Written Report

Name: Michael Luke Battle

Student Number: 20072560

As the quantity of data in the world continues to grow exponentially, so does the requirement for new technologies to effectively visualise it. Terapixel images are an emerging visualisation format that allows for images to be rendered at various levels, enabling the user to zoom in and out of the image whilst retaining clarity. This has the potential for a variety of applications, ranging from astronomy with the WorldWide Telescope, to breast cancer diagnosis (AAS WorldWide Telescope, 2021; Liu et al, 2017). However, with larger datasets comes higher computational cost, which hinders traditional computing methods in their ability to render terapixel images. An alternative is to use parallel computing along with public cloud platforms such as Microsoft Azure to substantially increase computing power.

The focus of this report is a project representing the third phase in a three-part process to create a terapixel image, depicting the city of Newcastle upon Tyne and its environmental data. Where the initial two phases of the wider project involved creating the necessary supercomputer architecture in the public cloud, followed by the production of the city visualisation itself, the third phase – and the aim of this project – is to rigorously evaluate cloud supercomputing in creating such terapixel images. This objective is particularly prudent, as a key requirement of the visualisation is that it should support daily updates, so that if it is presented to stakeholders the environmental data portrayed is up to date.

The data pipeline constructed as the basis of my response was done so using CRISP-DM methodology (Chapman et al, 2000). This approach provides a framework for data analysis. I found it particularly appropriate for this analysis where data understanding was key to performing the required preparation steps. In this aspect, CRISP-DM describes a thorough analysis template that begins with high level assessments before drilling down further into the data, whilst addressing potential issues such as duplicates or null values in the data. The CRISP-DM approach partitions the overall analysis pipeline into 6 components. These are business understanding, data understanding, data preparation, modelling, evaluation and deployment. Each of these phases is broken down into several clear steps. For instance, the steps involved in the data preparation phase are “select data”, “clean data”, “construct data”, “integrate data” then “format data”. For this project, where data preparation was key to performing a thorough analysis of the data, I found this to be a far more helpful guide than other data analysis frameworks, such as Microsoft’s Team Data Science Process (Microsoft, 2021). In contrast to CRISP-DM, this process has phases “business understanding”, “data acquisition and understanding”, “Modelling”, “Deployment” and “Customer Acceptance”. Both the lack of a clear data preparation section, as well as the clear sub-processes, render this framework less suitable for the analysis of this particular data than CRISP-DM.

By applying CRISP-DM methodology, I initially set out the key objectives to rigorously evaluate the rendering process. These were to “evaluate individual events performed as part of the total rendering process, explore the relationship between different performance metrics, identify GPU cards and tiles with unusual behavior and assess the efficiency of the task scheduling process”. After this, I performed some high-level exploratory data analysis on each individual data set. This involved assessments on the dimensions, structure, null values and duplicates in the data. The “data understanding” section of my report was concluded with some histograms of the numeric data in the “gpu” data set. In the data preparation phase of my analysis, I converted the “timestamp” variable to datetime format and used this to create the variables “runtime” and “task_no” - with the latter

providing the key to combining the “gpu” data with “application.checkpoints”. With this data, I performed my analysis in the “modelling” section. Here, I made several key findings. First, I found that the “Tiling” event made no contribution to the overall rendering time for each tile. This is because the “Tiling” event and “Uploading” event are started simultaneously for each task, and the “Uploading” event is always longer than “Tiling”, thereby driving the overall render time. With regards to the “Uploading” event, I also found that on around 80% of the virtual machines, the first two tasks had runtimes of approximately ten times the average runtime for the “Uploading” event. As these had the potential to skew the results in any subsequent analysis using the “runtime” variable, this was taken into consideration for the remainder of the analysis. During this section of the analysis, I also showed that the total render time is dominated by the “Render” event.

The next sections of the analysis concerned the interplay of the “runtime” variable with various performance metrics from the “gpu” data. I found positive correlations between runtime and GPU temperature, and runtime and the GPU power draw. Whilst the presence of the first two tasks skewed both correlations, they significantly skewed the runtime and temperature result. Further analysis showed that during the first two tasks, not only was the runtime significantly higher than average (due to the previously established long “uploading” events), but at the beginning of the total rendering process the GPU cores were colder than average. This meant that in some GPU cores these variables had a weak negative correlation when the first two tasks were included. Upon their removal, this correlation was moderately positive. Following this analysis, I examined the variation of the performance metrics across different tiles in the terapixel image. I did this both quantitatively, as well as visually with reference to the final image. This revealed that portions of the image with more shadows took longer to render, and conversely, that large areas of one colour took less time.

The final parts of the analysis involved finding the GPU’s, by serial number, that were perpetually slower than others. I presented the slowest 15 GPUs and proved that the inclusion of observations from the first two tasks did not skew this result significantly. The task scheduling process was then analysed, by finding the percentage of snapshots in the “gpu” data with % memory and GPU usage at 0, from the beginning of the first task to the end of the final one. This showed that the GPU cores are not being used for an average of 25% of the total rendering process and are awaiting a new task during this time. Many of the insights were then deployed using an interactive shiny app, that allows the user to explore the data and verify some of the findings from this project.

I have undertaken a rigorous analysis of the data and met all objectives set out during the “Business Understanding” section of my analysis. I have evaluated many different facets of each data set rigorously, and when I have found some interesting behaviour regarding a particular part of the data, I have endeavoured to prove that it is the case universally. Moreover, I have used both plots and numeric calculations to reinforce any assertions made during the analysis. Regarding the numeric calculations, I have implemented analytical techniques recommended by Hoeffler (2015) when analysing performance of parallel computing systems. These put emphasis on assessing the normality of results, where appropriate, to discern whether parametric or non-parametric techniques should be used.

Having said this, there are some weaknesses in my analysis and areas that could have been explored further. One such area is in making further use of the “gpu” data. In the data preparation phase of my analysis, I took the average of the performance metrics for each task where the % memory usage was non-zero and this was assigned to the corresponding task. Whilst this allowed for a less computationally costly data preparation, it also led to significant information loss from the resulting analysis. Making further use of the granular “gpu” data, may have allowed for the performance metrics to be used in an analysis of the individual events as part of the overall rendering process.

Furthermore, in contrast to most of the analysis, before taking the arithmetic mean of the performance metrics for each task, the normality of the data was assumed and not verified. Despite this being a computationally intense task, it would have made my resulting analysis more valid, and perhaps provided additional insights. Moreover, I could have analysed the two distinct groups of GPU cores found in my analysis of the GPU cards that were perpetually slow. Other issues with this analysis stemmed from the suitability of the CRISP-DM framework, in that there were several sections of the analysis template that were not applicable to this problem. Even in the data understanding and data preparation phases, which were the most relevant, aspects of the "gpu" data could not be fully understood without undergoing some basic data preparation with the "timestamp" variable. Moreover, much of the framework concerns the data mining aspects of a project, which were not applicable in this case, as the data was readily available. Additionally, as this project was primarily focused on exploration of the data, the "Modelling" section was not useful.

A key takeaway from this project is that there is significant room to improve the task scheduling process for cloud rendering. Whilst the terapixel image was rendered in a time frame that was suitable for the project requirements, by increasing the task scheduling process efficiency this could be performed using less GPU cores and so save money on the IaaS service. Furthermore, the rendering process could also be improved at the beginning, with a particular focus on the "Uploading" event and how on 80% of machines it takes approximately ten times longer to execute the first two tasks, than at any other point in the rendering process.

Overall, I found this process challenging, yet rewarding. Upon reflection, I have significantly improved my knowledge of R, the CRISP-DM methodology and of the terapixel rendering process using cloud supercomputing. I have become much more fluent in the dplyr and ggplot2 packages and find I am performing my data analysis without looking up functions as frequently as I had previously. I also improved my understanding of shiny and used reactivity within my app to promote user interaction with the data. I did this not only using the selectInput function, but also the sliderInput. The most challenging part of the project was most certainly trying to combine all three data sets into one, however I am pleased with my solution to this problem. Additionally, it was truly understanding that the "gpu" data consisted of snapshots from every two seconds for each virtual machine that enabled me to join the data. Ensuring that I fully comprehend the data and its meaning before fully beginning my analysis is an important lesson that I will make sure to apply to all future projects. I found most of my analysis straightforward, but there were some areas of it that were more challenging than others. Once such area was understanding the interplay between different performance metrics, which on the face of it, seemed a simple problem but considering the difference in environments between virtual machines made it far more difficult.

References

AAS WorldWide Telescope. (2021). *AAS Worldwide Telescope*. [online] Available at: <https://worldwidetelescope.org/webclient/> [Accessed 22 January 2021].

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C. and Wirth, R. (2000). *CRISP-DM 1.0 Step-by-step data mining guides*

Hoefler, T, and Belli, R. (2015) *Scientific benchmarking of parallel computing systems: twelve ways to tell the masses when reporting performance results.* In Proceedings of the international conference for high performance computing, networking, storage and analysis, p. 73. ACM

Liu, Y., Gadepalli, K., Norouzi, M., Dahl, G. E., Kohlberger, T., Boyko, A., Venugopalan, S., Timofeev, A., Nelson, P. Q., Corrado, G. S., Hipp, J. D., Peng, L. and Stumpe, M. C. (2017). *Detecting cancer metastases on gigapixel pathology images*. arxiv:1703.02442.

Microsoft. (2021). *Team Data Science Process Documentation*. [online] Docs.microsoft.com. Available at: <https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/> [Accessed 22 January 2021].