CSC8634 TeraScope Analysis”

Prathik Pradeep (220238007)

2023-01-12

1 Comparing Event Names with their Render Time and GPU Metrics

1.1 BUSINESS UNDERSTANDING

1.1.1 Business Objectives

Evaluating the GPU metrics and time taken for each event would provide valuable insight into the performance of the GPUs. The comparison of each GPU metric for an event name could give an insight into which event is computationally taxing. The time taken to complete each event also provides an insight into the duration of the sustained work load for each event. Being able to pin point the events that are the most and least computational taxing could help to configure the GPUs to function at their optimum level during these events. This is turn could help reduce the costs associated with computational power.

The performance of a GPU could be defined by metrics like GPU Power Draw, GPU Temperature, GPU Utilisation and GPU Memory Utilisation [1].

The success criteria for this business objective would be to identify a trend between GPU metrics and event names as well as time taken for each event.

1.1.2 Data Mining Goals

Based on the business objectives, looking for any relationship between event name and the GPU metrics is vital. The next step could be to find the time taken for each event and if there is any relationship between the GPU metrics and the time taken for an event.

1.1.3 Produce Project Plan

To carry out to analysis to achieve the business objective, the data would first need to be understood. This would mean to understand and analyse all the data to find all the relevant variables useful for this particular analysis. Upon which the data would need to prepared as required, which generally includes cleaning, creating additional variables from the existing variables, merging the data if needed and formatting the data based on the requirements. The next step would be to carry out the required analysis and produce observations based on the results of the analysis. The results of the analysis would then be compared to the business objective to check whether it has been answered.

The tools and techniques being used here are RStudio libraries like ggplot, dplyr, lubridate, ProjectTemplate and RMarkdown.

1.2 DATA PREPERATION

1.2.1 Selecting Data

The data being selected are the application_checkpoints and gpu data sets. These data sets are selected as they contain the necessary variables to solve the business objectives.

1.2.2 Format Data

Changing the format of the timestamp in the application_checkpoints and gpu into the correct date time format, that is, "YYYY-MM-DD HH:MM:SS.MSS". Converting the variable from character to date time datatype is also completed in this step.

The data type of the GPU Serial variable in the gpu data set would need to converted from numeric to character.

1.2.3 Integrate Data

Next, the two data sets need to be joined together. However, it can be observed here that the timestamp columns in the data sets are not the same, therefore matching the timestamps while joining the data sets would not be possible. Therefore, joining it on hostname and the nearest timestamp would be the best approach to join the two data sets. This however, is more of a rough method and not 100% accurate. The combined data set is called gpu_checkpoints.

After the data sets are joined, the columns timestamp and time are removed. The column "i.timestamp" is then renamed to timestamp. A new column called grouped_id is then created, which numbers each event type per event name per task ID. This means the START and STOP event type for each event has the same number.

Next the average of the gpu metrics is computed and put in place for each event type while the time difference is computed into a new column called "eventTime_in_ms" which is calculated in milliseconds. this replaces the two records for each event type and converts it into a single record. The unnecessary columns like grouped_id, START and STOP and removed.

1.3 DATA UNDERSTANDING

1.3.1 Describing and Exploring the Data

Below, a glimpse of the gpu_checkpoints data set can be found. This is how the data looks after the joins are completed. This also gives a glimpse of how application_checkpoints and gpu data sets have been joined.

```
## Rows: 330,200
## Columns: 11
## $ hostname        <chr> "0745914f4de046078517041d70b22fe7000001", "0745914f4de~
## $ gpuSerial       <chr> "325117173029", "325117173029", "325117173029", "32511~
## $ gpuUUID         <chr> "GPU-f3d43c08-eeb5-9a86-8783-3a3d27e418dc", "GPU-f3d43~
## $ powerDrawWatt   <dbl> 43.080, 31.630, 54.345, 42.895, 54.345, 52.370, 51.930~
## $ gpuTempC        <dbl> 46.0, 46.0, 46.0, 46.0, 46.0, 42.0, 41.0, 42.5, 41.5, ~
## $ gpuUtilPerc     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ gpuMemUtilPerc  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ eventName       <chr> "Render", "Saving Config", "Tiling", "TotalRender", "U~
## $ jobId           <chr> "1024-lvl12-7e026be3-5fd0-48ee-b7d1-abd61f747705", "10~
## $ taskId          <chr> "00004e77-304c-4fbd-88a1-1346ef947567", "00004e77-304c~
## $ eventTime_in_ms <dbl> 30239, 3, 974, 31405, 1163, 39721, 2, 994, 40730, 1007~
```

The unique number of key values, that is, hostname, gpuSerial, gpuUUID, eventName, jobId and taskId can be found in Table 1. This indicates that there are 1024 different hostnames, 1024 different GPU Serials, 1024 different GPU UUIDs, 5 different Event Names, 3 different Job IDs and 65793 different Task IDs. It can also be observed that the the same GPUs are used for each hostname.

Table 1: Unique Values

Summary	Value
Unique Hostnames	1024
Unique GPU Serial	1024
Unique GPU UUID	1024
Unique Event Name	5
Unique Job ID	3
Unique Task ID	65793

1.3.2 Data Quality

After joining the data sets, it can be observed that almost all the gpuUtilPerc and gpuMemUtilPerc values are 0. This could because these values have not been collected for these application checkpoints. These two variables are the most important metrics when it comes to evaluating the performance of a GPU as it indicates how hard the GPU is working to complete a given task [1]. Therefore, this limits the analysis as only powerDrawWatt and gpuTempC can be used to carry out the analysis in this case. Ensuring these values are present could provide very valuable insights.

1.4 ANALYSIS

1.4.1 Relationship Between Event Name vs Average Power Consumed in Watt and Event Name vs Average GPU Temp in C

From Figure 1, it is observed that the event consuming the lowest energy is Saving Config followed by TotalRender, Render, Uploading and Tiling with the highest power consumption. When it comes to the average GPU temperature, the same trend is observed, however the difference in temperature between the different events are marginal.

1.4.2 Relationship Between Event Name and Average Time Duration

From Figure 2, it is observed that the event that has taken the smallest time to execute is Saving Config, followed by Tiling, Uploading, Render and lastly Total Render.

1.5 EVALUATION AND FUTURE IMPLICATIONS

The evaluation being carried out is interesting because the event Total Render is actually the entire task, that is, it is a combination of all the other events. However, as seen in Chapter 1.4.1, the event TotalRender does not consume the most power or generate the most heat. One explanation for this could be due to the fact that the average GPU power draw and GPU temperature is being computed, the low values of saving config that is bringing the values of TotalRender down.

From Chapter 1.4.2, it is observed that the event with the highest event time is TotalRender, this is because it is a combination of all the other events that make up this task. Here, Saving config has a very low value which is almost negligible.

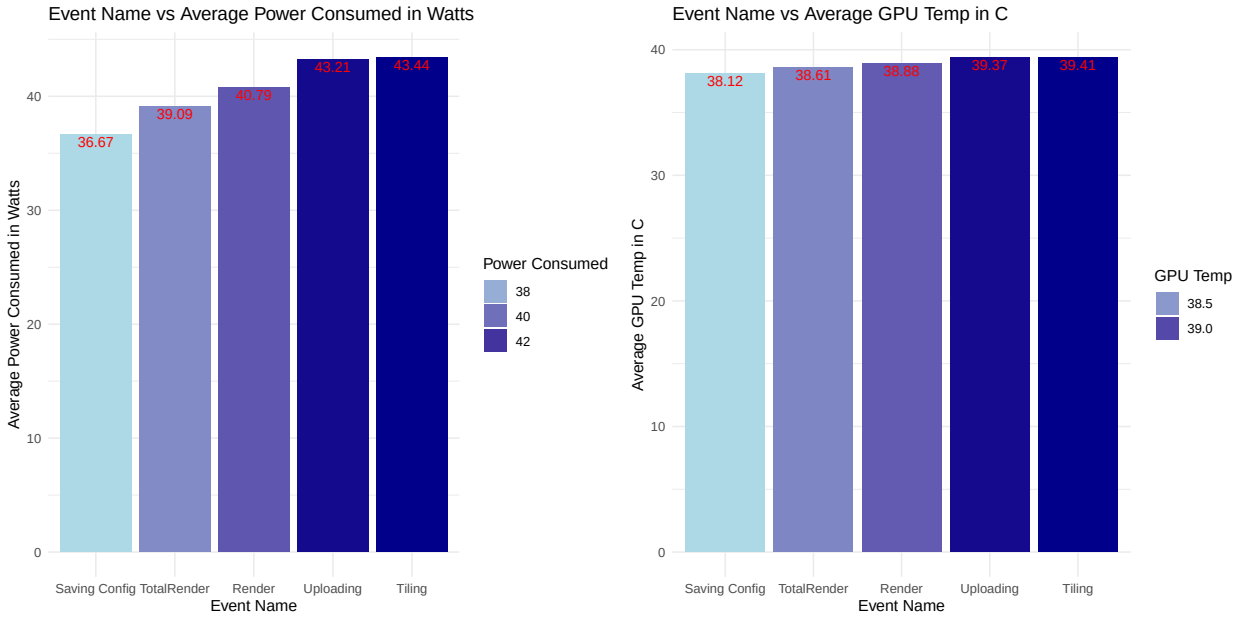

Figure 1: Graphical Summary of Relationship between Event Name vs Average Power Draw in Watt and Event Name vs Average GPU Tmep in C

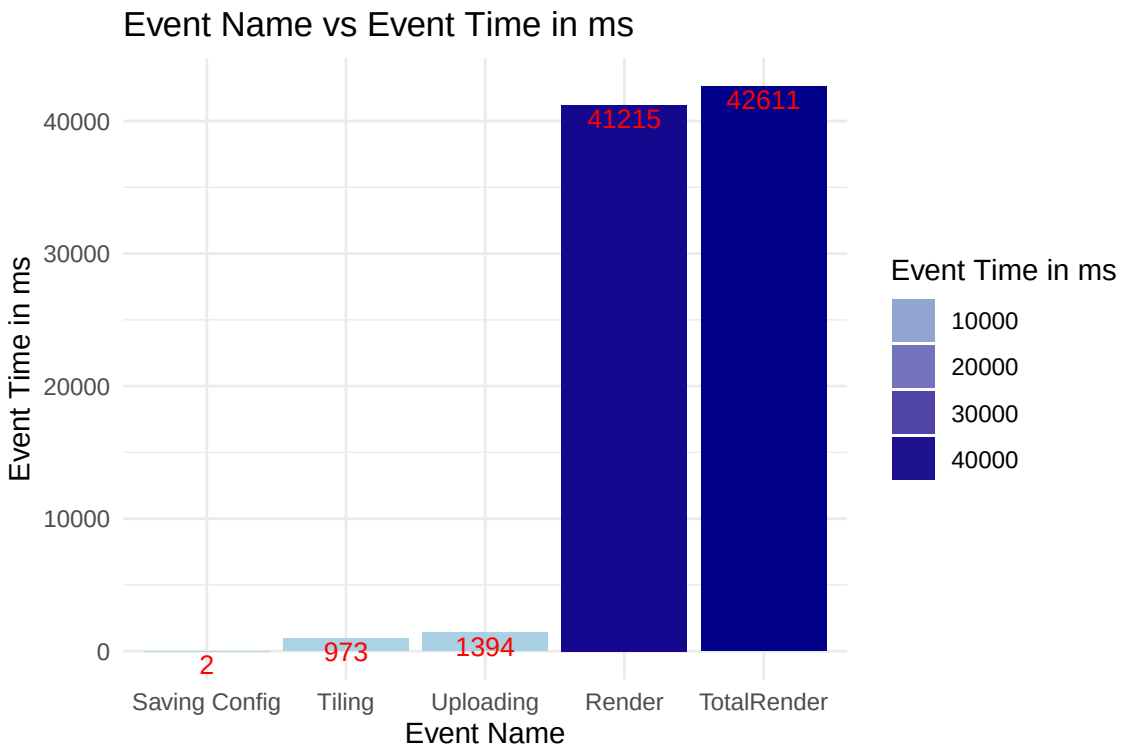

Figure 2: Graphical Summary of Relationship Between Event Name and Average Time Duration

From the analysis conducted in Chapter 1.4, it is evident that the events Render, Uploading and Tiling are the most computationally taxing, whereas, the events Tiling and Uploading do not take very long to execute. This means, the event time render is very computationally taxing for large periods of time, while the events uploading and Tiling are for a much shorter duration. Calibrating the GPUs to maximise performance for the Render event could lead to a more efficient execution of the TotalRender process. This could help bring down the costs associated with power consumption and also with cooling the GPU to maintain it in the optimal window.

Therefore, reflecting on the business objective from Chapter 1.1.1, it can be concluded that the success criteria has been met, useful insights have been provided and hence, the analysis is proven to be successful.

1.6 FUTURE SCOPE

As discussed in Chapter 1.3.2, having access to the GPU Utilisation and the GPU Memory Utilisation data of the GPUs would result in a more in-depth analysis and could indicate how hard the GPU needs to work to complete the required events.

2 Comparing Zoom Level with their Processing Time and GPU Metrics

2.1 BUSINESS UNDERSTANDING

2.1.1 Business Objectives

Evaluating the GPU metrics and time taken for zoom level would provide valuable insight into the configuration of the GPUs towards the zoom levels. The comparison of each GPU metric for a zoom level could give an insight into which level is computationally taxing. The time taken to complete each zoom level also provides an insight into the duration of the sustained work load for each event. Being able to pin point the zoom levels that are the most and least computational taxing could help to configure the GPUs to function at their optimum level during these events. This is turn could help reduce the costs associated with computational power.

As mentioned in Chapter 1, the performance of a GPU could be defined by metrics like GPU Power Draw, GPU Temperature, GPU Utilisation and GPU Memory Utilisation [1].

The success criteria for this business objective would be to identify a trend between GPU metrics and zoom levels as well as time taken for each level of zoom.

2.1.2 Data Mining Goals

Based on the business objectives, looking for any relationship between zoom level and the GPU metrics is vital. The next step could be to find the time taken for each event and if there is any relationship between the GPU metrics and the time taken for an event.

2.1.3 Produce Project Plan

To carry out to analysis to achieve the business objective, the data would first need to be understood. This would mean to understand and analyse all the data to find all the relevant variables useful for this particular analysis. Upon which the data would need to prepared as required, which generally includes cleaning, creating additional variables from the existing variables, merging the data if needed and formatting the data based on the requirements. The next step would be to carry out the required analysis and produce

observations based on the results of the analysis. The results of the analysis would then be compared to the business objective to check whether it has been answered.

The tools and techniques being used here are RStudio libraries like ggplot, dplyr, lubridate, ProjectTemplate and RMarkdown.

2.2 DATA PREPERATION

2.2.1 Selecting Data

The data being selected are the application_checkpoints, gpu and task_x_y data sets. These data sets are selected as they contain the necessary variables to solve the business objectives.

2.2.2 Format Data

Changing the format of the timestamp in the application_checkpoints and gpu into the correct date time format, i.e. "YYYY-MM-DD HH:MM:SS.MSS". Converting the variable from character to date time datatype is also completed in this step.

The data type of the GPU Serial variable in the gpu data set would need to converted from numeric to character.

2.2.3 Integrate Data

Next, the two data sets need to be joined together. However, it can be observed here that the timestamp columns in the data sets are not the same, therefore matching the timestamps while joining the data sets would not be possible. Therefore, joining it on hostname and the nearest timestamp would be the best approach to join the two data sets. This however, is more of a rough method and not 100% accurate. The combined data set is called gpu_checkpoints.

After the data sets are joined, the columns timestamp and time are removed. The column "i.timestamp" is then renamed to timestamp. A new column called grouped_id is then created, which numbers each event type per event name per task ID. This means the START and STOP event type for each event has the same number.

Next the average of the gpu metrics is computed and put in place for each event type while the time difference is computed into a new column called "eventTime_in_ms" which is calculated in milliseconds. this replaces the two records for each event type and converts it into a single record. The unnecessary columns like grouped_id, START and STOP and removed.

Finally, the task_x_y data set is joined to the gpu_checkpoints data set using a left join on task ID and Job ID. Next, the event name is filtered so only the the total render values are present. This is done so that the comparison can be done for the whole event rather than each individual event that takes place in total render.

2.3 DATA UNDERSTANDING

2.3.1 Describing and Exploring the Data

Below, a glimpse of the gpu_checkpoints_task data set can be found. This is how the data looks after the joins are completed. This also gives a glimpse of how application_checkpoints and gpu data sets have been joined.

```
## Rows: 66,040
## Columns: 14
## $ hostname        <chr> "0745914f4de046078517041d70b22fe7000001", "83ea61ac1ef~
## $ gpuSerial       <chr> "325117173029", "323617020500", "323617021461", "32361~
## $ gpuUUID         <chr> "GPU-f3d43c08-eeb5-9a86-8783-3a3d27e418dc", "GPU-35f4e~
## $ powerDrawWatt   <dbl> 42.895, 44.235, 41.775, 46.145, 40.020, 47.710, 40.365~
## $ gpuTempC        <dbl> 46.0, 41.5, 37.0, 41.0, 42.0, 47.0, 39.0, 38.5, 38.0, ~
## $ gpuUtilPerc     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ gpuMemUtilPerc  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ eventName       <chr> "TotalRender", "TotalRender", "TotalRender", "TotalRen~
## $ jobId           <chr> "1024-lvl12-7e026be3-5fd0-48ee-b7d1-abd61f747705", "10~
## $ taskId          <chr> "00004e77-304c-4fbd-88a1-1346ef947567", "0002afb5-d05e~
## $ eventTime_in_ms <dbl> 31405, 40730, 31899, 40138, 33693, 43938, 42013, 40902~
## $ x               <int> 116, 142, 142, 235, 171, 179, 255, 218, 241, 22, 114, ~
## $ y               <int> 178, 190, 86, 11, 53, 226, 61, 250, 166, 220, 152, 174~
## $ level           <int> 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12~
```

The unique number of key values, i.e., hostname, gpuSerial, gpuUUID, eventName, jobId and taskId can be found in Table 2. This indicates that there are 1024 different hostnames, 1024 different GPU Serials, 1024 different GPU UUIDs, 5 different Event Names, 3 different Job IDs, 65793 different Task IDs and 3 different zoom levels. It can also be observed that the the same GPUs are used for each hostname and each zoom level correlates to one job ID.

Table 2: Unique Values

Summary	Value
Unique Hostnames	1024
Unique GPU Serial	1024
Unique GPU UUID	1024
Unique Event Name	1
Unique Job ID	3
Unique Task ID	65793
Unique Levels	3

2.3.2 Data Quality

After joining the table, it can be observed that almost all the gpuUtilPerc and gpuMemUtilPerc values are 0. This could because these values have not been collected for these application checkpoints. These two variables are the most important metrics when it comes to evaluating the performance of a GPU as it indicates how hard the GPU is working to complete a given task [1]. Therefore, this limits the analysis as only powerDrawWatt and gpuTempC can be used to carry out the analysis in this case. Ensuring these values are present could provide very valuable insights.

The unique number of key values, i.e., hostname, gpuSerial, gpuUUID, eventName, jobId and taskId can be found in Table 2. This indicates that there are 1024 different hostnames, 1024 different GPU Serials, 1024 different GPU UUIDs, 5 different Event Names, 3 different Job IDs, 65793 different Task IDs and 3 different zoom levels. It can also be observed that the the same GPUs are used for each hostname and each zoom level correlates to one job ID.

Table 3: No. of Records of Zoom Level

Summary	Value
No. of Records of Zoom Level 4	1
No. of Records of Zoom Level 8	258
No. of Records of Zoom Level 12	65781

In addition to this, from Table 3, it can be observed that the zoom level 4 has only one record for the TotalRender event and zoom level 8 has 258 records. Whereas, zoom level 12 has 65781 records. This can result in an inaccurate analysis for the zoom levels 4 and 8.

2.4 ANALYSIS

2.4.1 Relationship Between Zoom Level vs Average Power Draw in Watts and Zoom Level vs Average GPU Temperature in C

From Figure 3, it is observed that the zoom level that consumes the least energy is zoom level 4 followed by zoom levels 8 and 12, with zoom level 12 having the highest power consumption. When it comes to comparing the zoom levels to the average GPU temperature, the same trend is observed, however the difference in temperature between the zoom levels 4 and 8 are marginal.

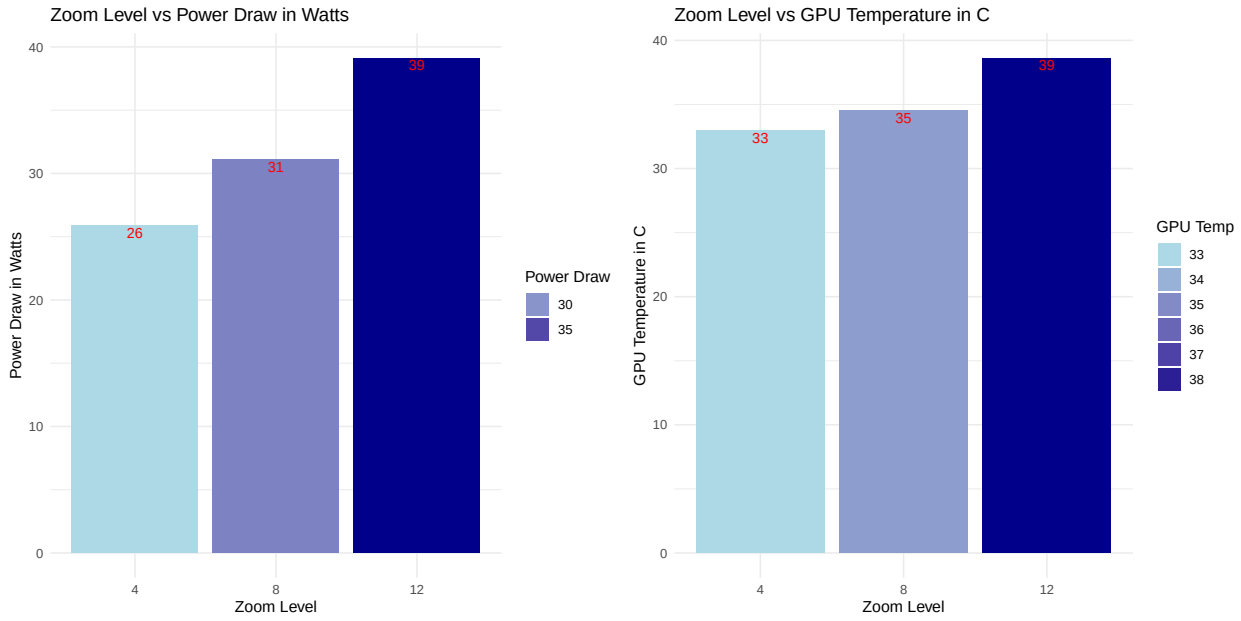

Figure 3: Graphical Summary of Relationship between Zoom Level vs Average Power Draw in Watt and Zoom Level vs Average GPU Temperature in C

2.4.2 Relationship Between Zoom Level vs Render Time in ms

From Figure 4, it is observed that zoom level 12 has the takes the least time, followed by zoom level 8 and lastly zoom level 4.

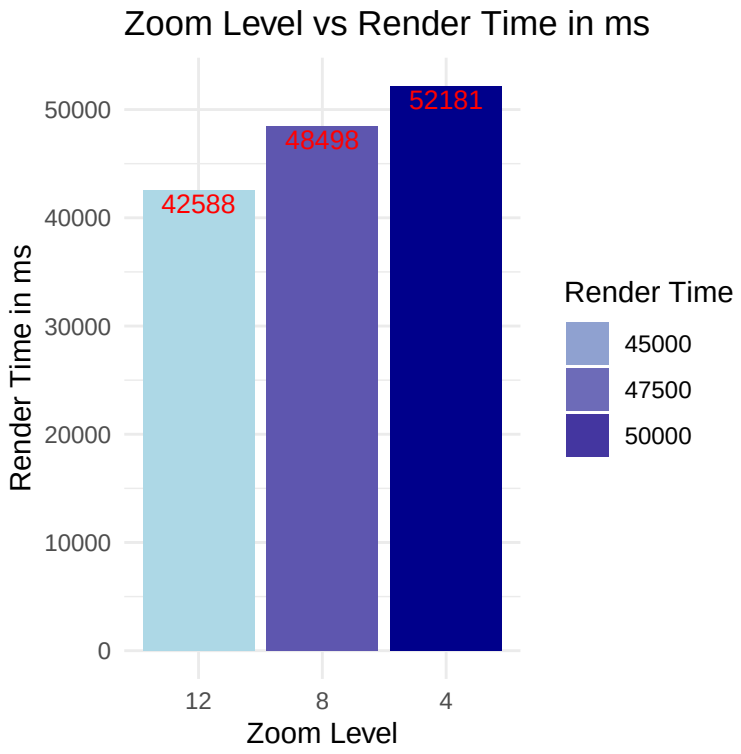

Figure 4: Graphical Summary of Relationship Between Zoom Level vs Render Time in ms

2.5 EVALUATION AND FUTURE IMPLICATIONS

As observed in Chapter 2.4, a similar trend can be observed when comparing the zoom levels to power draw and GPU temperature. Zoom level 12 consumes the most power and generates the highest temperatures. Whereas when comparing the zoom levels to their render times, it is evident that zoom level 12 has the smallest render time whereas zoom level 4 has the longest render time with zoom level 8 in the middle. This could be the result of the GPUs functioning higher on zoom level 12 and functioning at its lowest at zoom level 4. This could mean that the GPUs are already optimised since zoom level 12 has been rendered many more times than when compared to zoom level 8 and 4 by an enormous margin. This could indicate that zoom level 12 is the most common zoom level followed by zoom levels 8 and 4.

As observed in Chapter 2.4.1, when comparing zoom levels and GPU temperatures, the zoom level difference in temperatures between zoom levels 4 and 8 is very small. This could indicate that they are not very computationally taxing.

Therefore, reflecting on the business objective from Chapter 2.1.1, it can be concluded that the success criteria has been met, useful insights have been provided and hence, the analysis is proven to be successful.

2.6 FUTURE SCOPE

As discussed in Chapter 2.3.2, having access to the GPU Utilisation and the GPU Memory Utilisation data of the GPUs would result in a more in-depth analysis and could indicate how hard the GPU needs to work to complete the required events. More information into the data of zoom levels 8 and 12, including more data, could bring more accuracy and understanding to the analysis.

3 Comparing the Different GPUs with their Average Metrics and Average Total Render Time

3.1 BUSINESS UNDERSTANDING

3.1.1 Business Objectives

Drawing a comparison between the average GPU metrics and average total render time can indicate whether the higher GPU metrics actually correlate to a decrease in total render time. This could be one of the most important analyses as it directly compares the GPU metrics and total render time. Being able to pin point which GPU metrics maximize performance and which GPU metrics compromise performance could provide very beneficial insights into the GPUs and their performance metrics.

As mentioned in Chapter 1 and Chapter 2, the performance of a GPU could be defined by metrics like GPU Power Draw, GPU Temperature, GPU Utilisation and GPU Memory Utilisation [1]. However, the ultimate performance metric for GPUs is their render times [2].

The success criteria for this business objective would be to identify a trend between GPU metrics and the total render times.

3.1.2 Data Mining Goals

Based on the business objectives, looking for any relationship between GPU metrics and the total render time is vital. The next step could be to find the average total render time taken for each GPU and if there is any relationship between the GPU metrics and the total render time.

3.1.3 Produce Project Plan

To carry out to analysis to achieve the business objective, the data would first need to be understood. This would mean to understand and analyse all the data to find all the relevant variables useful for this particular analysis. Upon which the data would need to prepared as required, which generally includes cleaning, creating additional variables from the existing variables, merging the data if needed and formatting the data based on the requirements. The next step would be to carry out the required analysis and produce observations based on the results of the analysis. The results of the analysis would then be compared to the business objective to check whether it has been answered.

The tools and techniques being used here are RStudio libraries like ggplot, dplyr, lubridate, ProjectTemplate and RMarkdown.

3.2 DATA PREPERATION

3.2.1 Selecting Data

The data being selected are the application_checkpoints and gpu data sets. These data sets are selected as they contain the necessary variables to solve the business objectives.

3.2.2 Format Data

Changing the format of the timestamp in the application_checkpoints and gpu into the correct date time format, i.e. “YYYY-MM-DD HH:MM:SS.MSS”. Converting the variable from character to date time datatype is also completed in this step. The data from application_checkpoints is sorted grouped, spread and the

time difference for each event is calculated. Next the data set is filtered so it contains on TotalRender data. Finally, the columns “grouped_id”, “START”, “STOP”,“jobId”,“taskId” and”eventName” are removed.

The data type of the GPU Serial variable in the gpu data set would need to converted from numeric to character. The gpu data is then grouped and aggregated on all the GPU metrics so the mean metrics for each GPU is calculated.

3.2.3 Integrate Data

The applications_checkpoints data is then joined onto the gpu data by hostname and all the columns “hostname” and “gpuUUID” are removed.

3.3 DATA UNDERSTANDING

3.3.1 Describing and Exploring the Data

Below, a glimpse of the gpu_checkpoints_task data set can be found. This is how the data looks after the joins are completed. This also gives a glimpse of how application_checkpoints and gpu data sets have been joined.

```
## Rows: 1,024
## Columns: 6
## $ gpuSerial             <chr> "323217056165", "323617042956", "323617021222", ~
## $ powerDrawWatt         <dbl> 95.86895, 91.81369, 82.53780, 86.55858, 94.29239~
## $ gpuTempC              <dbl> 43.52533, 40.99200, 38.04819, 41.57628, 39.74284~
## $ gpuUtilPerc           <dbl> 63.60267, 64.63000, 61.54886, 60.12725, 64.54564~
## $ gpuMemUtilPerc        <dbl> 35.87600, 35.51000, 30.50535, 29.83278, 35.54963~
## $ avg_eventTime_in_secs <dbl> 44.35036, 45.69797, 40.27545, 40.82311, 45.96910~
```

Here it can be observed, for each hostname, a specific gpu is assigned which is referenced by its GPU Serial and GPU UUID.

3.3.2 Data Quality

After joining the table, it can be observed that almost all the necessary values are present however changing a few of the data types and formats is required Apart from this, the quality of the data required for this analysis is good.

3.4 ANALYSIS

3.4.1 Relationship Between Average GPU Metrics vs Average Total Render Time in secs

From Figure 5, looking into the relationship between power draw and total render time, it can be observed that there is a positive correlation between the them. This means, as the power drawn by the GPU increases, the total render time also increases. It can also be observed that two clusters are being formed, one cluster functioning better at lower power draw and one cluster functioning poorly at higher power draw.

Looking into the relationship between GPU temperature and total render time, it can be observed that there is a negative correlation between them. This means, as the GPU temperature increases, the total render time decreases. It can also be observed that two clusters are being formed, one cluster functioning better at lower temperature and one cluster functioning poorly at higher temperatures.

Looking into the relationship between GPU utilisation and total render time, it can be observed that there is a very strong positive correlation between them. This means, as the GPU utilisation increases, the total render time also increases. It can also be observed that two clusters are being formed, one cluster in between 61 and 63 percent GPU utilisation and another cluster in between 64 and 66 percent GPU utilisation.

Looking into the relationship between GPU memory utilisation and total render time, it can be observed that there is a very strong positive correlation between them. This means, as the GPU memory utilisation increases, the total render time also increases. It can also be observed that two clusters are being formed, one cluster in between 30 and 32 percent GPU memory utilisation and another cluster in between 35 and 37 percent GPU memory utilisation.

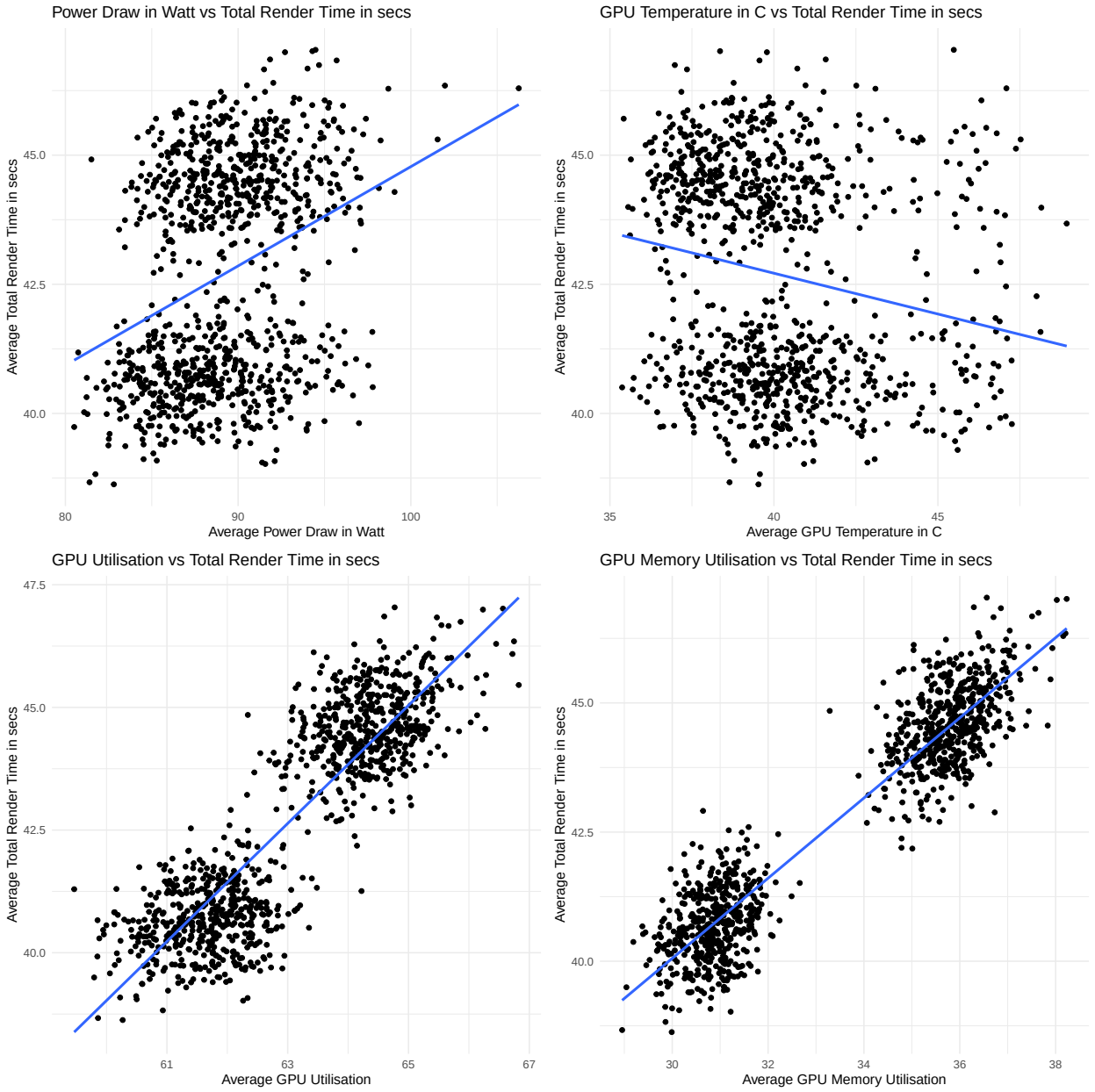

Figure 5: Graphical Summary of Average GPU Metrics vs Average Total Render Time in secs

3.5 EVALUATION AND FUTURE IMPLICATIONS

As observed in Chapter 3.4, clear correlations can be observed when comparing the average GPU metrics with the average total render time. When comparing power draw, GPU utilisation and GPU memory utilisation to the total render time, a positive correlation can be observed. However, when comparing the GPU temperature to the total render time, a negative correlation can be observed.

In this case, a positive correlation means, as the GPU metric value increases, the total render time taken also decreases. Therefore, where a positive correlation is present, it means that as the GPU metrics increases, the total render time decreases.

These trends could indicate that the GPUs that tend to function with a lower power supply, GPU utilisation, GPU memory utilisation and a higher GPU temperature leads to a lower total render time. The way the plots in Figure 5, it is also evident that there are 2 different GPUs used, which is why there are two significant clusters with different performance in each case.

Therefore, reflecting on the business objective from Chapter 3.1.1, it can be concluded that the success criteria has been met, useful insights have been provided and hence, the analysis is proven to be successful.

3.6 FUTURE SCOPE

As discussed in Chapter 3.3.2, having more performance data about the GPUs as well as more GPU processors could help in delivering more detailed analyses and insights. Discovering more metrics to evaluate performance could also prove useful.

4 REFLECTION

4.1 HOW THE TASK WAS APPROACHED

The assignment was challenging and quite large, however, it did not contain excess out of scope learning. The approach of the task was quite straightforward and the CRISP-DM framework was followed with a slight modifications based on the requirements of the assignment.

4.2 TOOLS AND TECHNIQUES USED

4.2.1 CRISP-DM

Learning how to automate an entire process leads to much more ease in reproducibility and reduces clutter. It also makes the execution run a lot smoother.

The CRISP-DM framework is a relatively easy to follow, the layout goes along the general project workflow. The various stages in each step brought out a lot of clarity while working. The evaluation step is one of the best steps as it comments on the results of the modelling step and relates to the business objective which can bring in a lot of insight.

4.2.2 Project Template

Project Template is an extremely useful feature. After using it for the second time, it was very easy and seamless without the learning curve. A very intuitive library to streamline large coding projects. One of the most useful features here is the ability to automate the loading of data frames and saving it in the cache, this made picking up with the project after closing it was extremely fast and simple. Another very helpful feature is the ability to execute the munge directly as the project is loaded. This made it very streamlined

and reduced the effort of running the preprocessing scripts while reopening the project. Another added benefit is that ProjectTemplate forces me to be more organized by saving different types of code in their respective folders.

4.2.3 dplyr

The dplyr library is extremely helpful and quick when it came to wrangling data. Throughout the assignment, a continuous learning cycle takes place with dplyr. Upon completion of the assignment, a realisation that dplyr's applications are endless. A lot of the required wrangling would not have been possible without dplyr.

4.2.4 ggplot

The ggplot library is one of the most useful libraries. Using ggplot effectively generally involves the dplyr library as well. Ggplot can produce much better plots when compared to the base plots produced in R. The entire plot can be customized as per my requirements and is also very logical and straightforward.

4.2.5 git

The use of git for version control was a first, and seemed very daunting and complicated to set up. Loading the data files onto git proved to be a haste. However, once the required setup was completed, committing and pushing the code proved to be very easy. It also provides the added benefit of the code being online and also in different stages of the project cycle. This would be a big if reverting to an older version is required for any particular reason.

4.3 THE TASK

The data in general was not too difficult to understand, however, connecting the data from application_checkpoints and gpu proved to be very tricky as the timestamps in the two tables were not the same. The analysis present in the report brought a lot of insight to the assignment. These would have brought out better insights into the business objectives used.

5 REFERENCES

[1] EXXACT (October 8, 2019). Top 5 Metrics for Evaluating Your Deep Learning Program's GPU Performance. Accessed on January 13, 2023. https://www.exxactcorp.com/blog/Deep-Learning/top-5-metrics-for-evaluating-your-deep-learning-program-s-gpu-performance

[2] Nicolas S. Holliman (February 2019) Petascale Cloud Supercomputing for Terapixel Visualization of a Digital Twin. Accessed on January 16, 2023. https://www.researchgate.net/publication/331086956_Petascale_Cloud_Supercomputing_for_Terapixel_Visualization_of_a_Digital_Twin

[3] Zhixiang Ren (September 2021). AIPerf: Automated machine learning as an AI-HPC benchmark. Accessed on January 12,2023. https://www.researchgate.net/publication/354283459_AIPerf_Automated_machine_learning_as_an_AI-HPC_benchmark

[4] Aggelos Ferikoglou (May 2021). Resource Aware GPU Scheduling in Kubernetes Infrastructure. Accessed on January 12,2023. https://www.researchgate.net/publication/351440179_Resource_Aware_GPU_Scheduling_in_Kubernetes_Infrastructure/figures?lo=1

[5] Tyler Allen (November 2016). Characterizing Power and Performance of GPU Memory Access. Accessed on January 13,2023. https://www.researchgate.net/publication/312966519_Characterizing_Power_and_Performance_of_GPU_Memory_Access