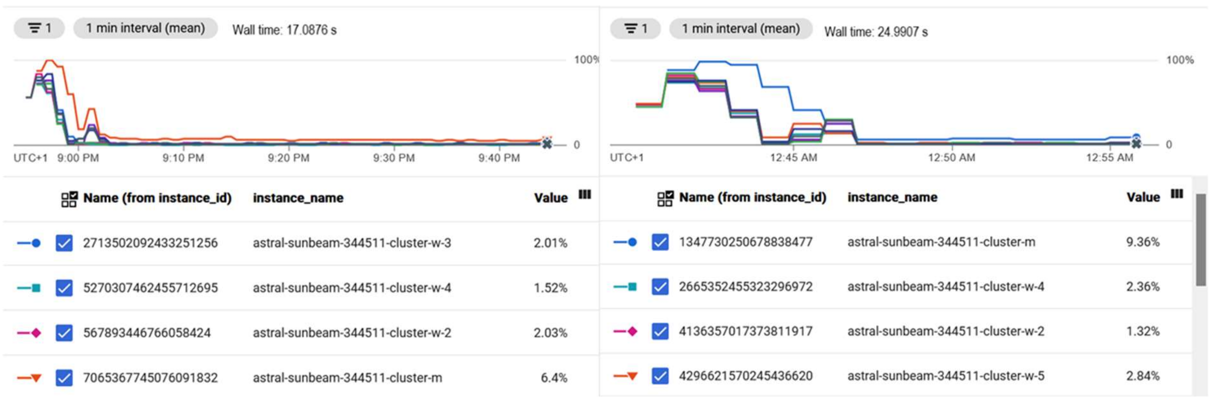
INM432 Big Data

Coursework Report

Student ID: 210049506

Name: RAJANI MOHAN JANIPALLI

Answer 1d(i):



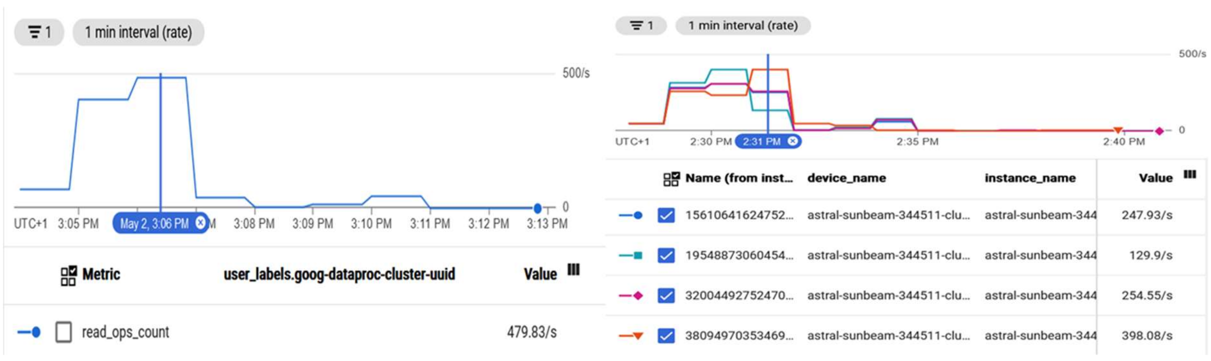
	Name (from instance_id)	instance_name	Value
● ✓	2713502092433251256	astral-sunbeam-344511-cluster-w-3	2.01%
■ ✓	5270307462455712695	astral-sunbeam-344511-cluster-w-4	1.52%
◆ ✓	567893446766058424	astral-sunbeam-344511-cluster-w-2	2.03%
▼ ✓	7065367745076091832	astral-sunbeam-344511-cluster-m	6.4%

	Name (from instance_id)	instance_name	Value
● ✓	1347730250678838477	astral-sunbeam-344511-cluster-m	9.36%
■ ✓	2665352455323296972	astral-sunbeam-344511-cluster-w-4	2.36%
◆ ✓	4136357017373811917	astral-sunbeam-344511-cluster-w-2	1.32%
▼ ✓	4296621570245436620	astral-sunbeam-344511-cluster-w-5	2.84%

Figure 1 CPU utilization for 1 Master, 7 Workers multi-node system with 2 partitions (left) and 16 partition (right).

In the first case, by default two partitions are created and in the second case 16 number of partitions were specifically created while parallelizing. From the figure, it can be interpreted that the CPU utilization for master is higher in the second case and for the workers there isn't a much perceptible difference between both cases. Using repartition, the data is shuffled and distributed among the 16 partitions in a balanced proportion, to avoid any kind of skewing of data distribution among the partitions. Computationally this is an expensive transformation operation. As a result, the CPU utilization in the second case was higher than the first case. So contrary to the general observation that partitioning improves computation due to higher parallelization, in this case the computation required for redistribution of data has nullified the advantage of partitioning. This was also evidenced by the additional 8 seconds of wall clock time in the second case.

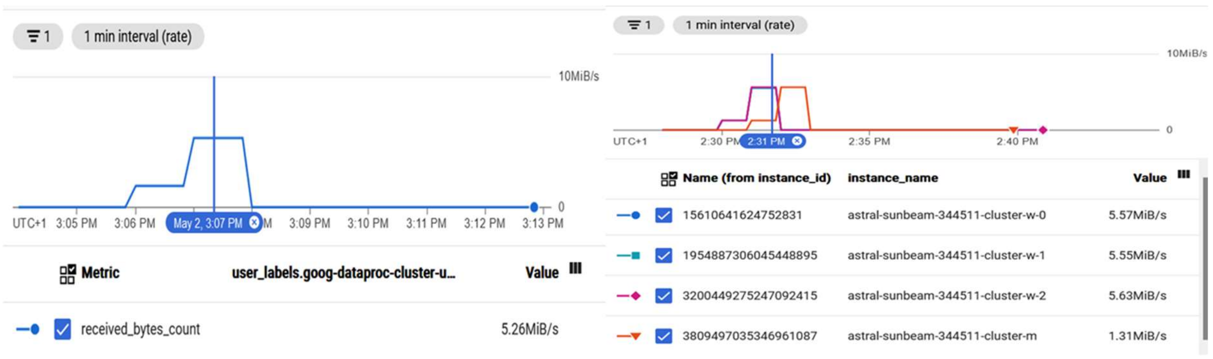
Answer 1d(ii):


1 min interval (rate)
500/s
0
UTC+1 3:05 PM May 2, 3:06 PM 3:08 PM 3:09 PM 3:10 PM 3:11 PM 3:12 PM 3:13 PM
1 min interval (rate)
500/s
0
UTC+1 2:30 PM 2:31 PM 2:35 PM 2:40 PM

	Metric	user_labels.goog-dataproc-cluster-uuid	Value
●	read_ops_count		479.83/s

	Name (from inst...	device_name	instance_name	Value
● ✓	15610641624752...	astral-sunbeam-344511-clu...	astral-sunbeam-344	247.93/s
■ ✓	19548873060454...	astral-sunbeam-344511-clu...	astral-sunbeam-344	129.9/s
◆ ✓	32004492752470...	astral-sunbeam-344511-clu...	astral-sunbeam-344	254.55/s
▼ ✓	38094970353469...	astral-sunbeam-344511-clu...	astral-sunbeam-344	398.08/s

Figure 2 Disk I/O read operations of 1Master-8vCPU (left) and 1Master-2vCPu, 3Workers-2vCPU (right).



Metric	user_labels.goog-dataproc-cluster-u...	Value
received_bytes_count		5.26MiB/s

	Name (from instance_id)	instance_name	Value
●	15610641624752831	astral-sunbeam-344511-cluster-w-0	5.57MiB/s
■	1954887306045448895	astral-sunbeam-344511-cluster-w-1	5.55MiB/s
◆	3200449275247092415	astral-sunbeam-344511-cluster-w-2	5.63MiB/s
▼	3809497035346961087	astral-sunbeam-344511-cluster-m	1.31MiB/s

Figure 3 Network bandwidth received of 1Master-8vCPU (left) and 1Master-2vCPu, 3Workers-2vCPU (right).

In Figure 2, the left half represents the I/O read operations for a single node cluster and the right half represents that for a multi-node cluster. The IOPS of master for single node is a bit higher than that of multi-node. But in the multi node cluster the IOPS of the workers sums up such that the overall IOPS is way higher than that of this single node cluster. Request for higher IOPS may lead to capping of performance which results in latency.

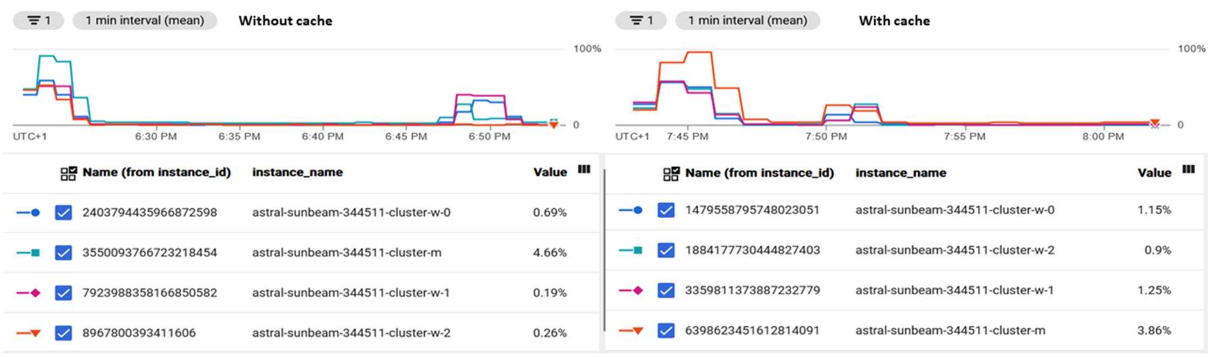
Figure 3 indicates network bandwidth allocation for this said two clusters in a similar pattern as figure 2. It can be seen that the received bytes for the master of single node cluster is less than that of each of the workers of the multi node cluster. This is because in the single node cluster, the data is stored in a local SSD which is physically available to the VM. Whereas in case of multi node cluster, the data is distributed among different standard persistent disks of the workers which are not physically available but are available over the network. So, data access for the workers happens only over the network transfer which causes latency.

Therefore, both the figures imply that the multi node cluster has higher latency and hence it is computationally less efficient.

<u>Answer 1d(iii):</u>

In terms of data storage location, only local SSDs are used in labs and most standard applications. Whereas in this case, standard persistent disks of respective nodes were used for storing data. Local SSDs are appropriate for temporary storage like cache etc., and hence they may not good from fault tolerance point of view. But they perform better than standard persistent disks due to lower latency.

<u>Answer 2c:</u>



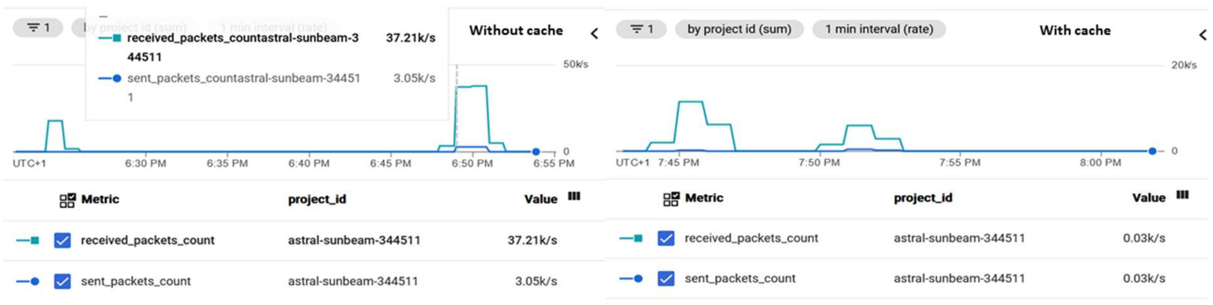
Without cache

	Name (from instance_id)	instance_name	Value
●	2403794435966872598	astral-sunbeam-344511-cluster-w-0	0.69%
■	3550093766723218454	astral-sunbeam-344511-cluster-m	4.66%
◆	7923988358166850582	astral-sunbeam-344511-cluster-w-1	0.19%
▼	8967800393411606	astral-sunbeam-344511-cluster-w-2	0.26%

With cache

	Name (from instance_id)	instance_name	Value
●	1479558795748023051	astral-sunbeam-344511-cluster-w-0	1.15%
■	1884177730444827403	astral-sunbeam-344511-cluster-w-2	0.9%
◆	3359811373887232779	astral-sunbeam-344511-cluster-w-1	1.25%
▼	6398623451612814091	astral-sunbeam-344511-cluster-m	3.86%

Figure 4 CPU utilization for 1Master-2vCPUs, 3Workers-2vCPUs



Metric	project_id	Value
received_packets_count	astral-sunbeam-344511	37.21k/s
sent_packets_count	astral-sunbeam-344511	3.05k/s

Metric	project_id	Value
received_packets_count	astral-sunbeam-344511	0.03k/s
sent_packets_count	astral-sunbeam-344511	0.03k/s

Figure 5 Network load for 1Master-2vCPUs, 3Workers-2vCPUs

Caching the RDDs of TFR and images immediately after their creation would help in reducing the computation cost for further process of mapping and reducing the RDDs. From figure 4, this can be seen, where without caching the second peak in CPU utilization sustains longer than with caching. During this second peak, master CPU utilization is less then workers in case of without cache and its not so in case of cache. Network load in figure 5 is in line with this as in case of without cache the network load is higher than that of cache case for the second peak. This is due to higher transfer of data packets to persistent disks of the nodes.

Answer 2d:

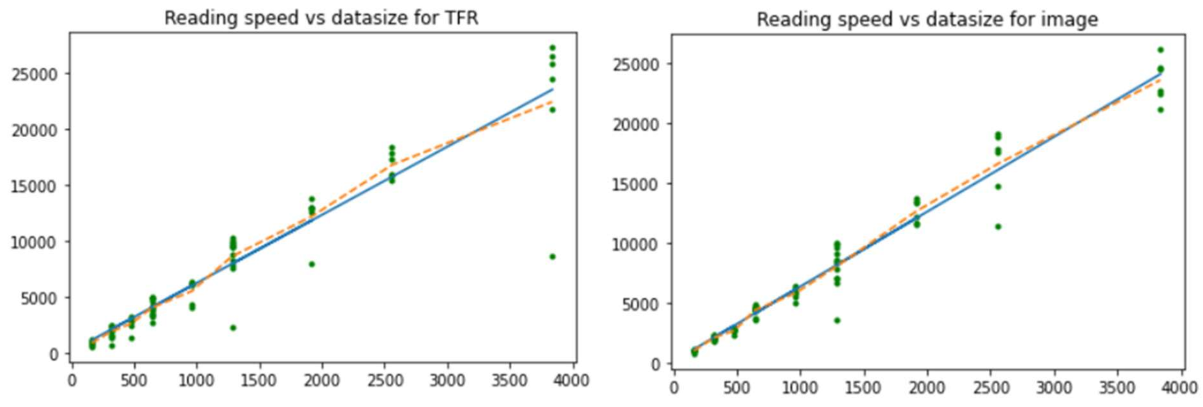


Figure 6 Reading speed variation with data size for TFR and image.

From the results obtained, shown in figure 6, there is clearly a linear relation between the reading speed and the data size. For large scale machine learning, the data size is one of the main factors that will be comparable to this case and it can be said that the reading speeds will increase further resulting in higher throughputs.

For a single machine, the linearity could be even more for this case of small data size specifically, due to the presence of data in the local SSD. For large data size, the capacity of the disk for each node would determine the requirement of network load needed for data access, based on which the throughput will change.

For speed tests in parallel on the cloud, one of the limits that relates to the case in the question is the limit for repeated writes to same object name, which can be addressed through object key indexing.

Except for reading speed variation with repetition, others show a considerable amount of linearity, which in practice can be useful for evaluating the effect of different parameters on the linearity.

Answer 3c:

Performance of machine learning task for different strategies with different batch sizes.

Sr. No.	Strategy	Batch size	GPU type	Final epoch accuracy	Wall clock time
1	OneDevice	16	Colab standard	0.2589	64.53
2	OneDevice	16	Gcloud AI Standard	0.2497	42.16
3	MultiWorkerMirrored	64	Gcloud AI Model L	Failed	Failed
4	Mirrored	128	Gcloud AI Model L	0.2897	59.60
5	CentralStorage	128	Gcloud AI Model L	0.2860	49.86
6	OneDevice	128	Gcloud AI Model L	0.2463	35.77
7	Mirrored	64	Gcloud AI Standard	0.2721	42.63
8	CentralStorage	64	Gcloud AI Standard	0.3713	37.54

From the table, it can be seen that for smaller batch size of 64, the synchronous type of strategy outperformed, whereas for larger batch size of 128, the asynchronous type gave the best accuracy. For both the batch sizes, the Mirrored strategy which is an asynchronous type, the time cost is higher.

Answer 4a:

Cherrypick Paper:

This paper mainly focuses on finding the near optimal configuration for cloud computation of a given task by isolating the best one from the rest. Task 1 was kind of in line with this, where same data pre-processing task was performed in different cluster configurations.

In the paper, different configurations were analysed based on the computational cost depending on the time for performing a given task. In Task 1, not just time was considered for analysis, but different working parameters of hardware were also analysed to understand computational cost.

The results of the paper showed computational cost increase for higher configurations and this is almost in line with the analysis of computation cost for the Task 1, where single node clusters appeared to be more computationally efficient.

The paper clearly mentions that its approach focuses mainly to figure out optimal configuration for performing repetitive tasks so as to reduce the cost for progressive iterations. In Task 2, repetition was one of the parameters for which the throughput of task was observed. But, unlike the paper, the Task 2 analysed the difference in terms of caching the RDDs rather than to check for different categories.

The paper also highlighted about saturation in performance of linear regression with upscaling, which even was not exactly performed in Task 2, but was a point of discussion in it. The paper concluded that beyond a point, the linear regression performance doesn't improve further with elevation of cluster configuration. A similar view was expected for linear regression performance comparison between single machine and multi-node.

Hybrid Parallel Training paper:

This paper is mostly relevant to Task 3, as both aim to compare the performance of Neural Networks for different types of distribution strategies.

In the paper, experiments were performed on both synchronous and asynchronous types of strategies. In Task 3 as well, experiments were done on both types of strategies.

In the paper, synchronous strategies included data parallelism, spatial parallelism and hybrid parallelism, in all of which the updated weights were fetch from the master to the workers and then the models in respective workers are retrained. In Task 3, this was included in OneDevice and CentralStorage strategies.

In the paper, asynchronous strategies included layer parallelism, filter parallelism and channel parallelism, in all of which the updated gradients were fetch from the master to the workers and then the models in the workers are retrained with updates of weights. In Task 3, this was done using Mirrored and MultiWorkerMirrored strategies.

In the paper, highest accuracy was obtained for data parallelism, which was then followed by two different types of hybrid parallelism. In Task 3, a similar pattern was observed for a batch size of 64 where Central device strategy gave the best accuracy, although a point to be noted is that the MultiWorkerMirrored strategy failed to run which could have changed the result. For a batch size of 128, the results showed contrary pattern from the paper, with Mirrored strategy producing the highest accuracy.

<u>Answer 4b:</u>

Batch data:

Batch data is available offline and there is not further addition of data to it once it is downloaded.

From the Cherrypick paper, a good strategy is to use single node cluster to store data in local SSD for small data size and to use multi node cluster to store data in persistent disk of respective nodes for larger data size.

From the Hybrid parallel training paper, an appropriate strategy would be to use asynchronous methods for small data size and synchronous methods for large data size.

Online data:

Online data is the case where additional data is fed to dataset at periodic intervals. These are usually written to durable storages rather than on memory ones, which is essentially writing to persistent disks rather than SSDs.

From the Cherrypick paper, a suitable strategy for such periodically increasing data would be to store in SSDs of master as back of the data before the periodic update of dataset would be avaliable.

From the Hybrid parallel training paper, using asynchronous methods would be computationally effective.

Stream data:

Stream data is the case where additional data is fetched to dataset on a continuous basis. Since the dataset is continuously updated, these are vulnerable to data loss due to down time. Due to continuous addition of data, the latency is high.

From the Cherrypick paper, the best strategy is to store the data into persistent disks of the worker nodes, which are fault tolerant and hence take care of data loss vulnerability.

From the Hybrid parallel training paper, the best strategy is to implement synchronous methods which would avoid the increase in latency caused by asynchronous ones.

Word count	Less than 1800
Colab notebook link	https://colab.research.google.com/drive/14IHNf9Orkj0tIh2qF-dB40Hdu1jN0an8?usp=sharing