Performance evaluation of Terapixel rendering in Cloud Computing

220554837

Siddhesh Shashikant Dhamale

1. Introduction

As we receive more information on our urban environment, it is critical to present it in ways that are educational, entertaining, and easily understood by the greatest range of stakeholders. Since it began collecting IoT-sensed environmental data about Newcastle-upon-Tyne more than three years ago, the Newcastle Urban Observatory has amassed more than one billion data recordings. This report's goal is to look at a performance evaluation component of cloud computing's terapixel rendering. The challenge we set out to solve with this project was how to supply the supercomputer scale resources necessary to create a true-to-life terapixel image of Newcastle upon Tyne and its environmental data as recorded by the Newcastle Urban Observatory. The three key objectives of this work are to: create a supercomputer architecture for scalable visualization using the public cloud; produce a terapixel 3D city visualization supporting daily updates; undertake a rigorous evaluation of cloud supercomputing for compute intensive visualization applications.

Monitoring and evaluating the efficiency and speed of rendering large images at a high resolution is necessary to evaluate the performance of terapixel rendering in cloud (super)computing. This can be accomplished using benchmarking, simulation, or modelling, and the resultant image quality can also be evaluated. Both the system's cost and its ability to grow must be taken into account. 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. Methodology

By using the processes of data cleaning and data pre-processing, we will first eliminate the redundant values, missing values, and null values from the dataset. After that, we'll do an exploratory data analysis to uncover information in our data that will be beneficial to the stakeholders. For the exploratory data analysis, we'll utilise Python as our programming language. Pandas, Numpy, Matplotlib, and more helpful libraries are available in Python. Python's Pandas package is used to manipulate data sets. It offers tools for data exploration, cleaning, analysis, and manipulation. The Python package NumPy is used to manipulate arrays.Additionally, it has matrices, fourier transform, and functions for working in the area of linear algebra.For Python and its numerical extension NumPy, Matplotlib is a cross-platform data visualisation and graphical charting package. As a result, it presents a strong open source substitute for MATLAB.

2.1 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.
The initial phase focuses on comprehending the business requirements and goals from a business standpoint, then translating this understanding into a data mining problem definition and an early strategy aimed at achieving these goals.The Business objective of this report is to study the following:-

1. Which hostnames consume more power draw?
2. What is the relationship between GPU utilisation percentage GPU memory utilisation percentage?
3. What is the average percentage of power draw consumption for each event name?

4. What is average GPU temperature of each GPU serial card and how is GPU temperature is distributed?
5. What is the relationship between GPU utilisation percentage GPU memory utilisation percentage?
6. What is the average percentage of power draw consumption for each event name?

Terapixel rendering dataset analysis makes the numeric and visual representation of the result and improves the business operations and develop more machine learning technique that can be integrated with cloud computing applications.

2.2 Data Understanding

The initial phase of data understanding starts with collection of data and proceeds with activities that enable you to become familiar with data, identify data quality problems ,discover first insights into the data,and detect interesting subsets to form hypotheses regarding the information.The dataset below was produced using inputs from system metrics and application checkpoints during the creation of a terapixel picture.The dataset is produced during a run using 1024 nodes. This run is split into 3 jobs to render, i.e. levels 4, 8 and 12 of the visualization output.The TeraScope dataset may be used to solve several different issues.

application-checkpoints.csv

This file contains application checkpoint events throughout the execution of the render job.

Field Name	Data Type	Description	Example
timestamp	timestamp	Start time and End time of each event type	"2018-11-08T07:41:55.921Z"
hostname	String	Hostname of the virtual machine	"0d56a730076643d585f77e00d2d8521a00000N"
eventName	String	Events occurring within the rendering application	"Tiling" "Saving Config" "Render" "TotalRender" "Uploading"
eventType	String	Type of events	"START" "STOP"
jobId	String	Azure batch job ID	"1024-lvl12-7e026be3-5fd0-48ee-b7d1-abd61f747705"
taskId	String	ID of the Azure batch task	"b47f0263-ba1c-48a7-8d29-4bf021b72043"

Table 1: Field Description of application-checkpoint dataset

gpu.csv

This file contains metrics that were output regarding the status of the GPU on the virtual machine.

Field Name	Data Type	Description	Example
timestamp	timestamp	Start time and End time of each event type	"2018-11-08T07:41:55.921Z"
hostname	String	Hostname of the virtual machine	"0d56a730076643d585f77e00d2d8521a00000N"
gpuSerial	int	The serial number of the physical GPU card	"0323217055910"
gpuUUID	String	The unique system id of the GPU unit	"GPU-1d1602dc-f615-a7c7-ab53-fb4a7a479534"
powerDrawWatt	float	Power draw of the GPU in watts	"121.35"
gpuTempC	float	GPU temperature in Celsius	"53"
gpuUtilPerc	float	Percent utilization of the GPU Core(s)	"85"
gpuMemUtilPerc	float	Percent utilization of the GPU memory	"65"

Table 2: Field Description of GPU dataset

task-x-y.csv

This file contains the x,y co-ordinates of which part the image was being rendered for each task.

Field Name	Data Type	Description	Example
jobId	String	Azure batch job ID	"1024-lvl12-7e026be3-5fd0-48ee-b7d1-abd61f747705"
taskId	String	ID of the Azure batch task	"b47f0263-ba1c-48a7-8d29-4bf021b72043"
x	int	X co-ordinate of the image tile being rendered	"10"
Y	int	Y co-ordinate o f the image tile being rendered	"125"
level	int	The visualization created is a zoomable "google maps style"	"8"

Table 3:Field Description of task-x-y dataset

2.3 Data Preparation

- The Data Understanding phase begins with the collection of the initial data and continues with activities that help you become familiar with the data, recognise data quality issues, gain preliminary understanding of the data, and identify interesting subsets to generate hypotheses about the information. A highly useful tool is the dplyr package. It offers numerous data-manipulation tools that can be used on the data frames. We must sanitise the data because it was raw. Missing data cause a number of issues, including reduced statistical power—the likelihood that the test would reject the null hypothesis when it is false—caused by the lack of data. Second, faulty parameter estimation may

result from lost data. Third, it might make the samples less representative. The dataset which contains duplicate values are application-checkpoint and GPU.

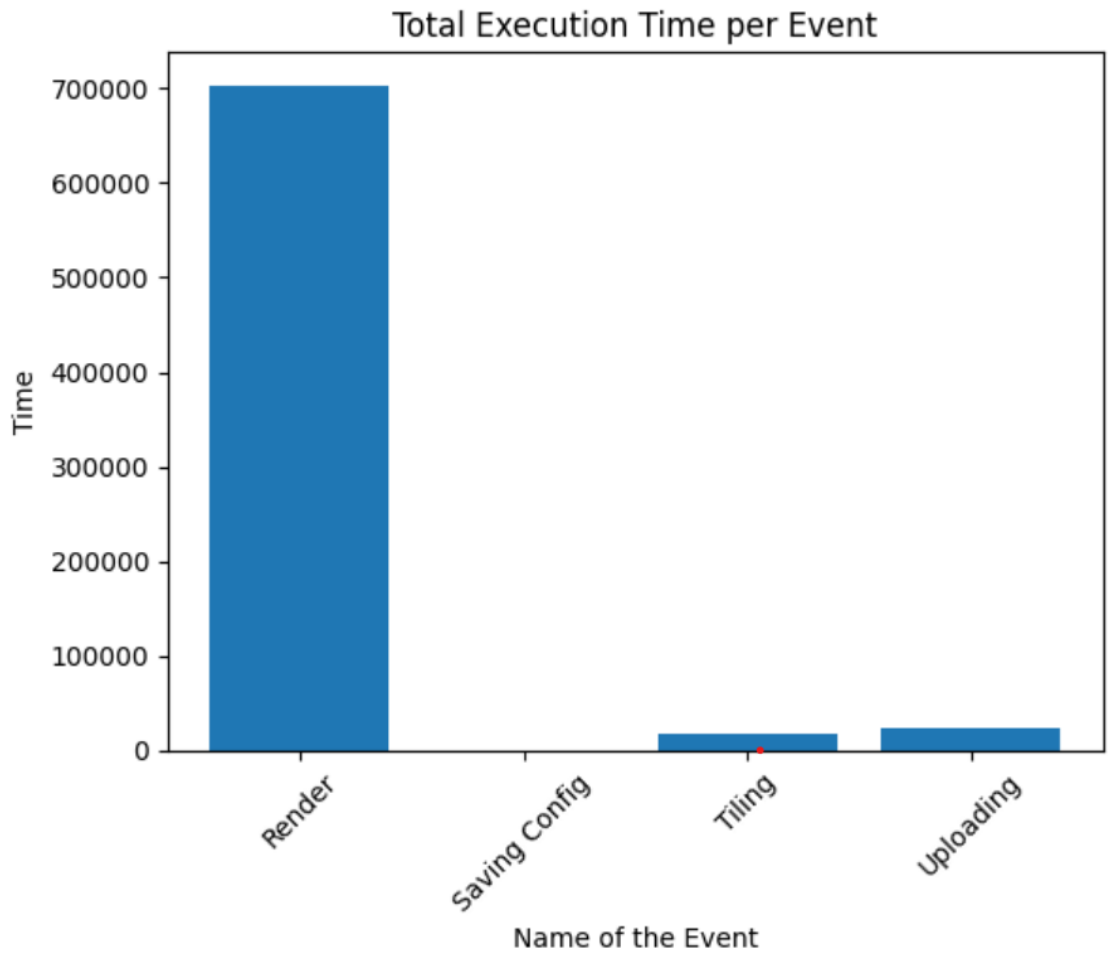
- The first step is to import the dataframes with the names df_app, df_gpu, and df task, respectively, from the CSV files application-checkpoints.csv, gpu.csv, and task-x-y.csv into the Jupyter notebook.
- We are focusing on data cleaning, wrangling, and merging as part of the pre-processing of the data. 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

2.4 Data Modeling

This procedure includes the modelling procedure itself, which might entail testing statistical hypotheses. It involves selecting a modelling approach, developing test designs, creating the model itself, and evaluating the outcomes. It largely depends on the problem's nature and the model being employed. The primary analysis, which takes into account both numerical and graphical performance, is performed by this component.

1. **Event Summed Execution Time by event name :-**
   The difference between the end time and start time is the execution time.We get the execution time by subtracting the start time from the end time.

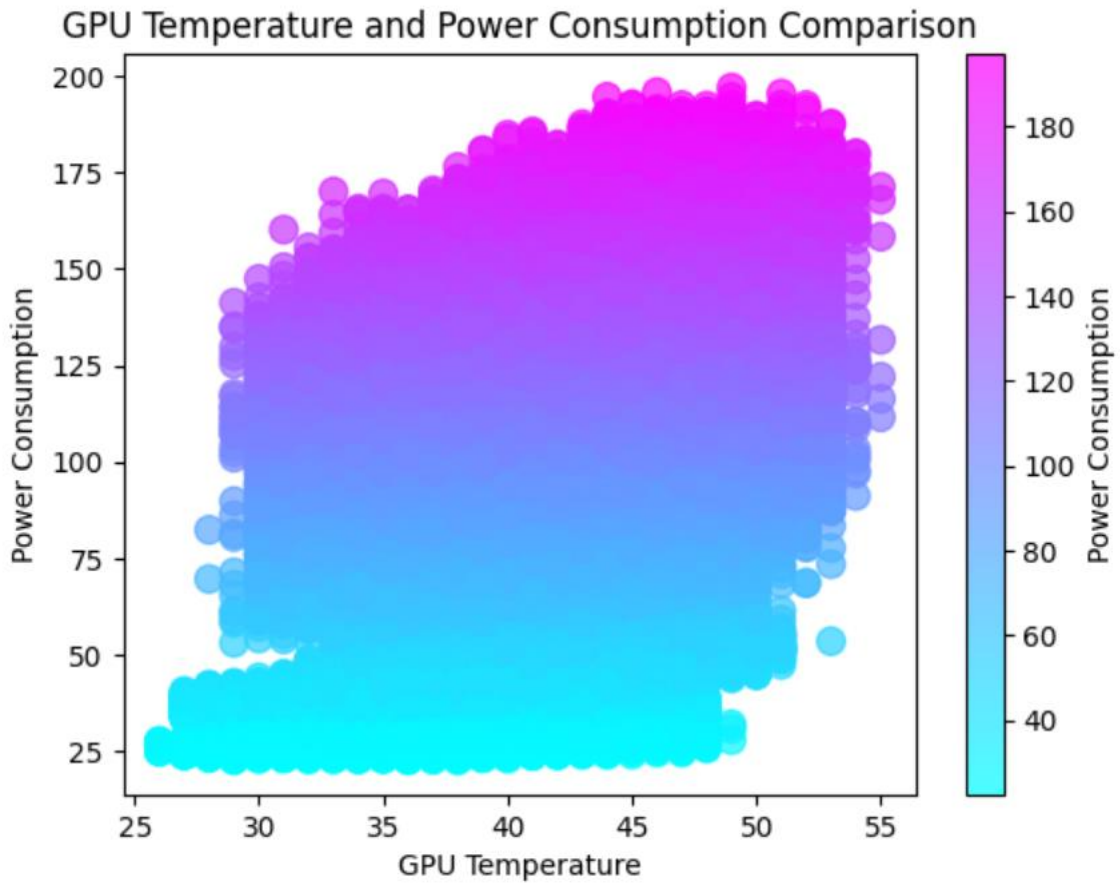


As can be seen from Fig. 1, the duration of each event is discussed, and it is quite clear that GPUs spend the vast bulk of their time rendering, with only a small amount of time going towards saving settings.

2. GPU Temperature v/s Power Drawn in Watt:-

The normal GPU temperature of a graphics card may alter depending on variables such as the particular model, its clock speed, and the task it is handling. The amount of electricity used by a system or device over a specific period of time or activity is known as power draw consumption.

Depending on the type of event or activity occurring, the system or device being used, and the quantity of power consumed, can differ. The typical power draw consumption for different events may vary greatly depending on the event and the equipment being used.



The scatter plot in Figure 2 shows that there is no linear relationship between the temperature of the GPU and the amount of power drawn, although the temperature range that accounts for the majority of the power drawn is between 30 and 50.

3.**Relation between GPU utilisation percentage GPU memory utilisation percentage.**

The GPU utilization percentage refers to the amount of time that the GPU's processing units are actively being used, while the GPU memory utilization percentage refers to the amount of the GPU's memory that is currently being used.

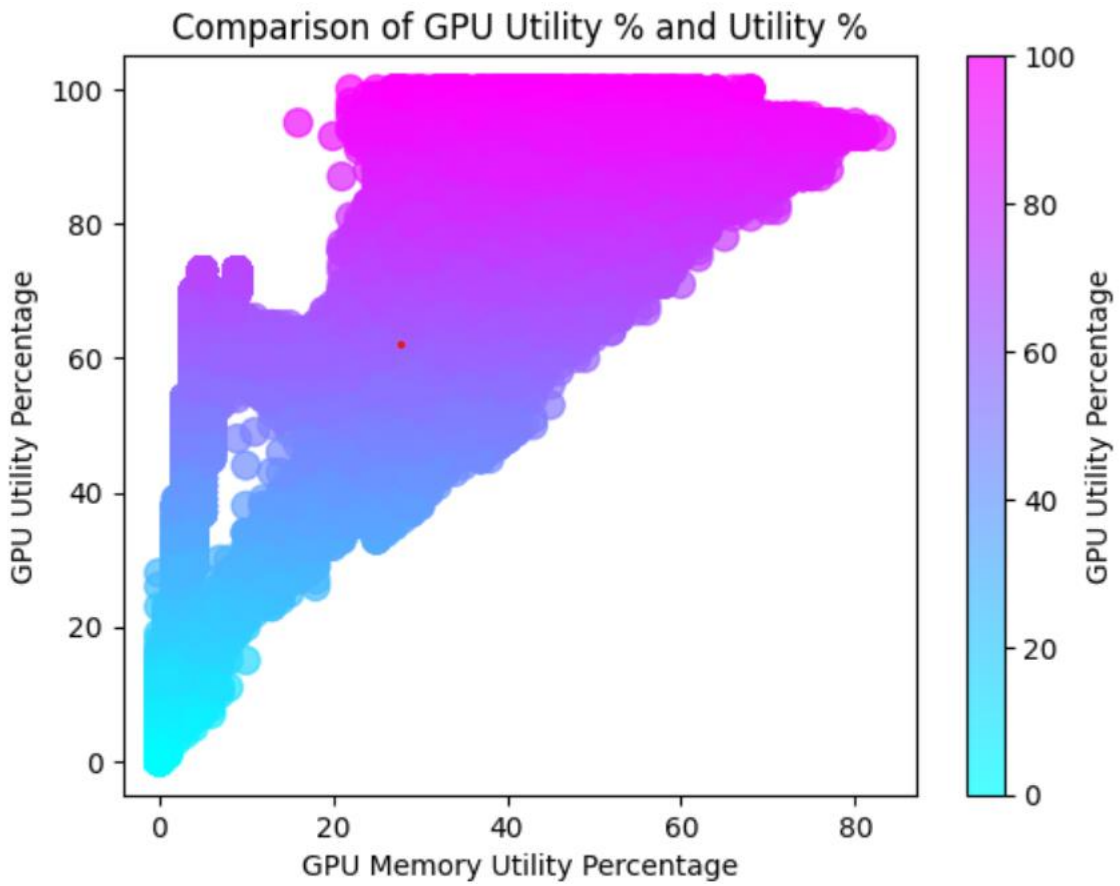
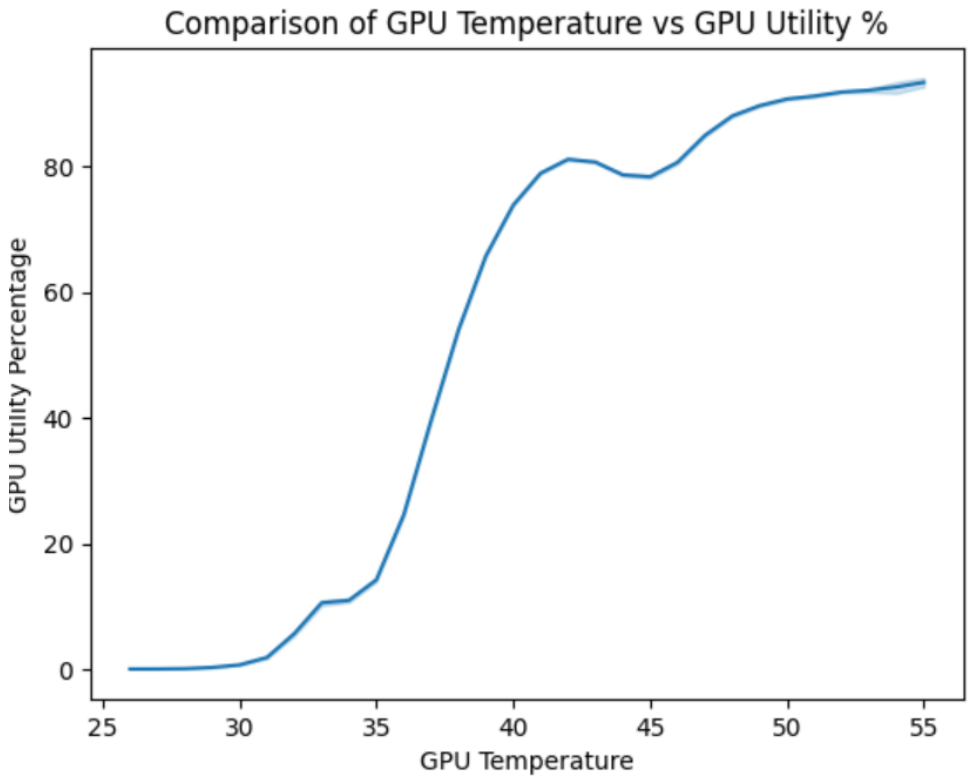


Figure 3: Relation between GPU Utilisation Percentage and GPU Memory Utilisation Percentage

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.The accompanying graphic shows that as the utilisation of a GPU core grows, the utilisation of GPU memory also increases. The two percentages are related because, because the GPU uses more resources to process data, a greater GPU memory utilisation percentage often follows suit.

4.Comparison between GPU Temperature and GPU Utility:-

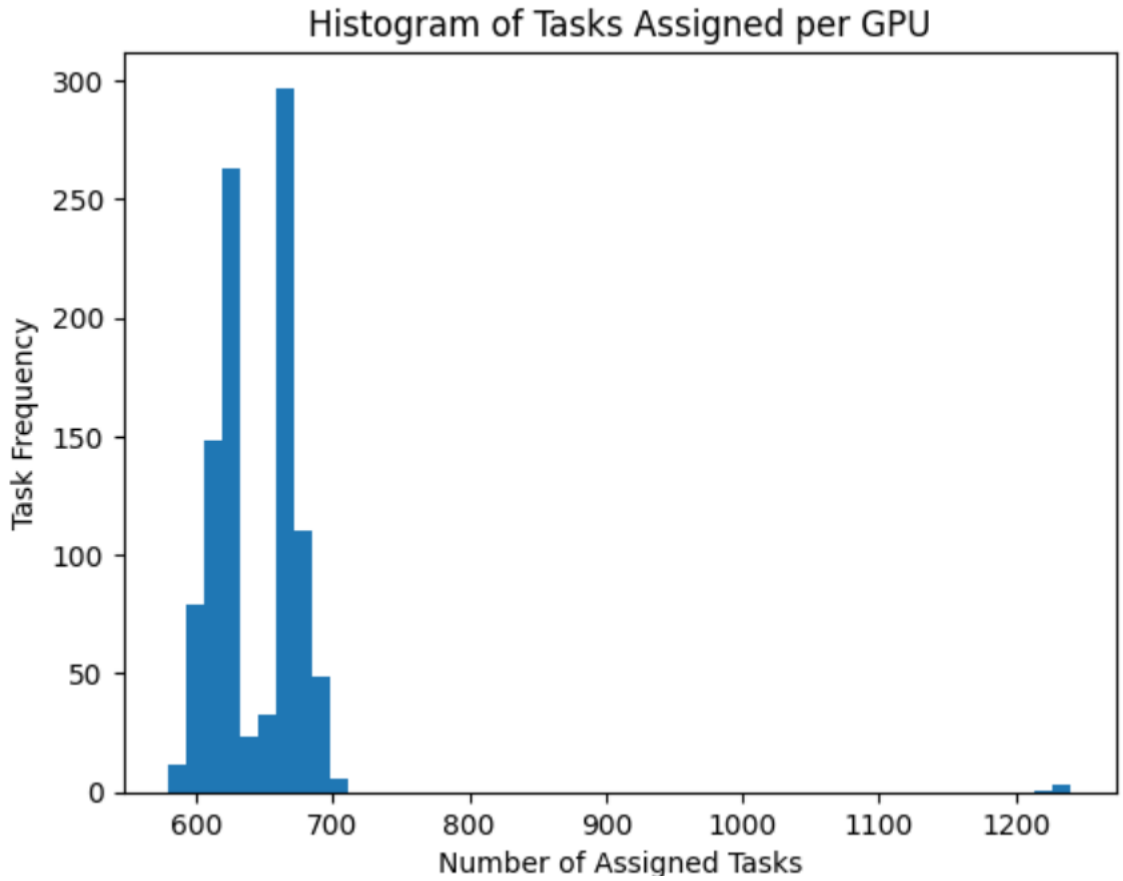
The GPU utility typically refers to the GPU's workload or consumption, which is expressed as a percentage of its maximal capacity. The temperature of the GPU when it is running, on the other hand, is referred to as GPU temperature.How the GPU's temperature changes as its workload or utilisation changes may be seen in the graph of GPU temperature vs. GPU utility. In general, the temperature tends to rise as GPU utility increases. Given that increased GPU utilisation frequently results in increased power consumption and heat generation, this association is expected.



The graph shows that there is exponential rise as temperatures increase from 33 to 45 degrees .There is a small dip after 43 ,but later on it reaches a saturation point.

5.Tasks assigned per GPU :-

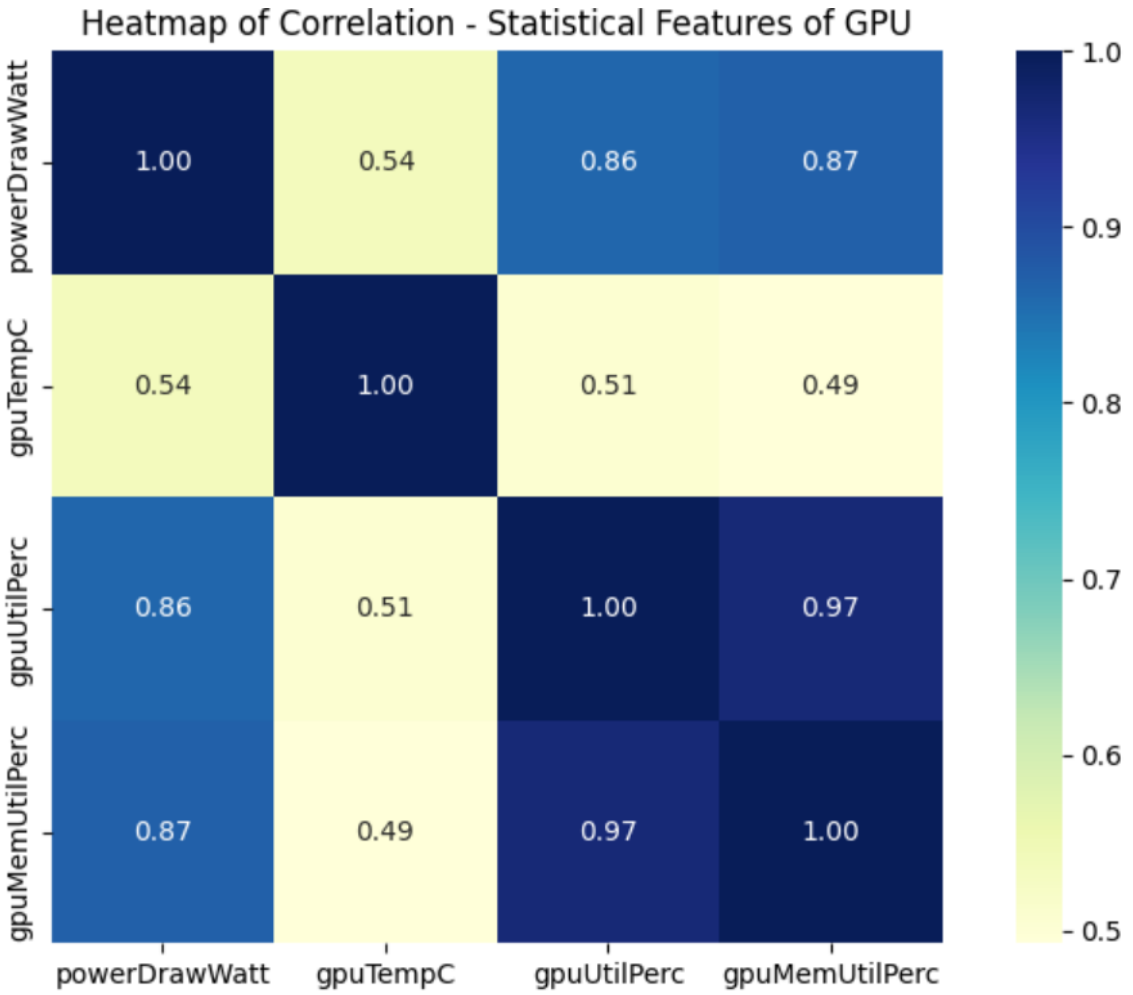
The workload distribution across various GPUs can be seen visually in a histogram that shows the number of jobs allocated per GPU. The height of each bar in the histogram corresponds to the number of jobs falling inside that range, and each bar represents a range or bin of GPU utilisation values.



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.The histogram indicates an unequal workload distribution since only a few GPUs are using the majority of the workloads, while others are underutilised. This knowledge can help decision-makers balance workloads, allocate resources, and plan tasks to maximise GPU usage and boost system performance as a whole.

6.Corelation Heatmap for GPU statistics:-

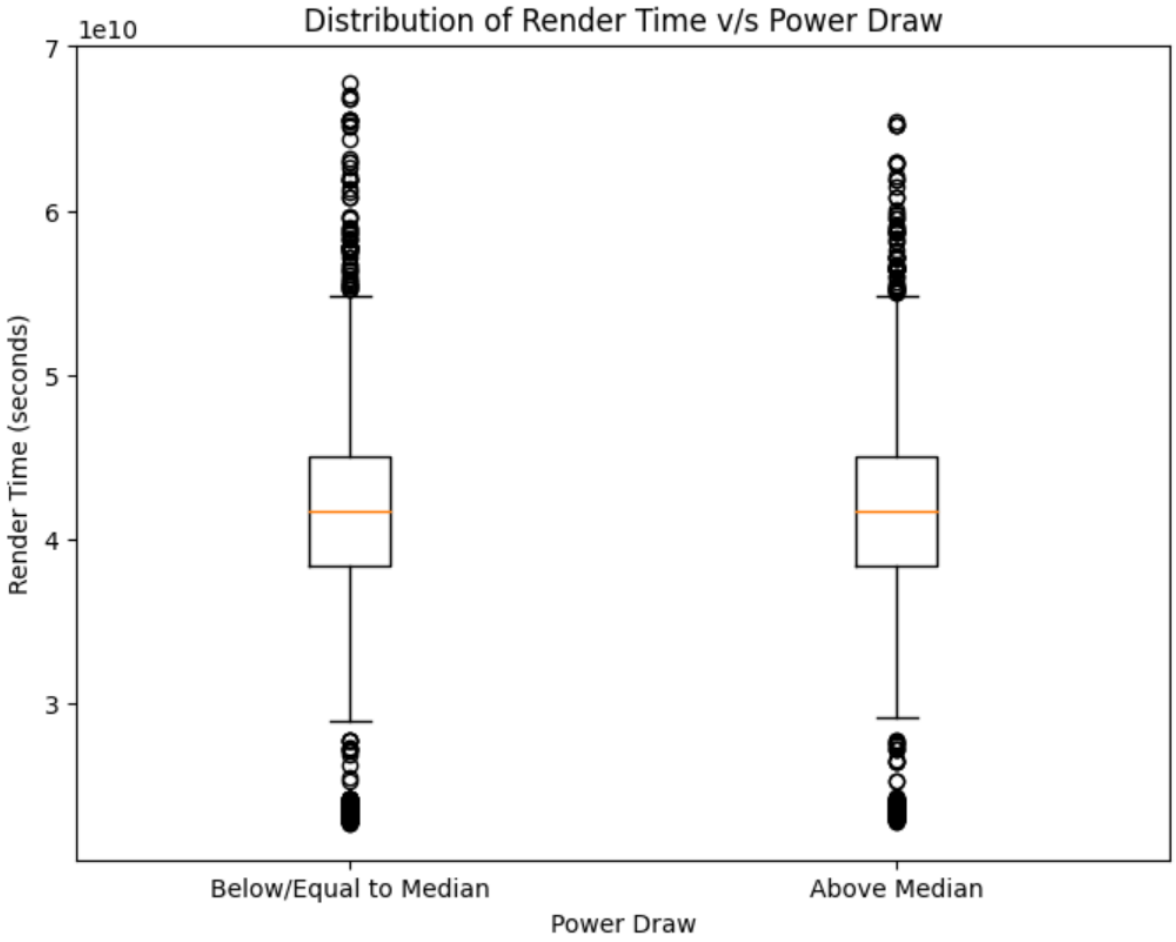
The correlation coefficients between two GPU variables are shown visually in a correlation heatmap for GPU statistics. Each heatmap cell indicates the correlation between two variables, with the strength and direction of the connection denoted by colours or shading. The degree and direction of the linear link between two variables are measured by the correlation coefficient. Its value falls between -1 and +1, with -1 denoting a strong negative correlation, +1 denoting a high positive correlation, and 0 denoting no correlation.



It can be seen in the figure below that the amount of power pulled has a significant relationship with the percentage of GPU utility and the percentage of GPU memory utility. This indicates that the graphics processing unit and memory are used to a greater extent as the amount of power drawn increases. This heat map also allows us to evaluate Fig. 5, which represents the percentage of GPU utilities and GPU memory. The utility % and the percentage are highly connected. Further, the temperature of the GPU has a poor association with the percentage of memory utility used by the GPU. This indicates that adjusting the percentage of memory utility used by the GPU has no discernible effect on the temperature of the GPU.

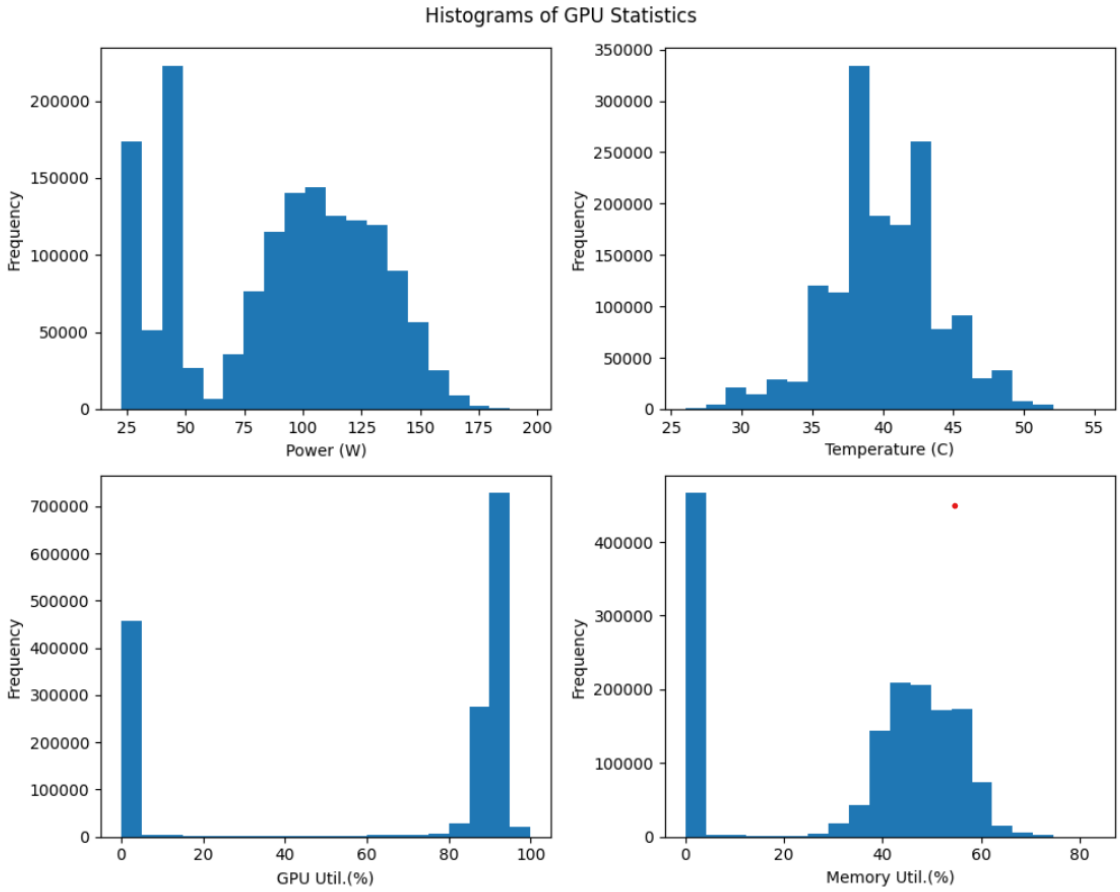
7. Box plot for render time execution and power drawn::-

The distribution of render time execution and power drawn can be better understood through the use of a box plot, which also sheds light on the relationship between the two variables.



According to Fig. 7, we are able to deduce that when the amount of power drawn increases, it goes beyond the median, whilst the average moves closer to the third quantile but does not reach the level of the third quantile of the other box. On the other hand, there is not a discernible rise in the number of outliers, in particular those that are greater than the maximum.

8.Histogram of GPU statistics:-



A histogram of GPU statistics can shed light on the range of possible values and the frequency with which they occur for a particular GPU variable.

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.

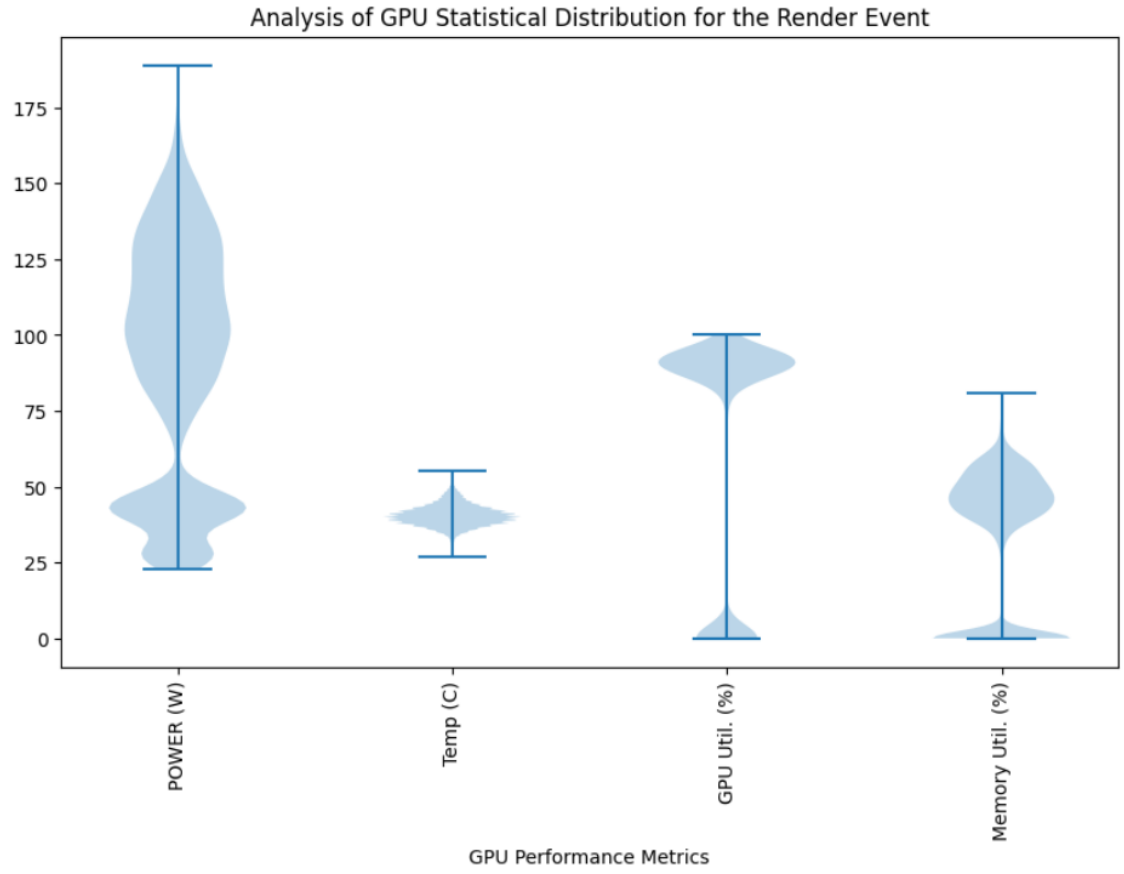
Memory Utility Pecentage :-

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

9. Box plot for GPU statistics for render event :-

We can obtain insight into the central tendency, spread, skewness, and outliers in the render event durations by viewing the box plot for GPU statistics linked to render events. This will show us the

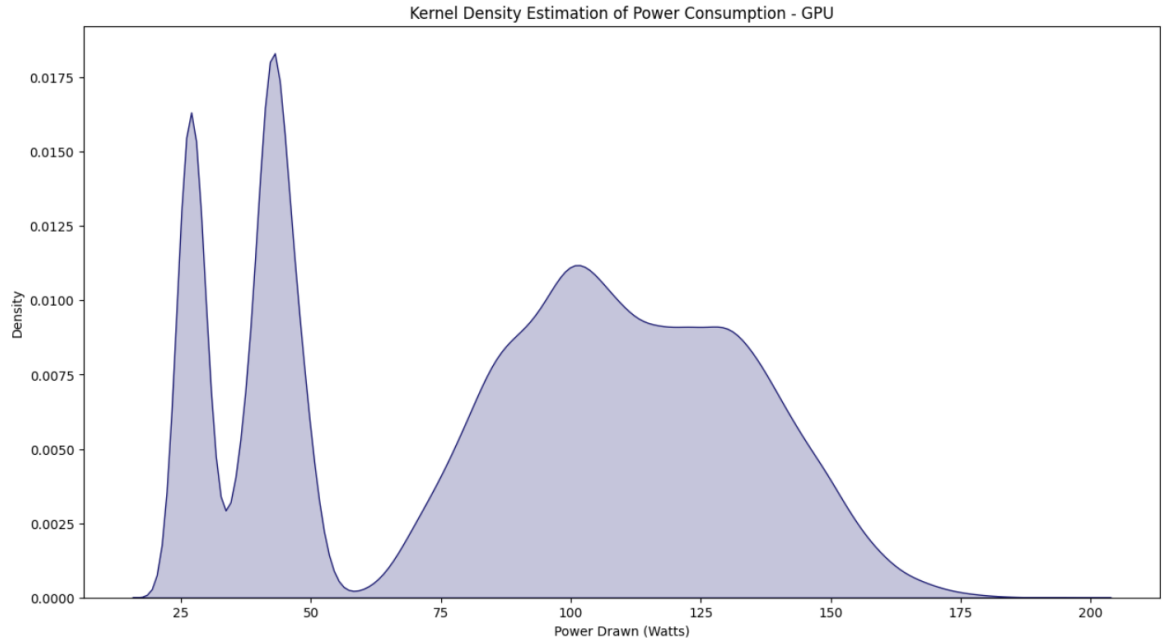
GPU statistics connected to render events. This information can be helpful in identifying performance patterns, locating anomalies, determining variability, and making decisions that are informed in relation to rendering performance optimisation or debugging.



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

10.Kernel Density Estimation of Power Consumption:-

It offers a smooth and continuous estimation of the probability density function, a KDE plot, also known as a Kernel Density Estimation plot, is an effective tool for visualising the power consumption distribution of a graphics processing unit (GPU). A KDE plot, in contrast to a histogram, which provides an approximation of the distribution by utilising discrete bins, makes it possible to conduct a more in-depth analysis of the power consumption values. It is helpful in identifying the structure of the distribution, as well as its peaks and probable modes, which provides insights into the range of power consumption levels and their likelihood. This information can be useful for recognising trends of power usage, locating anomalies, and making decisions that are informed in relation to power optimisation and management.



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

3.Results and Conclusions:-

• Event rendering consumes a greater portion of task runtime than Save Configuration, Tiling, and Uploading. It accounts for around 95% of the entire rendering process, even when the other three sub-tasks are included. Some procedures might go more quickly since the majority of their work involves metadata. Therefore, if the image to be produced is larger in size, the Render subtask will take a long time to process.

• The link between the various operating parameters of the GPU, including the performance time, is intriguing and may offer helpful insights for further research, even though most of them do not have a linear relationship and instead form some sort of clusters. If there is a high memory usage rate, rendering will take longer, which makes sense given that there will be more memory to process. Clustering and the link between performance time and GPU memory utility percentage are both linear.

• Additionally, the amount of zoom has an impact on how quickly an image is rendered. Based on this data, stakeholders can assign the appropriate hosts with the needed GPU resources.

• With proper execution management, the least time-consuming tasks—saving configuration and uploading—can be completed concurrently.

• As this experiment has shown, an image's characteristics can affect how long the rendering process takes.

4.Future Scopes:-

The results of this work may have future repercussions, one of which is the opportunity to implement prediction techniques in order to evaluate the dependability and performance of GPUs. This project has the potential to act as the basis for further planning on scalability. If resources for more intensive processing were available, such as 64 gigabytes of random access memory and a faster graphics processing unit, it's possible that this might be done with better results.

5. Personal Reflection:-

The dataset that was provided was quite decent, and it did not call for a significant amount of data cleaning or munging. The data went through a lot of manipulation in order to make it so that it could make sense on its own.

Working with this dataset was just like working with anything else; it came with both positive and negative experiences. I had a lot to learn by taking a cloud-based vantage point. A lot of new concepts and methodologies were brought to your attention. Became familiar with some recently developed tools such as Git and Jupyter notebook. Having access to all of these tools makes the process simpler and more efficient. Since I had no prior experience with Git, using it at first felt quite laborious. However, as the work went, I was able to get the feel of it, and these tools made my job much simpler.

Despite the fact that the majority of the items are satisfactory, I had a lot of trouble converting the ipynb file to pdf. Participated in this huddle and was able to come up with a workable solution that can be put into effect for this issue.

If I were given another chance in the same situation, I know that I would approach it in a different way the second time around. I would like to make further use of Git and the features of its version control system, both of which I was unable to investigate to a large extent this time around. Aside from all of that, working on this project was a rewarding experience, and I gained a lot of knowledge regarding the cloud infrastructure side of things, as well as the many tools and methodologies that are currently available.