Performance evaluation of Terapixel rendering in Cloud Computing

DARAKHSHA FATMA NAQVI

220518655

1. Introduction:-

For a range of computational needs, including the development of a three-dimensional city model, cloud computing has been used. For this project, a Terapixel rendering-based 3D image of Newcastle Upon Tyne was produced using supercomputers. We will be working with that dataset. Examining such data enables users to generate images more accurately while spending less time and money. The Crisp-DM model is used in this research to accomplish the same goal. This report will involve frequent data analysis, data merging, data cleansing, and pattern-finding for GPU performance using various methods.

2. Business Understanding:-

Cloud computing is a practical choice for advancing research and accelerating the creation of new products. With cloud computing, businesses can access the newest technologies and scalable resources without worrying about capital expenses or constrained fixed infrastructure. A poorly optimised workload in the cloud will increase total cost of ownership (TCO), which will have a negative impact on return on investment. As a result, it's essential to analyse performance data from cloud supercomputers in order to calibrate the procedure correctly.

3. Objective:-

This report's objective is to look into trends and bottlenecks in the use of GPUs for terapixel rendering operations in Newcastle Upon Tyne via a cloud computing platform. The anticipated criteria that will be looked at are listed below.

- Relationship between gpu statistics.
- Parallel workload distribution by the cloud architecture.
- Memory Usage.
- Event type runtime pattern and domination.

4. Tools and Methodologies :-

Tools – Python programming language is used in order to facilitate reproducibility. To create the report, a jupyter notebook is used in accordance with the readme file's instructions. This makes sure everything is in one place and is easily shareable.

In order to protect against or minimise the dangers of unintentional or erroneous changes, GitHub is used for version control. This allows for easy control over a variety of development paths without the worry of being unable to roll back changes.

5. Data Understanding:-

The datasets offered are based on the outcomes of a run of three 3 tasks (levels 4, 8, and 12 of the terapixel image) employing 1024 GPU nodes over a public cloud, which recorded rendering durations, GPU performance, and which image features were being rendered. Three separate datasets of this information, each recorded in a.csv file, are as follows:

- gpu.csv: GPU status metrics (temperatures, memory usage, etc.).
- application-checkpoints.csv: checkpoint events throughout render job saved by the application.
- task-x-y.csv: Co-ordinates of image parts being rendered in tasks carried by the application for the job.

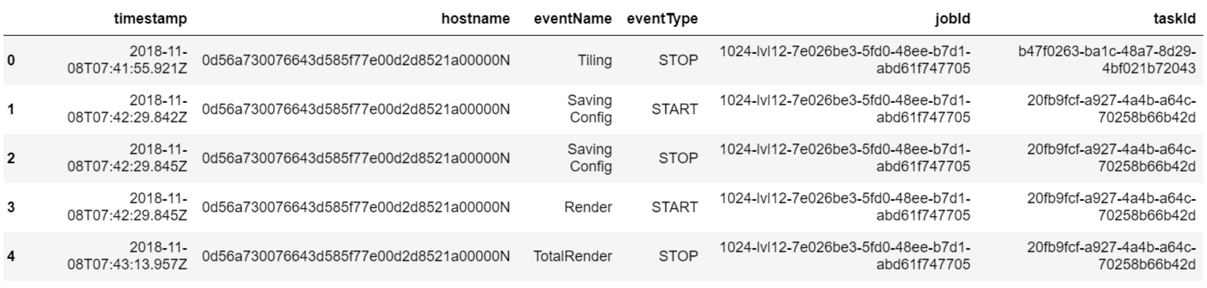
---application-checkpoints.csv---	
timestamp	object
hostname	object
eventName	object
eventType	object
jobId	object
taskId	object

---gpu.csv---	
timestamp	object
hostname	object
gpuSerial	int64
gpuUUID	object
powerDrawWatt	float64
gpuTempC	int64
gpuUtilPerc	int64
gpuMemUtilPerc	int64

task-x-y.csv--	
taskId	object
jobId	object
x	int64
y	int64
level	int64

6. Data Preprocessing:-

- The first step is to import the dataframes with the names df_app, df_gpu, and df task, respectively, from the CSV files application-checkpoints.csv, gpu.csv, and task-x-y.csv into the Jupyter notebook.
- Head of df_app:-



	timestamp	hostname	eventName	eventType	jobId	taskId
0	2018-11-08T07:41:55.921Z	0d56a730076643d585f77e00d2d8521a00000N	Tiling	STOP	1024-lvl12-7e026be3-5fd0-48ee-b7d1-abd61f747705	b47f0263-ba1c-48a7-8d29-4bf021b72043
1	2018-11-08T07:42:29.842Z	0d56a730076643d585f77e00d2d8521a00000N	Saving Config	START	1024-lvl12-7e026be3-5fd0-48ee-b7d1-abd61f747705	20fb9fcf-a927-4a4b-a64c-70258b66b42d
2	2018-11-08T07:42:29.845Z	0d56a730076643d585f77e00d2d8521a00000N	Saving Config	STOP	1024-lvl12-7e026be3-5fd0-48ee-b7d1-abd61f747705	20fb9fcf-a927-4a4b-a64c-70258b66b42d
3	2018-11-08T07:42:29.845Z	0d56a730076643d585f77e00d2d8521a00000N	Render	START	1024-lvl12-7e026be3-5fd0-48ee-b7d1-abd61f747705	20fb9fcf-a927-4a4b-a64c-70258b66b42d
4	2018-11-08T07:43:13.957Z	0d56a730076643d585f77e00d2d8521a00000N	TotalRender	STOP	1024-lvl12-7e026be3-5fd0-48ee-b7d1-abd61f747705	20fb9fcf-a927-4a4b-a64c-70258b66b42d

- We are focusing on data cleaning, wrangling, and merging as part of the pre-processing of the data. Here, a function is created to convert timestamps to our format, and the merge_check_task function is used to combine the application-checkpoints dataset with the task dataset using a left join on the taskId and jobId.

- After merging we have converted the timestamp column into our format and also df_gpu timestamp into our format.

- Finally, we have created a final_merge_gpu function to merge df_gpu with previous merged dataframe df_check_task using a inner join on timestamp. Head of final merged dataframe :-

# Column	Non-Null Count	Dtype
0 timestamp	337682 non-null	datetime64[ns]
1 hostname_x	337682 non-null	object
2 gpuSerial	337682 non-null	int64
3 gpuUUID	337682 non-null	object
4 powerDrawWatt	337682 non-null	float64
5 gpuTempC	337682 non-null	int64
6 gpuUtilPerc	337682 non-null	int64
7 gpuMemUtilPerc	337682 non-null	int64
8 hostname_y	337682 non-null	object
9 eventName	337682 non-null	object
10 eventType	337682 non-null	object
11 jobId	337682 non-null	object
12 taskId	337682 non-null	object
13 x	337682 non-null	int64
14 y	337682 non-null	int64
15 level	337682 non-null	int64

- After final merge we have separated the start and stop time in different columns based on timestamp.

7. Exploratory Data Analysis:-

7.1. Event Summed Execution Time By Event Name:-

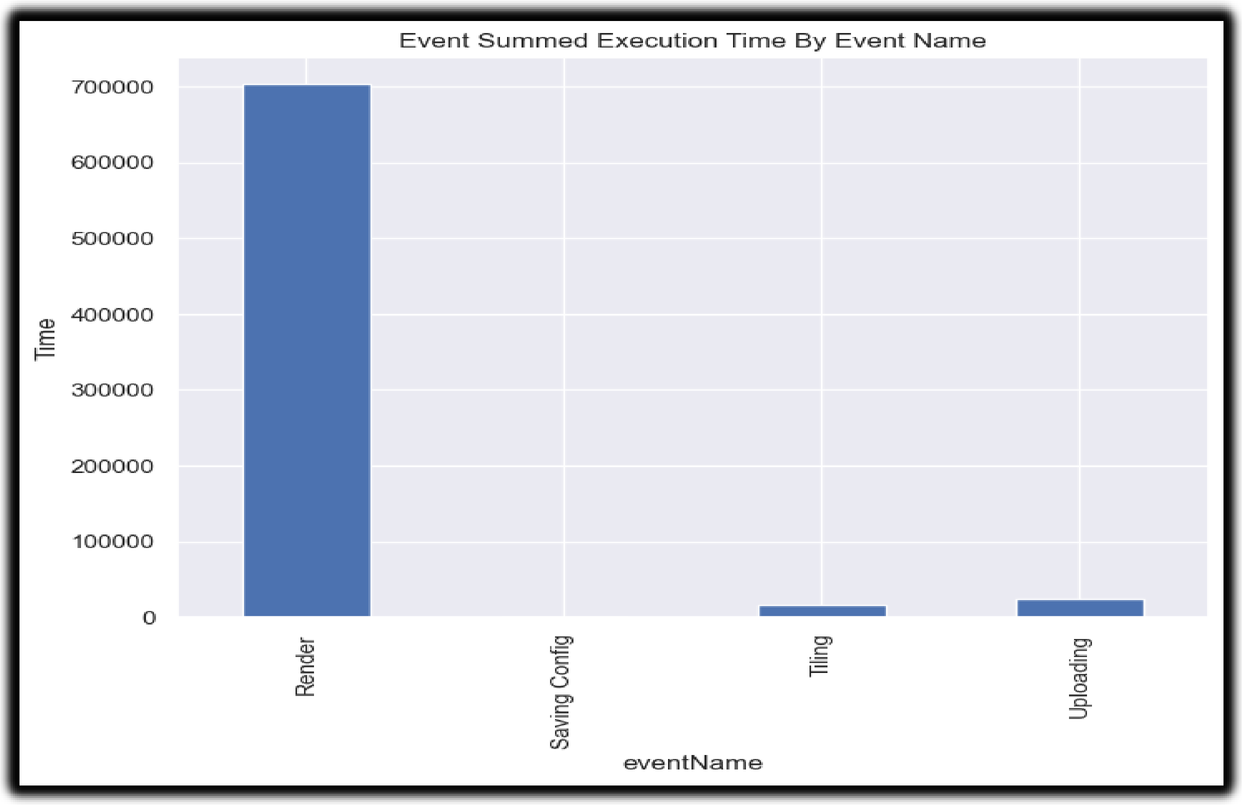


Fig 1

From Fig 1 duration of each event is mentioned and its abundantly evident that GPUs spend the majority of their time rendering, with little time spent saving configuration.

7.2 GPU Temperature v/s Power Drawn in Watt:-

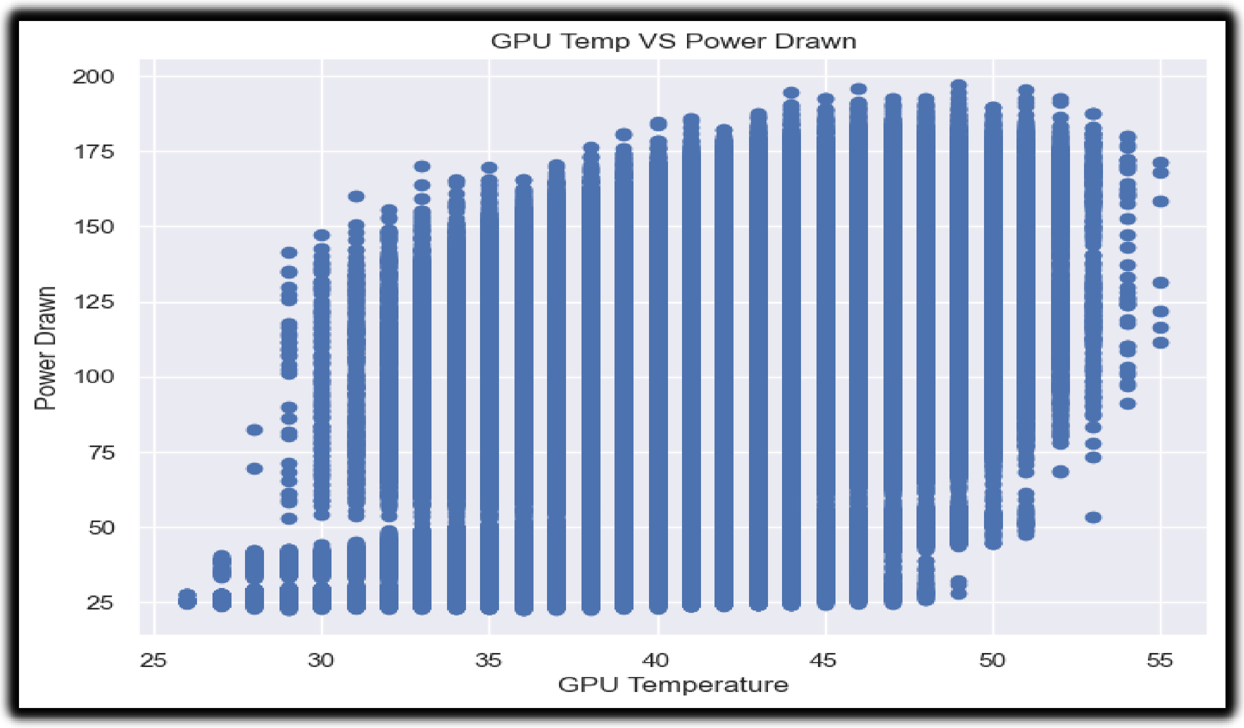


Fig 2

Scatter plot in Fig 2 indicates that there is no liner relationship between GPU temperature and Power Drawn but for most of the power drawn the temperature range is 30-50.

7.3 GPU Memory Utility v/s Utility Percentage:-

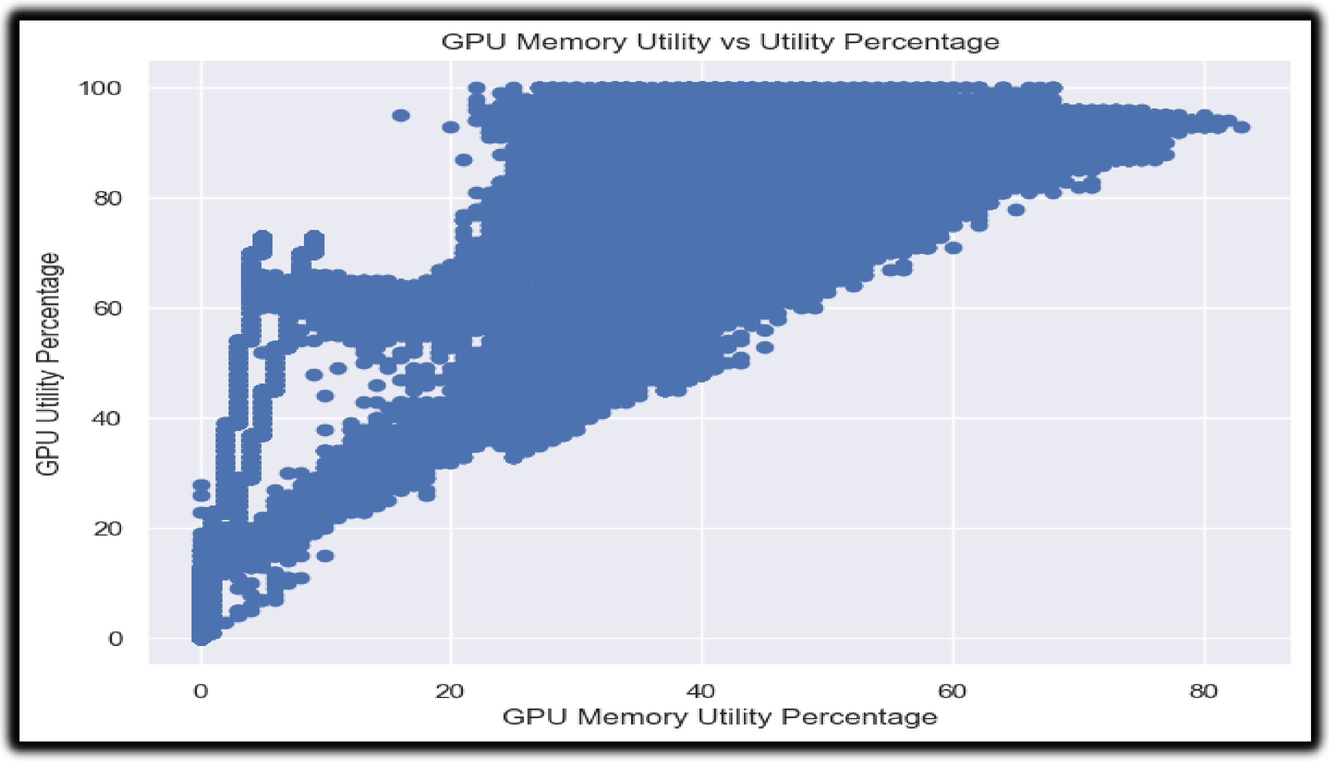


Fig 3

Scatter plot in Fig 3 indicates that there is linear relationship between GPU memory utility and GPU Utility percentage which means if the utility is increasing GPU is taking more space in memory.

7.4 Tasks assigned per GPU:-

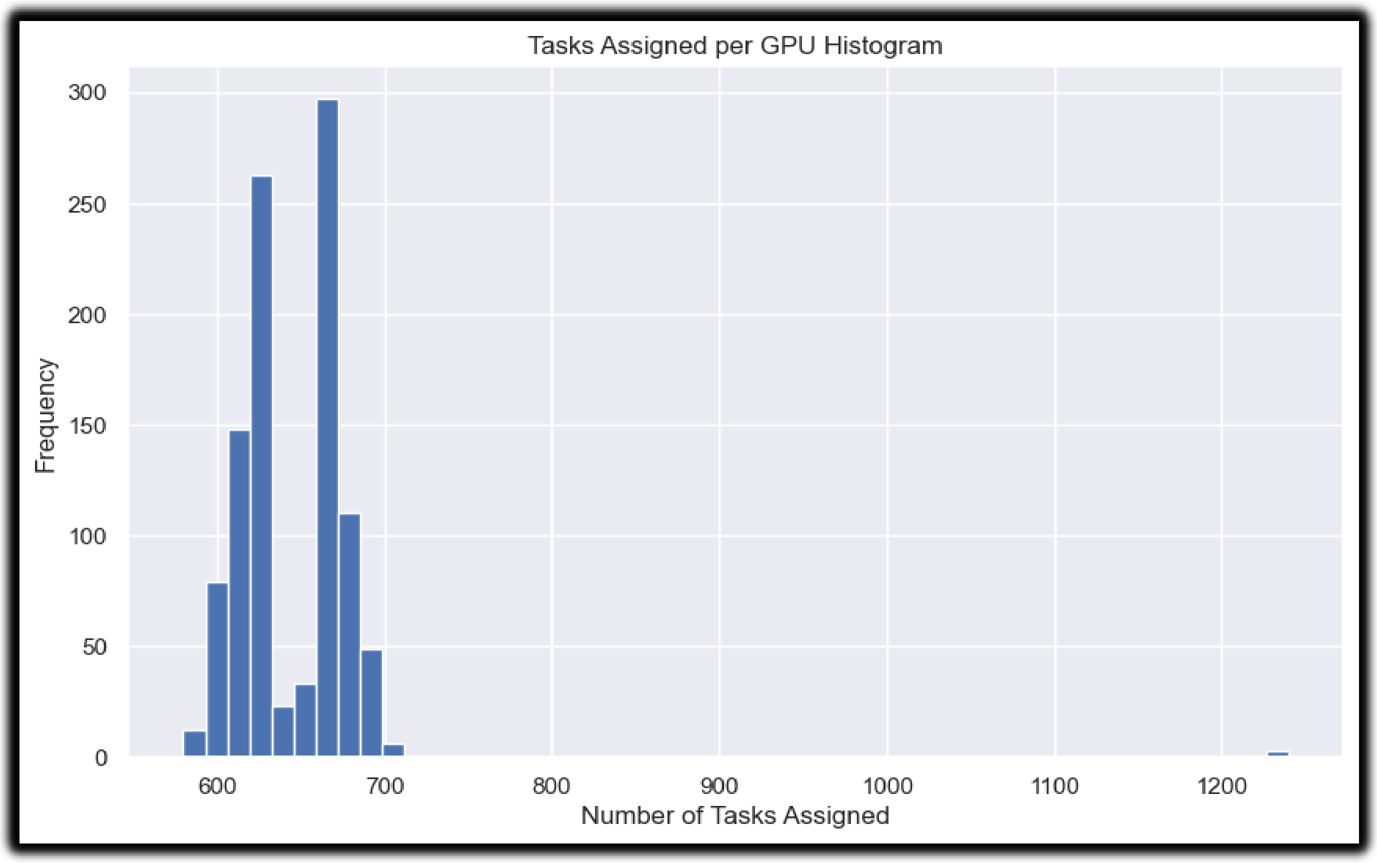


Fig 4

From Fig 4 its evident that the bulk of GPUs are allocated tasks between 600 and 700, with a very small number of workloads around 1200. The graph shows that task handling fluctuates between 600, 700, and even 1200, indicating that scheduling is uneven and could be improved to ensure an equal distribution of the workload.

7.5 Plotting pair subplots to obtain relationship between GPU variables:-

Scatter plot in fig 5 shows the relationship between GPU statistics, from this plot also its clear that among all 4 GPU statistics GPU memory util percentage and GPU utility percentage are highly related.

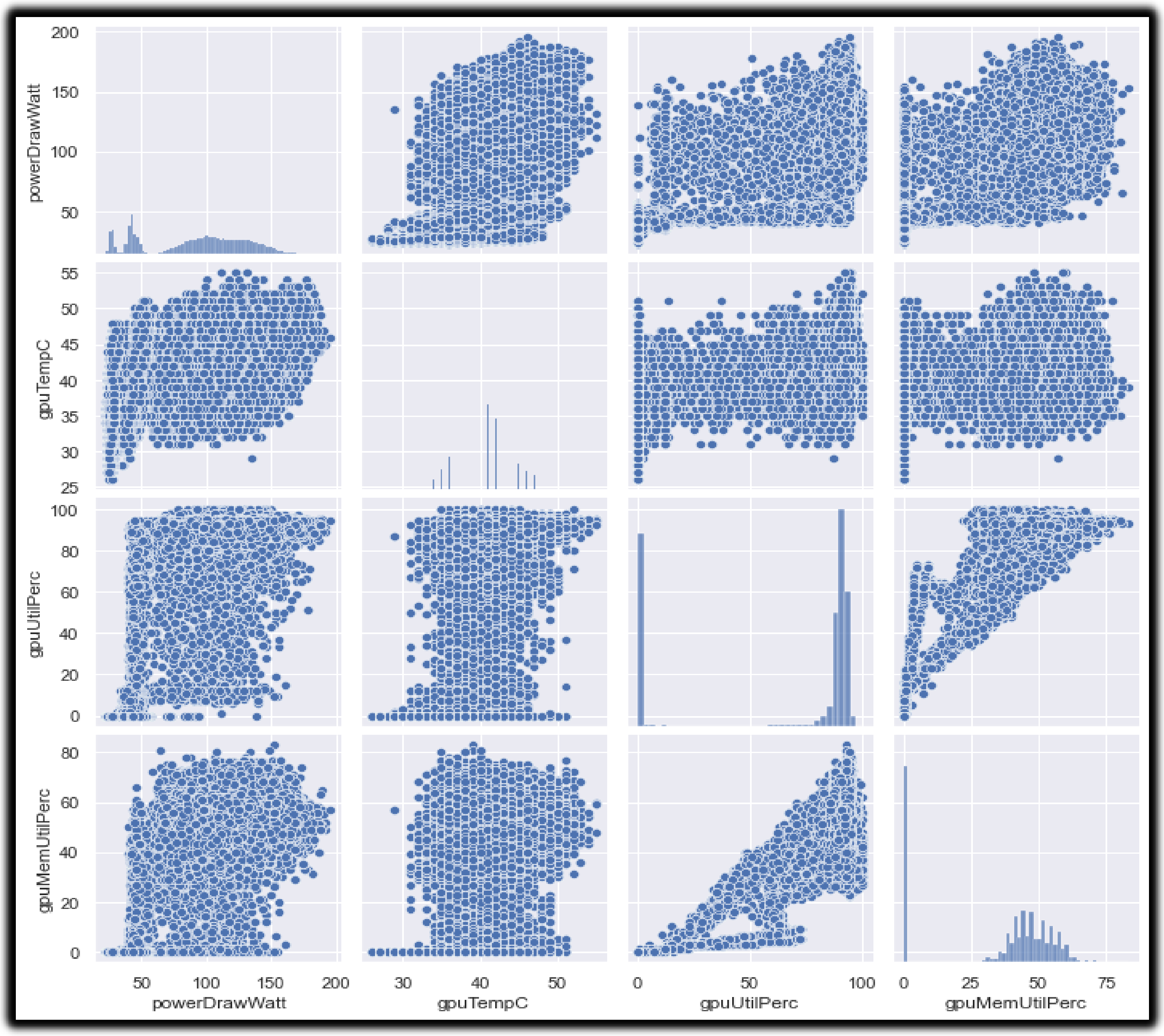


Fig 5

7.6 Correlation heat map for GPU statistics:-

The correlation coefficient matrix has been plotted below as a heatmap thereby making it easier to visualize the relationship with respect to colour intensity.

From below fig 6 its clear that power drawn is having strong relationship between GPU Util percentage and GPU memory util percentage which implies that as power is taken more, the graphics processing unit and memory are used. Also, from this heat map we can validate Fig 5 as GPU Util percentage and GPU memory util percentage are highly related. Further, GPU temperature is having weak relationship with GPU memory util percentage which shows that changing the percentage of memory utility has no discernible impact on the GPU's temperature.

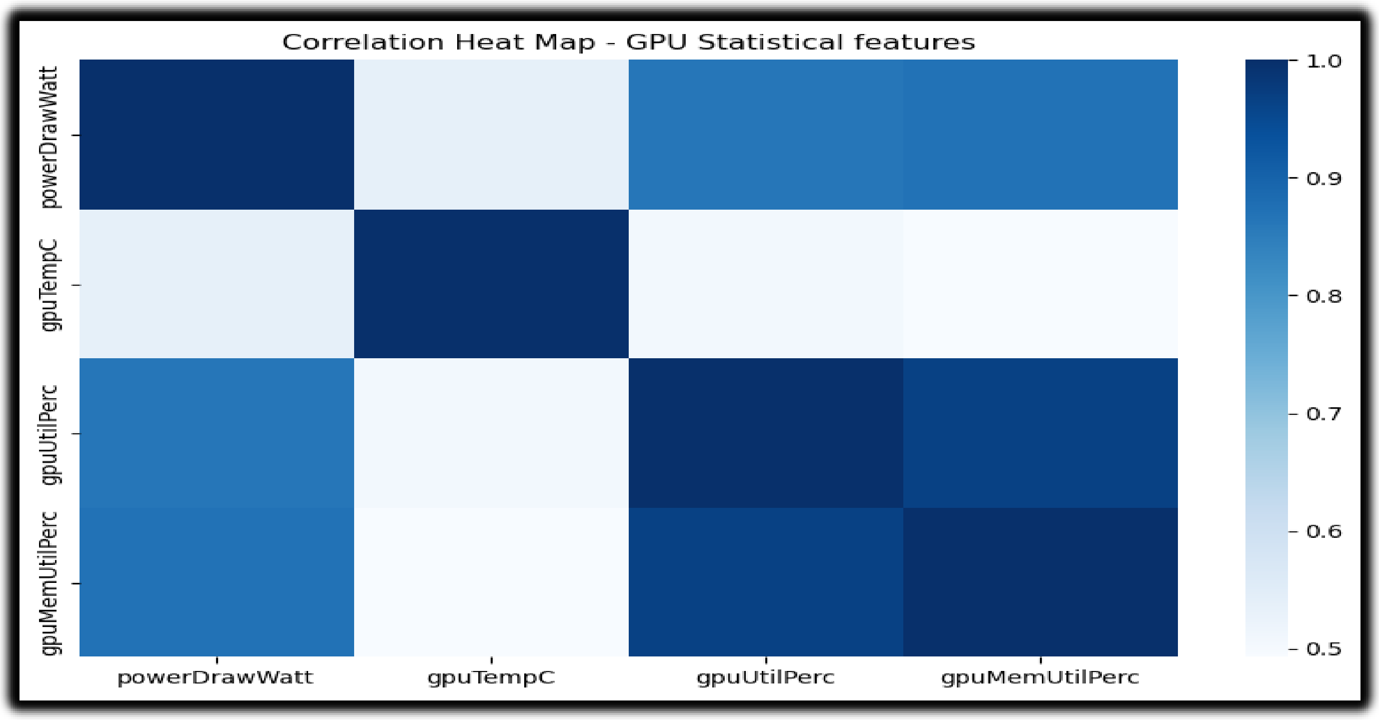


Fig 6

7.7 Plotting pair subplots between GPU variables and power drawn as hue:-

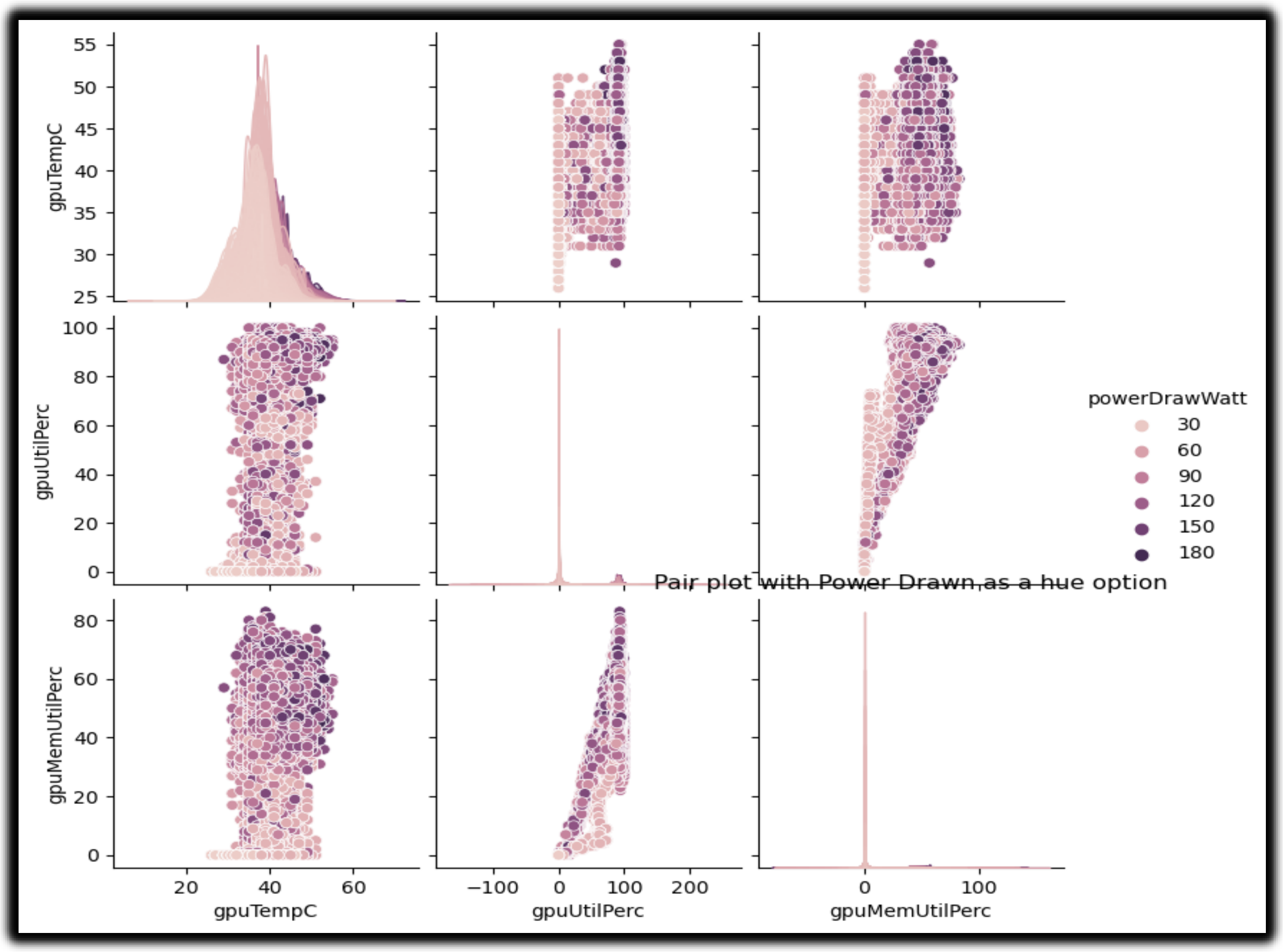


Fig 7

Fig 7 is a pair plot of GPU statistics with power drawn as hue option which shows that GPU utilisation and GPU memory utilisation have a strong correlation with power drawn, suggesting that when power consumption increases, the graphics processing unit and memory are employed.

7.8 Box plot for render time execution and power drawn:-

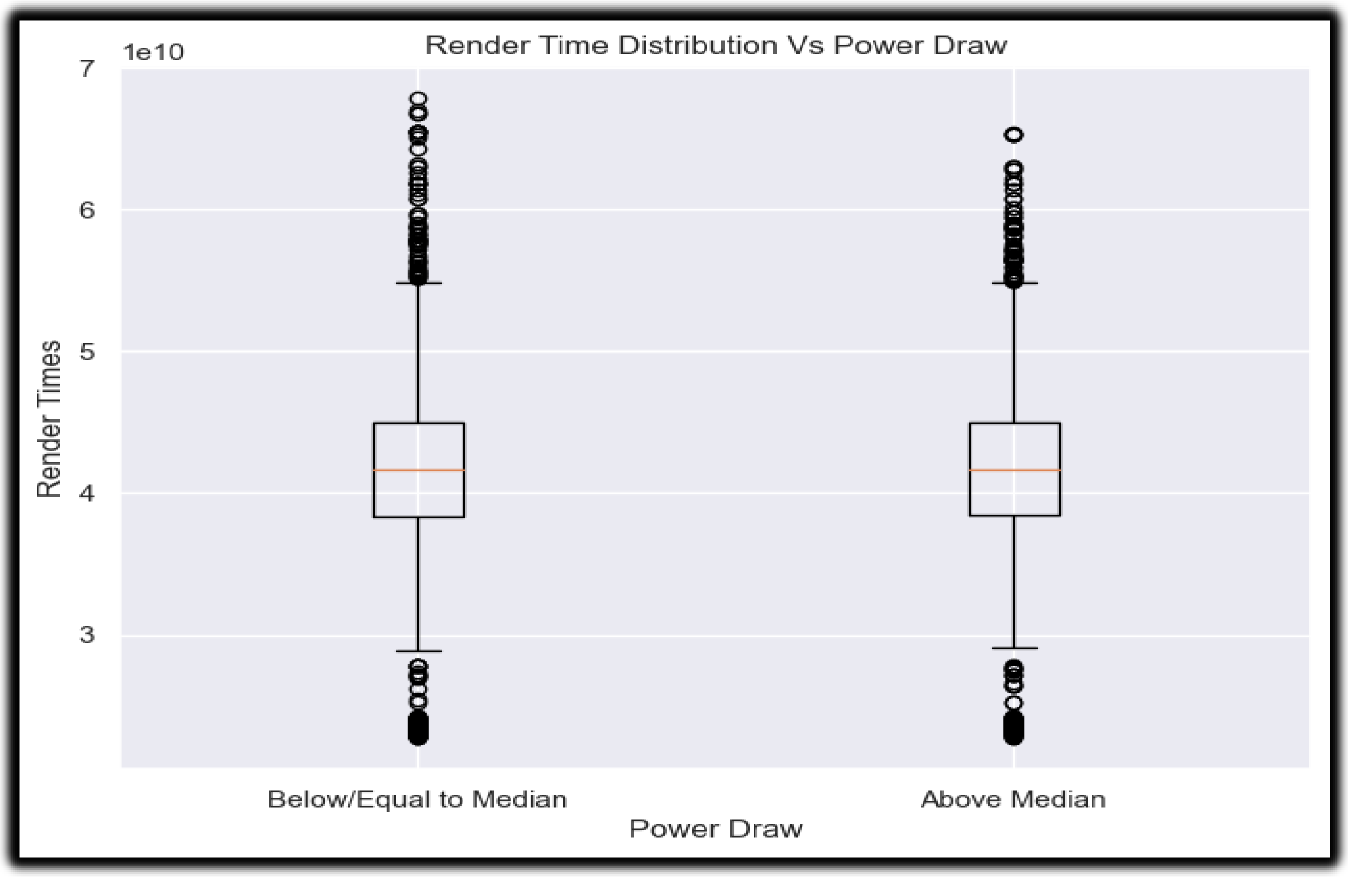


Fig 8

From fig 8 we can say that as the power drawn go higher than the median, while the average does go up the 3rd quantile but does not exceed the 3rd quantile of the other box. However, there is a no significant increase of outliers, in particular above the maximum.

7.9 Box plot for GPU statistics :-

The box plot of GPU statistics shown below in fig 9 shows the various GPU metrics that were recorded while the image was being rendered.

GPU Power Drawn

Power output ranges from 22.5W to 200W, with a typical range of 45 to 121W. These data can help engineers adjust the power supply unit for effective balance in a large-scale application. Additionally, it appears that the GPU utilisation averages at around 90% during the render runs, which is a positive sign that the processing resources are effectively deployed.

GPU Util Percentage

Since the average is between 85 and 90 percent, we can say that most graphics processors are used practically entirely, but because of the wide interquartile range, certain changes are still required for more effective handling.

Memory Utility Pecentage

The median is approximately 50%, and the interquartile range is lower and closer, indicating that either the rendering process is not memory intensive or that the task is not well optimised to use the memory. Another possibility is that GPUs, which use a lot of memory, are to blame for this waste.

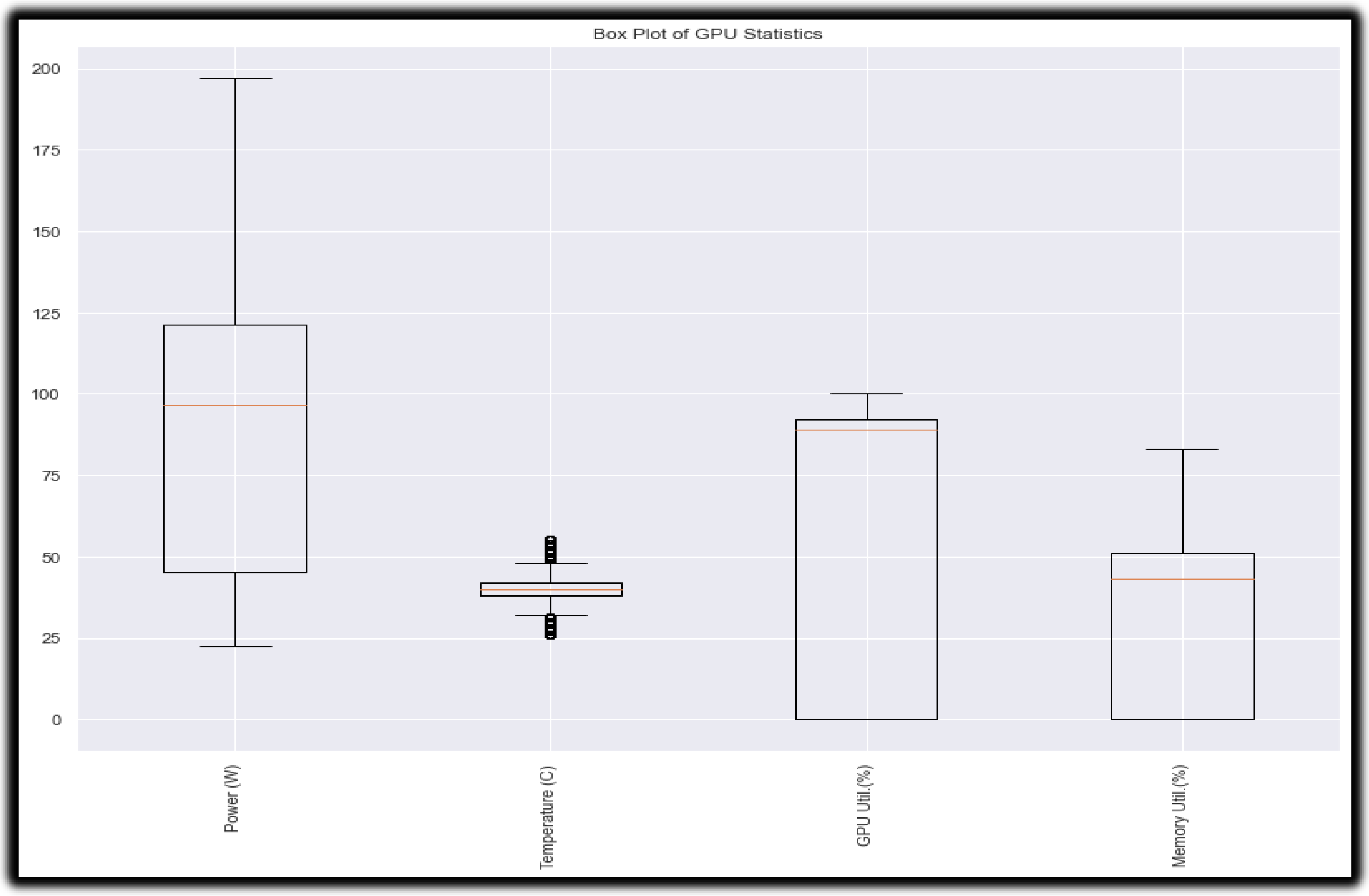


Fig 9

7.10 Box plot for GPU statistics for render event :-

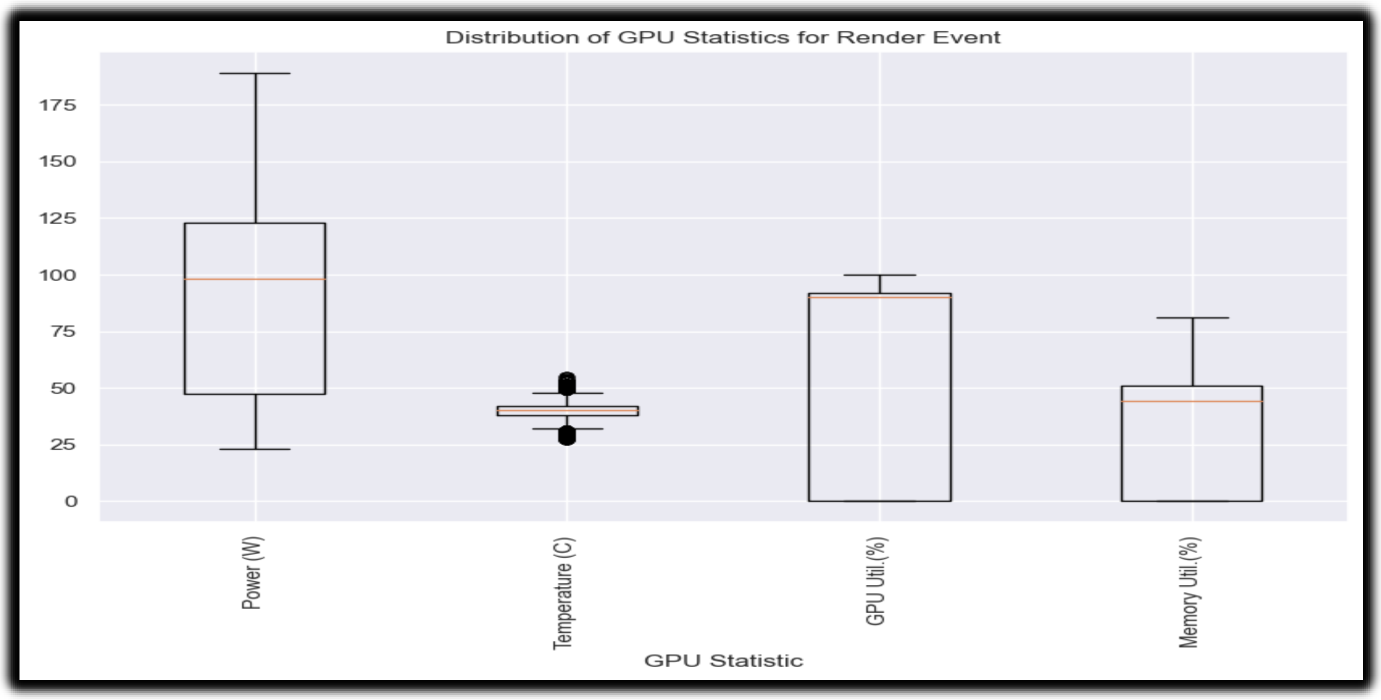


Fig 10

The box plot of GPU statistics for render event shown above in fig 10 shows the various GPU metrics that were recorded while the image was being rendered for render events only. The GPU statistics for render event is almost similar to whole as shown in fig 9.

7.11 Histogram to visualize the power consumption range :-

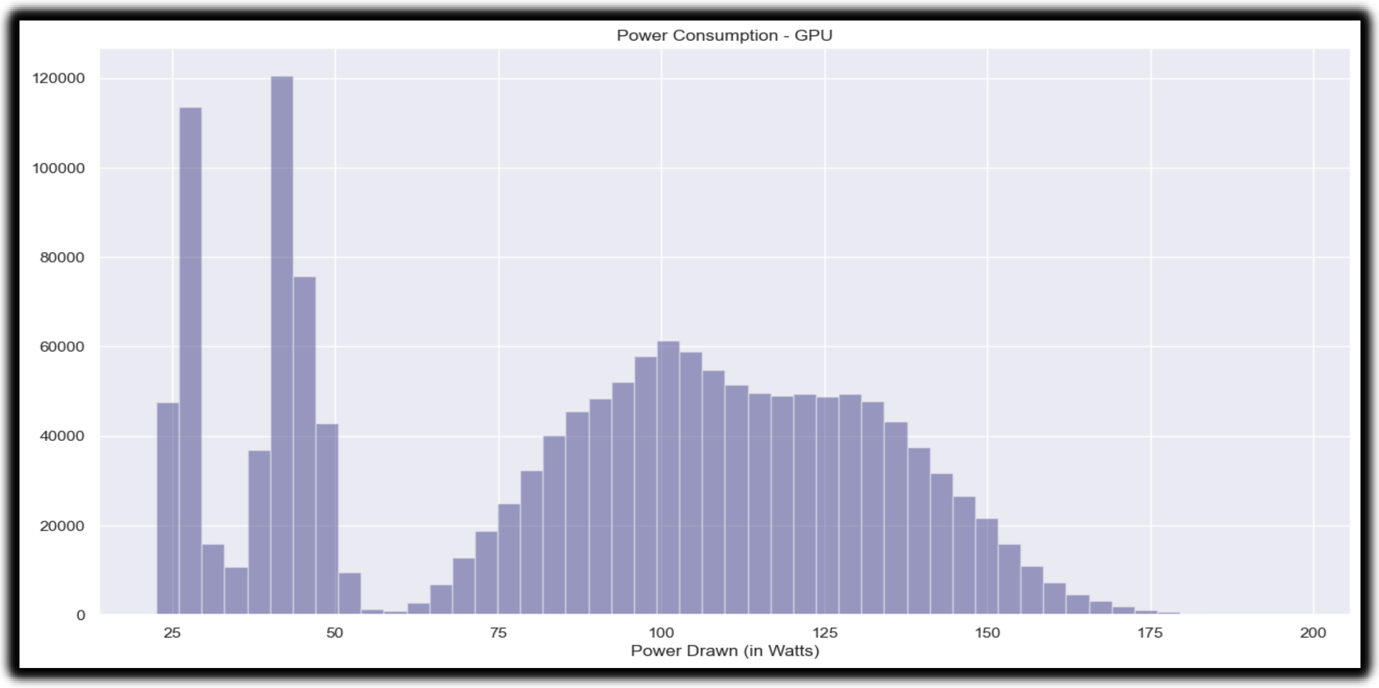


Fig 11

We can see from the above histogram that the majority of GPU clusters use 20 to 50W of power. The rest use energy in a normal distribution.

7.12 Box plot of GPU temperature:-

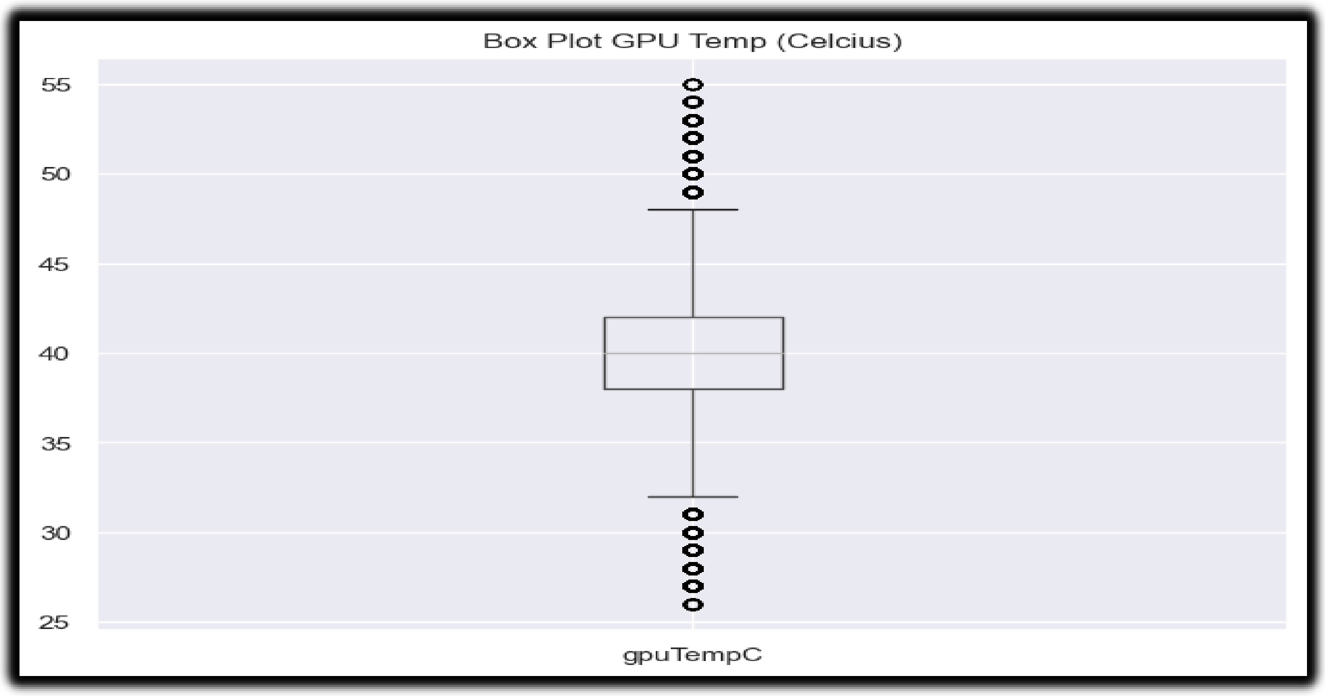
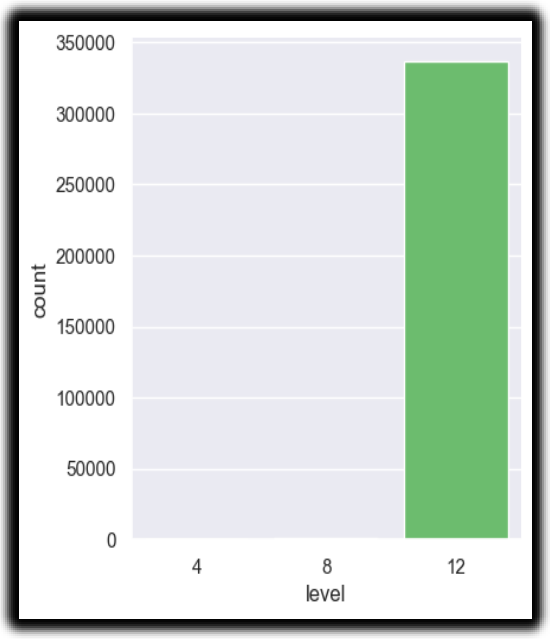


Fig 12

The GPU temperature is generally about 40 degrees Celsius, however there are a few outliers, ranging from 26 degrees to 55 degrees. Excellent temperature control is demonstrated in the GPU cluster.

7.12 Level count:-



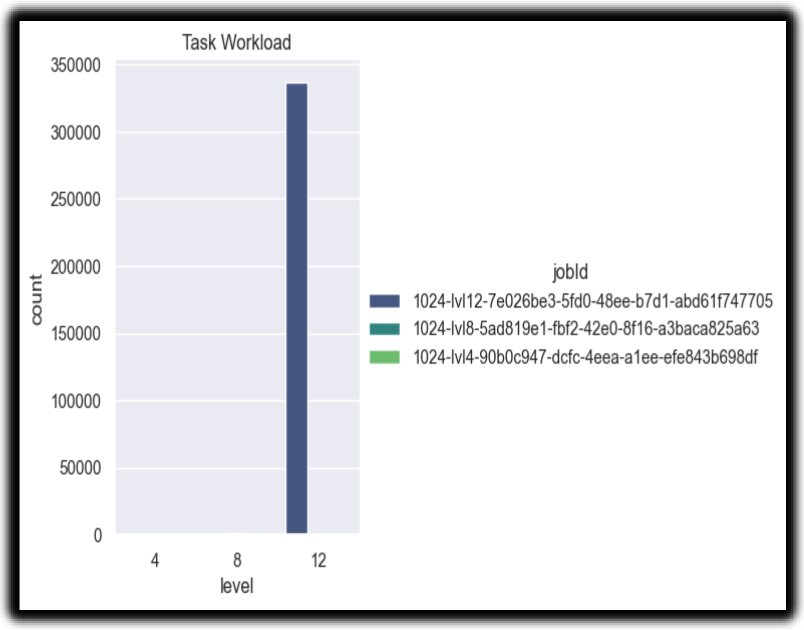


Fig 13 *Fig 14*

The two graphs above(Fig 13 and 14) shows that level 12, the layer of the image that is the most inwards zoomed, uses the most of the GPU resources.

6. Results and Conclusion:-

- Event render takes up most of the task runtime than on Save Config, Tiling, and Uploading. Even when the other three sub-tasks are taken into account, it makes up about 95% of the whole rendering process. Some processes might take less time since the majority of their work is related to metadata. Therefore, the Render subtask will take a long time to process if the image to be produced is greater in size.
- Although most of them do not have a linear relationship and instead form some sort of clusters, the relationship between the different operating parameters of the GPU, including the performance time, is intriguing and may provide useful insights for further investigation. The rendering process will take longer if there is a high memory utilisation rate, which makes reasonable given that there will be more memory to process. The relationship between performance time and GPU's memory utility percentage also has a linear relationship with clustering.
- Moreover, The degree of zoom also affects how long it takes to render an image. Stakeholders can assign the suitable hosts with the required GPU resources based on this information.
- Saving configuration and uploading are the least time-consuming tasks and can be done simultaneously with appropriate management of their execution.
- The properties of an image can have an impact on how long the rendering process takes, as this investigation has revealed.

7. Future Scope:-

Future ramifications on this work include the possibility to undertake prediction techniques to analyse the performance and dependability of the GPU. This project can also serve as the foundation for planning for scalability. If resources for more intensive processing were available, such as 64GB of RAM and a faster GPU, this might be done with better results.

8. Personal Reflection:-

My knowledge of how to analyse a system's performance and how to use a number of tools have both improved as a result of the project. My comprehension of how to use CRISP-DM has improved. Using the Python programming language and a literate programming framework, I have learnt a variety of analytical strategies for conducting data analysis and obtaining pertinent information. The project's use of Jupyter Notebooks, well-known libraries like Pandas (for data processing), matplotlib, and seaborn, together with GitHub for version control, has helped me improve my proficiency with these technologies. I was able to improve my report-writing and project management skills thanks to the assignment.