220238007 - CSC8634 TeraScope

Prathik Pradeep

2023-01-24

1 Question 1: Comparing Event Names with their Processing 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 effort could help in configuring the GPUs to function at their accordingly 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, this is because 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, 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.

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 shows how the values in the gpu_checkpoints are stored into the data frames. 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, ~
## $ gpuMemUtilPerc  <dbl> 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 ,i.e. 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.

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 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.

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 Tilling 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 takes 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 carries 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.

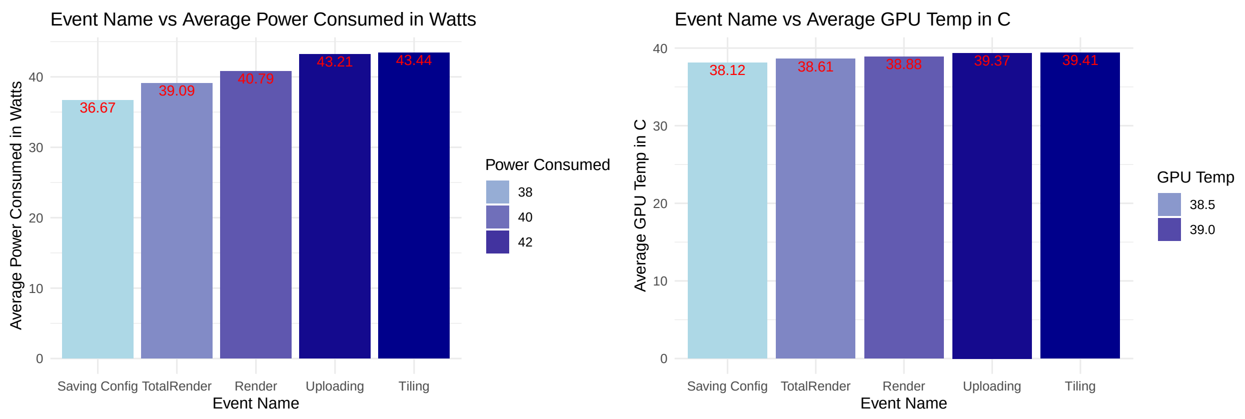

Figure 1: Graphical Summary of Relationship between Event Name vs Average Power Draw in Watt and Event Name vs Average GPU Tmep in C

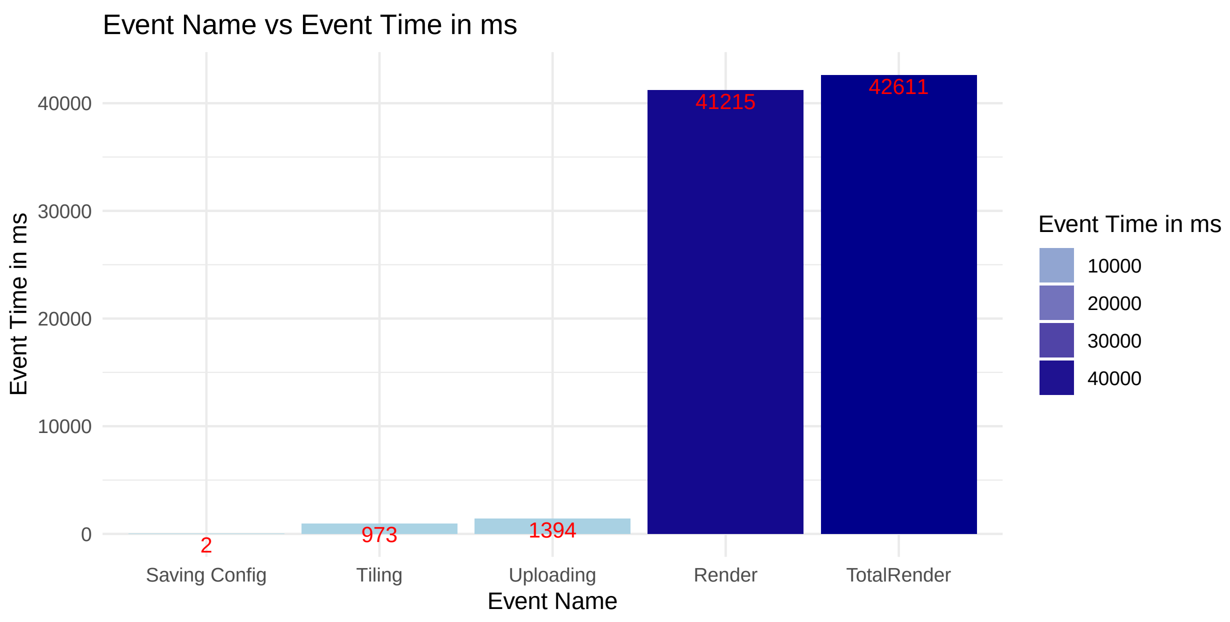

Figure 2: Graphical Summary of Relationship Between Event Name and Average Time Duration

From 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.

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 have been met, provided useful insight 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.

4 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