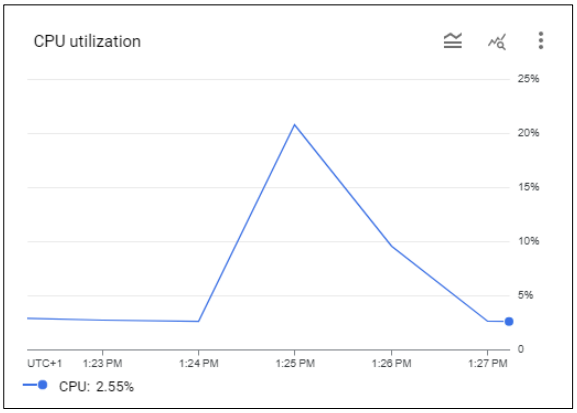
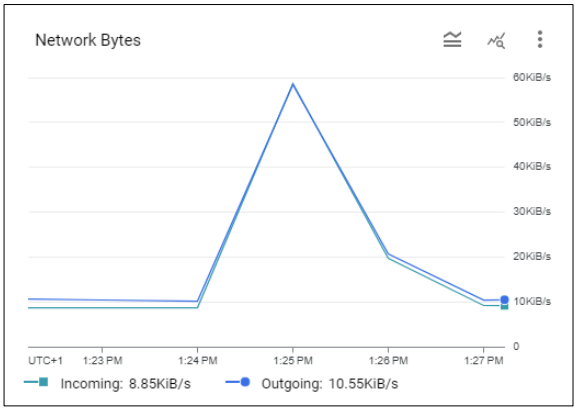
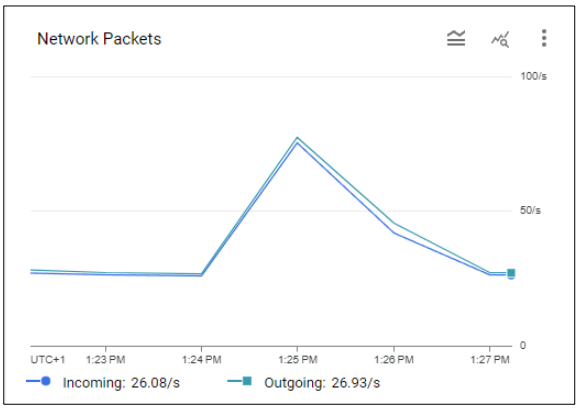
# INM 432 Big Data

Name: Dimitris Nikandrou Student ID: 200013603 Email: dimitris.nikandrou@city.ac.uk

## Task 1d i)





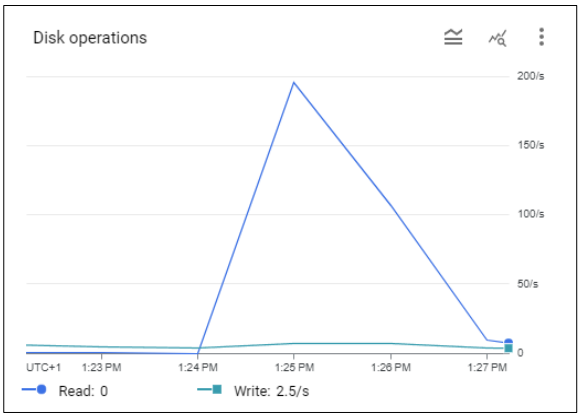



*Figure 1: Graphs showing different metrics, for the cluster with 8 machines. 7 worker nodes, 1 master node and maximal SSD size (100). This is the cluster before parallelization is implemented.*


*Figure 2: Graphs showing different metrics, for the cluster with 8 machines. 7 worker nodes, 1 master node and maximal SSD size (100). This is the cluster after parallelization is implemented.*

Before parallelization, the CPU utilization graphs showed low to moderate peaks, indicating underutilization of the cluster resources. After parallelization, the peaks in the CPU utilization graphs are more consistent and pronounced, indicating better distribution of the computational workload across multiple nodes. This behaviour would suggest that parallelization effectively improved the overall utilization of the cluster.

Before parallelization, the disk operations and disk bytes graphs showed minimal disk activity. This suggested that the cluster was not effectively utilizing disk I/O. After parallelization, the graphs show increased and sustained disk activity, which indicates improved utilization of disk resources. The increase in write operations and bytes, in particular, suggests that the processing was more evenly distributed across the nodes, leading to the better performance of the cluster.

Before parallelization, the network bytes and packets graphs indicated low network activity. After parallelization, there is a clear increase in network traffic, which aligns with the improved CPU and disk utilization. The increase in network packets and bytes reflects the improved distribution of data processing tasks across multiple nodes, which typically involves more data exchange between nodes.

**Task 1d ii)**


*Figure 3: Graphs showing different metrics, for the cluster with 4 machines and double the resources.*

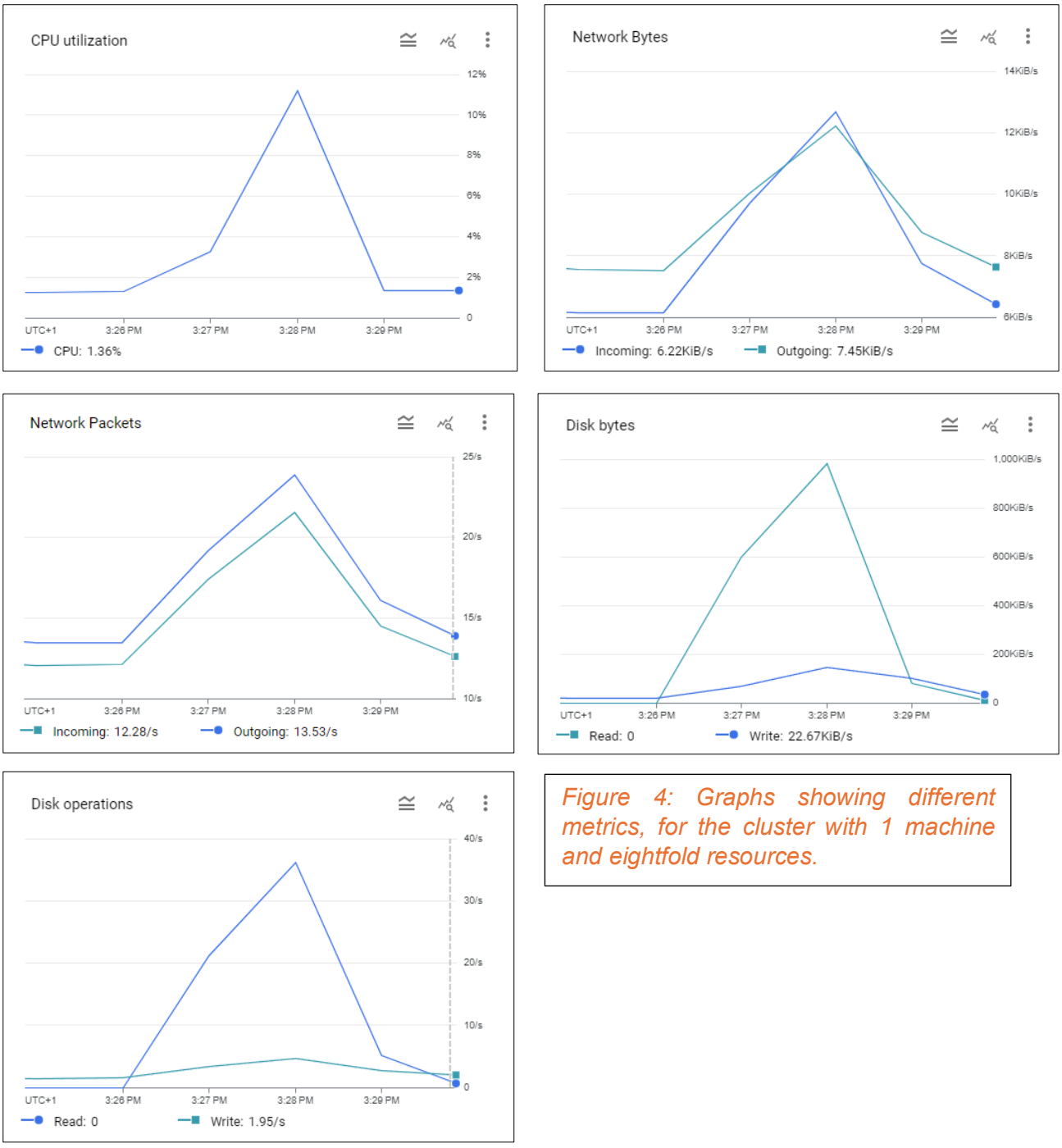



*Figure 4: Graphs showing different metrics, for the cluster with 1 machine and eightfold resources.*

When evaluating the different cluster configurations, it was clear that increasing the resources within each node reduced disk I/O and network bandwidth utilization.

In the configuration with seven standard nodes after parallelization, there was significant disk read and write activity, as well as high network usage, due to the communication between the nodes. This setup required frequent data exchanges and access to external resources, which resulted in increased network packets and bytes, as well as notable disk operations.

Switching to the configuration with four nodes, each with double the resources, the cluster experienced reduced disk activity and lower network usage. This change was likely due to the increased internal resources per node, which allowed more data processing to occur locally without relying on network communication or disk I/O as frequently.

In the final configuration, using a single node with eightfold resources, disk activity and network usage were minimal. With all the computation happening within a single, resource-rich node, there was little need for inter-node communication or external storage access. The internal resources were sufficient to handle the workload efficiently, which minimized the need for disk reads and writes, as well as network communication.

Overall, the experiments demonstrated that increasing internal resources per node reduced reliance on disk I/O and network bandwidth. This result suggests that investing in fewer but more powerful nodes could be an efficient approach for handling similar workloads in a cloud environment.

### Task 1d iii)

In our labs, we used Spark on single computers, storing data on those computers locally. However, Spark is made for handling large amounts of data across multiple computers using systems like HDFS (Hadoop Distributed File System). Spark's real strength is in distributing work across many machines within clusters through parallelization. Parallelization, which we used for this coursework, partitions the data, and processes these partitions across multiple nodes and virtual machines.

### Task 2c


*Figure 5: Graphs showing different metrics for the cluster containing 7 worker nodes, 1 virtual machine and maximal SSD size (100). These are the metrics for the for the cluster before caching is applied.*


*Figure 6: Graphs showing different metrics for the cluster containing 7 worker nodes, 1 virtual machine and maximal SSD size (100). These are the metrics for the for the cluster after caching is applied.*

Caching is a necessity in Spark as it helps to enhance efficiency and speed up the processing. Without caching, every time the program calls an RDD, which stands for Resilient Distributed Dataset, it has to recompute the data which is time-consuming and takes up resources. Once caching is enabled in Spark, the RDD is saved in memory and will be rapidly used in the future. This is especially helpful for frequently used data objects and where an RDD is used more than once, as it eliminates hundreds of disks reads as well as network transfers.

Using caching reduces disk I/O, network traffic, and overall time consumed in processing data. Because it minimizes the need for reading and writing from the disk, data is already accessible to Spark, which uses the system resources more effectively. Caching plays crucial roles in tasks that require extensive computation and where significant amounts of data are used. Thus, programs execute better and smoother.

After caching was applied, the system's efficiency noticeably improved. Disk operations and bytes declined, indicating reduced reliance on disk I/O as data was stored in memory. This minimized repetitive disk access, enhancing overall performance. CPU utilization became more stable after caching, showing a higher and more consistent usage. This suggests that the system made better use of its processing power by avoiding unnecessary computations. Network traffic also decreased, with fewer bytes and packets exchanged. This indicates less communication between nodes, as cached data was used within nodes rather than relying on network exchanges. The decrease in network traffic also aligned with improved efficiency in both CPU and disk operations.

In summary, caching enhanced the system's efficiency. By storing data in memory, the system reduced unnecessary disk and network activity while stabilizing CPU utilization. This led to more efficient resource usage and a smoother, faster performance.

## Task 2d

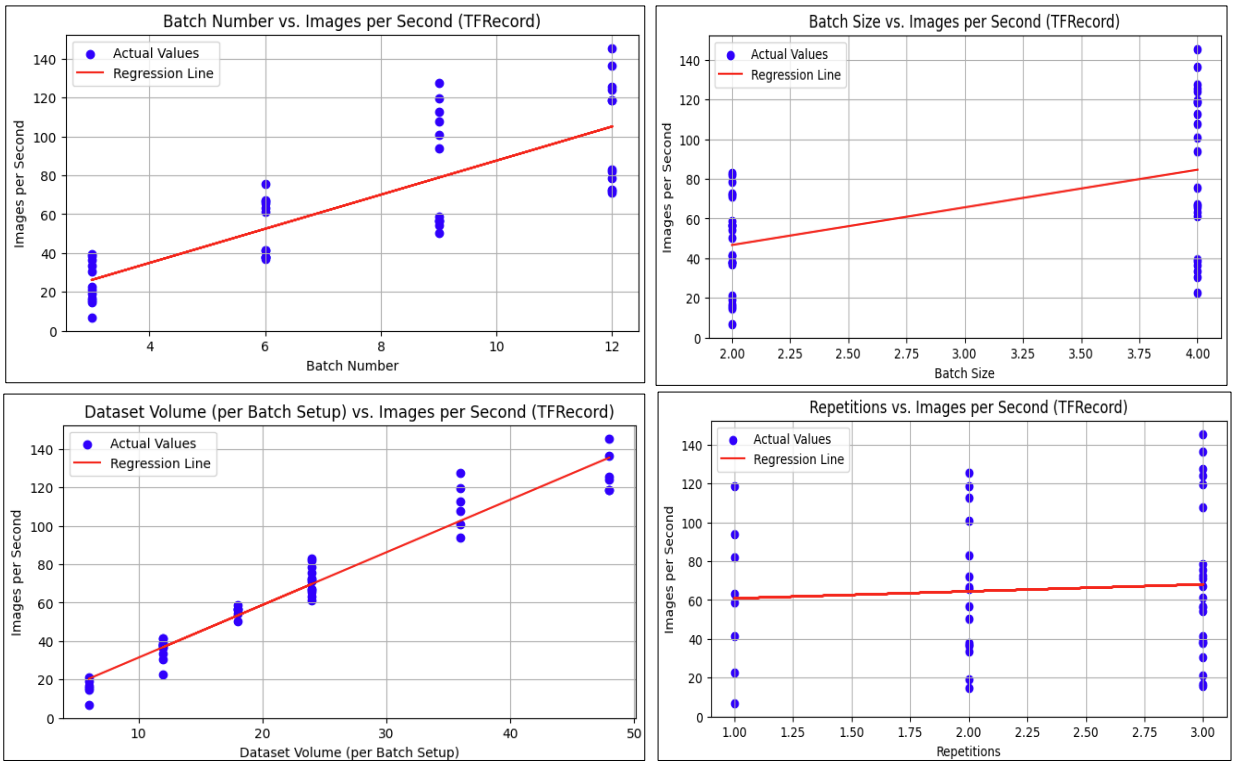



*Figure 7: Batch Number/Batch Size/Dataset Volume/Repetition vs Image per Second for the TFRecord Dataset.*

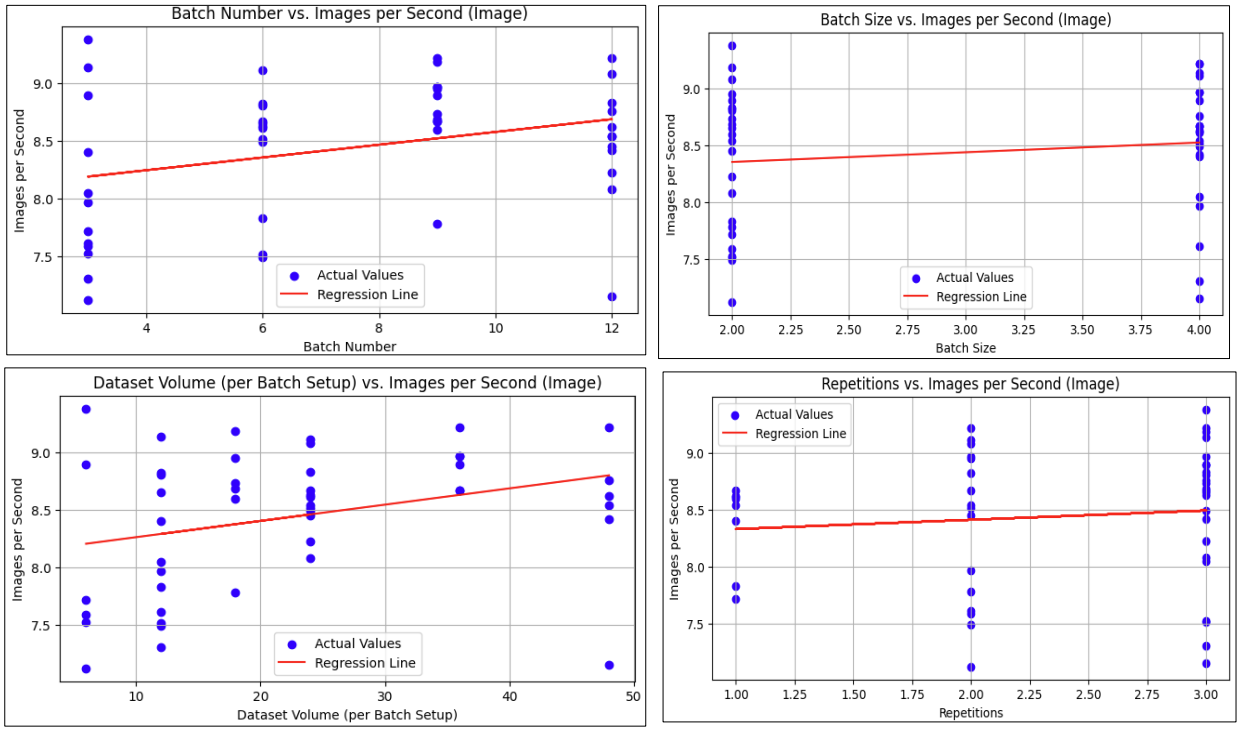



*Figure 8: Batch Number/Batch Size/Dataset Volume/Repetition vs Image per Second for the Image Dataset.*

### Implications for Large-Scale Machine Learning

Efficient data formats like TFRecord are crucial for large-scale machine learning, particularly in cloud environments where the data is stored across various locations. These formats reduce latency, as supported by cloud latency figures, and improve throughput, as seen in the graphs. For machine learning applications, using efficient data formats can significantly improve training speeds and scalability, especially when data is stored across multiple cloud regions.

### Single Machine vs Cloud

In cloud environments, data and computing are distributed, leading to variable performance as shown in the graphs. Single machines, on the other hand, offer more a consistent but less scalable performance. The cloud setup encounters challenges like network latency and distributed storage issues, while single machines generally provide steadier processing at the cost of limited scalability.

### Throughput and Disk Resources

Cloud providers link throughput to disk space to balance storage and computing power across different cloud servers. The changing performance in the graphs shows how crucial balanced resources are because disk problems can delay the process significantly, especially with big data. Improving disk space helps keep performance steady, aligning storage with processing needs.

### Reflections on Parallel Speeds

Parallel speed tests often highlight network bandwidth and disk I/O as potential bottlenecks. The varied disk and network activity in the graphs underscore the need for optimizing parallel reads to avoid congestion. When handling large datasets or working across multiple nodes, addressing these bottlenecks is crucial for maintaining high performance and effective resource utilization.

### Linear Modelling Considerations

Linear modelling aligns with the observed trends, showing performance improvements with increasing batch sizes and volumes. However, practical challenges like variable network conditions and disk I/O contention may not be fully observed in linear models. Balancing theoretical predictions with practical insights is key for accurate performance predictions and efficient system design in cloud-based applications.

## Task 3

## Contextualize

In this report, the focus was on evaluating different cloud computing configurations, particularly within the context of big data analytics using Spark. This paper introduces a method for predicting optimal or near-optimal cloud configurations using Bayesian Optimization. This aligns with the coursework, where various cluster configurations and the impacts of parallelization and caching on system performance, were examined.

The framework in this paper is designed to identify the best cloud configurations for recurring big data analytics jobs. This is directly relevant to the work done, where the aim was to optimize resource utilization and enhance system efficiency. The paper's emphasis on modelling performance and using Bayesian Optimization relates with the testing of different cluster sizes and resource allocations, where different metrics such as CPU utilization, disk I/O, and network traffic, were analysed.

CherryPick applies to cases where multiple configurations were evaluated to identify the optimal setup for a given task. This is consistent with the experiments in the coursework, where different cluster configurations were tested, such as varying the number of machines and adjusting resource allocations. CherryPick’s methodology is beneficial when there is a need to balance performance and cost in a range of configurations, as encountered during the coding part of the coursework.

In summary, the concepts and techniques used in the paper are applicable to this task where multiple potential configurations and performance balancing with resource utilization is crucial. The Bayesian Optimization approach used in this paper is valuable in scenarios where the testing of multiple configurations is impractical, meaning that the focus has to shift to predictive modelling to find the optimal options.

## Strategize

When planning for different types of applications, like batch processing and streaming, it's important to consider their specific needs. This paper offers a great way to optimize cloud setups using Bayesian Optimization, which can aid resource allocation in various situations.

For batch processing, where large amounts of data are handled regularly, the goal is to boost throughput while keeping costs low. The method in the paper works well for this type of work,

where tasks are usually repetitive and predictable. A good strategy here is to identify typical workloads that represent common needs, allowing CherryPick to make performance models that can accurately predict the best setups. The main focus should be on limiting costs since batch jobs often run repeatedly, leading to significant savings over time.

Streaming, on the other hand, involves constant data input and real-time analysis, where low latency is important. In such cases, keeping the system responsive while handling changing workloads is key. A strategy for streaming would include dynamic resource allocation to adapt to changing needs. CherryPick's flexible modelling can identify setups that handle variable workloads effectively, while also supporting automatic scaling. Since low latency is often crucial, CherryPick's ability to find near-optimal setups can help keep the system responsive while balancing resource use.

In both batch and stream processing scenarios, the key is to balance resource usage, cost, and performance. The CherryPick approach, with its focus on predictive optimization, offers valuable insights into how to achieve this balance across different application scenarios.

**References**

https://www.heavy.ai/technical-glossary/parallel-computing#:~:text=Parallel%20computing%20refers%20to%20the,part%20of%20an%20overall%20algorithm.

https://cloud.google.com/storage/docs/caching#:~:text=The%20Cache%2DControl%20metadata%20for,that%20are%20not%20publicly%20readable.

https://people.irisa.fr/Davide.Frey/wp-content/uploads/2018/02/cherrypick.pdf

https://medium.com/criteo-engineering/the-trade-offs-of-large-scale-machine-learning-71ad0cf7469f#:~:text=Focusing%20on%20the%20data%20and,the%20data%20and%20the%20task.

**Google Collab Link:**
https://colab.research.google.com/drive/115HmflZLRHM4b8QxaYVv3AEII-RcY9Kk#scrollTo=AYLysXrJm-9M

**Word Count:** 1833