**Performance evaluation of Terapixel rendering in Cloud (Super)computing**

**Background :**

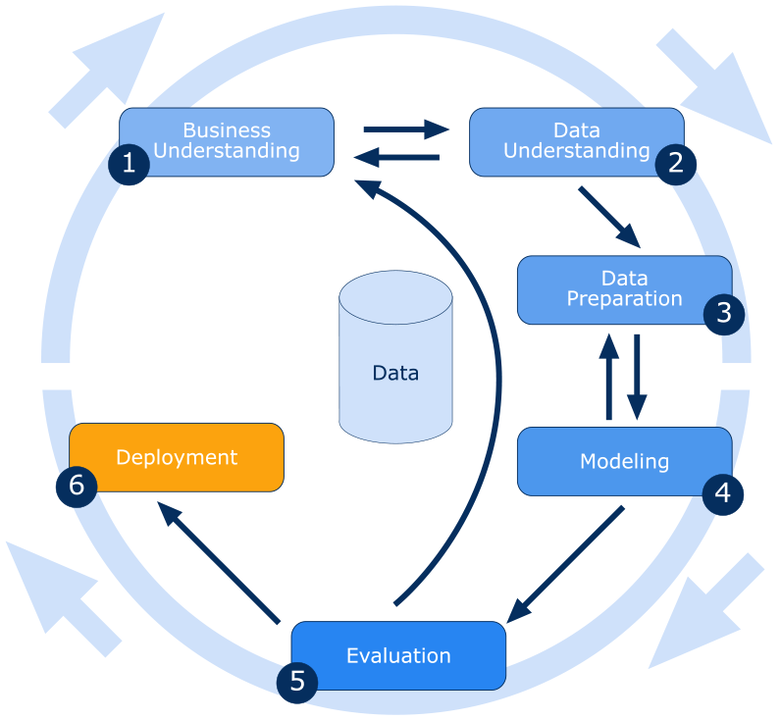
Photorealistic terapixel visualization requires a lot of computing power. A solution was demonstrated to terapixel visualization of the city of Newcastle upon Tyne using peak 14Pflops visual supercomputer with 1024 NC6v3 (6 core +1 Tesla V100 GPU) nodes with 1 Tbyte of Azure Blob storage, the solution is cloud-based deployment and can be accessed by wide range of thin clients used as pay as you use service. The efficiency of the super computer and the performance of the tasks, jobs rendering visualization is monitored through different statistics namely Application checkpoint, GPU statistics, Tasks rendered with reference to each tile, system metric output from the production of a terapixel image.

**Analysis Requirement:**

Exploratory data analysis (EDA) is a method used to analyze and summarize large datasets in order to gain insights and identify patterns. When applied to terapixel image rendering data and GPU metrics of the super computer which renders, EDA could be used to identify patterns or anomalies that may be present. This could include identifying areas of performance of the visualization, as well as analyzing the distribution of different tasks and jobs across nodes which renders and performance of the GPU Hosts. Some common techniques used in EDA include visualizing the data using histograms, scatter plots, and heat maps, as well as calculating summary statistics such as mean, median, and standard deviation. It’s important to keep in mind that EDA is an iterative process, and that insights and patterns discovered during the analysis may lead to further questions or areas of investigation. This Analysis will help us in identifying the GPU performance, efficiency of the GPU, power consumption, tasks and jobs management between nodes etc. Which can be used by the stake holders in future for building super computers for such tasks and implementing terapixel rendered visuals.

**Approach for Analysis:**

There are a number of different approaches that may be used to take a systematic approach to solving data analysis challenges. The Cross-Industry Standard Process for Data Mining, often known as CRISP-DM, is a technique that is frequently used in data mining and data analysis projects. It provides a systematic strategy for the entire analysis pipeline, beginning with the problem understanding and the collection of the data and continuing on through data preparation and to the construction of a model, the deployment of that model, and the monitoring of its performance. Figure 1 provides the depiction of the CRISP-DM process which is followed to address problems at hand.



*Figure 1 Graphical depiction of CRISP-DM methodology consisting of six stages*

There are six stages involved in the CRISP-DM process which is flexible and can be adapted for various problems. The flow is indicated by the direction of the arrows, and the scenario where it is necessary to revisit an earlier stage is indicated by the reverse arrows. The circle around the process represents a typical situation in which a deployed solution leads to the development of new requirements and the restart of the entire process based on updated problem formulation.

**Business Understanding**: Acquire an understanding of the problem at hand, as well as the objectives of the project, and specify the criteria for success.

**Data Understanding**: Data gathering, analysis, and evaluation of its various characteristics to establish its suitability to the problem.

**Data Preparation**: Prepare the data for model building by implementing the process such as data cleansing, transformation, feature extraction, feature selection and class balancing.

**Data Analysis and Exploring/Modeling**: Analysis is the process of selecting a suitable stastistical model and analysis on the prepared data.

**Evaluation**: This stage evaluates the model based on the evaluation criteria identified in the problem understanding stage.

**Deployment**: Deploy the model and analysis in a production environment and monitor its performance.

Tools Used :

* Jupyter notebook

**Analysis of Terapixel rendering in Cloud (Super)computing:**

**Business Understanding:**

The stake holders here want to understand the performance of the supercomputer and their nodes for rendering of the image and saving it in storage, they would like to under the resource consumption of these tasks to evaluate future projects on the same. Also, the task management by the azure system would be an addition to see if it is efficient or needs improvement.

**Objective of the Analysis:**

To perform EDA on the given data and unearth insights which might address the above business understanding. And also to address the below questions based on the analysis.

* Which event types dominate task runtimes?
* What is the interplay between GPU temperature and performance?
* What is the interplay between increased power draw and render time?
* Can we quantify the variation in computation requirements for particular tiles?
* Can we identify particular GPU cards (based on their serial numbers) whose performance differs to other cards? (i.e. perpetually slow cards).
* What can we learn about the efficiency of the task scheduling process?

**Success Criteria :**

Success criteria is do through analysis of the data and find useful insights based on above points.

**Data Understanding:**

Data:

The data files provided for is gathered by the monitoring systems and azure system in the supercomputer. The data is as follows

* Application checkpoints:

Detailing the events under tasks that is being run on different Host(GPU Node) with their timestamp.

* GPU:

GPU metrics like memory usage , Percent utilisation of the GPU Core(s),GPU temperature , power usage for different hosts monitored at different timestamps with their host details.

* Task x-y :

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

**Data Description:**

The data set can be described as tall as the values are more than variables.The column name and datatype is described below.

|  |  |
| --- | --- |
| -------------GPU------------- | |
| timestamp | object |
| hostname | Object |
| gpuSerial | int64 |
| gpuUUID | Object |
| powerDrawWatt | float64 |
| gpuTempC | int64 |
| gpuUtilPerc | int64 |
| gpuMemUtilPerc | int64 |

|  |  |
| --- | --- |
| ---APPLICATION CHECKPOINTS --- | |
| timestamp | datetime64[ns, UTC] |
| hostname | Object |
| eventName | Object |
| eventType | Object |
| jobId | Object |
| taskId | Object |

|  |  |
| --- | --- |
| TASK- X- Y | |
| taskId | object |
| jobId | object |
| x | int64 |
| y | int64 |
| level | int64 |

**Data Preparation:**

Data preparation is an important phase in the data analysis process since it guarantees that the data is in the correct format for the analysis. This process entails cleaning, transforming, and arranging data so that it can be comprehended and studied easily. The findings of the analysis may be erroneous or unreliable if the data is not properly prepared. Furthermore, data preparation can aid in identifying patterns and trends in data that would not have been obvious otherwise. Overall, data preparation is critical because it establishes the framework for accurate and useful data analysis. First step of any data is importing it and changing it to the required format. We have imported the csv files provided and have changed the

**Data Cleaning:**

Cleaning the data is so crucial that if it is not done, the analysis may yield incorrect or no results.

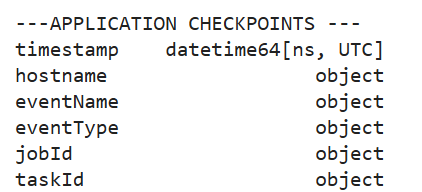
* The data has been checked for null values and there seems to be no null values
* The data has been checked for duplicate values and there seems to be duplicate values in Application checkpoint and gpu data. This is removed as the duplicates tampers analysis.



**Data Transformation :**

The data is transformed as per our requirement for our analysis.The data is checked for data type for our analysis and found that we have datetime value as object datatype and this has been changed to datetime. The below columns in data is transformed

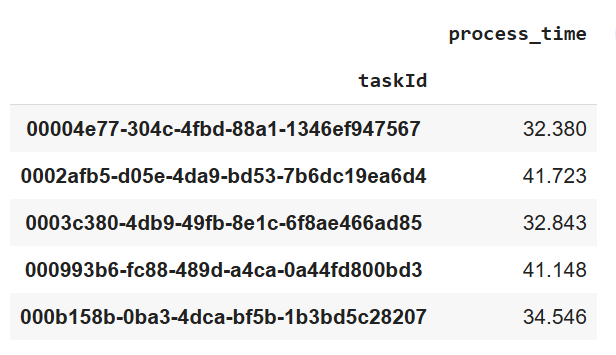
* The timestamp column in checkpoint data is transformed from object to datetime64[ns, UTC], as we need to do datetime calculation.



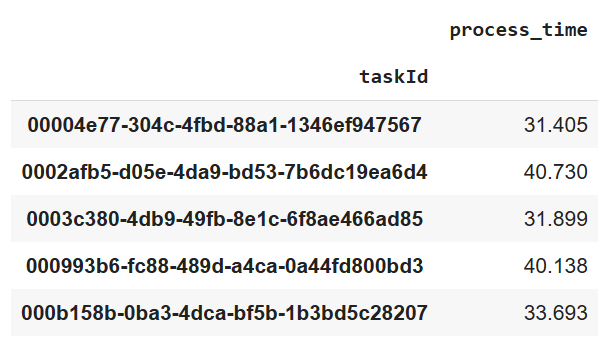
* The gpuserial is of int which is changed to string as this does not require any operations.
* Transforming the eventType column in application checkpoint data and adding start and stop time in two separate column from start event and stop event and the process time is calculated substracting the start and stop time and changing it to seconds and the data is saved in df\_app\_chk\_perf dataframe.The eventType column is dropped.



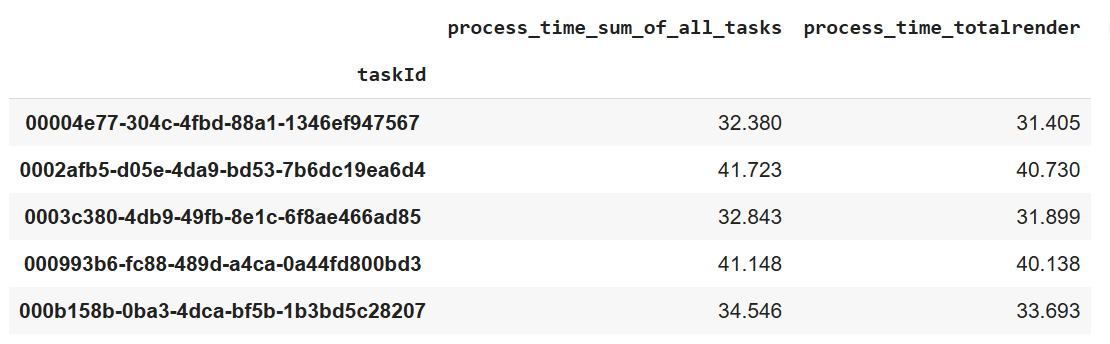
* calculating process type for all events except TotalRender and summing it by grouping by taskid and created data frame df\_app\_chk\_perf\_no\_tr\_grp from df\_app\_chk\_perf dataframe, with only task id and process time for the respective taskid.



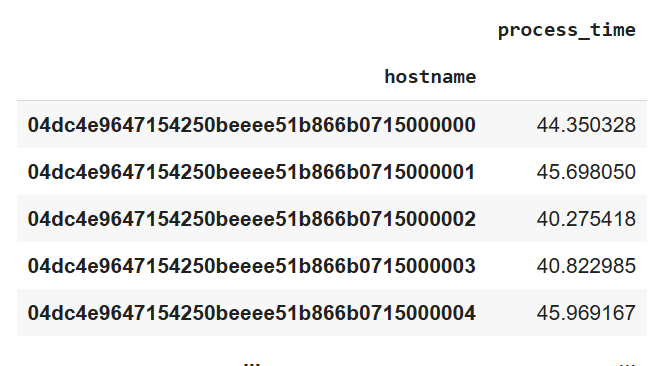
* calculating process type for event TotalRender and summing it by grouping by taskid and created data frame df\_app\_chk\_perf\_tr from df\_app\_chk\_perf dataframe, with only task id and process time for the respective taskid.



* Created df\_app\_chk\_perf\_merg data frame merging df\_app\_chk\_perf\_no\_tr\_grp and df\_app\_chk\_perf\_tr\_grp dataframe ,merged with their respective task id , the dataframe has three columns : taskid , process\_time\_sum\_of\_all\_tasks (calculated by summing), process\_time\_totalrender.



* Created host\_performance dataframe from df\_app\_chk\_perf by grouping the hostname coumn and calculating the mean of the processtime.This dataframe contains the mean of the processtime for every hostname



* The task xy datframe and filtered datasets with the process time are also combined.The combined dataset is then filtered by level to examine how long it takes to process different levels.

**Data Analysis and Exploring:**

**Exploratory Data analysis :**

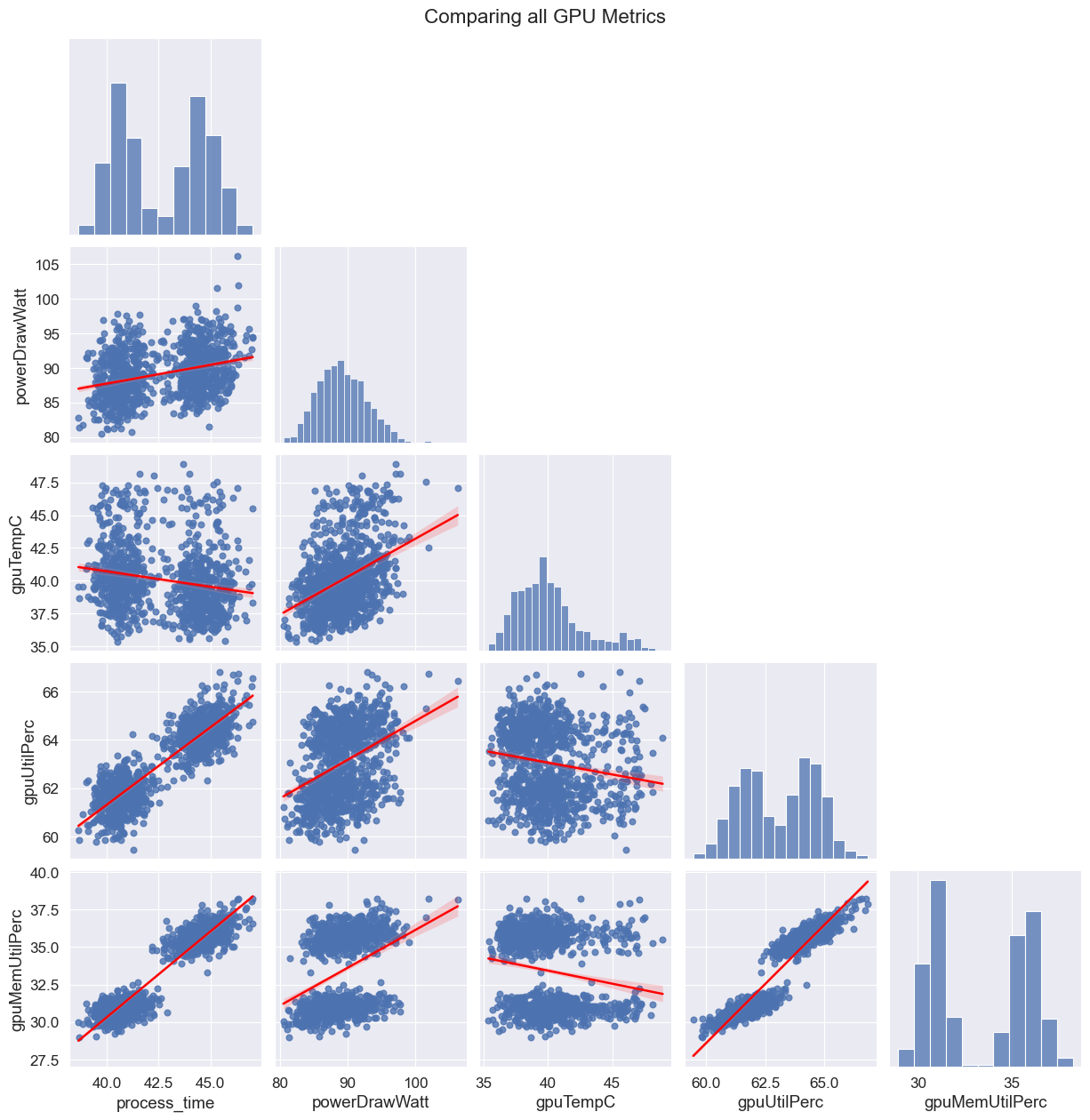


Fig : 1 Overall plot between GPU metrics and process time.

From the plot (Fig 1) we can infer that there is sharp increase in memory utilisation percent when there is a increase in processing time. Also there is sharp linearity between the two variables. The same can be said between power watt consumption and GPU Temperature which tells us that when there is increase in power consumption there is increase in temperature of the GPU.

Also whenever there is increase in GPU Memory utilisation or GPU Utilisation percent there is increase in power draw

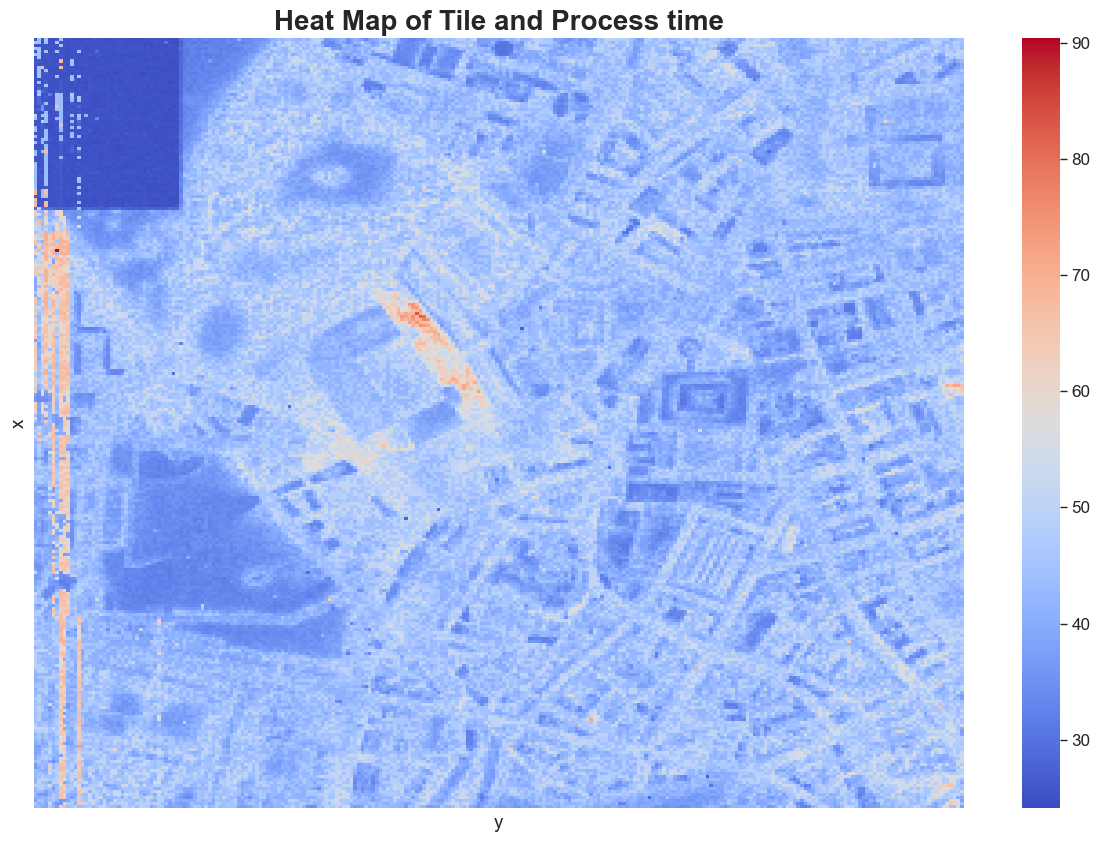


Fig 2 : Heatmap between tile and process time for each tile.

From fig 2 we can interpret that most of the tiles takes process time between 30 to 50 seconds for their processing.

**Which event types dominate task runtimes?**

From Fig 3 we can see that the total render time is the most dominant task which takes more processing time. The bar plot is plotted using the data set where the name is grouped and the process time is summed using the start and end time from eventype. we cannot consider total render time as this is the sum of all events. Hence Render is considered to be the most dominant event which required more processing time followed by Uploading and tiling. saving config is the most least dominant or less time-consuming event.

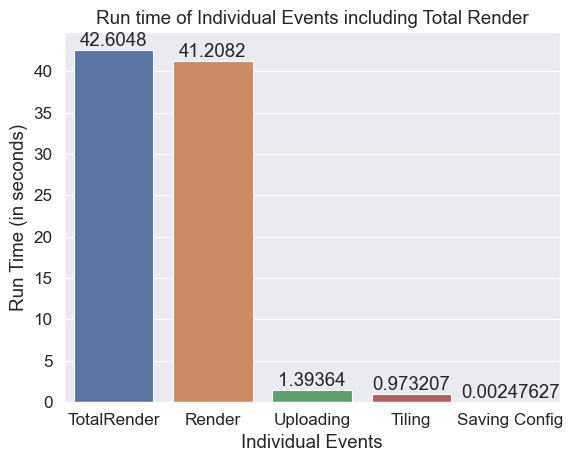


Fig 3. Run time of Individual Events including Total Render

**What is the interplay between GPU temperature and performance?**

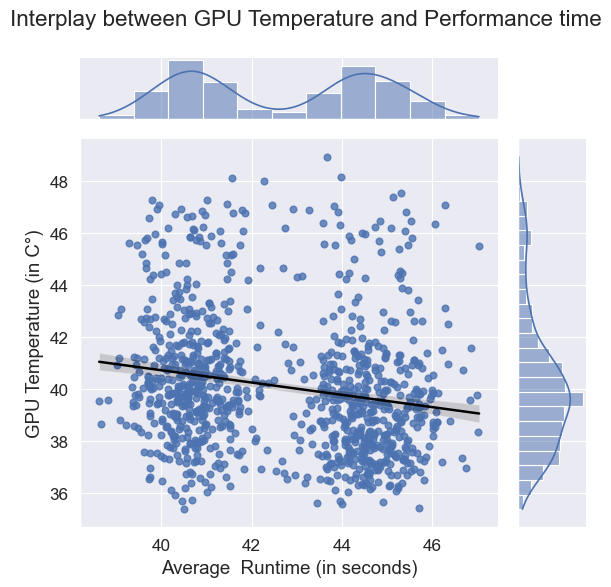


Fig.4 Interplay between GPU Temperature and Performance time

From Fig 4 we can see that there outliers and also the plot is clustered into two groups between time 40 to 42 seconds and 44 to 46 seconds. Also if you consider 40 to 42 seconds and 44 to 46 seconds we can interpret that there is a rise in temperature.

**What is the interplay between increased power draw and render time?**

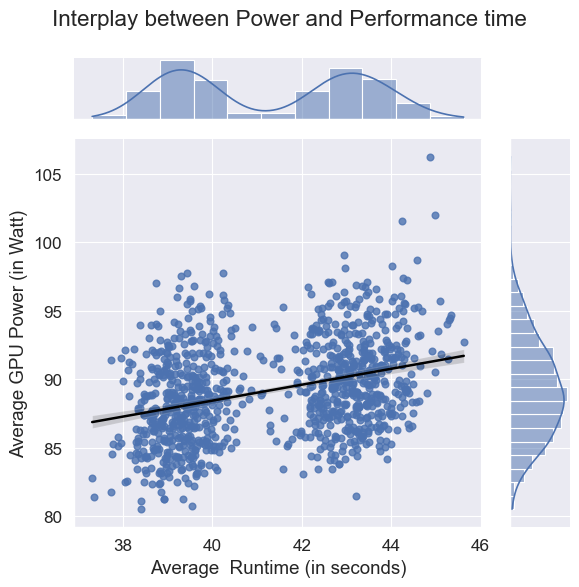


Fig.5 Interplay between Power and Performance time

From Fig 5 we can see there is rise in Gpu power when the runtime gets high. Also the cluters are formed between runtime 38 to 40 seconds and 42 to 44 seconds. Also from the plot we can see there is a line between runtime/performance time and gpu power. This can be seen as that when there more processing happening there is more power consumption . However the linearity is not very high as per the plot.

**Can we quantify the variation in computation requirements for particular tiles?**

For analysing this we consider only level 12 of the imaging as there are very less or negligible amount of data for level 4 and 8 to be considered. From fig 6 we can the heat map plotted between tile coordinates and the GPU Memory utilisation. The heat map shows that the Memory utilisation differs between each tile and we can see more tiles used memory between 62 and 65 Percent.

Fig 7 is the heatmap between power draw and tile , the plot clearly shows the poer draw is mostly between 85 to 90 watts.

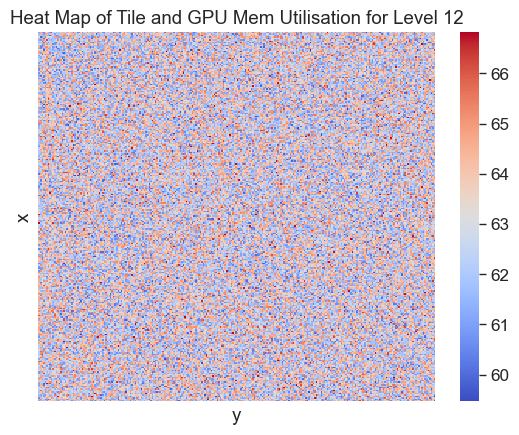


Fig 6 : Heat Map of Tile and GPU Mem Utilisation for Level 12

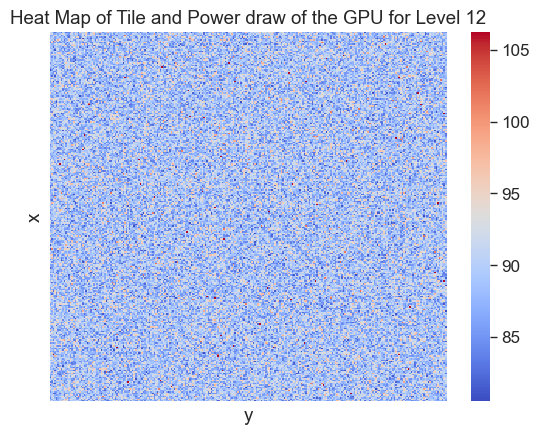


Fig : 7 Heat Map of Tile and Power draw of the GPU for Level 12

**Can we identify particular GPU cards (based on their serial numbers) whose performance differs to other cards? (i.e. perpetually slow cards)**

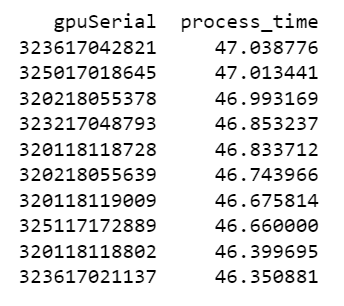
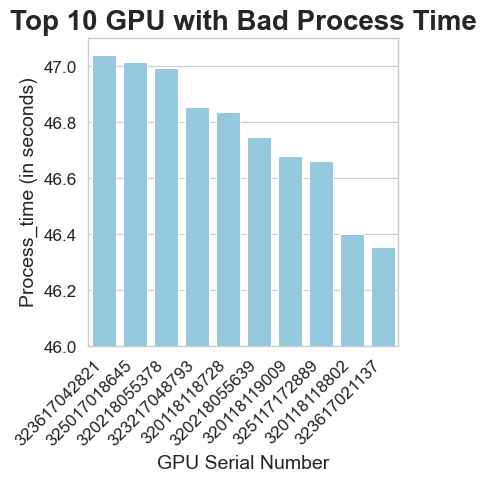


Fig 8: Top 10 GPU which used more processing time and bar plot about the same

From fig 8 we can see the top 10 GPU which had more process time and since our process time is the average we can assume that these are the worst performing GPU cards. The Fig also lists the GPU cards with their serial numbers.

**What can we learn about the efficiency of the task scheduling process?**

The Task scheduling process is handled by AZURE’s task schedular which uses a separate GPU Node. Processing of every tile is assigned a task and the tasks are for different events.every node under the 1024 nodes handles a multiple tasks and the tasks are related to different events. At any given a single node handles only one task.so the calculation time between the start of the next task subtracted by end time of the current task gives us the wait time the node waited for allocation of tasks. The below Fig.9 shows us there are delay time between tasks.

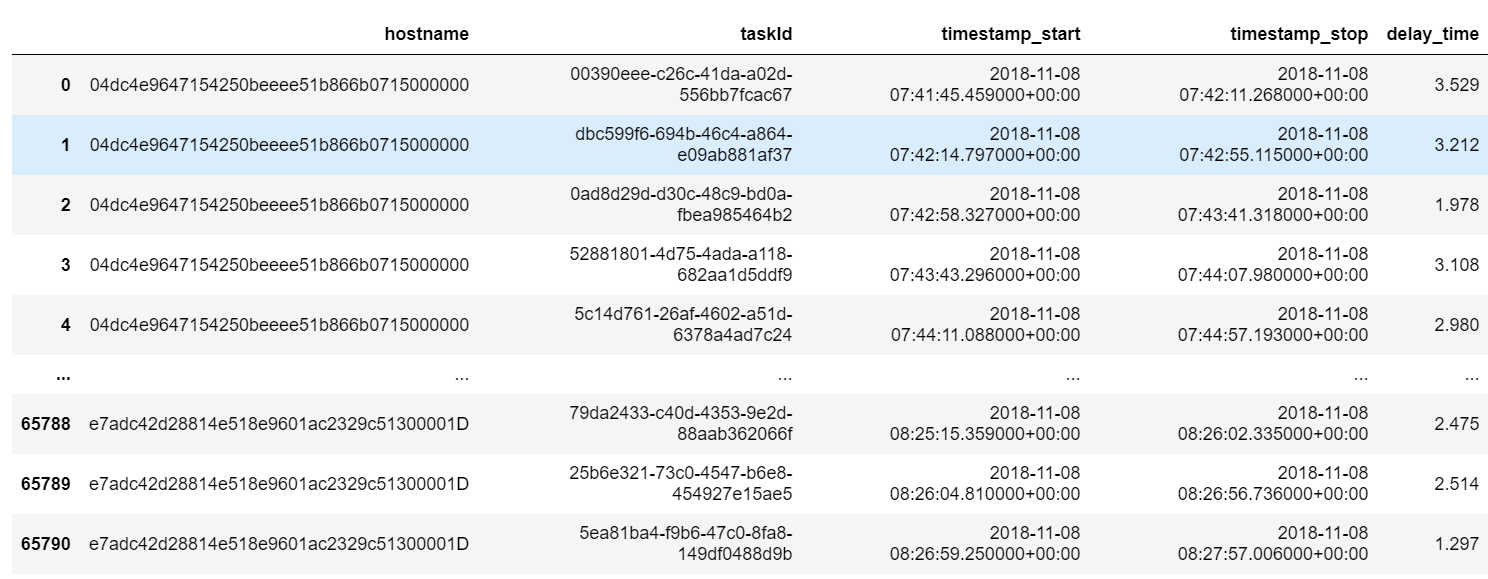


Fig 9 : Table of delay time between each task.

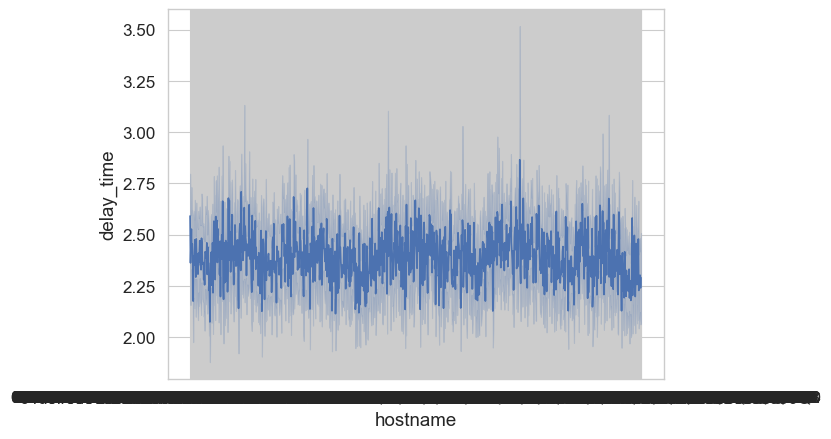


Fig 10 : line plot of the delay between tasks in seconds

From fig 10 we can see that there is delay between every tasks and this can because the task scheduler handles 1024 nodes and also manages the failed tasks and pushes it back. Also average delay time is between 2.25 to 2.75 seconds and highest is around 3.5 seconds.

Evaluation :

The analysis was done and the interpretation is summarised , on evaluating it can be said that to analyse further we need more data and more parameters from the source systems.

Usage of Github

A private repository is created for the project and code file with the changes were commited to utilise the Github.

Link : https://github.com/Ksivamurugan2022/Cloud\_Terapixel