Big Data Report

Nicholas Tsioras

Nicholas.Tsioras@city.ac.uk

Task 1d i

- Maximal Cluster 1 Master 7 Worker Nodes Before Parallelization

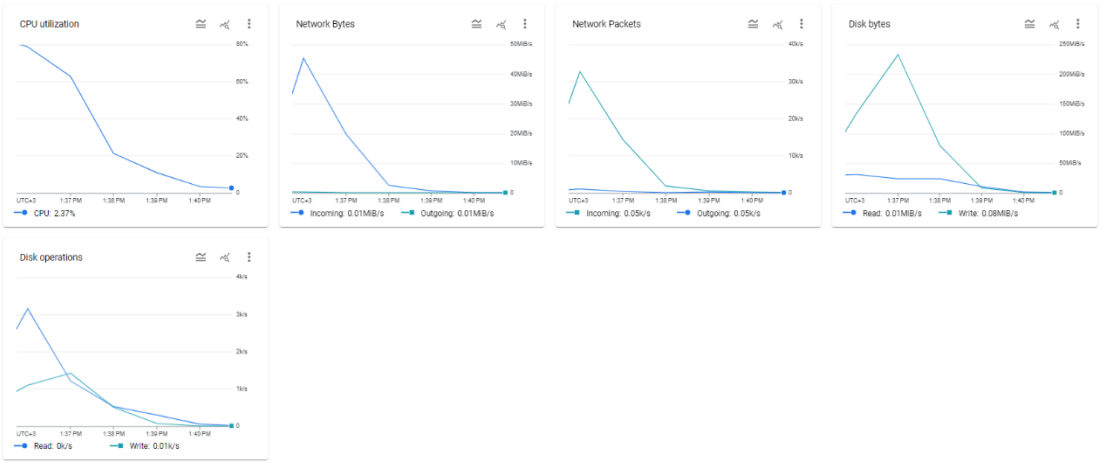

Figure 1. Job for Maximal Cluster with 1 Master and 7 Worker Nodes Before Parallelization

- Maximal Cluster 1 Master 7 Worker Nodes After Parallelization

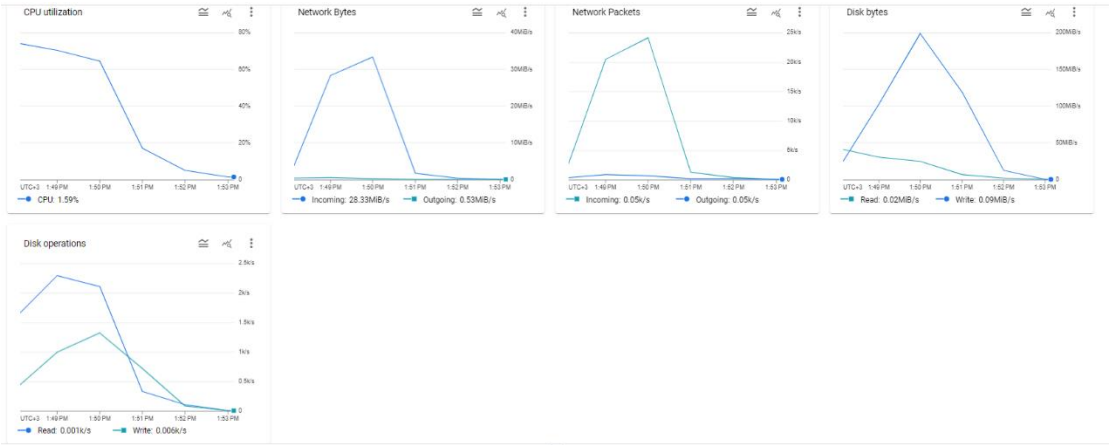

Figure 2. Job for Maximal Cluster With 1 Master and 7 Worker Nodes After Parallelization

The CPU Utilization shows a sharp decline from around 80% to 0% This suggests that initially there's a burst of CPU activity which then tapers off, indicating that not all of the nodes are being used effectively. On the other hand, after parallelization the CPU also starts around 80% and decreases more gradually to around 20%. The more gradual decrease indicates a better distribution of load across more nodes over a longer period of time, which suggests improved parallelization.

The Network Bytes graph before parallelization peaks at around 50 MiB/s but then rapidly drops to 0MiB/s. On the other hand after parallelization the Network Bytes peak at 40MiB/s with a significant sustained level of outgoing traffic, which indicates that there are ongoing data exchanges between nodes and the parallelization improved network performance. As for the Network Packets, similarly to the Network bytes, after parallelization, there is a more pronounced peak and sustained activity which aligns with increased network bytes, confirming that more nodes were consistently used in the network.

Before parallelization the graph of the disk bytes peak around 250MiB/s but then drops rapidly indicating a large amount of data being processed at once but not maintained. On the other hand, after parallelization, the peak is around 200MiB/s but with a slower decrease, suggesting more sustained disk operations. As for the disk operations, there is a slight increase after parallelization but its still quite low, which indicates that the I/O is not the primary activity in the cluster's workload.

Overall, we can see that after parallelization the cluster's resources are being better utilized, including a more sustained and higher CPU Usage, more consistent network traffic, and in general a better parallel task distribution and coordination among the nodes of the clusters.

Task 1d ii

For this task, we experimented with different cluster configurations. We are going to compare the performance metrics of a cluster with 8 machines (Figure 2), a cluster with 4 machines that have double the resources in each machine, and a cluster with 1 machine with eightfold resources.

- Cluster 4 Machines Double Resources

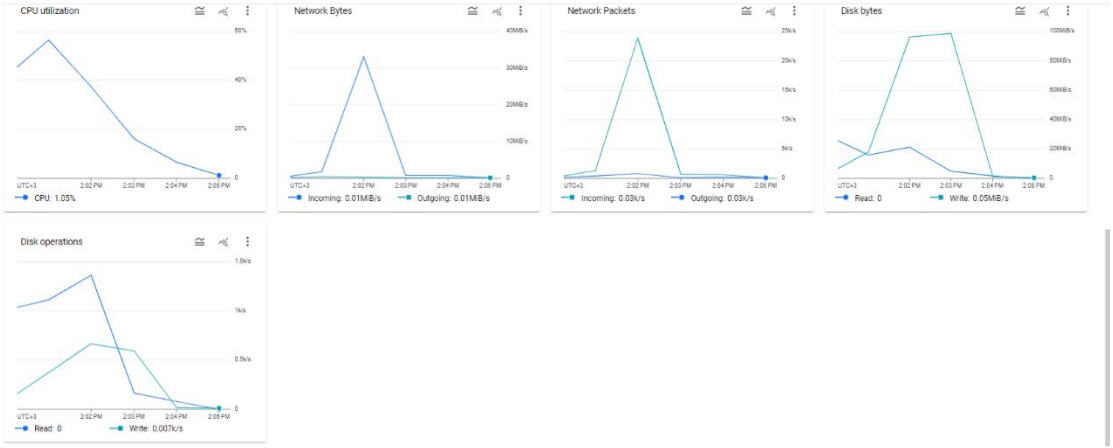

Figure 3. Job for Cluster with 4 Machines with Double Resources in Each Machine

- Cluster 1 Machine Eightfold Resources

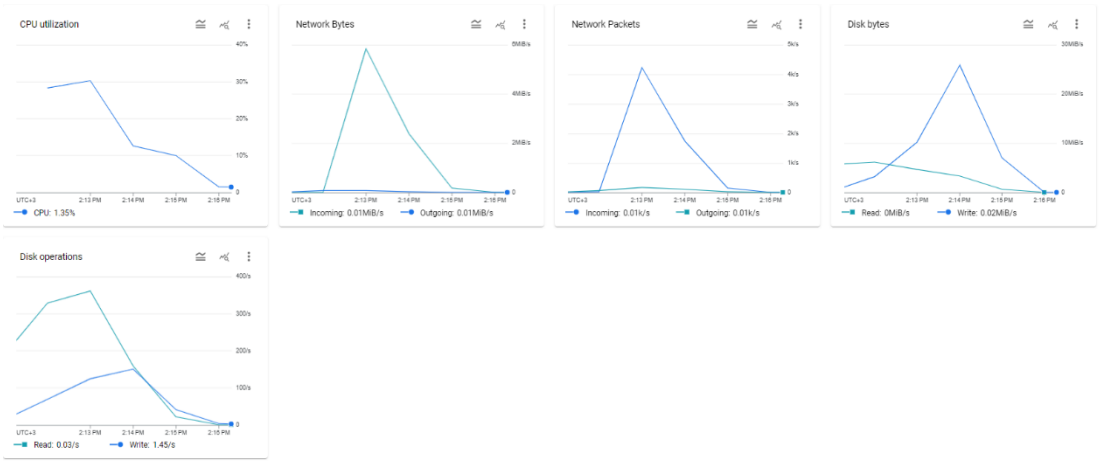

Figure 4. Job for Cluster with 1 Machine with Eightfold Resources

In terms of the Disk Bytes and Operations (I/O), there are higher peaks in disk bytes for the 8 machine cluster, showing that there are considerable I/O operations across multiple nodes. The cluster setup with 8VMs the system showed high disk and network activity due to the inter-connection between multiple nodes. Then, after reducing the number of nodes to four, but also using double the resources for each node, we notice reduced disk activity and network traffic, as more processing was handled locally. On the other hand, the single VM with eightfold resources shows significantly higher disk bytes throughput especially in write operations but much lower disk operations and network usage but with more efficiency in consolidating resources into a single node. This indicates that there are larger, less frequent write operations due to better resource allocation and fewer I/0 bottlenecks with a single machine handling all operations. The internal resources were sufficient to handle the workload efficiently, which led to minimized need for disk read and writes and network communication. The 4VMs cluster is considered a middle ground with moderate levels of disk and network activity. It provides a more balanced option for a wide range of tasks by leveraging increased node resources in comparison to the 8VMs.

Overall, this analysis indicates that using less, but more powerful nodes will reduce the complications related to managing data between multiple machines and accessing data from outside sources. Our analysis shows that for tasks that can be handled well with less machines, it can be more efficient and cost-effective to use fewer, but stronger machines with more resources. This approach could lead to better performance and lower costs in the cloud environment.

Task 1d iii

In our laboratory sessions, the tasks were mainly performed directly on local machines. Our current use of Spark operates primarily in the cloud, exploiting cloud resources to manage and distribute processing loads. The parallelization approach that we are using, involves dividing the data into multiple partitions and processing these partitions simultaneously across multiple nodes. This method is one of the main strengths of Spark when it is deployed in Cloud environments [3].

Task 2c

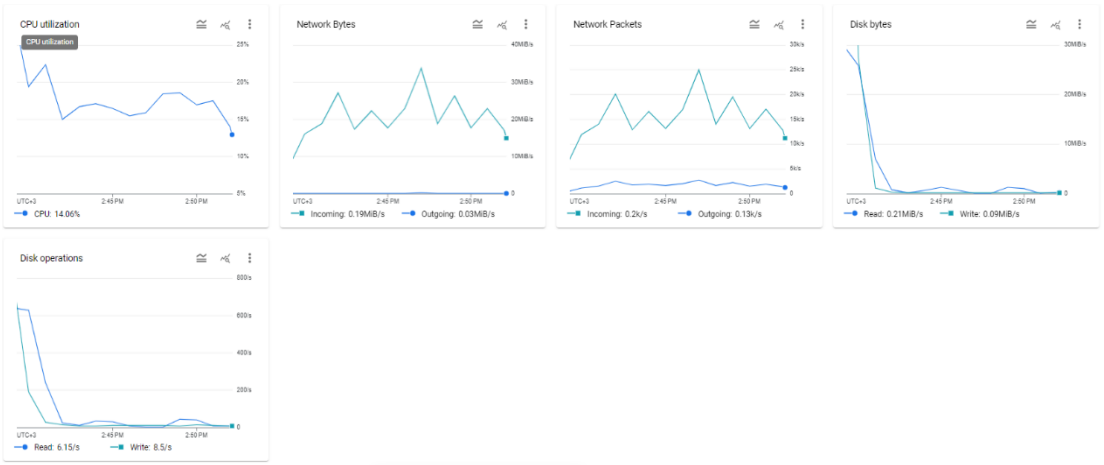

Figure 5. Job for Cluster Showing Speed and Efficiency Before Caching

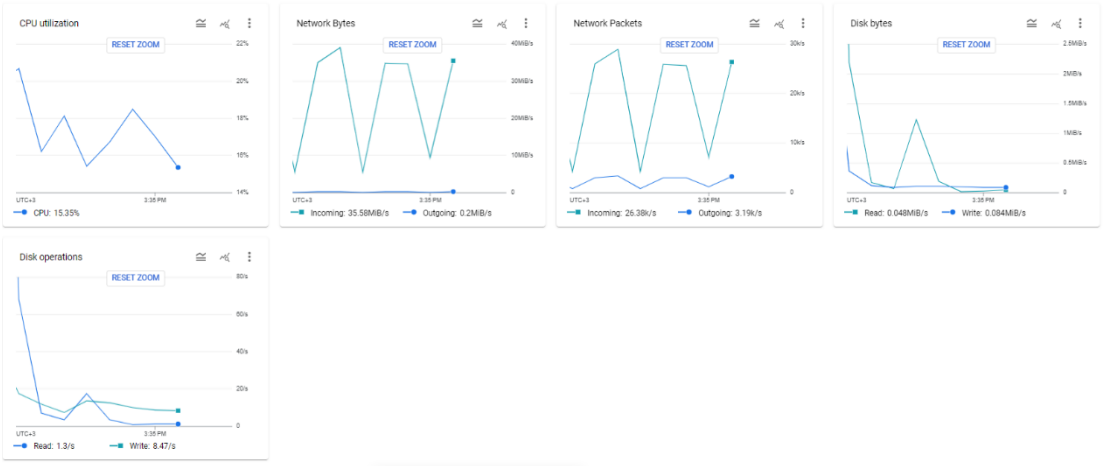

Figure 6. Job for Cluster Showing Speed and Efficiency After Caching

- **Reasons to Use Caching**

Caching is a useful optimization technique in distributed computer environments like Apache Spark when dealing with iterative algorithms or multiple read options in a dataset. In Spark, caching an RDD can improve performance by storing the intermediate RDD in memory after the first computation, making it easier to access in subsequent actions without the need for re-computation [4].

The main reason to use the cache function in an RDD is that it avoids re-computation each time it is accessed. This is useful when the same RDD is used multiple times for counting different statistics from a dataset. Another benefit of the cache function is that it improves performance. When an RDD is cached, subsequent read operations are served directly from memory, leading to faster execution times [1]. Finally, caching helps the system use the resources more effectively. By storing data that is frequently accessed in memory, the cluster's CPU cycles and I/O operations are reduced, which allows for better thoughtput.

- **Implementation and Impact**

In the provided figures (figures 5,6) significant differences can be observed in the speed and efficiency of the clusters. In Figure 5 before caching, it is noticeable that multiple read operations on the RDDs lead to inefficiencies. It is likely that each operation in the script without caching had to go through all of the processing steps from start to finish, including reading from the disk and using the network. This makes the process a lot slower and inefficient. After introducing the cache, it is observed from the performance metrics of Figure 6 that there is significant improvement in the processing times highlighted by the reduction of CPU and disk I/O spikes, which aligns with the improvement of efficiency. The CPU Utilization graph shows that after caching there is a more stable and possibly lower average usage which indicates that the system did not spend that much time for re-computation. Also, after caching, there is a decrease in the read and write disk operations per second which indicates a reduced access to physical storage thanks to data being served from memory. Also, the Network Bytes and Packets show less variability and lower peaks of network traffic because less data needs to be shuffled across the different nodes after caching. Overall, it is noticeable from the 2 graphs that caching optimized performance and decreased execution time by reducing the need for repeated data generation processes. The performance metrics after caching are smoother, showing the effectiveness of caching in distributed data processing tasks.

Task 2d

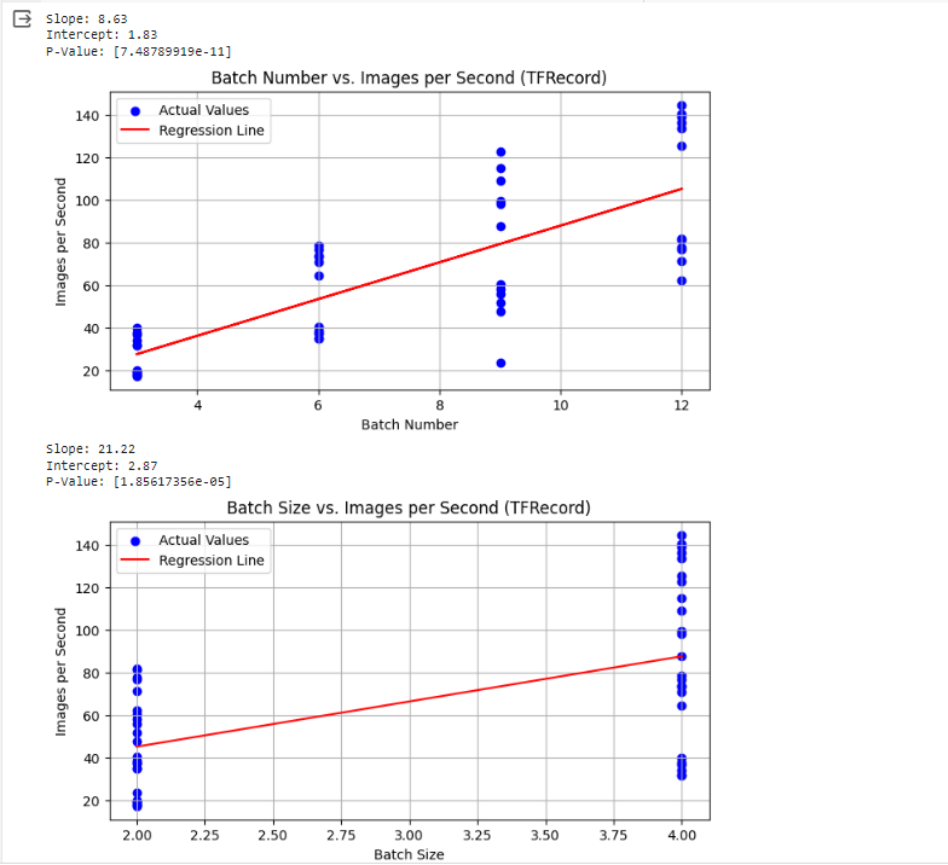

Figure 7. Batch Number/Size vs Images per Second (TFRecord Dataset)

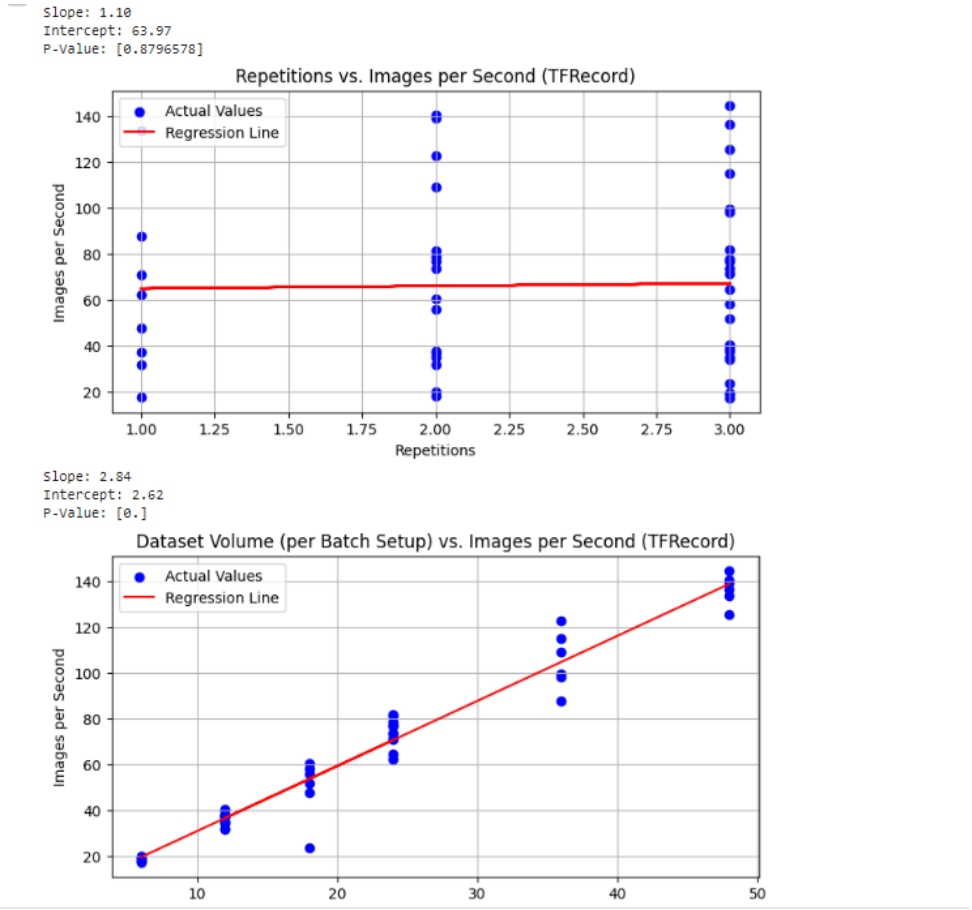

Figure 8. Repetitions/Dataset Volume vs Images per Second (TFRecord Dataset)

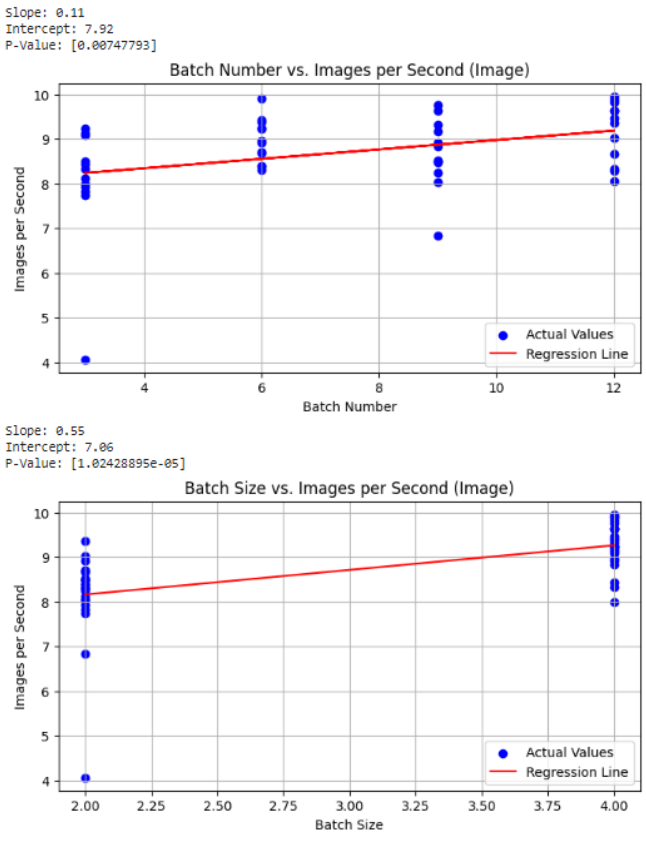

Figure 9. Batch Number/Size vs Images per Second (Image Dataset)

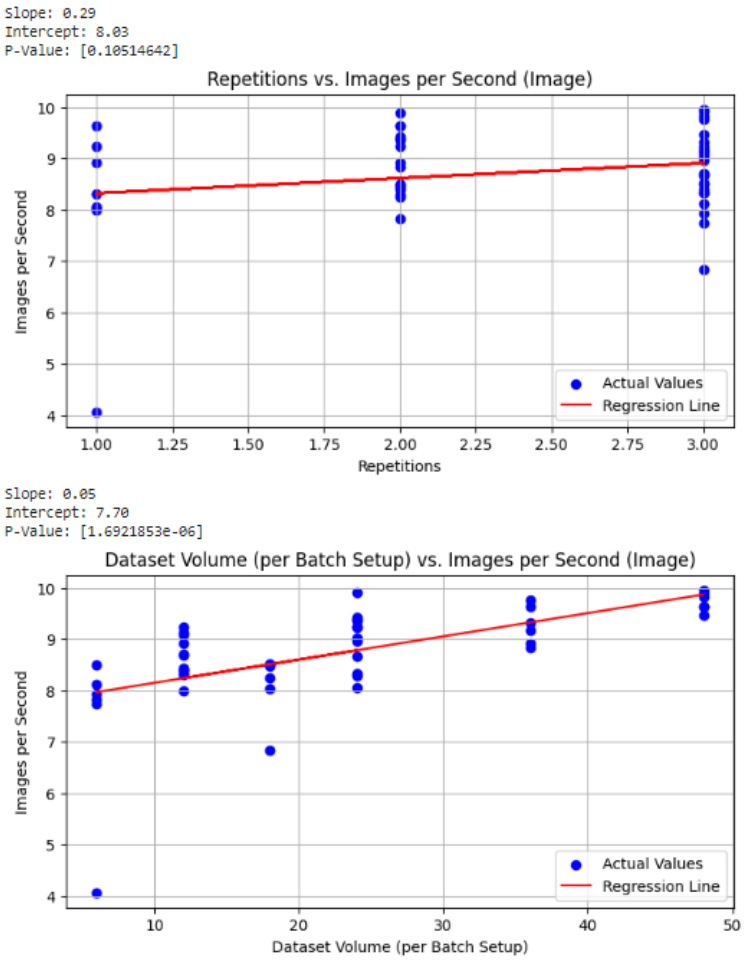

Figure 10. Repetitions/Dataset Volume vs Images per Second (Image Dataset)

- **Linear Regression Analysis**

From the graphs above where we can see the regression analysis for the images per second for each parameter separately, we can observe that increasing the batch numbers shows a steady thoughtput with a small slope. This suggests that the thoughtput is not impacted from the number of batches. This indicates that the system's handling of larger numbers has been efficiently parallelized. We also notice that as the batch size increases, the thoughtput of images per second also increases for both the TFRecord and Image datasets. This indicates that the system can effectively handle larger volumes of data at a time. For both datasets, especially for the TFRecord dataset, as the number of repetitions increases, there is still very little variance in the images per second. That shows that the repeatability of the tasks does not affect the thoughtput. Finally, the positive slope in both datasets and the high slope value for the TFRecord dataset suggests that increasing the dataset volume processed per batch improves throughtput, likely due to better utilization of I/O and better processing capabilities.

- **Cloud-Based Machine Learning and Throughtput Optimization**

In cloud environments, data is often stored in different locations. So being able to handle larger batches is very important because it reduces processing time, and it

makes large-scale machine learning tasks to be more practical and cost-effective. Cloud providers link throughtput to disk resource capacity to ensure that larger tasks which need more resources can be managed effectively with good performance being maintained. This strategy is essential for maintaining consistent quality by optimizing data processing workflows in cloud-based systems.

- **Considerations in Parallel Speed Tests**

In parallel speed tests, on the cloud, the potential bottlenecks and related to network latency, I/O disk speeds, and resource allocation should be considered. The linear improvement in thoughtput with increased batch sizes might not always hold if the bottlenecks become significant. For example, even if more batches are being processed at once, if the disk reads the data slowly and the network is slow, performance could still get worse.

- **Reflection on Linear Modeling and Theoretical and Practical Perspectives**

Linear models are essential for identifying trends and potential bottlenecks in data processing in cloud environments. However, they often fail to account for non-linear behaviors due to practical limitations such as system bottlenecks. Theoretically, consistent scaling behavior is expected as the parameters increase, but practical limitations often lead to deviations from this model. Therefore, understanding these deviations is important for optimizing systems and predicting realistic performance outcomes in practical applications. In our case, the observed results reflect well on the system's capability of handling large-scale operations in the cloud, with significant implications for the efficiency and scalability of machine learning workflows.

Task 3a

The core idea of the Cherrypick paper is about finding the best cloud configurations for big data analytics tasks without over-searching. It proposes a model-based approach to predict the performance of different configurations efficiently. This enables the selection of an optimal configuration based on different cost or performance metrics. This approach aligns well with our tasks as we evaluated how different configurations and optimizations impact the performance of data processing tasks. To be more specific, we performed a variety of tests and optimizations such as parallelization strategies, efficiency improvements like caching, and evaluated them using different cluster configurations (CPU Utilization, disk memory etc). The experiments we did with different cluster sizes and resource allocations reflect the approach that the Cherrypick paper uses to find effective configurations that optimize a more balanced resource use and operational speed.

Task 3b

- **Defining Strategies for Application Scenarios**

    - **Batch Processing:** Batch processing refers to the processing of high volumes of data all at once in a group (batch) within a specific time. For batch processing, the optimal cloud configurations can influence both performance and cost. It is important to opt for configurations with high computational and memory since these are better at processing larger datasets more efficiently. In addition, integrating SSD Storage can accelerate the process of accessing data which are important in batch jobs [2]. Also, beyond hardware

specifications, it's also important to employ predictive models, like those proposed in the Cherrypick paper, which can identify the most cost-effective configurations that align performance needs. This strategic approach ensures that resources are not only powerful but are also utilized in the most efficient, economical way.

- **Stream Processing:** Stream processing refers to the processing of a continuous stream of data right after it is produced. Stream processing is a lot faster than batch processing and it requires configurations that can manage continuous data flow and provide real-time responses. For this to be done, it is important to implement a cloud setup that can scale based on the volume of the data and the processing needs, ensuring optimal resource utilization without overcrowding the system [2]. Ensuring high availability and redundancy is crucial for maintaining data processing continuity without disruptions. Finally, optimizing network configurations for improved throughtput and reduced latency is essential since stream processing involves significant data transfers across the network. These strategies ensure that the system can support the demands of stream processing with efficiency and reliability.

Both batch and stream processing benefit from the adaptive methodology in the Cherrypick paper. By exploiting predictive modeling and performance simulation, users can predict and adapt configurations based on real-time data processing requirements which can lead to better resource utilization and cost savings. Also, experimenting with different configurations in a simulated environment before deploying can help find a balance between cost and performance that other methods might miss.

References

[1] C. Patidar, ‘’How Cache Works in Apache Spark,’’ Medium, 2023, [Online]. Available at: https://medium.com/@charchitpatidar/how-cache-works-in-apache-spark-aea6eeb3fd03 . [Accessed: 30-04-2024].

[2] ‘Difference Between Batch Processing and Stream Processing,’’ GeeksforGeeks. [Online]. Available at: https://www.geeksforgeeks.org/difference-between-batch-processing-and-stream-processing/ . [Accessed: 01-05-2024].

[3] ‘’Parallel Processing in Python, ‘’ GeeksforGeeks, 2019, [Online]. Available at: https://www.geeksforgeeks.org/parallel-processing-in-python/ . [Accessed: 01-05-2024].

[4] ‘’RDD Programming Guide,’’ Apache Spark, [Online]. Available at: https://spark.apache.org/docs/latest/rdd-programming-guide.html#rdd-persistence . [Accessed: 01-05-2024].

Word Count: 1995

Google Colab - Link:
https://colab.research.google.com/drive/1suI7ZPn1JgGcV_6uCz2SqigHyM8tPGs_?authuser=1#scrollTo=C_uNa6VeTLSN