Performance Evaluation Of Terapixel Rendering in Cloud(Super) Computing Project

Cahyadi/200511881

1/23/2021

Introduction

Cloud technology, a relatively new technology that is based on utilising the internet infrastructure to the fullest has seen a major increase in popularity in recent years. This success could be attributed to the huge growth of the Tech Industries that seen many new Unicorns; a term first mentioned by Aileen Lee on 2013, used to describe privately owned start-up companies that valued more than 1 billion USD. One of many reasons why the emergence of these "Unicorns" of Tech world brought popularity to Cloud technologies is that Cloud technologies provide a huge benefit to a start-up companies to further expanse and scale their business with greater flexibility and relatively minimum cost and commitment. Start-up companies now have an option to rent the necessary hardware infastructure instead of investing huge capital on it, since many of the established Tech companies such as Google and Amazon provide access to their extensive range of hardware to be rent. These infrastructures could then be run via virtualisation methods in which provide a great flexibility of scaling.

Cloud technologies have been utilised in many parts of the modern society, one of them being for rendering high-resolution images. The implementation of the cloud technologies in rendering task, would eliminate the hardware constraint for the client or the user. An example of this implementation could be found in rendering model in "Image-Based Network Rendering of Large Meshes for Cloud Computing" (Okamoto et al. 2010). This report would evaluate the performance of GPU nodes in a scalable cloud architecture that are used to produce a Terapixel 3D City visualization of Newcastle Upon Tyne.

To gain the information adaquate to create an evaluation on the performance of the GPU nodes capabilities to render the terapixel visualizations, a data mining process following a CRISP-DM best data mining practice would be conducted. The tools that would be used to carry out the data mining process are Rstudio combined with the Project Template package to streamlined the data mining process.

CRISP-DM Methodology

Business Objective

The scalable cloud architecture that was created to support the terapixel images rendering in order for it to be accessible for users without putting a constraint on the users hardware. Therefore it might be crucial to be able to extract performance information out of the hardware, in this case, the GPU, to further re-evaluate the amount of GPU nodes and its type needed to run the current rendering process or to create a plan for future usage. The goal of this data mining project is to be able to extract the performance evaluation information out of the current cloud-architecture that is currently use to support the terapixel image rendering. The information extracted would then be able to be used to improve the cost efficiency of running the current cloud-architecture or create future planning to prepare for an expansion.

To help achieving the goals mentioned above, this data mining project would be done using the best practice CRISP-DM methodology, and would be done entirely on R environment supported by the Project Template package and git version control to further streamline the data mining process.

Data understanding

There are three data frames available in which the data mining process could be implemented. These data frames are extracted from the application checkpoint and system metric output from the production of terapixel image. Each data frames are in the comma-separated value format in which it could be loaded directly into R by placing the data frames into the data folder of the Project Template, and loading the project. The three data frames mentioned consist of Application Checkpoint data, GPU data, and Task X Y data. Before any process could be made, an initial data exploration would be done on these data frames.

Application Checkpoints Data

Data Preview

```
## # A tibble: 6 x 6
##   timestamp     hostname        eventName  eventType jobId          taskId
##   <chr>         <chr>           <chr>      <chr>     <chr>          <chr>
## 1 2018-11-08T0~ 0d56a730076643~ Tiling     STOP      1024-lvl12-7e~ b47f0263-ba~
## 2 2018-11-08T0~ 0d56a730076643~ Saving Co~ START     1024-lvl12-7e~ 20fb9fcf-a9~
## 3 2018-11-08T0~ 0d56a730076643~ Saving Co~ STOP      1024-lvl12-7e~ 20fb9fcf-a9~
## 4 2018-11-08T0~ 0d56a730076643~ Render     START     1024-lvl12-7e~ 20fb9fcf-a9~
## 5 2018-11-08T0~ 0d56a730076643~ TotalRend~ STOP      1024-lvl12-7e~ 20fb9fcf-a9~
## 6 2018-11-08T0~ 0d56a730076643~ Render     STOP      1024-lvl12-7e~ 3dd4840c-47~
```

Data Summary

```
##   timestamp           hostname          eventName          eventType
##  Length:660400      Length:660400      Length:660400      Length:660400
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##     jobId              taskId
##  Length:660400      Length:660400
##  Class :character   Class :character
##  Mode  :character   Mode  :character
```

The Application Checkpoints Data consist of 660400 rows and 6 columns as could be seen from the first 6 rows of the data and the summary table above. For all the columns that are present on the data frame, all of them are character class, and there are no missing values detected. However, after checking for duplicates, there are 130821 duplicated individual within the data set. Therefore, moving onwards with the analyses, the data would be made sure to carry no duplicates.

Assessing each columns of the data, a following interpretations could be extracted.

- ***Timestamp*** = Contains the date and the current time of the event down to the milliseconds. The class of this column is that of a character, therefore for further analysis some processing might be needed to convert the timestamp into a numerical format.
- ***Hostname*** = Contains the Hostname of the virtual machine. The hostname is auto-assigned by the azure batch system. There are 1024 hostname recorded on the application checkpoints data.

- ***Event Name*** = Contains the name of the events that are happening on the task. There are 5 distinct events recorded on the application checkpoints data. These events are
  - Total Render = The entire task itself
  - Saving Config = Saving the configuration
  - Render = The rendering process of the image tile
  - Tiling = Post-processing of the rendered tile
  - Uploading = Uploading the output of the post-processed image tile to the Azure Blob Storage
- ***Event Type*** = Contains two distinct event Type which are STOP and START. This event type are meant to describe the checkpoint of the event name. Meaning, a data point with event name "Tiling" and event type "START" refer to the start of "Tiling" process of a certain task.
- ***Job ID*** = Contains the Azure batch job. There are three distinct value of the Job ID column, and it refers to the level of zoom (4,8,12) of the rendering proces.
- ***Task ID*** = Contains the ID of the Azure batch task. There are 65793 distinct task ID. Each task id would have all of the five event names respectively.

GPU Data

Data Preview

```
## # A tibble: 6 x 8
##   timestamp hostname gpuSerial gpuUUID powerDrawWatt gpuTempC gpuUtilPerc
##   <chr>     <chr>        <dbl> <chr>           <dbl>    <int>       <int>
## 1 2018-11-~ 8b6a0ee~   3.23e11 GPU-1d~         132.        48          92
## 2 2018-11-~ d824187~   3.24e11 GPU-04~         117.        40          92
## 3 2018-11-~ db871cd~   3.23e11 GPU-f4~         122.        45          91
## 4 2018-11-~ b9a1fa7~   3.25e11 GPU-ad~          50.2       38          90
## 5 2018-11-~ db871cd~   3.23e11 GPU-2d~         142.        41          90
## 6 2018-11-~ 265232c~   3.24e11 GPU-71~         120.        43          88
## # ... with 1 more variable: gpuMemUtilPerc <int>
```

Data Summary

```
##   timestamp           hostname           gpuSerial           gpuUUID
##  Length:1048575     Length:1048575     Min.   :3.201e+11   Length:1048575
##  Class :character   Class :character   1st Qu.:3.236e+11   Class :character
##  Mode  :character   Mode  :character   Median :3.236e+11   Mode  :character
##                                        Mean   :3.240e+11
##                                        3rd Qu.:3.250e+11
##                                        Max.   :3.252e+11
##  powerDrawWatt       gpuTempC      gpuUtilPerc     gpuMemUtilPerc
##  Min.   : 22.55   Min.   :26.00   Min.   :  0.00   Min.   : 0.00
##  1st Qu.: 45.15   1st Qu.:38.00   1st Qu.:  0.00   1st Qu.: 0.00
##  Median : 96.68   Median :40.00   Median : 89.00   Median :43.00
##  Mean   : 89.32   Mean   :40.07   Mean   : 63.23   Mean   :33.47
##  3rd Qu.:121.34   3rd Qu.:42.00   3rd Qu.: 92.00   3rd Qu.:51.00
##  Max.   :197.01   Max.   :55.00   Max.   :100.00   Max.   :83.00
```

Extracting the information from the data preview and the data summary above, the GPU data consist of 1048575 rows and 8 columns. The columns are consisted of Timestamp, Hostname, GPU serial number,

GPU UUID, Power Draw on GPU in Watt, GPU Temperature in Celsius, Percent Utilisation of GPU Cores, and Percent Utilisation of the GPU Memory. For the timestamp, hostname, and GPU UUID, the class of column is listed as characters and for the rest of the columns, they are listed as continuous data. However, upon closer inspection it could be deducted that GPU Serial should not be regarded as a continuous data since it refers to serial number of the physical GPU card. It seems that this data set also suffers from duplicated values, and after some thorough look at the data set there are 662118 duplicated values found on the data. Similar to the application checkpoints data, the information regarding the duplicated values within the data should be taken into consideration for further analyses to make sure that these duplicated values are not carried over.

Assessing each columns of the GPU data, a following interpretations could be extracted.

- ***Timestamp*** = Similar to the Application Checkpoints Data,the timestamp column for the GPU data contains the current time in which the condition is being recorded. It shows the date as well as the time down to the milliseconds. There is an interesting feature that is found on the timestamp for the GPU data. The timestamp shows that for each hostname, the GPU condition of each hostname is recorded every **2 Seconds**.
- ***Hostname*** = Similar to the Application Checkpoints data, contains the hostname of the virtual machine that is auto-assigned by the Azure Batch system. The number of unique hostname recorded on this data is equal to the number of hostname that is recorded for the Application Checkpoints data, which is 1024 unique hostnames. Therefore, this information could prove to be useful for the analyses on the later part of this report.
- ***GPU Serial*** = Contains the serial number of the physical GPU card. Even though on the data summary the GPU serial column is treated as a continous variable, on reality, this column should be treated as a categorical variable.
- ***GPU UUID*** = Contains the unique system id which is assigned by the Azure system to the GPU Unit. Note that GPU UUID is unique for each hostname since the number of unique GPU UUID is equal to that of the number of hostname at 1024 unique data points.
- ***GPU Power Draw in Watt*** = Contains the current recorded value of the Power Draw of the GPU in Watt. This variable is a continuous variable. It has a minimum recorded value of 22.55 Watt and a maximum recorded value of 197.01 Watt.
- ***GPU Temperature in Celcius*** = Contains the current recorded Temperature of the GPU in Celcius. This variable is a continuous variable in which the minimum recorded value for this variable is 26 degree Celsius and the maximum recorded value of 55 degree Celsius.
- ***GPU Util Perc*** = Contains the current recorded value of the Percent Utilisation of the GPU cores. This variable is a continuous variable ranging from 0% to 100%. It has a minimum recorded value of 0% and a maximum recorded value of 100%.
- ***GPU Memory Util Perc*** = Contains the current recorded value of the Percent Utilisation of the GPU memory. This variable is continuous variable ranging from 0% to 100%. It has a minimum recorded value of 0% and a maximum recorded value of 83%

Task X Y Data

Data Preview

```
## # A tibble: 6 x 5
##   taskId                       jobId                                   x     y level
##   <chr>                        <chr>                               <int> <int> <int>
## 1 00004e77-304c-4fbd-88a1-1~ 1024-lvl12-7e026be3-5fd0-48ee-b7~   116   178    12
## 2 0002afb5-d05e-4da9-bd53-7~ 1024-lvl12-7e026be3-5fd0-48ee-b7~   142   190    12
## 3 0003c380-4db9-49fb-8e1c-6~ 1024-lvl12-7e026be3-5fd0-48ee-b7~   142    86    12
## 4 000993b6-fc88-489d-a4ca-0~ 1024-lvl12-7e026be3-5fd0-48ee-b7~   235    11    12
## 5 000b158b-0ba3-4dca-bf5b-1~ 1024-lvl12-7e026be3-5fd0-48ee-b7~   171    53    12
## 6 000d1def-1478-40d3-a5e3-4~ 1024-lvl12-7e026be3-5fd0-48ee-b7~   179   226    12
```

Data Summary

```
##     taskId             jobId                x             y
##  Length:65793       Length:65793       Min.   :  0   Min.   :  0
##  Class :character   Class :character   1st Qu.: 63   1st Qu.: 63
##  Mode  :character   Mode  :character   Median :127   Median :127
##                                        Mean   :127   Mean   :127
##                                        3rd Qu.:191   3rd Qu.:191
##                                        Max.   :255   Max.   :255
##      level
##  Min.   : 4.00
##  1st Qu.:12.00
##  Median :12.00
##  Mean   :11.98
##  3rd Qu.:12.00
##  Max.   :12.00
```

Extracting information out of the Data Preview and the Data Summary above, the Task-X-Y Data consist of 6579 rows and 5 columns. The number of rows are equal to the number of task as shown by the taskId column. It has recuring columns from the Application Checkpoints data in taskId and JobId. This information should be taken into consideration when finding the correlation between the Application Checkpoints data and the Task-X-Y data. The Task-X-Y data columns that shows coordinate "X" and "Y" is deemed by the R system as a column of continouus variable, since the values inside the column are consisted of numerical value. However, note that since it only shows the coordinate of an object, the stats value extracted from these two columns might not be that beneficial for the analyses, although it still could be useful for determining whether a certain coordinates affect the performance of the rendering task. The "level" column is shown as a continuous variable on the data summary even though the "level" column represent zoom level of the rendered visualisation.

Assessing each column of the Task-X-Y data, a following interpretations could be extracted.

- ***Task ID*** = Similar to the Application Checkpoints data, this column contains the ID of the Azure Batch task. Since it shows the same number of unique taskId as that of the Application Checkpoints data with 65793 unique task ID, it should be taken into consideration to link the task id from the Task-X-Y data with the task id in the Application Checkpoints data.
- ***Job ID*** = Similar to the Application Checpoints data, this column contains Azure batch job Id
- ***X*** = Contains the X coordinate of the rendered image tile.
- ***Y*** = Contains the Y coordinate of the rendered image tile.
- ***Level*** = Contains the zoom levels of the render. The visualisation is created with 12 zoom levels starting with 1 until 12, however, for the data set in this project, only zoom level 4,8, and 12 that are present since the intermediate level are derived in the tiling process. Note that the zoom level is also reflected in the JobId column, as jobId also mentions the level of zoom in its character.

Data Preparation

Before analyses would be made, the data frames mentioned and described on the Data Understanding section, would need to be processed first. The process includes but not limited to data wrangling, removing duplicates, merging data frames, and extracting new features from the data. In this part of the report, the processes that are applied to the data frames would be discussed and explanations would be given.

  Before any data merging could be done, each individual data frame would be processed first. The data processing is done based on the information gained from the assessment of each data frame that is highlighted in the Data Understanding section of this report.

Application Checkpoints Data

Referring to the initial data assessment of the Application Checkpoints data frame, data points duplicates would be removed. Removing the duplicates on the Application Checkpoints would leave us with a data frame which has 657930 data points and removing 130821 duplicates. The reason to remove any duplicates is to avoid having any error during the analyses or producing an analysis that is skewed.

The ***"timestamp"*** column of the Application Checkpoints could not be directly used as a method of analyses since the type of values that are present in the column are characters. Therefore we would process the ***"timestamp"*** column and extract only the numerical time value out of the ***"timestamp"***. The dates that are recorded on the ***"Timestamp"*** are all showing the same date which is **2018-11-08** with varying times. Therefore, for the analyses, only the time would be extracted and would be converted into the Hour:Min:Sec format.

For the *Event Name* column, the number of data points for each event names seems to be the same, indicating that for all the task that is present in the data, each task includes all possible event names. The Distribution table of the *Event Name* column is shown as below

```
##
##        Render Saving Config        Tiling   TotalRender     Uploading
##        131586        131586        131586        131586        131586
```

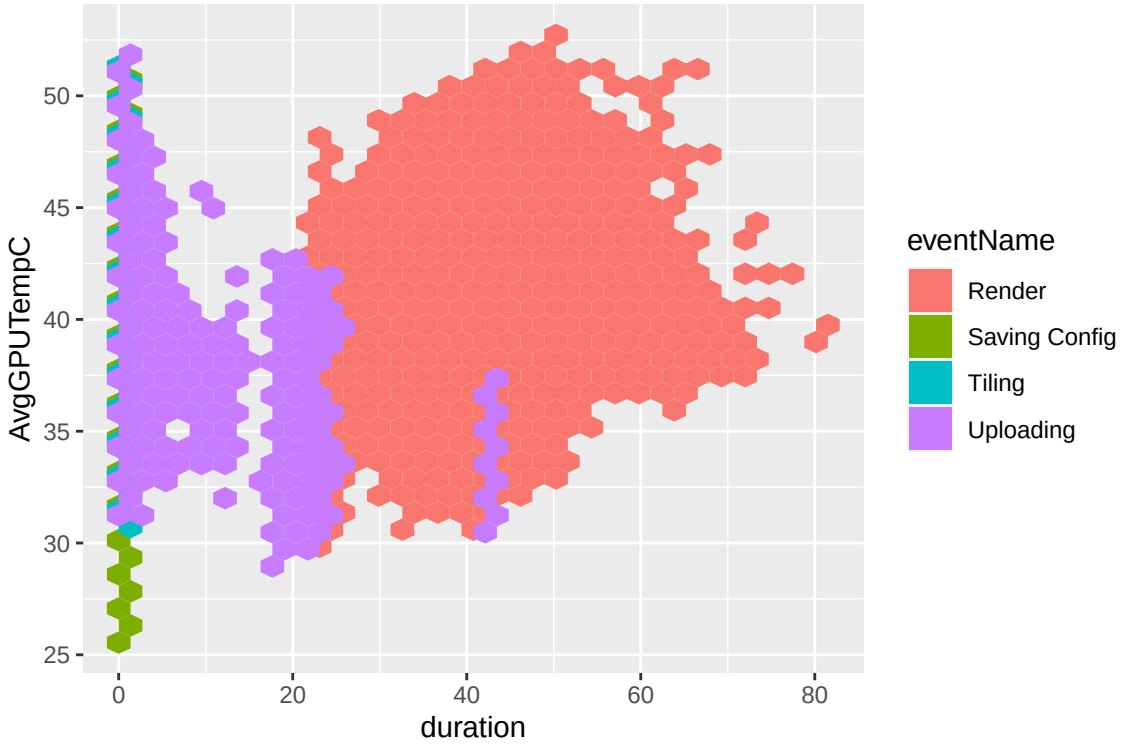
Each event name recorded on the data 131586 times, which if divided by the number of **Event Type** of each Event Name which is 2, would equal to the amount of distinct **Task ID**. With these information at hand,we would proceed with wrangling the data, to prepare it for analyses. The wrangling process are done in these manners

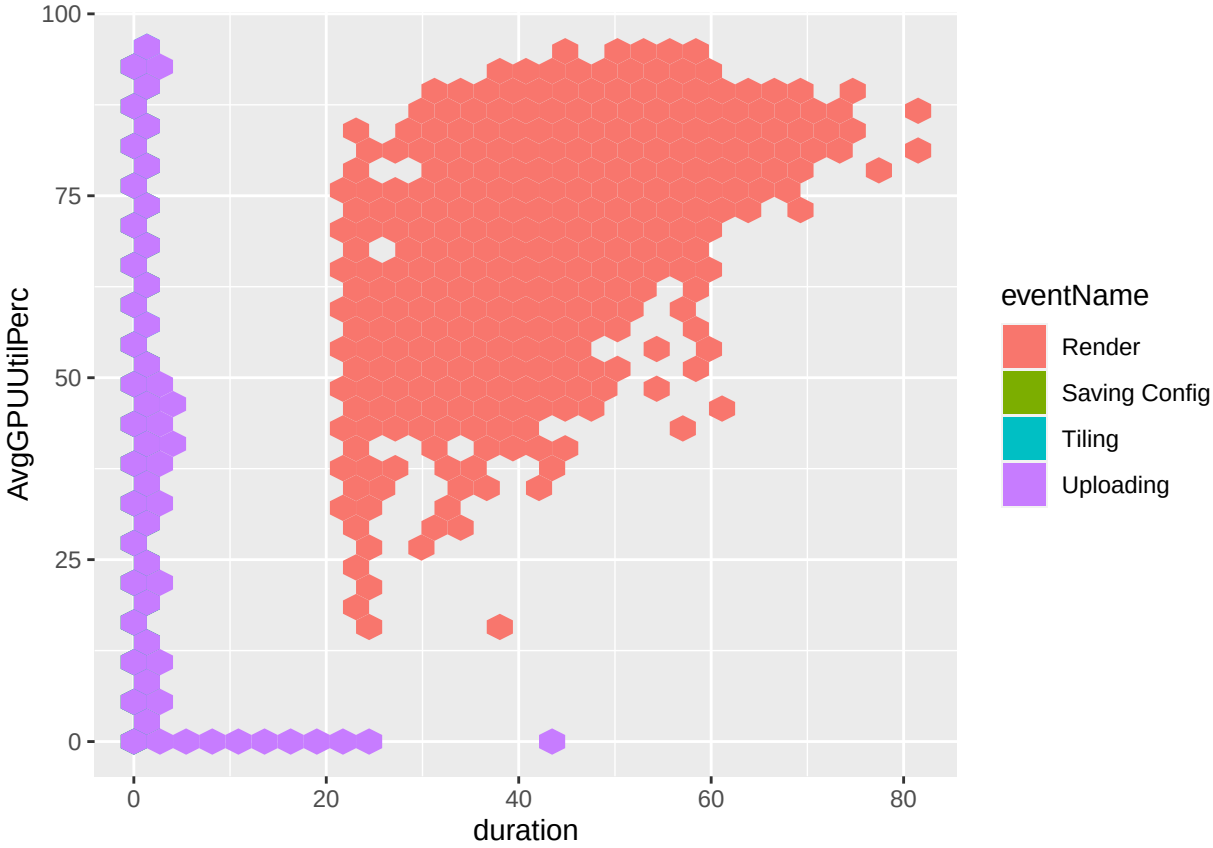
1. Ensure that the Applicaton Checkpoints data does not contains any duplicated data points. If there are any, remove the duplicated data points.
2. Extract only the time characters out of the ***Timestamp*** column and change the format to that of an Hour:Minute:Second format.

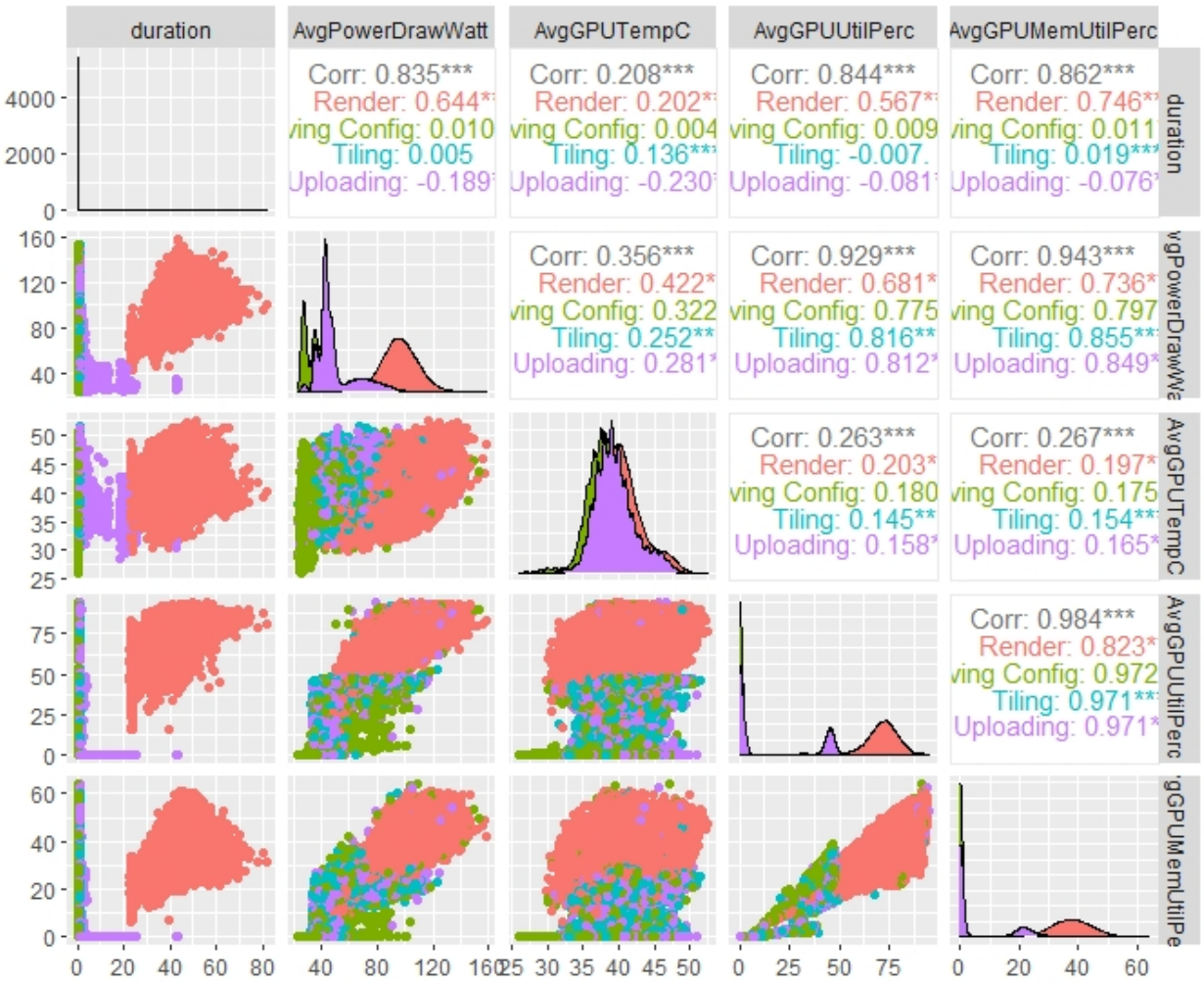
3. Transform the ***Event Type*** row into two separate columns which are ***START*** and ***STOP*** .
4. Fill these new columns ***START*** and ***STOP*** with the **Timestamp** for each **Event Name** which has **Event Type** that matched each of these two new columns.
5. Create a new feature called **Duration** which would store the duration of each event, which is calculated by subtracting the ***STOP*** and ***START*** column. This newly made feature **Duration** is what would be used in analyses as a metric that describe the length of an event.

The resulting data frame would be

```
## # A tibble: 6 x 7
##   hostname eventName jobId taskId STOP           START
##   <chr>    <chr>     <chr> <chr>  <Period>       <Period>
## 1 0d56a73~ Tiling    1024~ b47f0~ 7H 41M 55.921S 7H 41M 55.2S
## 2 0d56a73~ Saving C~ 1024~ 20fb9~ 7H 42M 29.845S 7H 42M 29.842S
## 3 0d56a73~ Render    1024~ 20fb9~ 7H 43M 10.965S 7H 42M 29.845S
## 4 0d56a73~ TotalRen~ 1024~ 20fb9~ 7H 43M 13.957S 7H 42M 29.842S
## 5 0d56a73~ Render    1024~ 3dd48~ 7H 43M 56.239S 7H 43M 16.506S
## 6 0d56a73~ Uploading 1024~ 3dd48~ 7H 43M 57.245S 7H 43M 56.239S
## # ... with 1 more variable: duration <Duration>
```


AvgGPUTempC
50
45
40
35
30
25
0
20
40
60
80
duration
eventName
Render
Saving Config
Tiling
Uploading


AvgGPUUtilPerc
100
75
50
25
0
0
20
40
60
80
duration
eventName
Render
Saving Config
Tiling
Uploading


duration
AvgPowerDrawWatt
AvgGPUTempC
AvgGPUUtilPerc
AvgGPUMemUtilPerc
Corr: 0.835***
Render: 0.644**
ving Config: 0.010
Tiling: 0.005
Uploading: -0.189
Corr: 0.208***
Render: 0.202**
ving Config: 0.004
Tiling: 0.136***
Uploading: -0.230
Corr: 0.844***
Render: 0.567**
ving Config: 0.009
Tiling: -0.007.
Uploading: -0.081
Corr: 0.862***
Render: 0.746**
ving Config: 0.011
Tiling: 0.019***
Uploading: -0.076
Corr: 0.356***
Render: 0.422*
ving Config: 0.322
Tiling: 0.252**
Uploading: 0.281*
Corr: 0.929***
Render: 0.681*
ving Config: 0.775
Tiling: 0.816**
Uploading: 0.812*
Corr: 0.943***
Render: 0.736*
ving Config: 0.797
Tiling: 0.855**
Uploading: 0.849*
Corr: 0.263***
Render: 0.203*
ving Config: 0.180
Tiling: 0.145**
Uploading: 0.158*
Corr: 0.267***
Render: 0.197*
ving Config: 0.175
Tiling: 0.154**
Uploading: 0.165*
Corr: 0.984***
Render: 0.823*
ving Config: 0.972
Tiling: 0.971**
Uploading: 0.971*