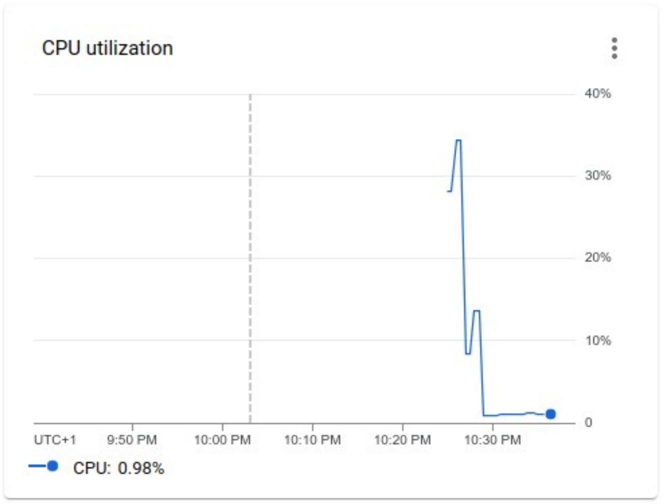
1d-Part II

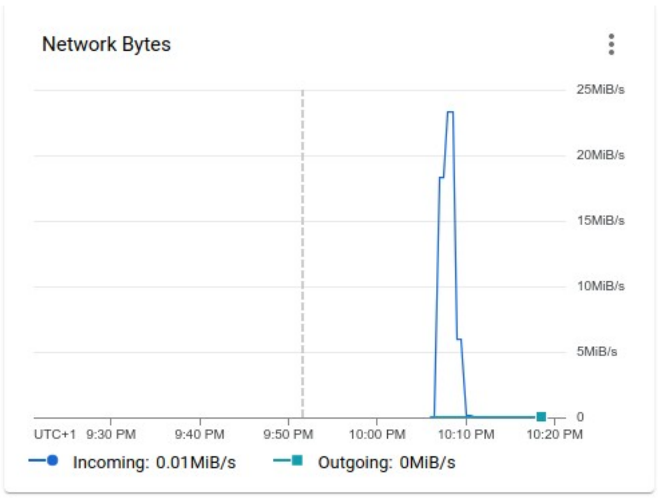
This experiments are done after changing the code in part 1d-i.

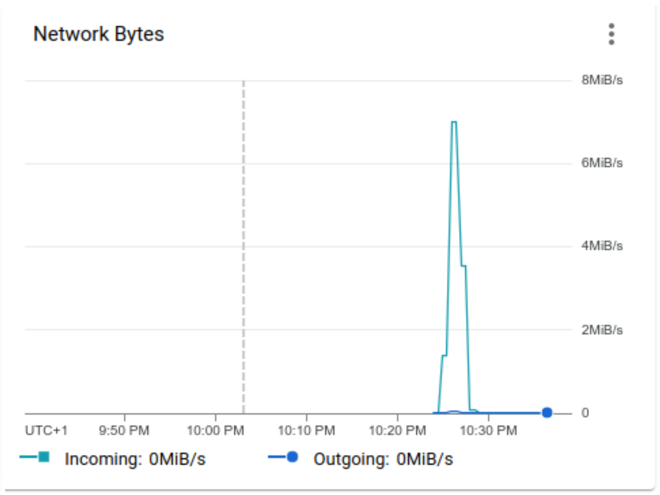
4 Machines (1 Master & 3 Workers each with 2 CPUs)

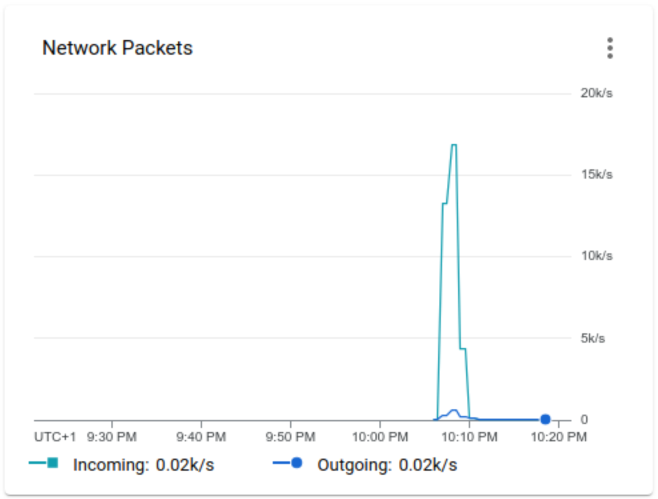
1 Machine with Eight-fold Resource & No Worker.

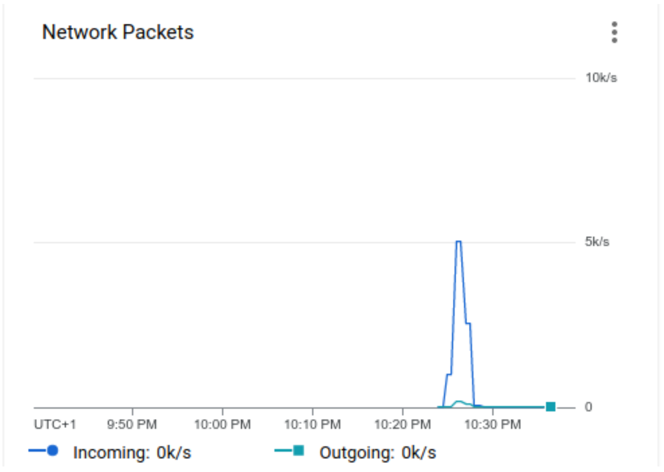


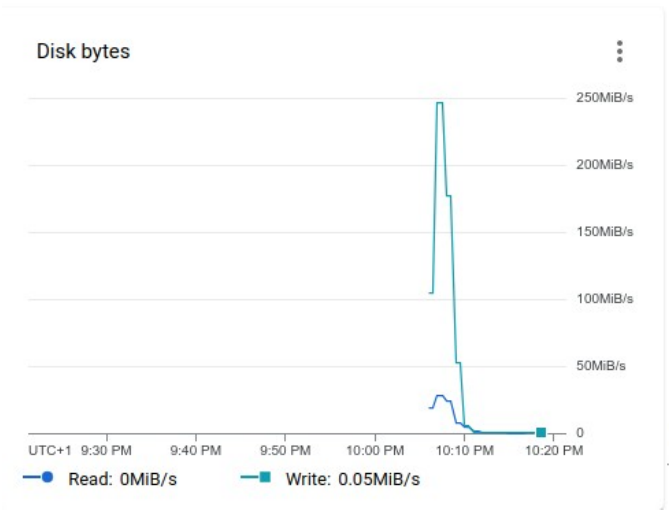


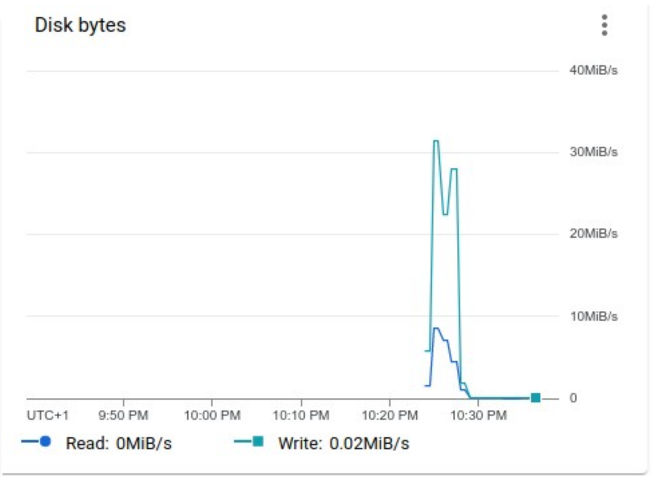


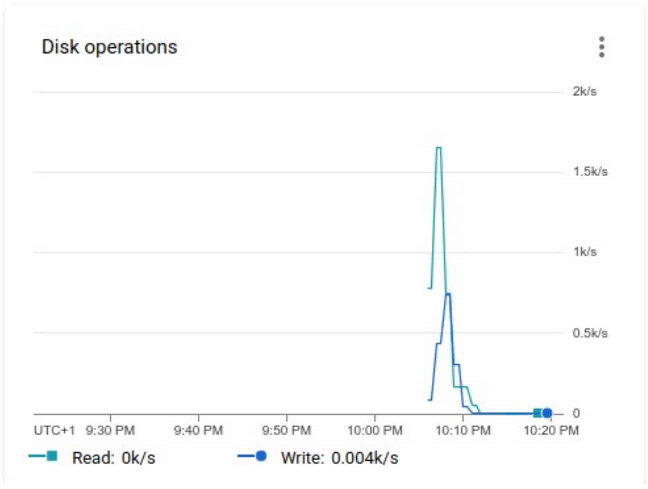


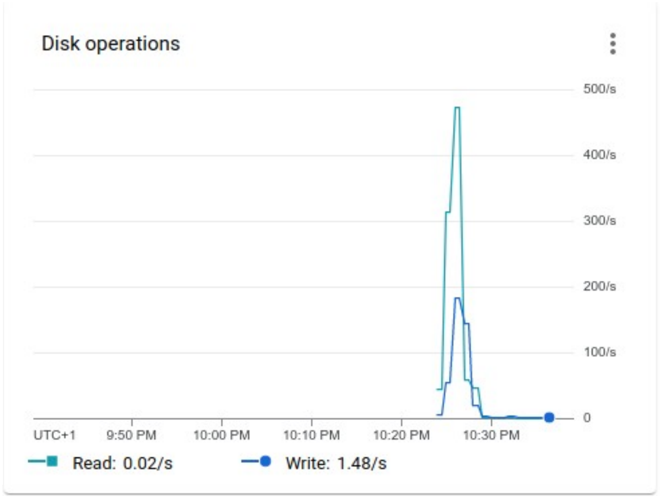






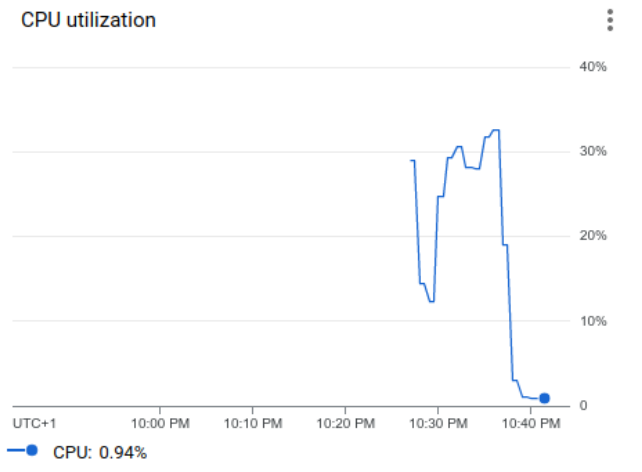


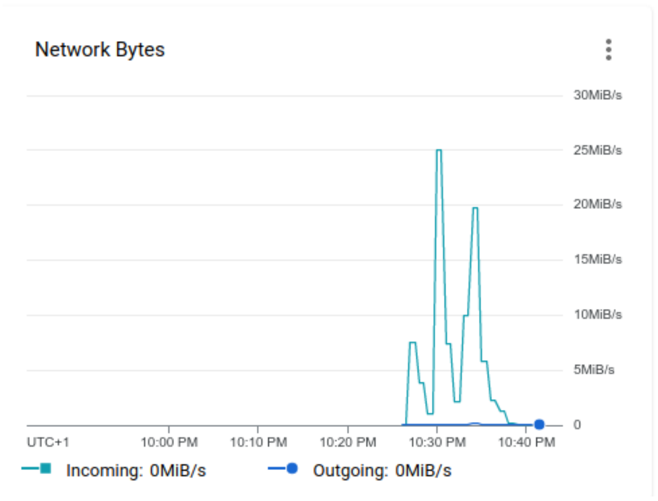


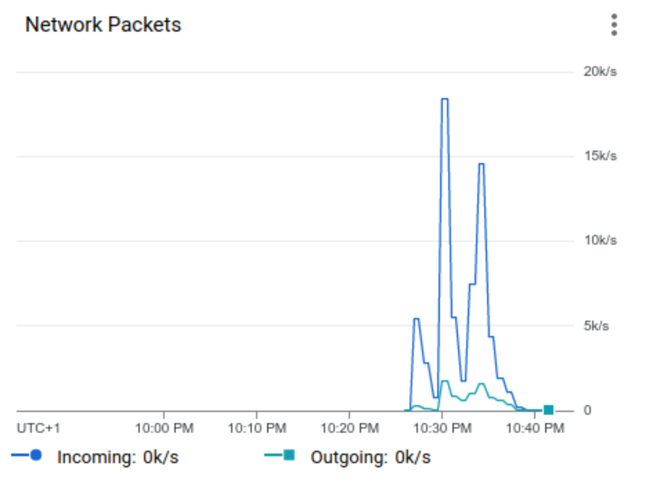


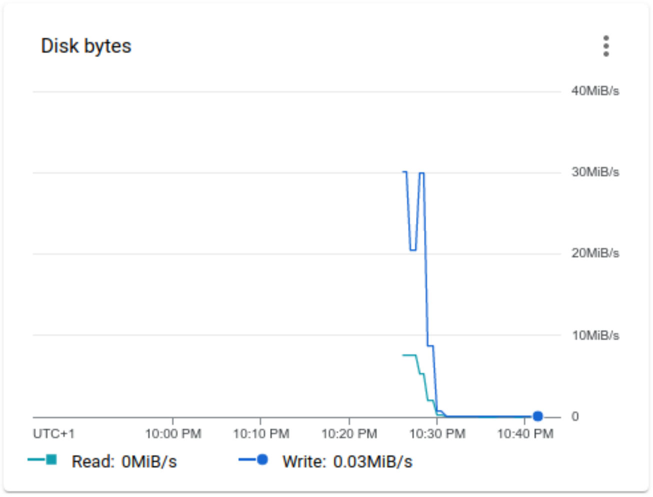
In term of CPUs utilization, 4-machines experiment used double the amount of 1-machine experiment. This is totally logical since we have more machine running.
In term of Network Bytes and Network Packets (network bandwidth allocation in the cloud), the value for 4-machine experiment is slightly more than triple the value of 1-machine experiment.
In term of Disk Operation, the amount for 4-machine experiment is also slight more than triple the amount of 1-machine experiment. However, the value for Disk Bytes the 4-machine experiment is almost tenth-fold the value of 1-machine experiment. It means that by adding 3 more machines we can speed up disk I/O almost tenth-fold. This point should be keep in mind when doing cloud configuration. From the above two experiments, we can see that standard application on 1-machine could not take full advantage of the cloud, while adding more machine could make the usage of cloud resources more efficient.

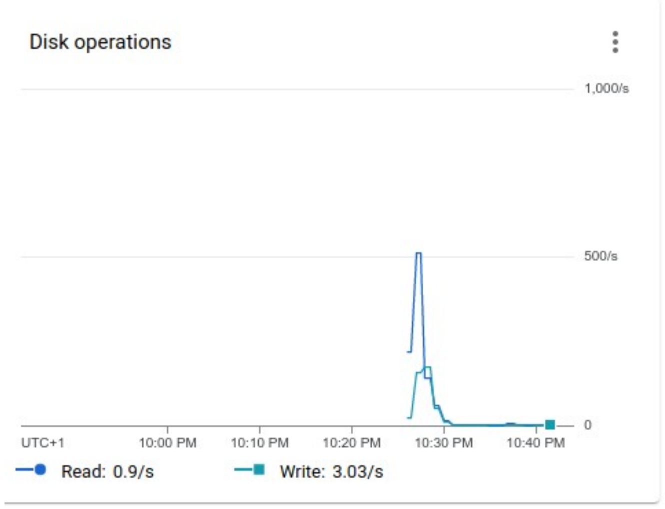
2b) Testing The Code & Collecting Results



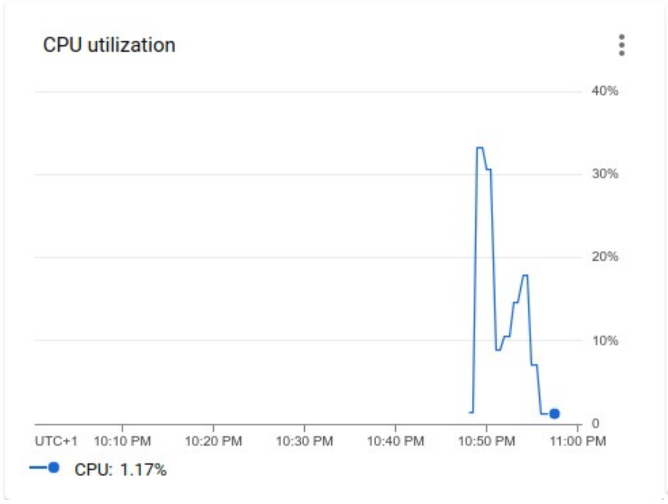


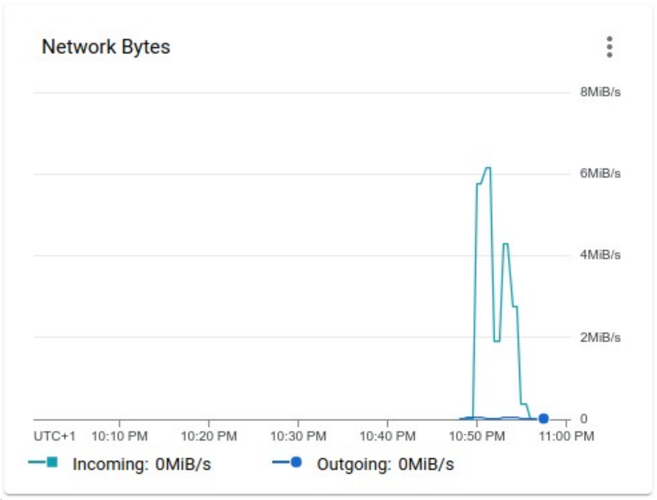


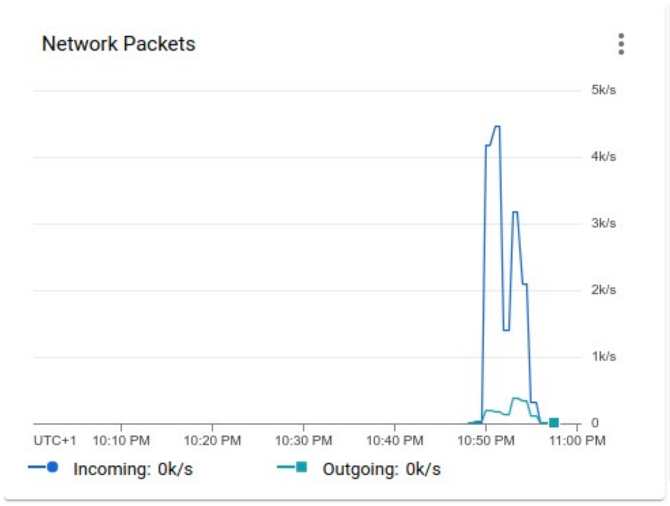


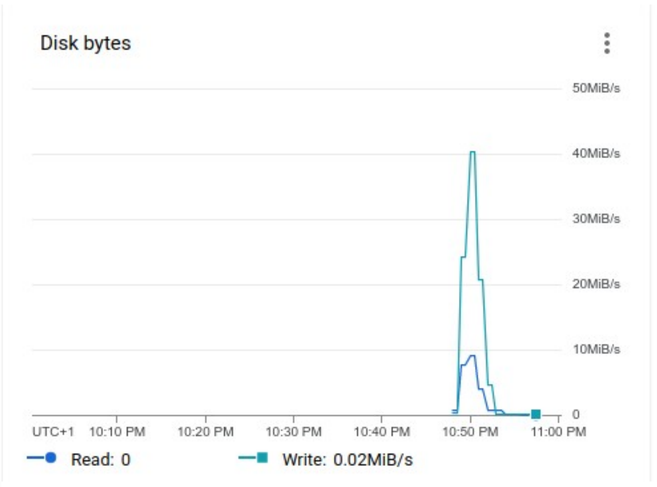


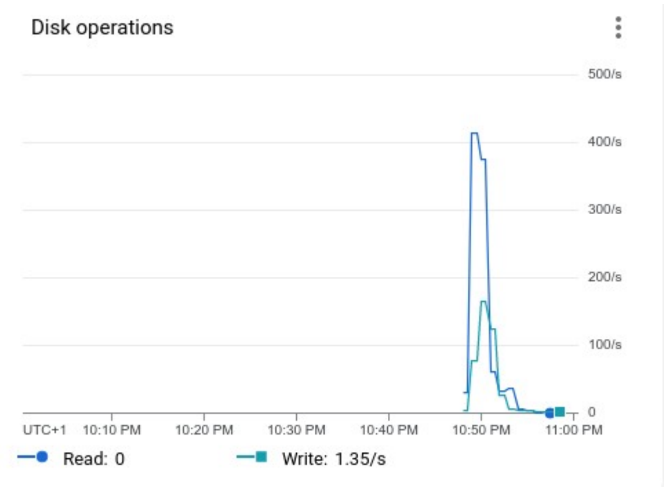
2c) Improve Efficiency









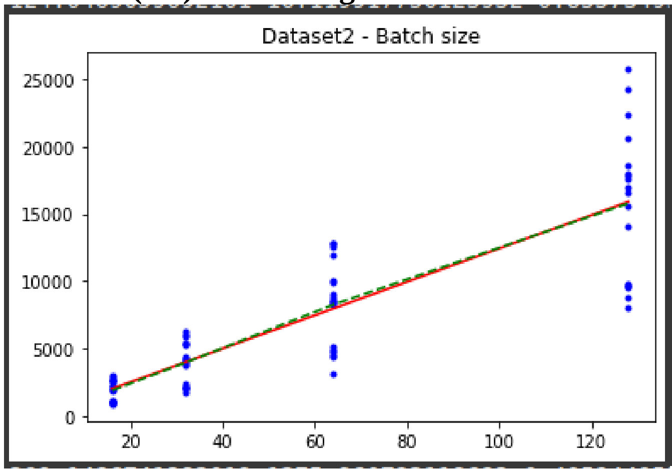


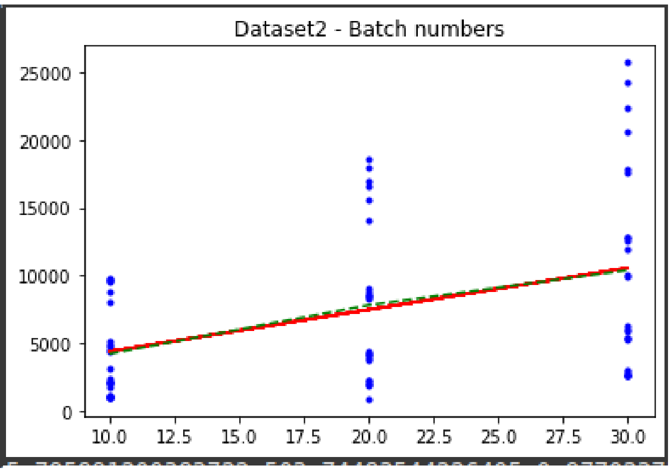
The reason why we run into an inefficiency in our code is because we are reading multiple times from an RDD to read the values for different parameters and their average. Therefore, to improve the efficiency, we apply caching to the existing results because it helps to speed up the the process which access the RDD multiples times. The change in the code has been marked in the notebook. Basically, we applied caching to the RDDs of two data sets for the map functions, which are the RDD with different parameters combinations and corresponding throughput for both image files and TF record files. By applying caching the RDD, each individual worker will store its partitioned data into memory or disk and reuse them and thus help to speed up the operations. The above images are the screenshots of the CPU and network load over time before and after applying caching. We compare the changes in 5 key metrics: CPU utilization, Network Byte, Network Packets, Disk Bytes, and Disk Operation. We can see that there is quite a different in most of the metrics after applying caching. It shows that caching helps to improve the inefficiency and it also helps to reduce the operation time from 25 minutes to 16 minutes.

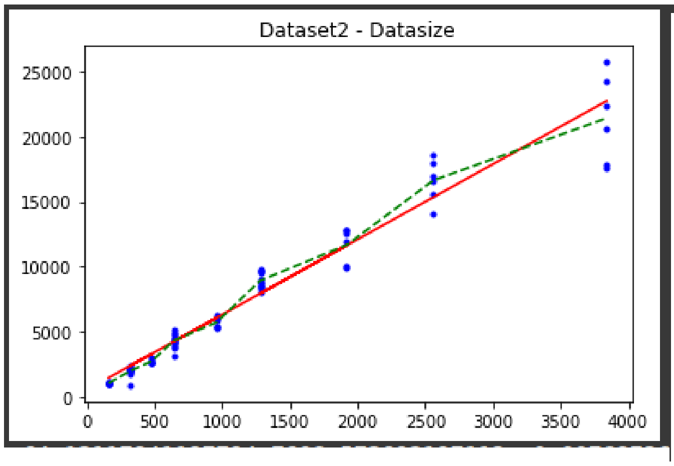
2d) Retrieve, Analyses & Discuss The Output

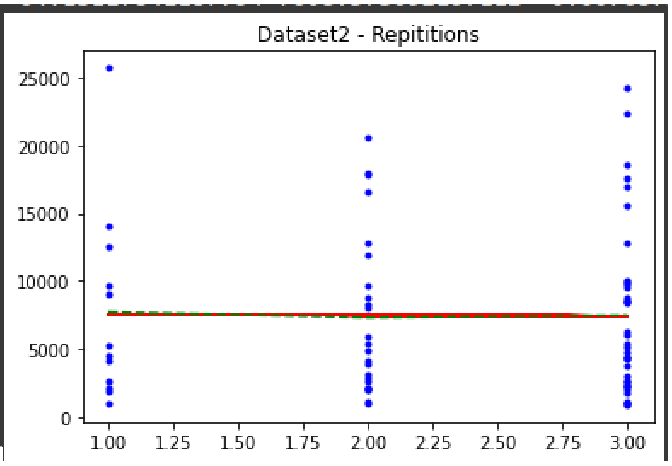
Data set	Parameter	Slope	Intercept	P-value
Dataset2 (TF)	Batch Size	124.04	16.12	1.03e-21
Dataset2 (TF)	Batch Number	309.15	1275.96	0.00040
Dataset2 (TF)	Data Set Size	5.8	503.74	9.83e-49
Dataset2 (TF)	Repetition	-64.13	7608.57	0.95
DatasetDecoded (IMG)	Batch Size	8.85	-3.45	4.84e-86
DatasetDecoded (IMG)	Batch Number	0.96	508.52	0.86
DatasetDecoded (IMG)	Data Set Size	0.30	167.72	3.91e-19
DatasetDecoded (IMG)	Repetition	6.44	512.74	0.916

Dataset2 (TF) Linear Regression

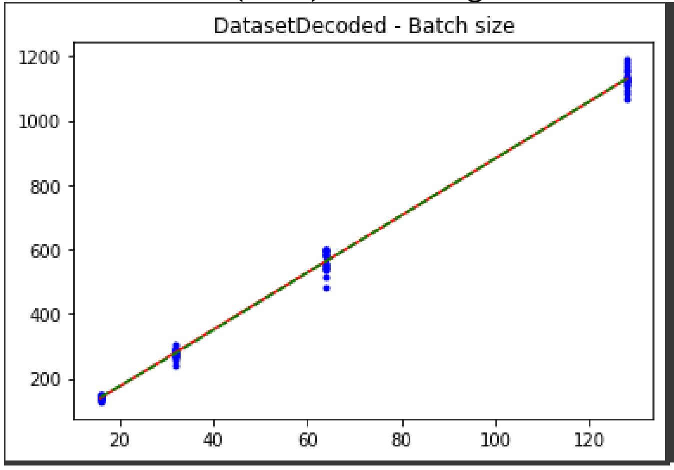


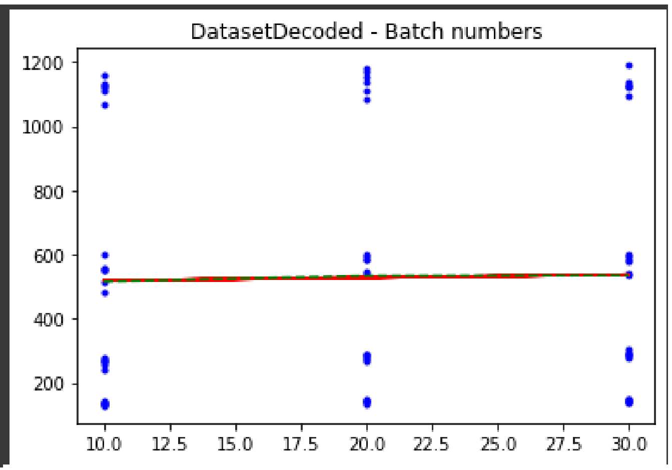


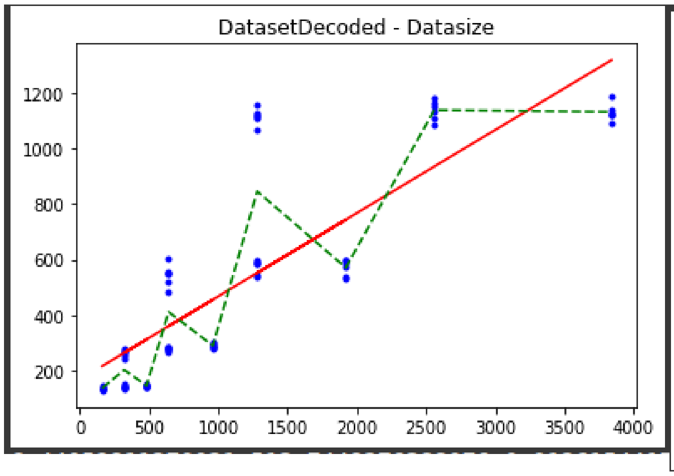


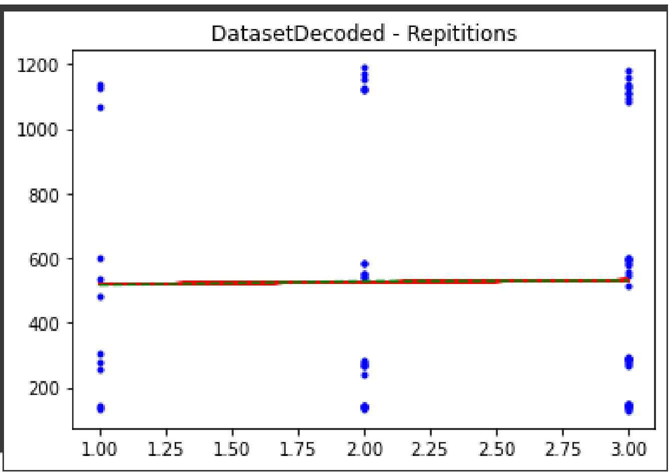


DatasetDecoded (IMG) Linear Regression





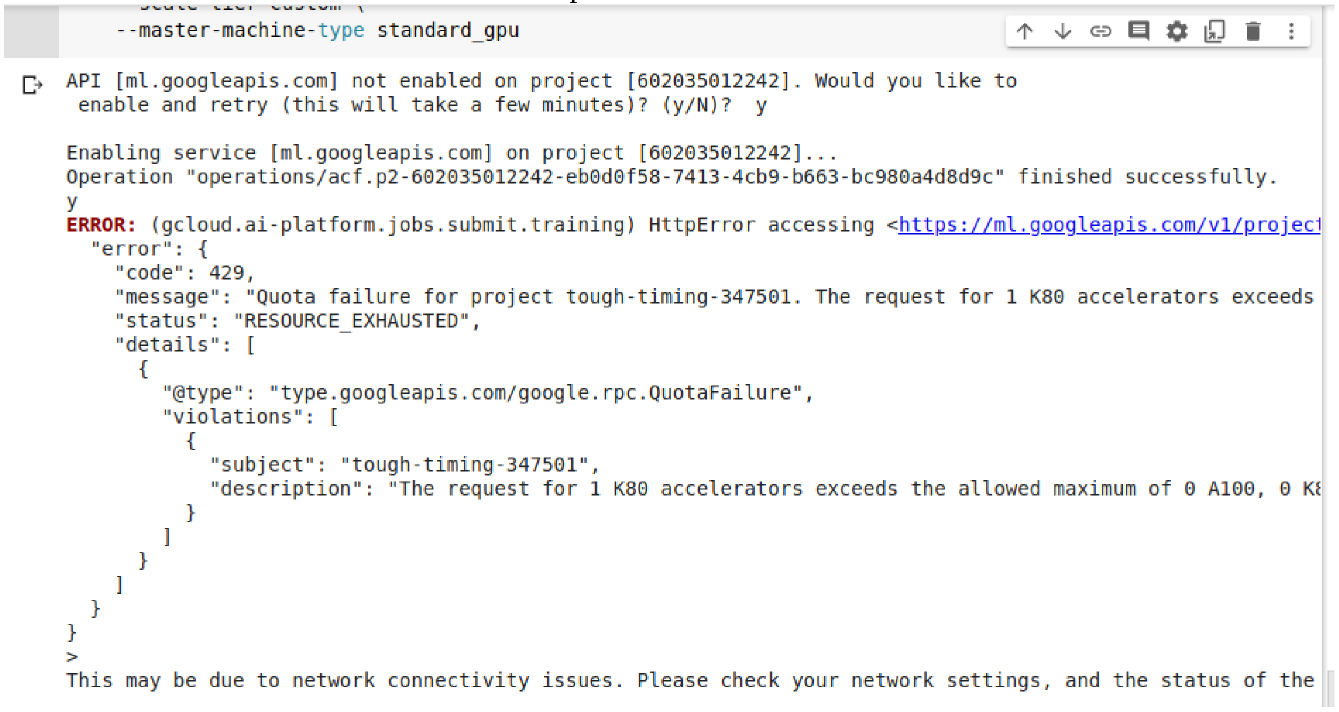




When inspecting the graphs, we can easily see that there is a positive linear relationships between all parameters with throughput, especially in the case of TF record files. Batch size parameter seems to have the better correlation with throughput in general. If we look at the P-values, repetition parameters has the most statistical significant on throughput.

The implication for large-scale machine learning are: large-scale machine learning models or applications need to have large disk size to ensure the optimal throughput and Cloud data may be stored in distant physical location. From the latency number provided to the coursework, a latency for a round trip within the same data center takes 500,000 nanoseconds. This number drops significantly to 0.5 nanosecond and 7 nanoseconds if we use L1 cache and L2 cache respectively. Therefore, caching is strongly recommended to implement to improve the running time.
The reason why cloud providers tie throughput to capacity of disk resource is because batch size has the most impact on throughput as we have mentioned several times. The positive correlation between throughput, batch size and disk size is the reason for this.

3b) Run The Training on AI-Platform.
We initially had the problem with submitting the job to google cloud platform due to the problem of accelerator. Please see the screenshot of the problem below.



```
  --master-machine-type standard_gpu

API [ml.googleapis.com] not enabled on project [602035012242]. Would you like to
 enable and retry (this will take a few minutes)? (y/N)?  y

Enabling service [ml.googleapis.com] on project [602035012242]...
Operation "operations/acf.p2-602035012242-eb0d0f58-7413-4cb9-b663-bc980a4d8d9c" finished successfully.
y
ERROR: (gcloud.ai-platform.jobs.submit.training) HttpError accessing <https://ml.googleapis.com/v1/project
  "error": {
    "code": 429,
    "message": "Quota failure for project tough-timing-347501. The request for 1 K80 accelerators exceeds
    "status": "RESOURCE_EXHAUSTED",
    "details": [
      {
        "@type": "type.googleapis.com/google.rpc.QuotaFailure",
        "violations": [
          {
            "subject": "tough-timing-347501",
            "description": "The request for 1 K80 accelerators exceeds the allowed maximum of 0 A100, 0 K8
          }
        ]
      }
    ]
  }
}
>
This may be due to network connectivity issues. Please check your network settings, and the status of the
```

With the help of other classmate we managed to submit the job on google cloud platform and continue with this task and also other tasks. However, this means that we can not experiment with different type of machine and cloud configuration such as complex_model_m_gpu (4xK80), complex_model_l_gpu (8xK80), and even standard gpu (1xK80) but this does not stop us from providing the code in the notebook for evidences and marking references. The error messages we got when running the scripts are the same as above picture. Since it is the same, we will not paste them here again, just to make the report look more tidy. Please see the notebook. The training time is 371 seconds.

3c) Distributed Learning
Due to the aforementioned issue with google cloud account of accelerator, we would not be able to perform the distributed learning with different machine configurations and strategies. We can only perform the distributed learning for different strategies with one type of machine. The table below is the detailed results:

Strategy	Machine Configuration	Batch Size	Training Accuracy	Validation Accuracy	Train Loss	Val Loss	Time
One Device	--master-accelerator	64	0.367	0.4656	1.46	1.412	360s

	count=1,type=nvidia-tesla-k80						
Multi-worker Mirrored	--master-accelerator count=1,type=nvidia-tesla-k80	64	0.2520	0.2437	1.591	1.600	385s
Multi-worker Mirrored	--master-accelerator count=1,type=nvidia-tesla-k80	128	0.2476	0.2406	1.6042	1.604	375s
Multi-worker Mirrored	--master-accelerator count=1,type=nvidia-tesla-k80	256	0.3284	0.3222	1.5415	1.548	367s
Mirrored	--master-accelerator count=1,type=nvidia-tesla-k80	64	0.375	0.425	1.468	1.459	354s

From the above results, we can clearly see that One Device Strategy is more suitable to this CNN machine learning model based on the values of different key metrics such as time, training accuracy, validation accuracy, training loss, validation loss, and finally batch size. Since this is a small data set, it is possible that Multi-worker strategy will give better result than One Device strategy in a larger and more complex data set. The different experiments with different batch size in Multi-worker strategy shows that increasing the batch size helps to give better performance in almost all the metrics. This is one point that we will use in theoretical task. We also need to keep in mind that we can only run the distributed strategy on one simple type of configuration. The result may be different if our google cloud account setting allows us to run more power configuration cloud setting, but we expect the result should not be much of a different.

4a) Contextualize

Cherrypick is a prediction system that allows people to find an optimal or near-optimal cloud configuration for recurring and non=recurring big data tasks or application. According to the paper, doing cloud configuration for big data tasks or application is an troublesome jobs because there are many different combinations It is not possible or convenient for the person who do the tasks to carry out all the combinations because of time and resources constrains. Finding optimal or near-optimal cloud configuration is significantly important for big data tasks or application because it helps to save lots of time and resources in order to achieve the best performance models or applications. This is what we have been doing repeatedly in task 1, 2 and a bit of task 3. We run and experimented the task with different cloud configuration cluster setting using google cloud AI platform by using different combinations of machines and cluster setting in order to find an optimal configuration to improve performance and reduce running time.

An Oracle is a model for predicting the effectiveness of different parallelization methods for training neural networks. According to The Oracle paper, with the current trend in Deep Learning of large data samples and large models, the common approaches are large scale parallel training in high performance computing: data parallelism and model parallelism. Data parallelism is the go-to method since it is easy and straight forward to implement with OK-performance but it is not feasible for all cases due to its limitation such as memory capacity, communication overhead and degradation in convergence accuracy because data parallelism involves replicating the model and giving a different batch size to each different batch to each model as we increase the number of CPUs we increase the mini-batch size and at the very high batch size there is an issue with performance degradation in term of accuracy. Model parallelism also has its own problems. There is no existing method to determine the right parallel strategy for a given model, data-set and too much time spent trying out different strategies. Even if you found a strategy, it can becomes obsolete if model architecture or system changes. And that is the purpose of this Oracle paper. To find the correcting parallelism strategy. The purpose behind the two papers is similar to each other. It is to find the correcting or optimal ways or methods to any big data tasks or application by trying with different settings and combinations. And this is what we have been doing from task 1 to task 3 or even task 0. In task 0, we tried to re-processed image files by resize, decode or compressed the image and write it to TF record files to improve the speed of reading the data. In task 1 and 2, we tried to write the TF files to the cloud and using data parallel to increase the speed. In task 3, this is CNN machine learning model. This task 3 is much more relevant to the Oracle paper. In here we perform different strategy from One device strategy to mirrored strategy and multi worker strategy to find the with different cloud settings to find the optimal configuration to improve the overall performance of the model such as accuracy, time, CPU utilization, RAM etc.

4b) Strategies

We have been consistently involved with performing and testing data parallelism and machine learning model parallelism using different cloud configurations and parameter such as batch-sizes and analyze the different between them to find the optimal combinations of them. The most important metrics when running big data analytic jobs are performance (accuracy) and overhead (cost). Cherry pick is a system that allows you to pick a near optimal cloud configuration for our big data recurring applications such as batch , online, and streaming. Many of the big data jobs are recurring, around 40% reported by Microsoft. They are executed periodically on similar queries and data. One example is to analyzing server logs processing adds data or even scientific computation. There are lots of options to run these jobs with using different cloud providers. Within each provider, there are tens of machines and cluster size that we can pick. The combination of machine type and machine count gives hundreds of possible configuration that we can pick from. When choosing a cloud configuration for experiments, once has to be careful because by choosing a good cloud configuration resulted in up to three times lower running time for the same cost and up to 12 times lower cost for the same performance target. It means that choosing a good cloud configuration can save us a significant amount of time and it is even more important for recurring and long-term jobs. The key to identify the near optimal cloud configuration is to be able to determine the total workload and which application processing we want to use. This applied to both recurring and non-recurring tasks in which the later should be more difficult to do. We also want our cloud configuration should adapt to different application processing and the different between its actual performance versus the expected performance should also be constantly monitored so that we can make changes when needed. From the results of above tasks, we can see that by deploying more virtual machines reduces the running time. However, this is not always true if we keep adding more virtual machines to what the application needs than we will have a problem of diminishing returns due to the sequential portion of the application. Therefore, when trying to find the optimal configuration, which is highest performance for a lowest costs, we should always consider this point.
Batching processing handles large parts of data set or entire data set at once, meaning all data is available and does not change for example history of sale data. And because of this, batch processing needs to have the largest dish size to be able to perform this task. Online processing or real-time processing is where the data is still changing or growing. Process whenever a new data point comes in. Streaming processing or micro-batch processing is between the two. Group new incoming data together at regular intervals or data sizes and process those together. Online and stream processing need to have more RAM.
The choice of batch, online or streaming strategy is determined by our need. When we need to process a large amount of data set and and does not require instant results or response, we can go with batch processing. We need to optimize it to handle large volumes of data and complex models. Batch processing is asynchronous strategy. Batch processing requires huge computational resources such as RAM, disk size, cpu etc. A positive correlation between the batch size and computational power is needed to keep in mind. For that reason, batch processing is suitable for small data sets. If we are looking for a strategy which provides response in a short period of time while data is being transacted in nature then online and streaming strategy are more suitable. Because the need to instant response, those strategy need to be optimized to minimize the latency. Streaming application needs sequential processing and incrementally on sliding time window on a continuous basic. Because of minimizing

the latency in online and streaming processing, a maximum value of batch size should be chosen to reduce the training time and maximize the response for the outputs.

TOTAL WORDS COUNT = 2500 WORDS