Parallelizing Data Processing with Spark on Google Cloud Report

Owner: Ling-Yun, Huang

Task 1: Write TFRecord files to the cloud with Spark

1d) Optimisation, experiments, and discussion

(i) Improve parallelisation: demonstrate the difference in cluster utilisation before and after the change based on different parameter values with screenshots from Google Cloud and measure the difference in the processing time.

With default setting, the RDD runs on only two partitions across the cluster. By adjusting the partitioning parameter in the code to 16, we can increase parallelism and make more efficient use of cluster resources. Figure 1 shows the impact of RDD partitioning on cluster utilisation and performance metrics before and after increasing the number of partitions from 2 to 16 on an 8-machine cluster.

When using only 2 partitions, the cluster usage is limited and only 2 machines are effectively used to process tasks. In contrast, increasing the partitions to 16 allows all 8 machines to be fully utilized, thereby increasing the overall efficiency of the cluster. In addition, YARN memory utilization shows more efficient resource allocation through higher partitions. Furthermore, the total processing time is reduced to less than half when using 16 partitions compared to using 2 partitions, highlighting the benefits of increased parallelism and efficient resource utilization.

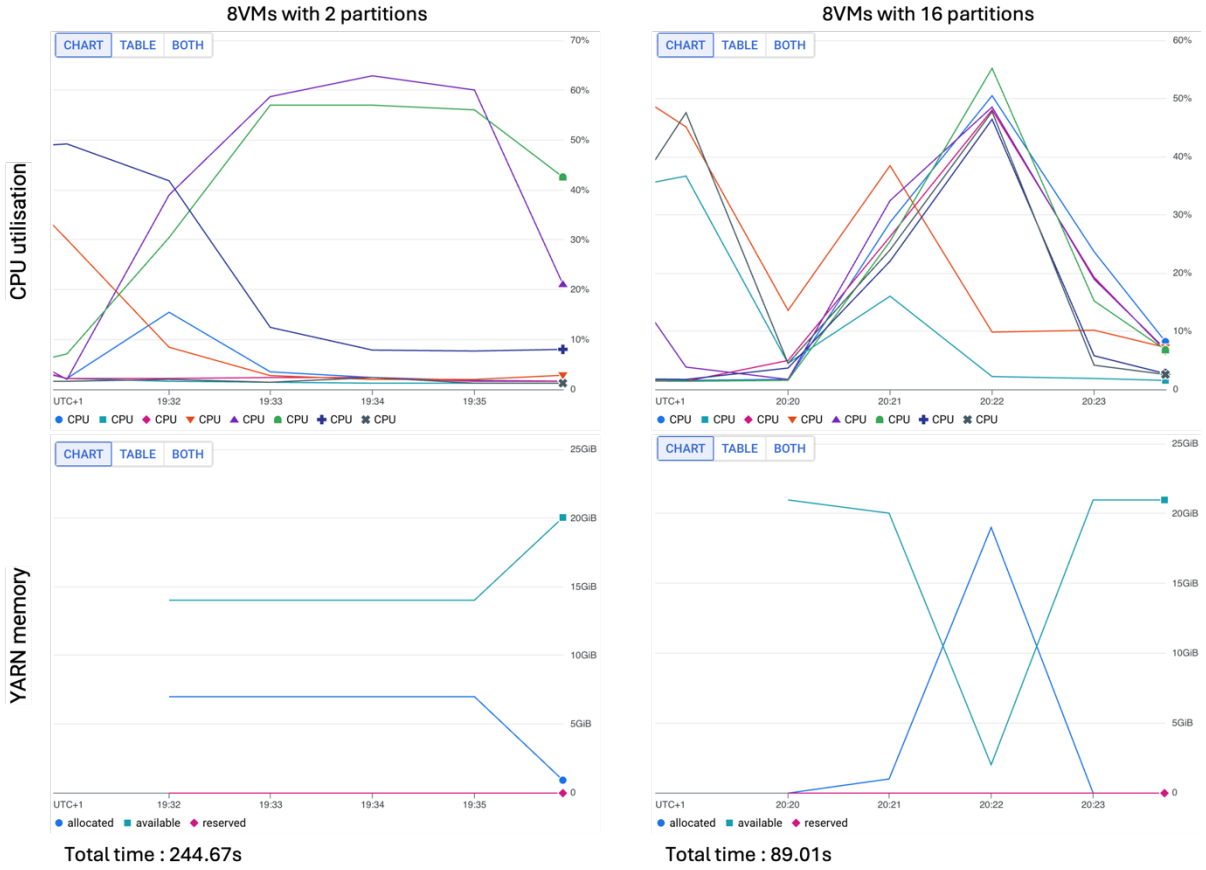

Figure 1. Cluster Utilization Comparison: RDD Partition Sizes (2 vs. 16)

(ii) **Experiment with cluster configurations:** in addition to the experiments above (using 8 VMs), test your program with 4 machines with double the resources each (2 vCPUs, memory, disk) and 1 machine with eightfold resources. Discuss the results in terms of disk I/O and network bandwidth allocation in the cloud.

In this section, we explored three distinct cluster configurations, including a single machine with 8 vCPUs, 4 virtual machines (VMs) with 2 vCPUs each, and 8 VMs with 1 vCPU each. Figure 2 shows disk I/O and network bandwidth allocation in the cloud environment.

Despite differences in the number of VMs and resource allocation, the total network bandwidth usage across all configurations shows a consistent pattern. This means that network traffic in a cloud configuration is more affected by workload characteristics than by the number or type of VMs.

Examining the disk I/O performance across these configurations highlighted interesting trends. The single machine with 8 vCPUs showed lower disk read speeds, peaking below 1 MiB/s. In contrast, the 4 VMs with 2 vCPUs showed significantly higher disk read rates of approximately 7 MiB/s, while 8 VMs with 1 vCPU showed the highest peak disk read rates. The disk read rate is slightly over 10 MiB/s.

Further analysis of the detailed disk read rates per VM revealed that each VM within the 4 and 8 machine setups displayed similar disk read rates, peaking around 2 MiB/s per VM, while the peak value set by a single machine is 0.8 MiB/s. Additionally, the 4 VMs configuration showed more stable disk I/O performance across all machines compared to other configurations.

Figure 2. Disk I/O and Network Bandwidth Utilization Across Cluster Configurations

Task 2: Parallelising the speed test with Spark in the cloud

2c) Improve efficiency: explain the reasons for the change of RDD.cache() and demonstrate and interpret its effect.

Using "RDD.cache()" into Spark job effectively prevents redundant computations by storing intermediate results. To effectively utilize this technology, it is important to apply caching to RDDs that are repeatedly used, such as parameter combinations, speed tests with image or TFRecord files, and raw data for parameter values and results. By caching these RDDs, there is no longer necessary to collect the results (as done in part(iii)) before creating an RDD with all results for each parameter to ensure alignment.

Figure 3 shows the impact of using caching versus not using it on CPU utilisation and Network bytes in a cloud environment. The overall patterns are similar between these two jobs. However, in the final minute, the performance is completely different. When not using caching, CPU utilisation increases by around 10% and network bytes spike significantly up to 30MiB/s. In contrast, when using caching, there is only a slight increase in both metrices. Additionally, overall execution time benefits from caching, being approximately 2% faster than without caching.

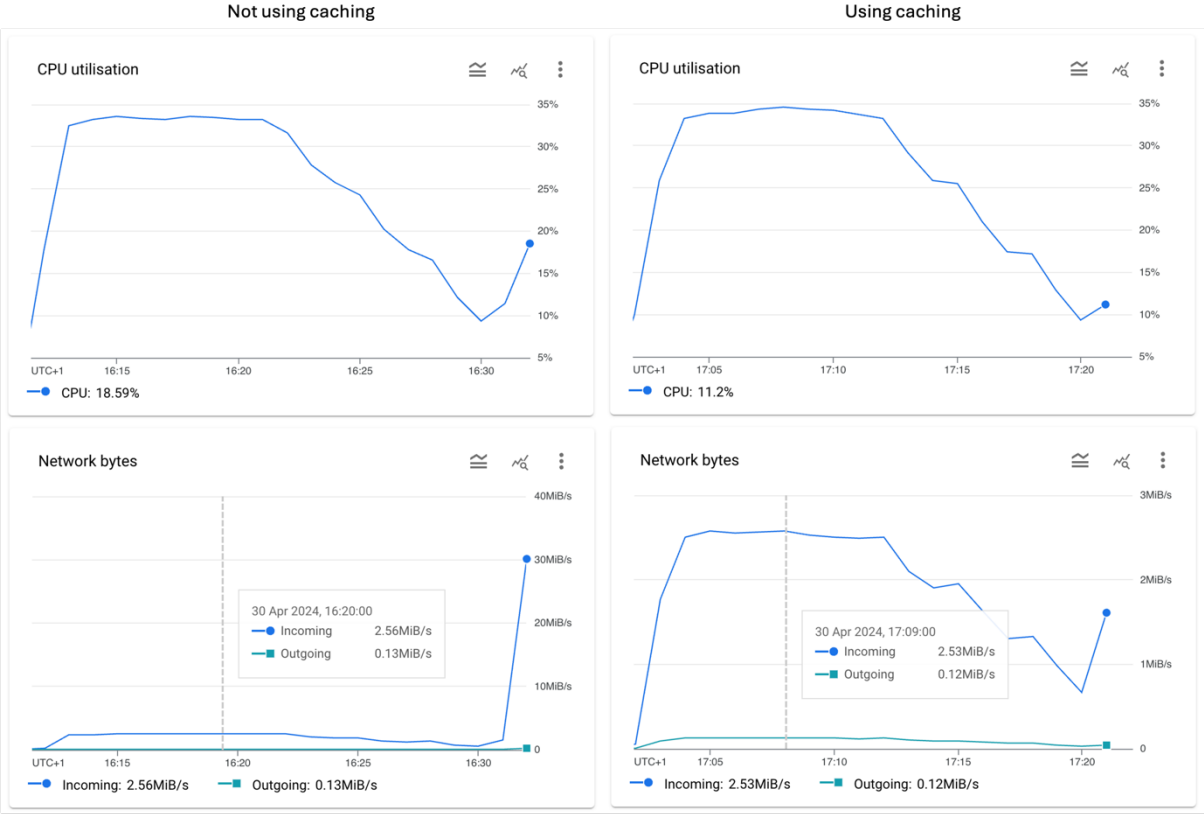

Figure 3. CPU utilisation and Network bytes of using caching or not using it

2d) Retrieve, analyse and discuss the output

Run the tests over a wide range of different parameters and list the results in a table. Perform a linear regression (e.g. using scikit-learn) over the values for each parameter and for the two cases (reading from image files/reading TFRecord files). List a table with the output and interpret the results in terms of the effects of overall. Also, plot the output values, the averages per parameter value and the regression lines for each parameter and for the product of batch_size and batch_number

Discuss the implications of this result for applications like large-scale machine learning. Keep in mind that cloud data may be stored in distant physical locations. Use the numbers provided in the PDF latency-numbers document available on Moodle or here for your arguments. How is the observed behaviour similar or different from what you'd expect from a single machine? Why would cloud providers tie throughput to capacity of disk resources?

By parallelising the speed test we are making assumptions about the limits of the bucket reading speeds. See here for more information. Discuss, what we need to consider in speed tests in parallel on the cloud, which bottlenecks we might be identifying, and how this relates to your results.

Discuss to what extent linear modelling reflects the effects we are observing. Discuss what could be expected from a theoretical perspective and what can be useful in practice.

In this section, we analyse the results of speed tests and discuss the output. Table 1 details the linear regression analyses results for each parameter across different cases. Additionally, Figure 4 shows the output values, average values, and regression lines for each parameter.

Table 1: The linear regression results for speed tests

Cases	Parameter	Intercept	Coefficient	p-value
Image files	Batch size	8.536	0.054	<.0001
Image files	Batch number	8.564	0.034	<.0001
Image files	Repetition	8.977	0.017	0.5412
Image files	Batch size x Batch number	8.663	0.003	<.0001
TFRecord files	Batch size	152.878	23.671	<.0001
TFRecord files	Batch number	158.615	15.355	<.0001
TFRecord files	Repetition	360.272	2.256	0.8118
TFRecord files	Batch size x Batch number	177.532	1.550	<.0001

The linear regression results show that increasing batch size and batch number improves data reading efficiency for both file types. However, repeating the data reading process does not contribute to speed improvement. Notably, reading from TFRecord files is 40 times faster than reading from image files, as evidenced by the regression intercepts related to repetition.

These findings highlight the importance of efficient data retrieval in large-scale machine learning application. With the provided latency numbers in file[1], we also know the impact of storage location on reading speed. For example, a road trip within same datacentre (0.5 ms) is 40 times faster than reading 1 MB sequentially from disk (20 ms). Given these latency differences, prioritising optimised data storage and retrieval methods is critical in large-scale machine learning applications.

Due to local data access and processing, running speed tests on a single computer may done more predictable results. Under this setting, increasing the batch size and batch number can improve data reading efficiency. However, resource constraints on a single machine may limit the test's ability to

effectively handle large-scale machine learning applications. The reason why cloud providers tie throughput to capacity of disk resources is to ensure efficient use of resources and prevent performance bottlenecks.

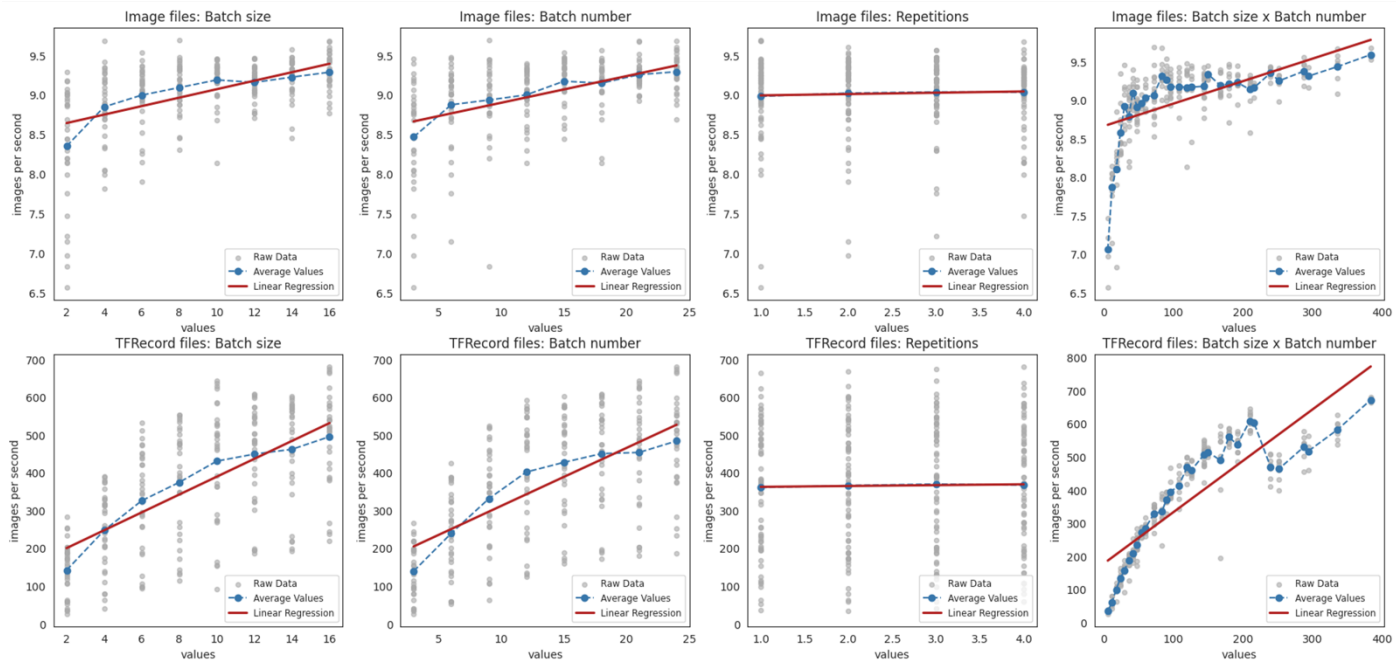

Figure 4. The linear regression results with raw data and average values for speed tests

When performing speed tests in parallel on the cloud, we assume that cloud server can efficiently handle multiple read requests simultaneously. However, several important considerations in the provided document[2] must be considered, such as load redistribution time, bucket IO capacity limits, and object key indexing. In the results, the observation that higher values of batch size and batch number, along with its product, the average values are lower than the regression line, indicates the possibility of encountering bottlenecks or restrictions imposed by the cloud servers. However, this possibility needs to be further investigated to help find the specific limitations affecting performance.

Linear modelling is a practical tool for speed test assessments; however, it also implies that parameter values influence image reading speed linearly, which may ignore or overlook the non-linear relationships. Theoretically, it might be helpful by employing polynomial regression to compare its fitting with linear regression or applying multiple linear regression with parameter combinations as factors could provide insights into fit and combined effects. These methods might offer more complex interactions beyond linear assumptions.

Task 3: Discussion in context

In this task we refer an idea that is introduced in this paper: - Alipourfard, O., Liu, H. H., Chen, J., Venkataraman, S., Yu, M., & Zhang, M. (2017). Cherrypick: Adaptively unearthing the best cloud configurations for big data analytics.. In USENIX NSDI 17 (pp. 469-482).

Alipourfard et al (2017) introduce the prediction an optimal or near-optimal cloud configuration for a given compute task.

3a) **Contextualise:** relate the previous tasks and the results to this concept. (It is not necessary to work through the full details of the paper, focus just on the main ideas). To what extent and under what conditions do the concepts and techniques in the paper apply to the task in this coursework?

It is not an easy task to choose the right cloud configurations for the big data analytics jobs, and selecting the right settings can significantly impact costs and performance. The CherryPick system, introduced in the provided paper[3], is specifically designed to address this challenge by leveraging Bayesian Optimisation to minimize costs, ensure high performance, and reduce search overhead when determining the optimal cloud settings for specific task.

In the coursework, the focus is on optimising Spark jobs in a cloud environment to increase efficiency and maximise performance. Tasks such as adjusting RDD partitions, applying caching strategies, and experimenting with different cluster configurations are align closely with the approach in the paper. These actions aim to identify optimal setups to specific tasks, improving overall job performance and resource utilisation.

Moreover, by observing disk I/O and network bandwidth across different cloud configuration settings, a deeper understanding is gained of how these settings impact job execution. This understanding allows for more informed decision when applying different configurations to optimise job performance further.

Another shared concept is the developing of performance models to predict the impact of processing time and resource utilisation. In Task 2, these models are used to estimate how changes in parameter values and data types affect job performance. In the paper, CherryPick employs performance modelling to estimate costs and runtime for varying cloud configurations, which provides valuable insights into the optimisation process.

Overall, the coursework aims to effectively optimise cloud configurations and develop performance models for big data analytics tasks. This is related to the concepts and techniques discussed in CherryPick to achieve optimal cloud settings and performance outcomes for applications.

3b) **Strategise:** define - as far as possible - concrete strategies for different application scenarios (batch, stream) and discuss the general relationship with the concepts above.

For batch processing, the goal is to efficiently process large volumes of data that do not require real-time data updates. In this scenario, the following strategies can be considered.

1. Explore different setups of computing resources within a cluster, such as varying the number of machines, CPU allocations, and memory allocations per machine. Similar to tasks we had done in the Task 1. By exploring these settings can help us to identify the most efficient configurations and reduce the processing time significantly for specific batch jobs.
2. Implement caching techniques to avoid redundant computations. By storing results in memory, subsequent computations can retrieve cached data instead of recomputing it. When

processing large datasets, caching frequently used data or intermediate results can speed up processing times and ensure efficient resource utilisation.

3. Create models that predict how changes in settings impact processing time and resource utilisation. These performance models can simulate different settings and predict which setups will achieve optimal processing time and cost efficiency for specific batch jobs. By leveraging these models, better decisions can be made to optimise batch processing workflows and allocate resources effectively.
4. Apply the CherryPick system to guide the selection of configurations based on the prior selections. This system leverages Bayesian Optimisation and learns from pervious choices to recommend the next optimal configuration. This can help to reduce the search costs related to choose the optimal configuration, leading to more efficient and effective batch processing workflows.

For stream processing, the goal is to analyse the continuous data streams in real-time or near real-time to generate immediate insights and support timely decision-making. In this scenario, the following strategies can be considered.

1. Adjust computing resources based on data volumes and processing demands. For example, during peak periods when more data is generated, can ensure jobs are completed quickly. On the other hand, the amount of data will decrease during off-peak, using fewer machines or reducing resources can avoid resource waste and reduce costs.
2. Stream processing often requires timely insights. Minimising processing delays becomes crucial. Optimising the location of data storage and analysis can significantly reduce processing delays. Placing data closer to the processing units or using high-speed network connections can minimise latency for the processing workflows.
3. Regularly monitor performance metrics and data patterns to make configuration adjustments. For example, analyse new data on a daily or weekly basis and evaluate system performance to optimise configurations. This ensures that the stream processing system remains efficient and effective over time.

In batch processing, strategies such as exploring varying computing setups, employing caching to avoid redundant computations, predicting performance impacts, and using CherryPick for setup choices directly relate to optimising how resources are used and making processing more efficient. Similarly, for stream processing, adjusting resources based on data volume, minimising delays through optimal data placement, and regularly monitoring performance align with concepts and techniques discussed in coursework and the research paper. These strategies demonstrate practical applications for optimising both batch and stream processing in big data analytics workflows.

References

[1] P. Norvig, “Latency numbers every programmer should know.” Accessed: May 03, 2024. [Online]. Available: https://gist.github.com/hellerbarde/2843375

[2] “Request rate and access distribution guidelines,” Google Cloud. Accessed: May 03, 2024. [Online]. Available: https://cloud.google.com/storage/docs/request-rate

[3] O. Alipourfard, H. H. Liu, J. Chen, S. Venkataraman, M. Yu, and M. Zhang, “CherryPick: Adaptively Unearthing the Best Cloud Configurations for Big Data Analytics,” in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, Boston, MA: USENIX Association, Mar. 2017, pp. 469–482. [Online]. Available: https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/alipourfard