Big Data Final Coursework Report

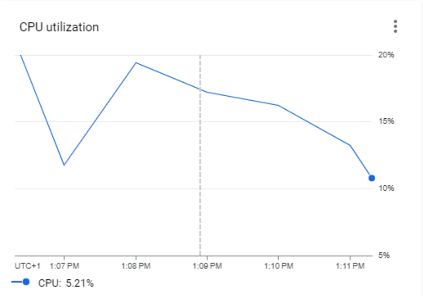
Syed Mahbubul Huq (ID 220033725)
MSc Data Science

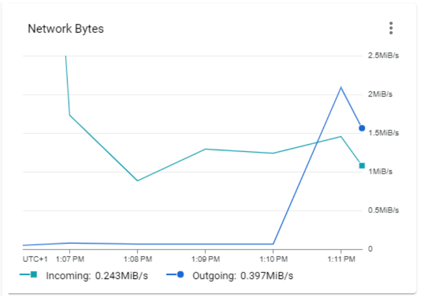
Colab link:
https://colab.research.google.com/drive/1mbPUH8lbVCBreIK9sD6urlXUIpRAvnXp?usp=sharing
*****Please note, I have talked with our lecturer, Dr. Alex, with my issue. I used to google colab linked with academic g-suite (Gmail) account of my former institution, BRAC University. They gave me a notice suddenly on 6th May that all files and storage associated with BRAC University Gmail account would be cleared and deleted by 21st May. I also did not have enough time to do the coding part in google cloud again on a new account on 6th May. In that case there is a high risk of my colab file to be deleted and I mentioned this to Dr. Alex. He told me to mention my case in the report and ensure that my .ipynb file has saved output, I have ensured that *****

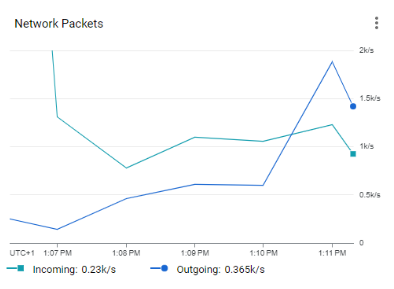
Task 1d

(i) After implementing our parallelize with default settings and after passing second argument, partition we can see significant difference in resource usage and performance and elapsed time for the job to be completed in cloud. For experiment, in the second argument, I passed 2 different partition value. Hence, we can see difference with default settings, 4 partition and 8 partition respectively.

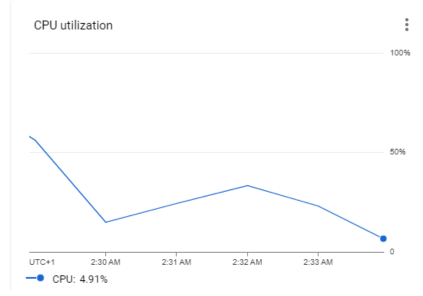
Default Settings (Elapsed time for the job: 4 min 37 sec):

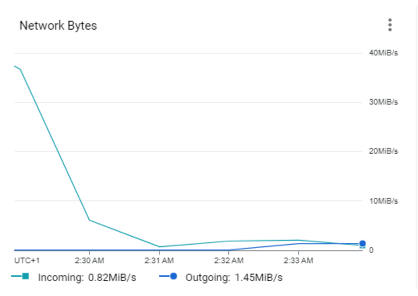


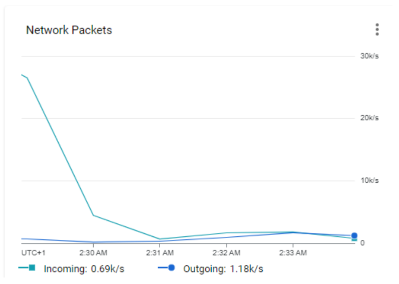




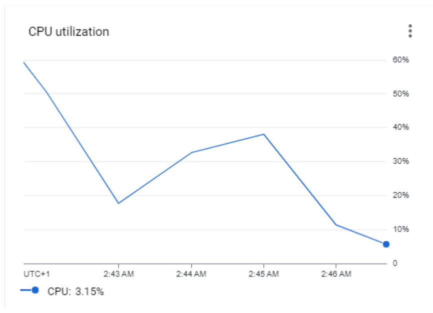
4 Partition (Elapsed time for the job: 2 min 47 sec):

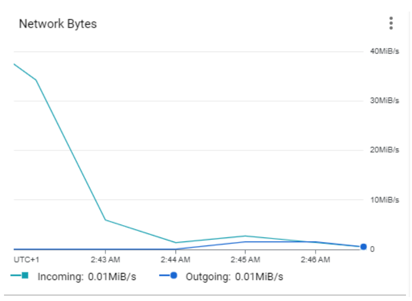


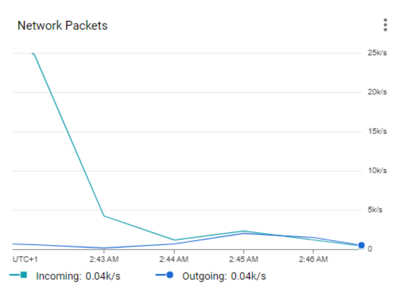




8 Partition (Elapsed time for the job: 2 min 38 sec):



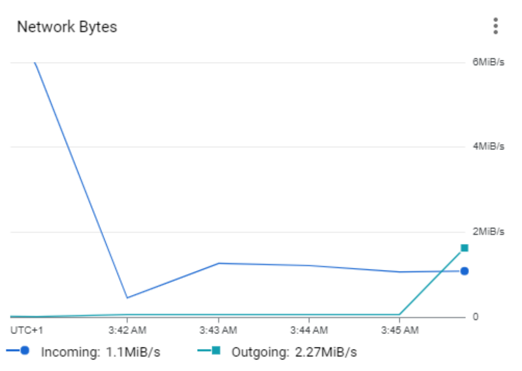


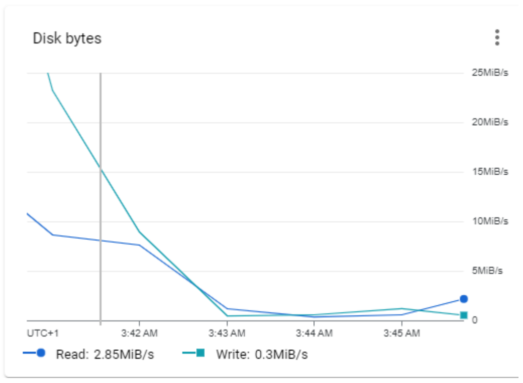


From this we can observe that with the increase of partition number, maximum CPU utilization gradually increases. With the passage of partition argument, another interesting change can be observed in Network Bytes and Network Packets value, the graph looks similar for 4 partition and 8 partition compared to default settings. Another important difference in the above 3 different scenarios can be observed in Elapsed time for job execution, although the CPU usage increase, but execution time gradually falls down. A drastic change in elapsed time can be observed from default settings to the introduction of partition argument.

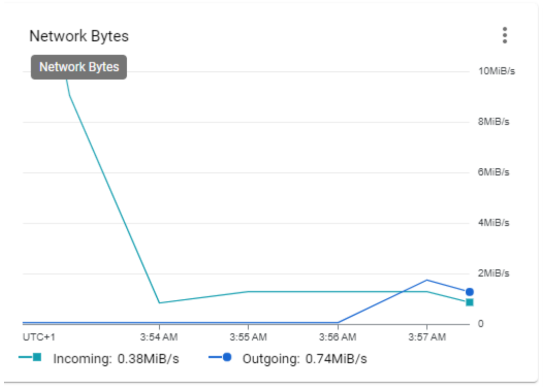
(ii) Here we experiment with different cluster configuration on our default setting parallelize code. We experiment the same code on 3 different clusters using 8 VMs, 4 machine (2 vCPUs, memory disk) and one machine with 8-fold resources. Screenshot of result in terms of disk I/O and network bandwidth allocation is given below:

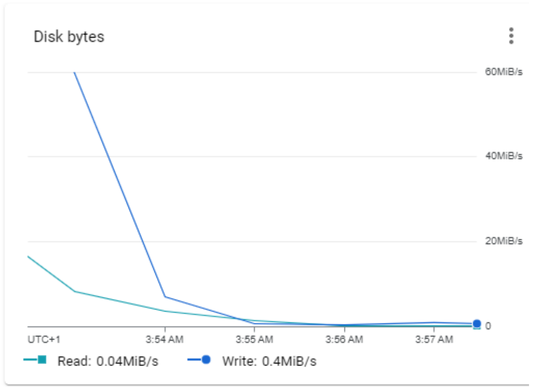
1 Master-7 worker:



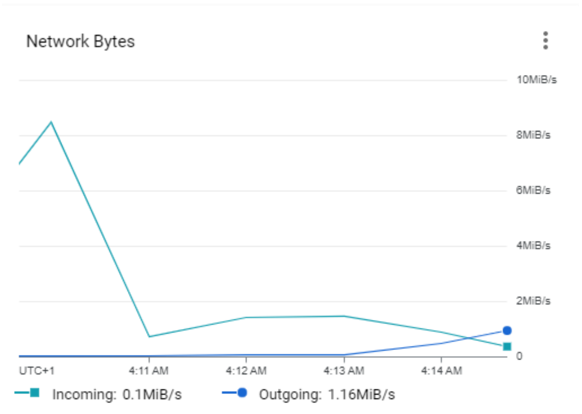


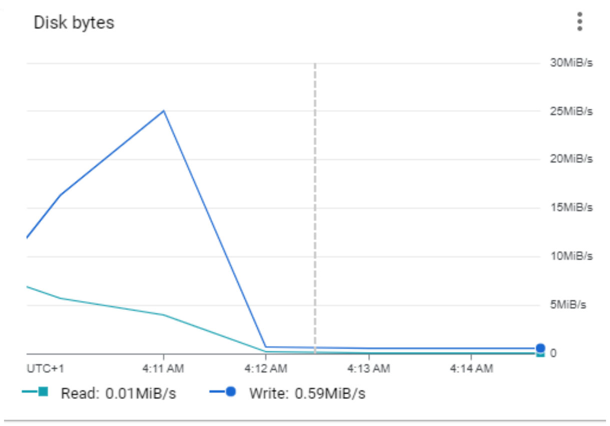
4 machine (2 vCPUs, memory disk):





1 Machine with 8-fold resources:





Here in the above figure, we can observe about Network Bytes and Disk Bytes usage for different distributed machines. It can be observed that Network Bytes of 4 machine had the highest peak compared to the other two. Similarly, for Disk Bytes the system with the more distribution of machines had the highest peak. Among the three sceneries, the last system, 1 machine working with 8-fold resources had the least peak. Overall, the highest peak for Network and Disk Bytes it reached was the lowest compared to other system. This shows about one important characteristic. When proper distribution of system is done in case of parallelizing, highest peak can be achieved in network and disk bytes. This is because a single machine has throughput limit and are limited in sending more data, but if it distributes the task among multiple machines, more data transfer can be done.
Therefore it can be concluded that based on a particular task it is important to configure clusters according to a certain task and its requirements.

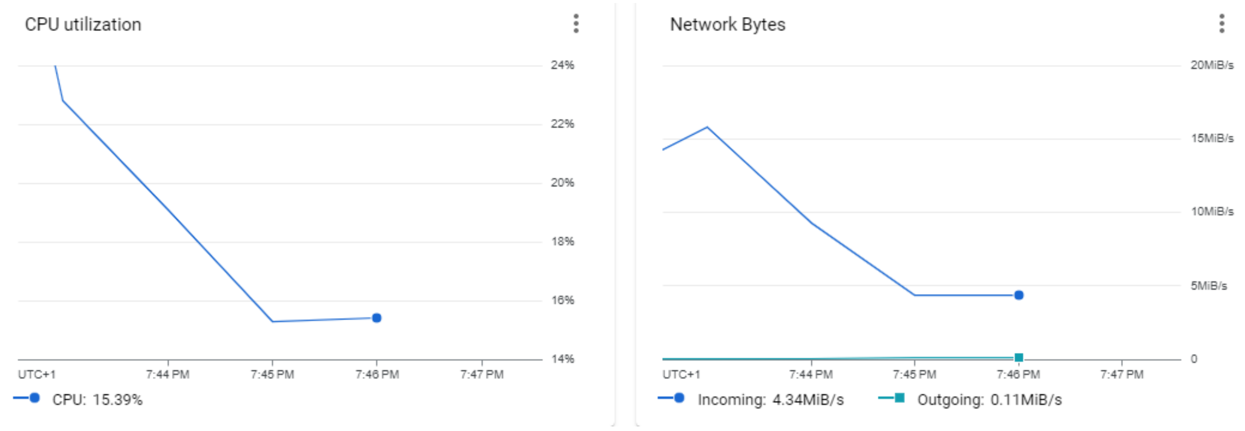
(iii) In case of using Spark in cloud, parallelizing of task is possible which is one of the best features of using spark in cloud compared to a single PC which we are used to in using in Labs. Distribution of task among different systems can be done in cloud because of its distribution architect.
In cloud, Spark have lower memory consumption and can process more data compared to a single PC which we use in Labs. It is also possible to get higher performance and network transfer when clusters are created with multiple machines. As in cloud, Spark uses parallelizing technique, it can make use of multiple devices and multiple devices when reached to their maximum throughput, when combined together would result in better and efficient performance. In cloud architecture, data are stored in Buckets which can easily be accessed by multiple systems which makes using Spark in cloud unparallel in performance compared to using with a single PC. Besides, it is also evident from about the job completion time that a system takes when proper partition is done while parallelizing using Spark. This is not possible while using a single device.
Overall, an optimal and efficient solution can be achieved using cloud architecture compared to doing a task using a single PC. Memory distribution, task distribution, parallelizing are the benefits of using Spark in cloud compared to normal usage like the use that we did in our Lab PCs.

Task 2c

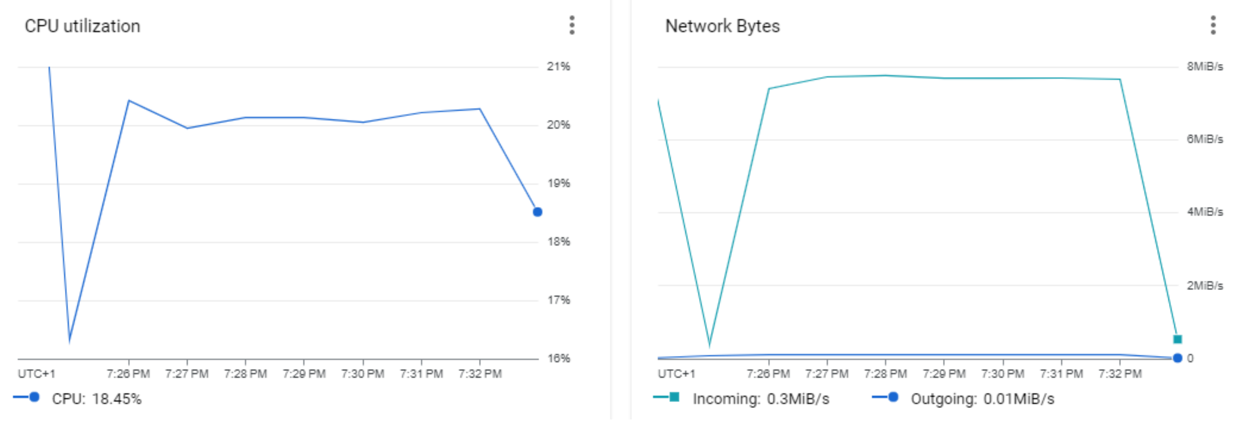
When a spark action like, count(), show(), take(), collect() etc takes place on the same RDD which was executed before, it is necessary to call cache() after execution of the RDD that would store it's value in the memory of cluster's worker, hence when the spark action like collect() is executed, the system can refer to the cache memory and this makes the overall execution much faster.

Hence, we kept the cache() part in our code before execution of a certain task and in our case it was collect().
We can observe evidence in our system performance before and after applying cache(). A comparative analysis in term of CPU utilization and Network Bytes transfer in given below:

Task with cache() (Job Execution time: 1 min 49 sec):



Task without cache() (Job Execution: 8 min 52 sec):

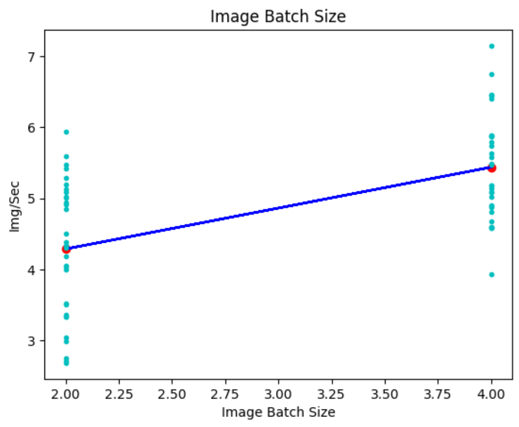


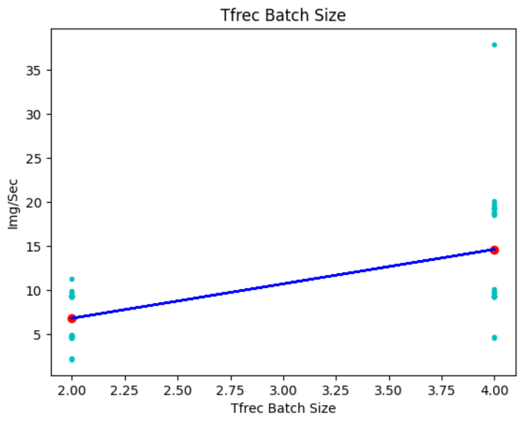
We can observe in difference of CPU usage, Network Bytes transfer and even drastic change in Job Execution time when cache() is applied. We can observe, in case of CPU utilization, after cache() is applied, proper CPU utilization is made. Even in case of Network Bytes transfer for the task with cache() applied, the highest peak the Network Bytes reached was comparatively higher when compared task without cache(). To conclude, we can say that by applying cache() in proper part of the code, we can make the best use of resource utilization and it can also be time efficient for us.

Task 2d

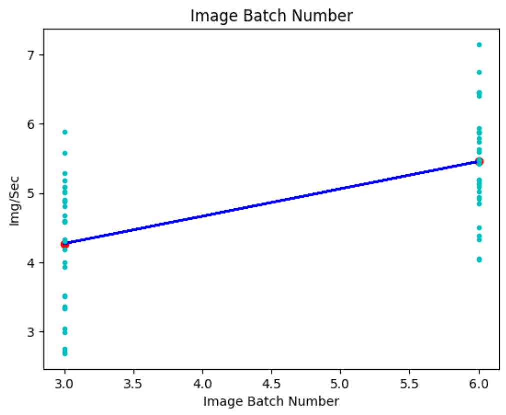
Linear Regression with output values, the averages per parameter value and the regression lines:

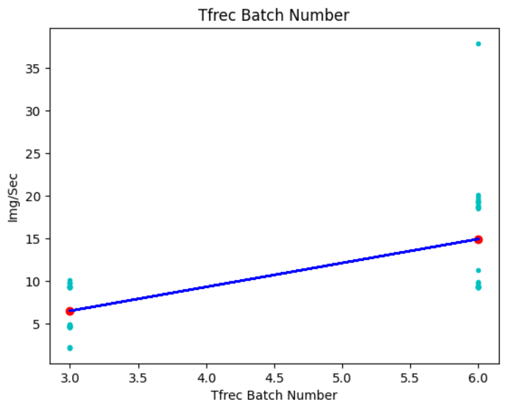
Batch Size:



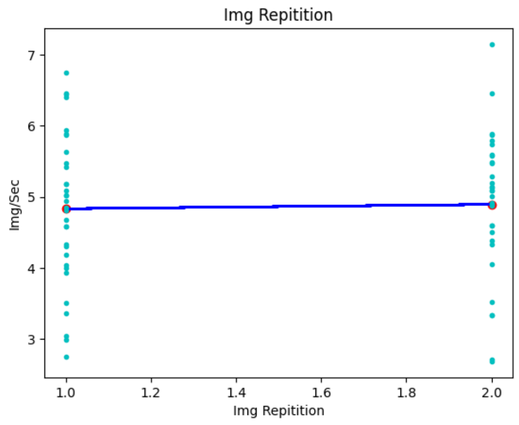


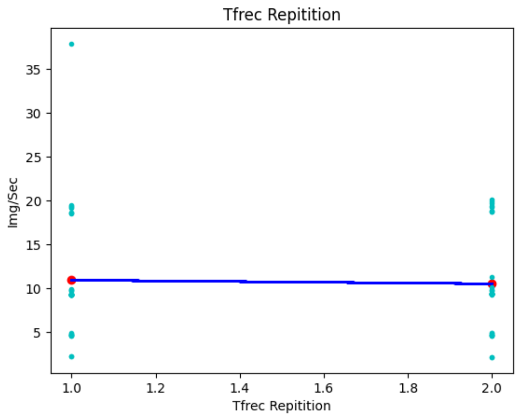
Batch Number:



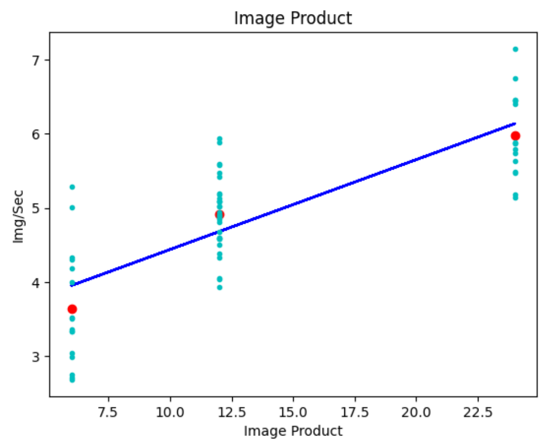


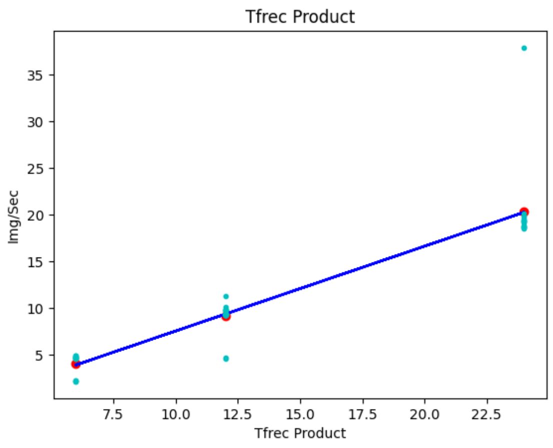
Repetition:





Product:





Above graphs shows us the result that we got while conducting our experiment on Google cloud service. The table of the above results are presented below:

	Img Batch Size	Tfrec Batch Size	Img Batch No.	Tfrec Batch No.	Img Rep	Tfrec Rep	Img Pro	Tfrec Pro
Slope	0.57	3.9	0.39	3.8	0.6	-0.41	0.12	0.9
Intercept	3.13	-0.96	3.08	-1.89	4.72	11.36	3.22	-1.49
p-value	1.00E-06	1.09E-07	3.90E-07	4.50E-09	0.81	0.8	2.3 e-14	3.8 e-27

From the above table we can see about the linear regression. In most of the result we had a good p-value which indicates how statistically important the outcome is.

The overall experiment indicates about using partition, parallelization, storage management etc. of cloud in a good way. In many cases, we might face difficulty as location wise CPU utilization power can be limited. Also, a good amount credit would be required for model requirement more than: 100GB SSD persistent disk, 2000GB standard persistent disk, 8 vCPUs, GPUs resources. Latency should also be considered while designing remote cloud architecture. Because of latency a random SSD read can take over a day to execute.
The observed behavior is different because of the dependency and diversity of resources; both are similar in term of being human dependent system. To limit disk usage and maintain disk overall health providers tie throughput to capacity of disk resource.
In speed test in cloud we need to consider some good practices in order to avoid errors and inefficiency. For example, reduce request rates while facing error or naming convention that distributes load evenly across key ranges can be some of good practices.
Linear modeling can be a good way to understand our findings. For example, in the above linear regression modeling, we can observe, increasing Batch Number, Batch Size and Product in case of both Image and Tfrec can have good impact on our overall performance in cloud. But increasing Repetition can be a bad option in our case and should be avoided.

Task 3a)
In the previous tasks that we did, we were given the task of experimenting with different cluster configurations, parameter values and analyze and observe the usage and try to optimize the speed and cost of the system. With regards to the given task that we completed, the above mentioned paper mentions a system called CherryPick which can be used by us to improve our cloud computing process.

CherryPick focuses in reducing the cost and remove unwanted resources from our list. In the paper the authors mentioned and compared about the memory usage architecture of TPC-DS and Regression. They mentioned one can give better performance, but the other with similar configuration may not be beneficial. We can relate this with our task, as in most of the task we configured cluster without considering the task and architecture of the task in hand. For example, they also mentioned for Regression, the cluster configuration best for TPC-DS is 2.5 times costlier for regression. To solve this problem, their system CherryPick can be used which give almost optimal configuration. It is built on Bayesian Optimization which also has a defined stop search which can be used by optimally search for best configuration.
As our task involved in experimenting with different configurations, we can use the concept of the paper and implement in for cost efficiency. One condition while applying this can be considering the limitation of BO, who's computation complexity is O(N4). Therefore if the dataset is very large this can be a very big problem while using CherryPick.

Task 3b)

Batch and stream are both performed by Spark and are different concepts. In case of batch data is available to us and it is generally converted to a larger compression before processing the task. This is a useful task which we used in our previous task. Converting batch to large compression makes it efficient and faster for computation. For batch data we know the size of it beforehand. Different Spark techniques like Map, Reduce and Lambda are used a lot for Batch data. For this kind of data the bottleneck is the disk performance.
On the other hand for stream data, rather than disk performance, CPU resource is one of the most important thing to consider. For stream data latency is a very big and important issue and while building remote architecture for this kind of data latency should always be taken into consideration. Considering the task, domain and architecture in hand we should chose the correct system otherwise total efficiency would not be achieved.

Words
1786

References

https://www.databricks.com/blog/2018/05/03/benchmarking-apache-spark-on-a-single-node-machine.html#:~:text=Spark%20can%20often%20be%20faster,set%20into%20memory%20before%20processing.

https://kb.databricks.com/scala/best-practice-cache-count-take#:~:text=cache()%20is%20an%20Apache,memory%20of%20your%20cluster's%20workers.