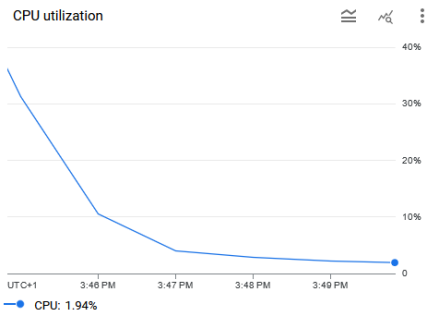
Big Data Report

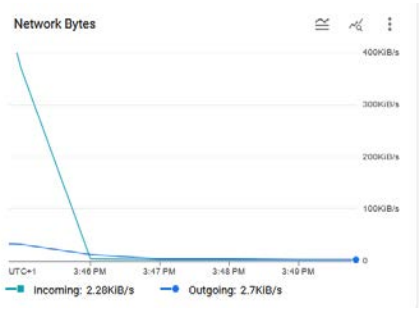
Andreas Ioannides

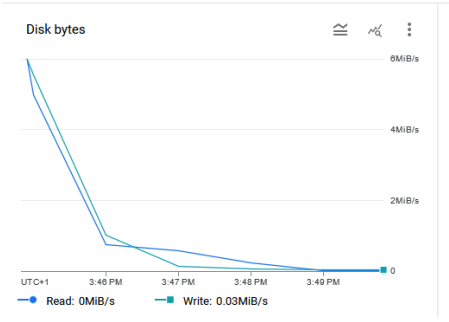
City University of London

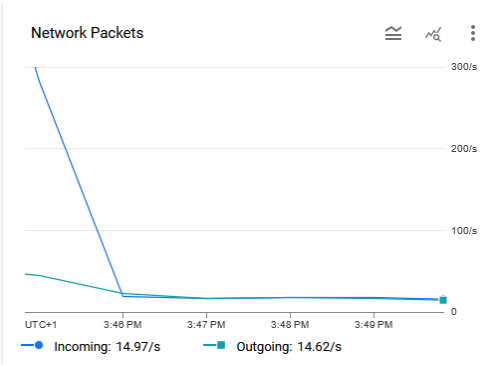
Andreas.Ioannides@city.ac.uk

Answer to task 1di









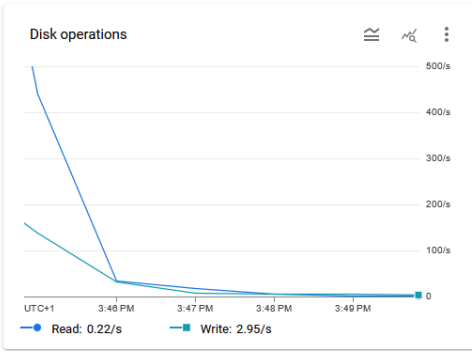


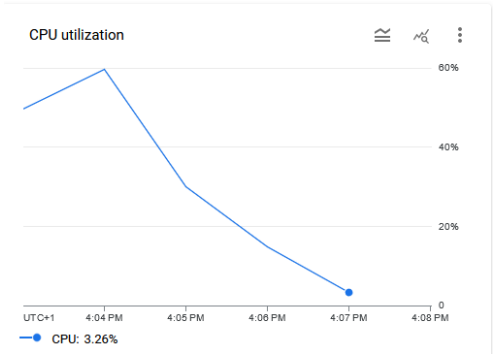
Figure 1 shows the jobs of a maximal cluster consisting of 3 worker nodes and 1 master before parallelization

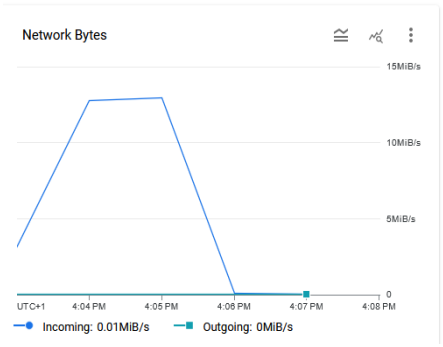
The CPU graph before parallelization, shows a rapid decline from 40% to 2% within a short time frame. The workload completes most of its processing very quickly, showcasing that the tasks are lightweight and not accordingly distributed across the nodes, indicating idleness. Network bytes decline in incoming and outcoming bytes, showing instances where there was a lot of activity, and instances when there was no activity. For network packets, there is a sharp drop in network packets, indicating a workload that quickly reduces communication needs. Disk bytes show a sharp decrease before, which might be due to a task that initially loads data, and then processes it in memory, requiring more disk reads. Disk operations had a decrease in read and write operations, showing uncoordinated disk access.

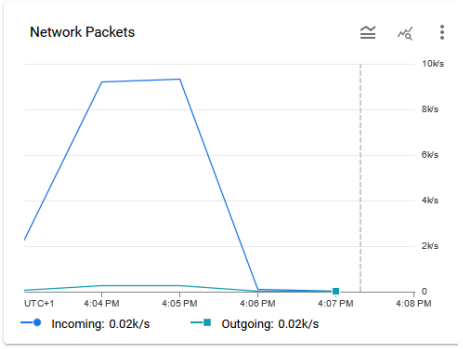
Figure 2 shows the jobs of a maximal cluster consisting of 3 worker nodes and 1 master after parallelization

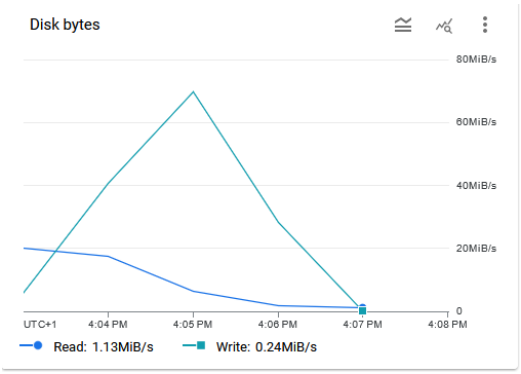
After parallelization, the decline of the CPU is less steep and has a minor bump before stabilizing. This shows a more even distribution of tasks. Network bytes have a spike in both incoming and outgoing bytes. This means that higher levels of data were exchanged between nodes, due to the tasks being distributed better. This suggests that parallel tasks may be distributed across multiple nodes requiring a higher volume of data transmission. Network packets have a sharp peak showing a rise in communication and workload between the nodes, which are necessary to perform parallel tasks. For disk bytes, we see a noticeable decrease in disk reads and a slight increase in writes, showing more frequent access to disks. This shows lower reliability of disk operations meaning more data is retained in memory reducing disk I/O latency. Finally, disk operations, show an increase in write operations indicating that more data was logged.

Answer to task 1dii









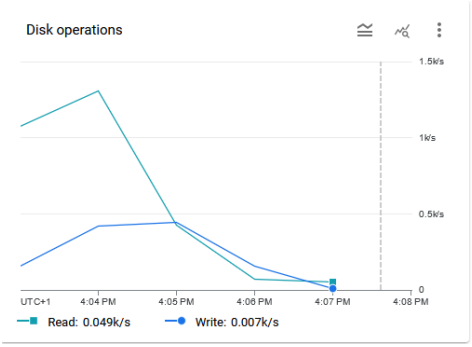


Figure 3 shows the jobs of a cluster consisting of 3 worker nodes 1 master and double the resources after parallelization

CPU utilization has a sharp decline from 60% to about 3% indicating a higher initial load, which is processed very quickly. Parallelization is intensive at the start, but quickly completes tasks effectively, which reduces the overall CPU load. Network bytes have a sharp spike in network activity which shows that the exchange of data between multiple nodes happens at the same time, followed by little activity as transfers complete. This shows that the distribution of tasks requires substantial data transfers. Efficient scaling strategies are needed to tackle peaks in networks to align the allocation of resources with usage patterns. Network packets show a fall in packet transfers, indicating that there was a period where parallel tasks faced a burst of packets, but then gradually fell off. Disk types show that there is a peak in disc reads indicating that data was read simultaneously across processes. More advanced disk management techniques are used, allowing for faster access. The tapering off of disk I/O activity shows that data is processed faster. Finally, disc operations initially spike, suggesting a quick fetch of data.

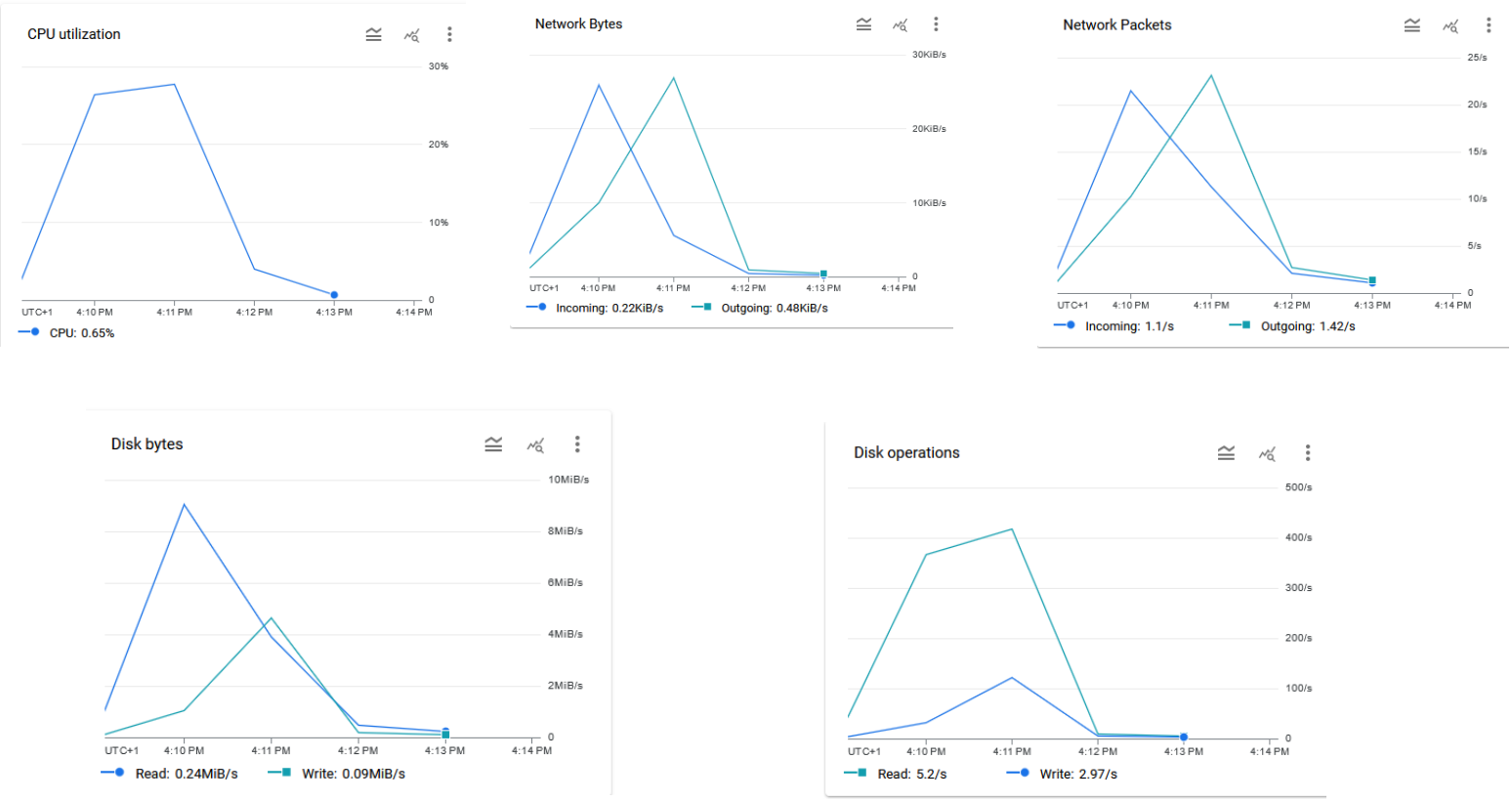


Figure 4 shows a cluster consisting of 0 worker nodes 1 master and eightfold resources after parallelization

CPU utilization has a significant spike, peaking at around 25% before gradually dropping to around 1%. This again shows us a burst of CPU activity, where the CPU handled a wider range of tasks simultaneously due to parallel processing. The rapid decline shows us that tasks were completed very quickly. The peaks in CPU showcase the urgent need for dynamic resource allocation. In network bytes, there are significant spikes in both incoming and outgoing network bytes. This shows us that parallelization caused more network activity as different communicated and transferred more loads of data. For disk bytes, disk read and write rates had an initial sharp peak, which gradually declined. We can see write operations after parallelization which indicates that tasks involved more data recording activities and logging activities. Network packets peak in both incoming and outgoing network packets, showing higher levels of network communication during parallelization. Finally, for disk operations, we can see an increase in both read and write operations, indicating more active and intensive disk usage. This could be due to parallel tasks simultaneously accessing and storing data on the disk. Managing spike peaks can help optimize costs related to resources and introduce autoscaling and resource management tools.

Answer to task 1diii

Apache Spark is designed to handle large datasets using a parallelization approach. Data is divided into partitions and the same operations are performed on each partition. This model is different from the traditional approach where different tasks can run on different processors and operate on the same dataset. In Spark, we use RDDs or data frames. Spark differs from standard applications in how it stores data and processes it. Standard applications rely on data storage whilst Spark uses a distributed storage system and applies parallelism to efficiently process large datasets.

Answer to task 2c

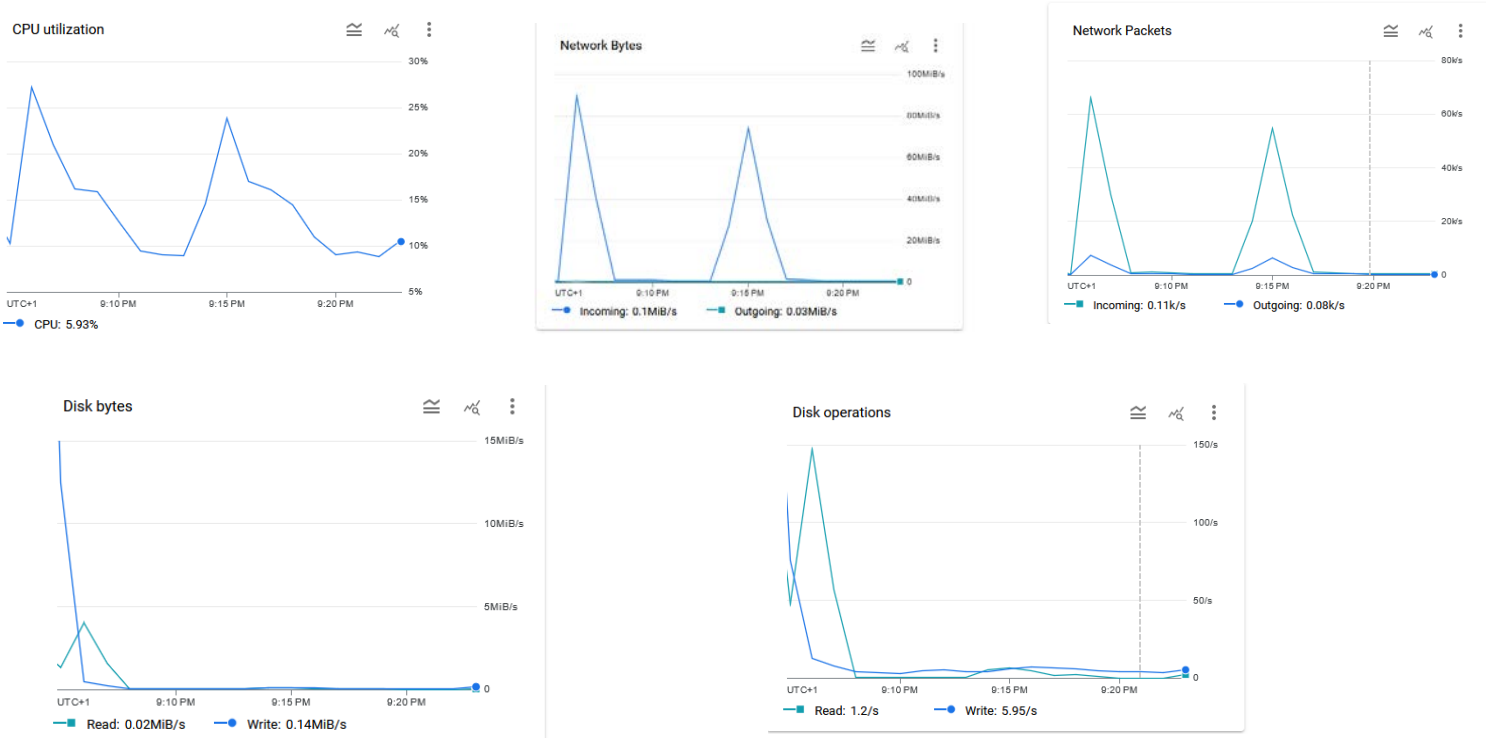


Figure 5 shows the jobs of a cluster consisting of 3 worker nodes 1 master and double resources after performing the speed test

CPU jobs showcase significant variability with a range of 10%-25% and a notable peak. This suggests frequent loading of data into CPU-intensive operations. For network bytes, we can see high peaks especially in incoming bytes, indicating large amounts of data being transferred. The peaks of network packets are synchronous for incoming and outgoing at around 60 packets per second. For disk bytes, write peaks are at around 150MB/s and read are at lower rates. Disk bytes show high write activity indicating substantial data writing to disks. For disk operations, read operations peak sharply at the beginning and stabilize later.

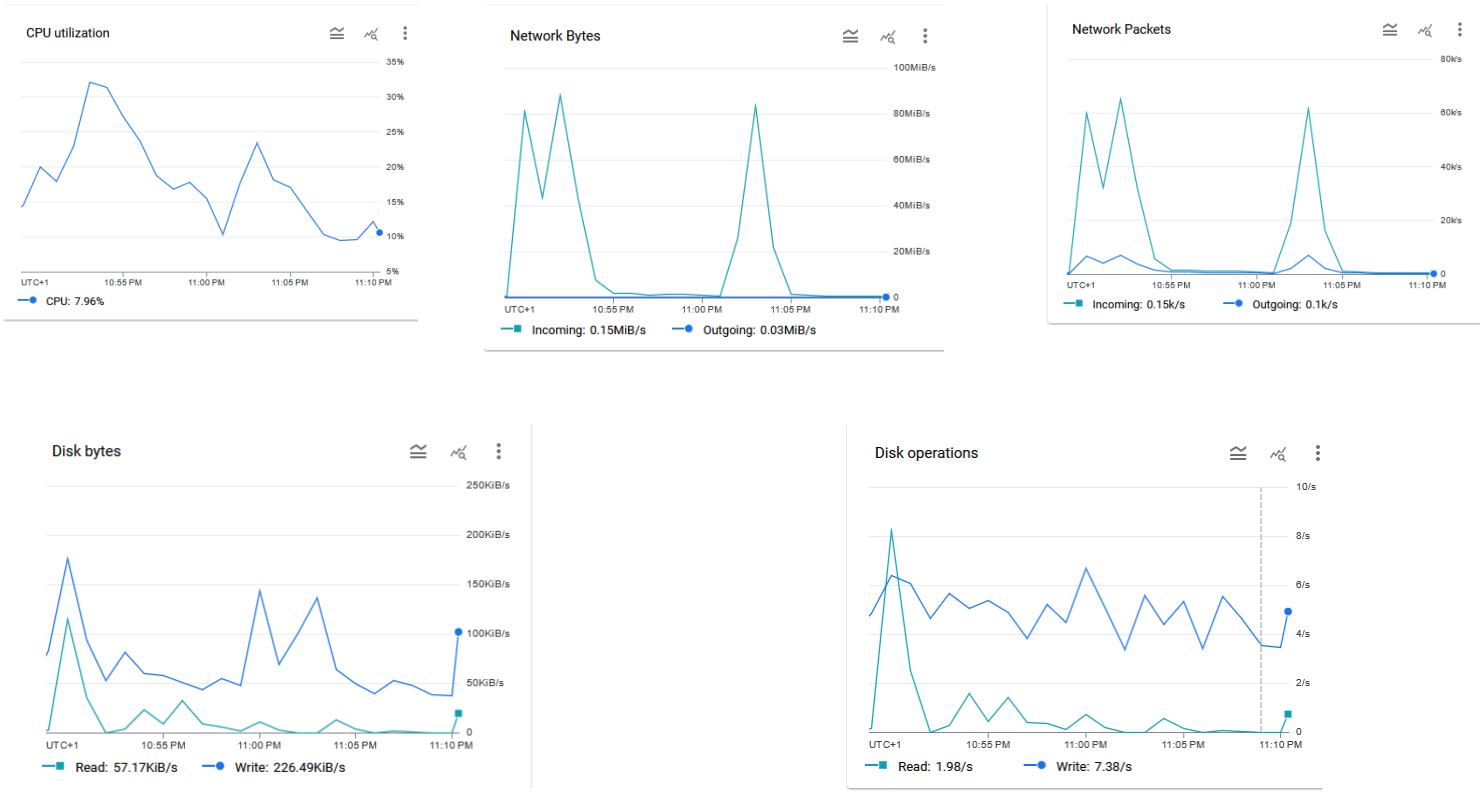
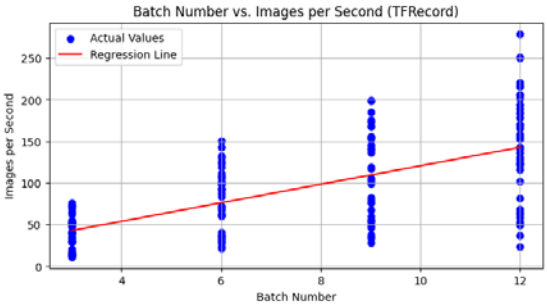
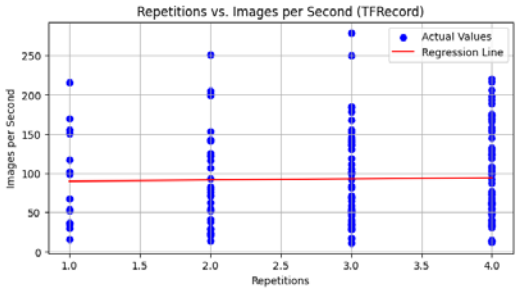


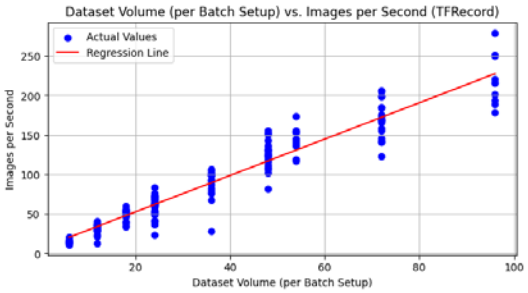
Figure 6 shows the jobs of a cluster consisting of 3 worker nodes 1 master and double resources after performing the efficiency test

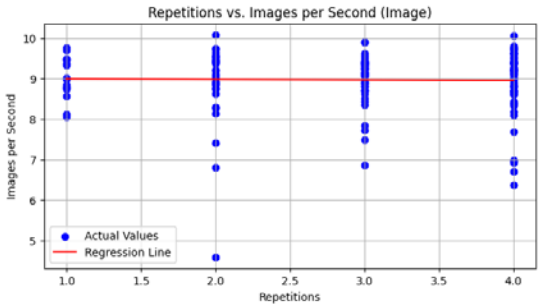
Considering the CPU graph, we can see a slight initial peak, followed by a more consistent usage, mostly ranging between 10%-15%. There is a smaller range of fluctuations when compared to the speed test. Network bytes show a reduction in both peak values and variability, indicating that less data is being transferred when implementing caching. For disk bytes, we can see a balance between read and write operations, with write operations being lower than those of the speed test. Considering disk operations, both read and write operations are more consistent, with lower peaks, which indicate a reduced load on disk due to caching. Both read and write operations show regular patterns. Implementing caching for Spark RDDs demonstrates an effective use of in-memory computation to reduce dependency on storage subsystems like disks and networks. This approach leads to more consistent performances by minimizing variability. Also, caching can optimize operational costs by managing the computational resources of clusters. In conclusion, caching is very effective in Spark where optimizing resources leads to better performance and efficiency of costs.

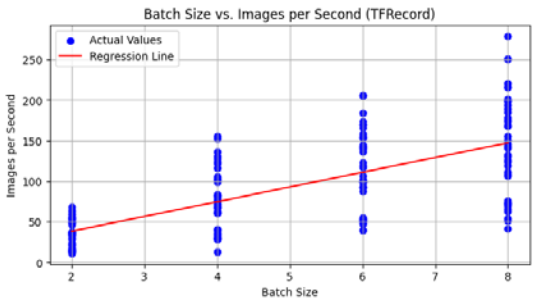
Answer to task 2d

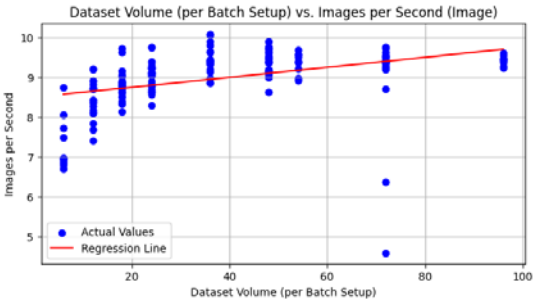












In large-scale machine learning applications, understanding relationships is crucial for optimizing performance. Higher batch size leads to higher throughput. This is a positive for cloud-based environments since processing large batches is more efficient due to reduced loading and network communications. Dataset volume and throughput suggest large volumes of data in each batch can improve efficiency. This is relevant for machine learning models where data can be distributed across multiple physical locations. On the other hand, large batches tend to require more powerful computing resources that are more expensive, therefore, we have to balance the cost and gains in speed and efficiency to find an optimal point that satisfies both performance and budget. On a single machine, a higher batch size leads to higher throughput to a point, where beyond that point, returns diminish due to hardware limitations. This is because systems can only handle a specific amount of data simultaneously, before running into bottlenecks. Cloud environments can scale horizontally by distributing workloads across multiple machines. This can also increase throughput in large batch sizes and dataset volumes, as observed in our regression plot. In single machines, latency is lower and more predictable, as data transfers are confined to internal buses and storage systems. On cloud-based systems, network latency can impact performance, especially when data has to travel very long distances between nodes. In conclusion, the behavior observed in cloud environments highlights the differences from single-machine setups and demonstrates the ability of the cloud to handle larger workloads more efficiently. When speed tests run in parallel, demand on I/O bandwidth increases. Disk I/O operations can become a bottleneck especially when the read/write capacity is exceeded. The regression graphs show the relationship between volume and throughput, where increasing throughput with larger volumes suggests good scaling and disk I/O operations. The positive correlation observed in the network bandwidth and latency might be attributed to efficient network infrastructure. Also, the generally consistent increase in throughput indicates minimal contention and highly effective resource management. For scalability, we can see that our results imply that the cloud environment scales well with increasing demand, as indicated by improved throughput with larger batch numbers and data volumes. Linear models assume a constant relationship between dependent and independent variables, this simplification exhibits non-linear behaviors due to system constraints and resource interactions. Assumptions of independence may also not be true, in scenarios where previous configurations affect performance. Linear models provide a straightforward method to predict outcomes and are easy to understand and interpret. Usually, linear models are used as baseline models to facilitate understanding and guide initial resource allocation decisions in cloud environments such as determining a starting point for batch sizes. However, linear models fail to capture thresholds or saturations, and complex behavior in general. Linear models also cannot adapt to cloud computing environments, where resources and networks change rapidly. Possible alternatives are machine learning models and polynomial regression since they can model and capture non-linear relationships and interactions between multiple variables. Real-time monitor tools and systematic testing can determine optimal configurations and adjust models based on performance.

Answer to task 3a

The cherry-pick research paper utilizes Bayesian optimization to model and predict the performance of cloud configurations. This optimization process significantly reduces costs as well as time. For our coursework, we used similar optimization techniques, to select the optimal number of nodes and type of machines, and other Spark cluster parameters to maximize the efficiency of image processing tasks. The research paper adapts configurations based on the application's demands and characteristics. In the coursework, on the other hand, we adapted Spark clusters based on demands of image resizing, recompression, and TF Records. The cherry-pick research paper uses Bayesian optimization to minimize unnecessary trials and focus only on the most important configurations. In our coursework, we have used partitioning and data distribution methods such as map partitions with index, to minimize shuffling and maximize parallel processing efficiency. The research paper evaluates configurations under conditions that ensure efficiency and robustness. For our coursework, we tested Spark under different cluster configurations which mirrors the experiments of the research paper. In conclusion, our coursework and the research paper deal with big data processing, making optimization of resources crucial.

Answer to task 3b

For batch processing scenarios, we involve processing large datasets that are not time-sensitive. We start by profiling typical jobs to understand their resource patterns. We identify the CPU that is most critical to maximize performance. Considering the research paper, we can apply its methodology to different configurations, and then narrow down to the configurations that offer the best tradeoffs between cost and performance. For example, batch jobs that don’t require real-time processing can use cheaper instances. Analyzing the cost and performance gains across different configurations can ensure that the selected setup will remain optimal over time. We can use SSD to speed up the time it takes to access data. We can also use Bayesian optimization to explore and evaluate different configurations, considering those that will offer the best improvements in performance. Considering streaming processing, we can use configurations prioritizing fast CPU and ample network bandwidth to handle frequent data transfers. We can use the research paper’s model to adapt configurations based on volumes of data and processing needs. Choosing cloud configurations that allow easy scaling would be crucial to handling peak loads efficiently. We can also consider the Bayesian model from the research paper to predict future resource needs based on data patterns and trends. Finally, we can consider constraints such as maximal allowance latency and use these as boundaries for optimization. In conclusion, these strategies can help manage cloud resources dynamically and cost-effectively, improving service reliability and performance.

WORD COUNT:

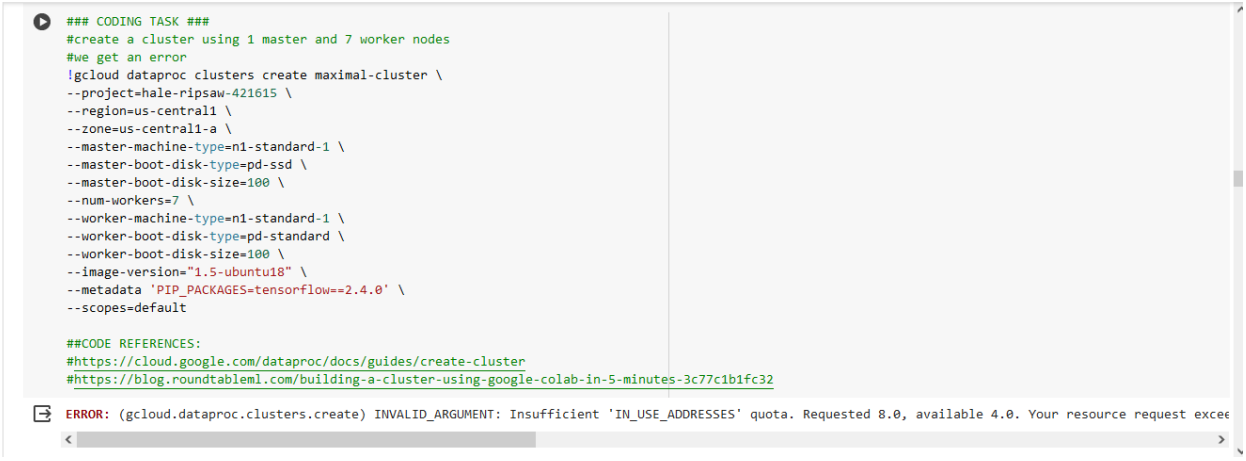
1955/2000

COLLAB LINK:

https://colab.research.google.com/drive/1wK1sIfpd8K_AFs-vqFeArgDm07DJub8U?usp=sharing

Screenshots of Google Collab errors

Maximal cluster error



```
### CODING TASK ###
#create a cluster using 1 master and 7 worker nodes
#we get an error
!gcloud dataproc clusters create maximal-cluster \
--project=hale-ripsaw-421615 \
--region=us-central1 \
--zone=us-central1-a \
--master-machine-type=n1-standard-1 \
--master-boot-disk-type=pd-ssd \
--master-boot-disk-size=100 \
--num-workers=7 \
--worker-machine-type=n1-standard-1 \
--worker-boot-disk-type=pd-standard \
--worker-boot-disk-size=100 \
--image-version="1.5-ubuntu18" \
--metadata 'PIP_PACKAGES=tensorflow==2.4.0' \
--scopes=default

##CODE REFERENCES:
#https://cloud.google.com/dataproc/docs/guides/create-cluster
#https://blog.roundtableml.com/building-a-cluster-using-google-colab-in-5-minutes-3c77c1b1fc32
```

```
ERROR: (gcloud.dataproc.clusters.create) INVALID_ARGUMENT: Insufficient 'IN_USE_ADDRESSES' quota. Requested 8.0, available 4.0. Your resource request exce
```

Initial cluster with 8VM's error

```
#i) we create an initial cluster with 8VM's
#we get an error
!gcloud dataproc clusters create task-1d-cluster \
--project=hale-ripsaw-421615 \
--region=europe-west2 \
--master-machine-type=n1-standard-1 \
--master-boot-disk-type=pd-ssd \
--master-boot-disk-size=100 \
--num-workers=7 \
--worker-machine-type=n1-standard-1 \
--worker-boot-disk-type=pd-standard \
--worker-boot-disk-size=100 \
--image-version="1.4-ubuntu18" \
--metadata 'PIP_PACKAGES=tensorflow==2.4.0' \
--scopes=default
#tried every single region and i still get a quota error
#lets try with 4 VM's instead like i have been told in an email
#Hello Andreas, As previously discussed,
#if running it with 1+7 machines isn't feasible, please attempt it with 1+3 machines instead.

#CODE REFERENCES:
#https://cloud.google.com/dataproc/docs/guides/create-cluster
#https://blog.roundtableml.com/building-a-cluster-using-google-colab-in-5-minutes-3c77c1b1fc32
```

```
ERROR: (gcloud.dataproc.clusters.create) INVALID_ARGUMENT: Insufficient 'IN_USE_ADDRESSES' quota. Requested 8.0, available 4.0. Your resource request exce
```