Big Data Analysis using Cloud Technology

Matthew Parker

Abstract

This report analyses the effectiveness, scalability, and limitations of a cloud computing method for analysing data from the Large Hadron Collider. The method involved splitting the provided datasets into chunks which were then distributed equally between worker nodes. This aims at getting the workers to finish their individual analyses at similar times. Potential bottlenecks were carefully considered and minimised as much as possible, with slow compute time of workers and build-up of message queues between master and worker nodes determined as the most likely limitations depending on the scenario. With more time, Kubernetes should be implemented to ensure tolerance of failed nodes.

Introduction

The Large Hadron Collider (LHC) in Switzerland collects vast amounts of data, as the outputs of tens of millions of collisions per second between subatomic particles are recorded. This data must be analysed to help physicists better understand our universe on the smallest scale. One of the key breakthroughs from the LHC is in its discovery of the Higgs Boson in 2012. This is a subatomic particle that was predicted to exist but cannot be measured directly due to it only existing for very short periods. The proof of its existence helped confirm a decades-old theory about the standard model – a model of the elementary particles of the universe, shown in Figure 1.

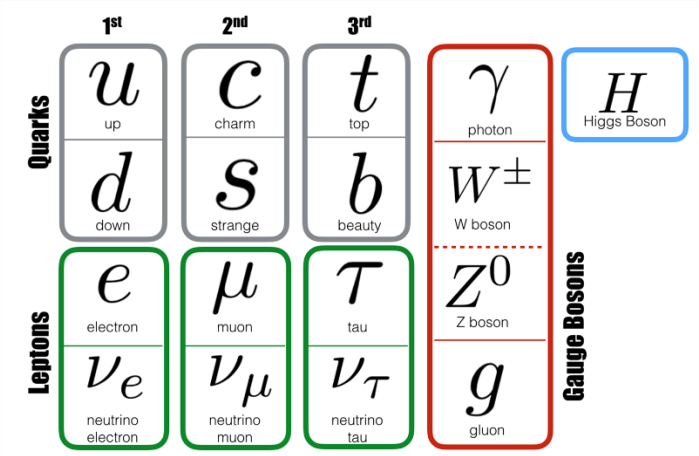

Figure 1: The elementary particles that make up the current version of the Standard Model.[1]

To discover the Higgs Boson, a huge amount of data had to be analysed. The LHC produces 15 petabytes of data each year, which requires a huge amount of compute power to analyse.[2] Cloud computing is a method that can be used to analyse a large amount of data across a network, with each node of a network used in analysis. The Worldwide LHC Computing Grid (WLCG) is the cloud computing network used for analysis of data from the LHC. It combines approximately 1.4 million computer cores from nearly 180 sites across 42 countries.[3] This allows storage of a huge amount of data (1.5 exabytes), with scientists able to access this data from wherever they are in the world.

Some of the data produced by the LHC is available for public use, with the ATLAS Open Datasets providing small example versions of the data produced by the LHC. These small datasets can be used to produce effective methods to analyse the LHC data. However, to produce useful methods, the analysis should be able to scale effectively. This means the method must be able to handle much greater volumes of data than those provided in the ATLAS datasets, including when much greater compute power is available.

The following report explores a method to analyse one of the Higgs decay channels using data from the ATLAS Open Datasets.[4] It is a relatively small dataset, but careful consideration was given to how a larger volume of data would be handled.

Analysis Method

The Higgs Boson decays via the decay channel shown in figure 2, to a Z and Z* boson, which in turn decay to four *l* leptons. This decay of the Higgs Boson occurs rapidly so its existence cannot be measured directly.

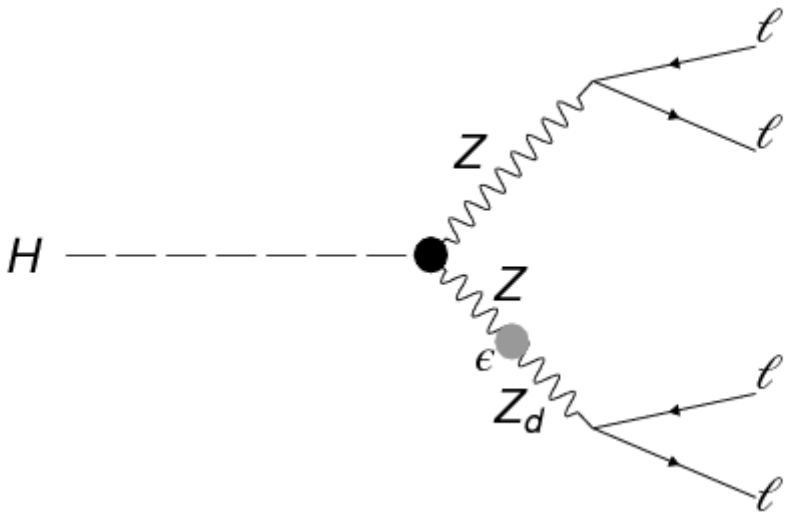

Figure 2: Decay channel of a Higgs Boson to Z, Z* bosons and four *l* leptons.[5]

The products of the decay can be detected to determine the existence of the Higgs Boson. However, this is not the only decay process that can produce these outputs, with four other processes producing the same outputs. These background processes mean that simple detection of these decay products does not guarantee the existence of the Higgs Boson.

The main background signal is from the decay of ZZ* bosons to four *l* leptons. This background is irreducible, as it has the same components as the decay of the Higgs Boson. To differentiate between this decay and the decay of the Higgs Boson, the invariant mass of the lepton products must be calculated. The invariant mass of the lepton products must equal that of the Higgs Boson, while other background processes would have different invariant masses.

The data provided by the ATLAS Open Datasets contains real data collected from the LHC, referred to as “data”. It also contains Monte Carlo (MC) simulations of the relevant background

processes, referred to as “Background ZZ*”, “Background Z, ttbar”, and “Signal”. The real data must be compared to the MC data to determine the statistical significance of any differences between the real and simulated data.

A method involving cloud computing was developed to run the required analysis. The scripts were configured automatically using Docker Compose. The network was configured with one producer/master script which distributes the data to multiple works nodes, ensuring parallel processing. The number of worker nodes can vary depending on how many are available to the user. The master node splits the data up into chunks and distributes the chunks to each worker. Each worker then analyses their chunk of data and sends the outputs back to the master node.

Once all chunks have been distributed to the workers, the master then receives the analysed data from the workers. This data is sent in a separate “worker to master” queue, which is accessed by the master node. On receiving the data from the workers, it combines the outputs into an overall dataset containing all the relevant data. This overall dataset is then used in the final analysis. Figure 3 shows a plot of the analysed data, and how the real LHC data compares to the MC data.

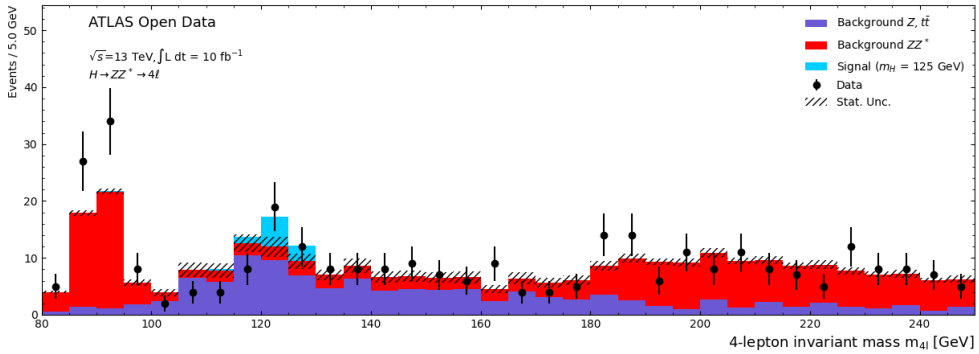

Figure 3: Expected output of the analysis. The data spikes at approximately 125 GeV and 90 GeV are due to the extra invariant mass measurements of the Higgs Boson and its decay products (Z,Z* bosons).

Cloud Computing Method

Figure 4 shows the quantity of data in each sample from the ATLAS Open Dataset. It can clearly be seen that the “Background ZZ*” and the “Signal” samples have significantly more data than the “Background Z ttbar” and “data” samples. Splitting the data up by sending one subsample to each nodes would be an inefficient method, as less time is needed to process the smaller samples compared to the “background ZZ*” sample.

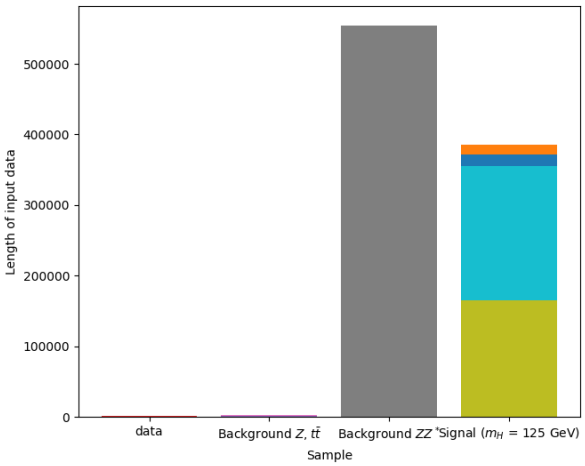

Figure 4: Quantity of data in each sample and subsample.

Instead, the distribution method used in this analysis involved splitting up each subsample into as many chunks as there are number of workers. Therefore, if there were three workers, each subsample would be split into thirds, ensuring each worker has the same volume of input data to analyse from each sample. This means the workers will finish the analysis of each subsample at approximately the same time.

The order of the samples was set with the largest samples analysed first. This was intentional so the smaller samples would be distributed to workers that finished analysing the largest samples earliest. This enables any workers who analyse the largest samples rapidly to then take on larger proportions of the smaller samples.

With this method, problems may arise if you have a much larger number of containers than the quantity of data, for example, if you had the dataset shown in figure 4, but with 300 available workers. If the above method were used as is, there would be an ineffective distribution of data between each worker, with each worker given very small chunks of data. This would likely increase the total run time of the analysis, as the time taken to send and receive the messages would be the limiting factor, rather than the time of each worker to analyse the data. To counteract this, a minimum chunk size was set as 10,000 bits of data. This means each worker has a significant chunk of data to iterate through and analyse. The number 10,000 is arbitrary and may not be the optimal minimum chunk size. However, it was chosen to ensure each worker has a significant amount of data to analyse and can easily be varied to improve efficiency.

Other methods for splitting the data into chunks were also contemplated. It was debated whether each set of data should be split into twice the number of workers, rather than matching the number of workers. This would allow more flexibility in redistributing the data chunks if workers take significantly different lengths of time to analyse their data chunks.

It was also considered that as the samples are iterated through, the number of chunks should be increased – the first sample split into three chunks, the second sample split into chunks of 4, third sample split into 5, and so on. This would again allow more flexible redistribution, as if one worker finished significantly before another worker, this worker would then receive more of the later chunks of data.

However, neither of these methods were used as these methods significantly increase the number of messages, without an obvious increase to efficiency.

Results and discussion

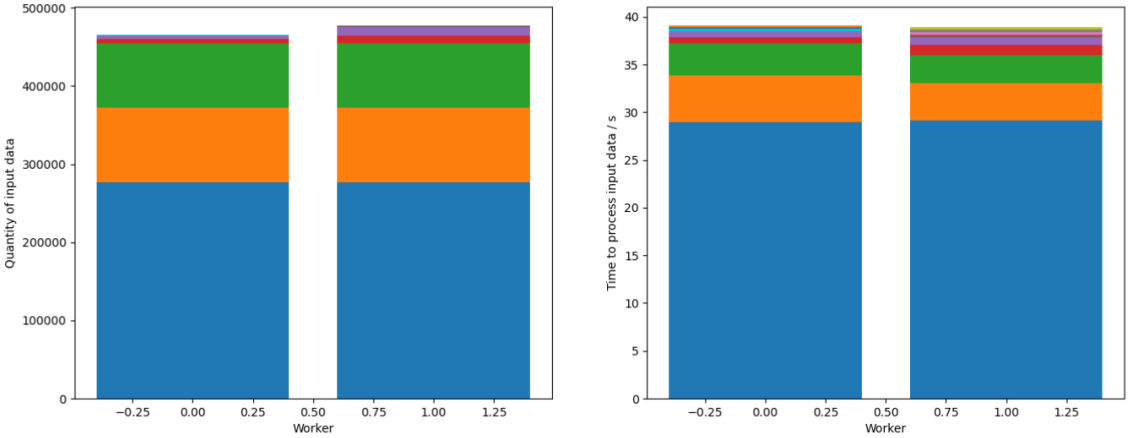

Figure 5: Quantity of data and time taken to analyse each sample distributed to two worker nodes.

Figure 5 shows an example distribution of the data between two workers. The left-hand plot shows the raw quantity of data sent to each worker, while the right-hand plot shows the time taken to analyse each set of data. It can clearly be seen that the workers are sent slightly different quantities of data but end up taking a similar length of time to analyse the total results. This shows the effectiveness of the method, as "worker 1" received more data while "worker 0" was taking longer to analyse subsamples represented in orange and green.

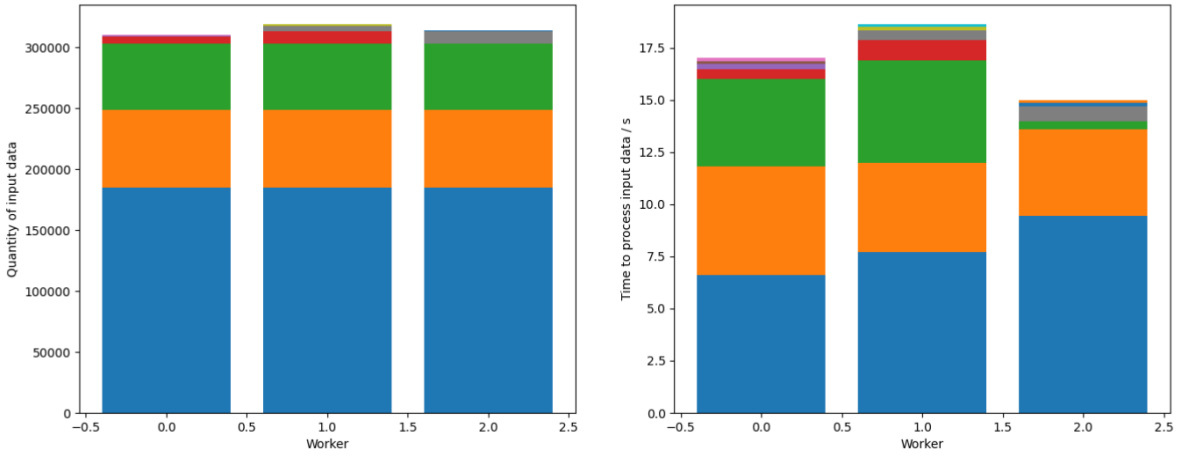

Figure 6: Quantity of data and time taken to analyse each sample distributed to three worker nodes.

Figure 6 shows an example distribution of the data sent between three workers. It is important to note that the timing in the right-hand plot may not be very accurate because they are only

recording the time taken between the worker node receiving the data and sending the outputs. This means it does not include the time spent in the queue between nodes.

Furthermore, with the small datasets used in this experiment, the sample data is not always distributed effectively between workers. Rapid analysis of the final samples can leave both workers free, causing one to receive multiple consecutive chunks of data while the other does nothing. This can lead to slight discrepancies in the time recorded for the workers to analyse the data but would not be a problem with larger datasets as longer processing times would allow better message distribution.

Limitations

Overall, this method should be an effective, scalable way of analysing the available data. However, there are some limitations with this method that must be considered.

First of all, the system must determine the number of worker nodes. This is done in the master script by counting the current number of connections between the master and worker nodes. Sometimes, the workers are not yet activated when the initial count is done. This means the larger samples which are analysed first are at risk of being analysed using fewer workers than could be available. However, this risk has been minimised by pausing the master node for 1 second before counting the number of workers, to allow them extra time to activate. The number of worker nodes is also updated before analysing each sample to ensure any workers that are activated late will be used.

The main bottleneck for the method used is the time taken for the analysis of each sample from each worker. This will particularly be the case as the volume of data per worker increases, as the data analysis time will take longer. However, this bottleneck is unavoidable, as the workers must analyse the data at some point, and the data is already as parallel as possible.

If there are both a large number of containers and a large volume of data, the system should handle the large volume of data effectively, with it distributed evenly between workers. In this scenario, the bottleneck would likely by the “worker to master” queue. This is because the master container must distribute all the data to the workers before it can start combining the results sent back. In this situation, the workers may finish their tasks rapidly. This means they would send lots of the results back before the master is done distributing, causing a large backlog of messages in the “worker to master” queue. This queue will not diminish until all the data has been published by the master node. On the other hand, this issue is minimised by the setting of the minimum chunk size to 10,000.

This bottleneck at the “worker to master” queue is difficult to eliminate. However, the chosen method does limit the maximum number of messages to the number of workers multiplied by the number of subsamples. To fully utilise a large number of available workers there is always going to be a large number of messages, so this bottleneck is hard to avoid. One method considered to avoid this problem would be to have the analysis done in a separate “analyser” node. This node would consume the outputs from the worker and combine them into the final data to be analysed. However, this method was not used as it was believed the master node should return the results, as this is the node that the user has access to.

With more time, Kubernetes would've been implemented. This would allow better control of nodes that fail. This is currently a major inefficiency in the project, as the Docker Compose system is set to abort when one container finishes. This generally works fine on a small scale as it is unlikely a node will break and is added as a convenience to allow the system to close once the master script has finished the analysis. However, if a worker node does break then the whole system terminates. This could be fixed with the use of Kubernetes. Furthermore, there may be a command to terminate a Docker Compose system once a specific script is finished, but it was not found.

Conclusion

Overall, this project provides an effective and scalable method to analyse the provided data. The main limitations and bottlenecks are generally unavoidable, as they would likely either be due to the time taken for the workers to analyse their chunks of data, or a backlog in the "worker to master" queue. When each worker receives a large quantity of data, analysis time would likely be the bottleneck. On the other hand, when the workers each receive a small quantity of data the bottleneck would likely be caused by the message queue, in particular if there were a large number of workers. To mitigate this, a minimum chunk size of 10,000 was included to ensure each worker receives a suitable amount of data to analyse.

To conclude, this project provides an effective, scalable method for analysis. However, with more time, Kubernetes should be implemented to ensure system reliability and resource management.

Appendix

The GitHub repository for this project can be found at:
https://github.com/M4ffff/LHC_cloud_computing

References

[1] Standard Model, Universität Zurich, https://www.physik.uzh.ch/en/groups/serra/outreach/StandardModel.html (accessed 19/03/2025).
[2] Facts and figures about the LHC, CERN, https://home.cern/resources/faqs/facts-and-figures-about-lhc (accessed 19/03/2025).
[3] The Worldwide LHC Computing Grid (WLCG), CERN, https://home.cern/science/computing/grid (accessed 19/03/2025).
[4] 13 TeV ATLAS Open Data physics analysis examples, ATLAS Open Data, https://opendata.atlas.cern/docs/physics/intro (accessed 19/03/2025).
[5] ATLAS Collaboration, *JHEP*, 2018, **6**, 166.