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

Peiqiang Li - 200987503

## Introduce

"Cloud computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the data centers that provide those services"[1].Cloud computing concentrates computing and storage at the heart of the service, and high-bandwidth connections are adopted to link high-performance machines, so that all these resources could be carefully managed. The user sends a computing request to the cloud computing service and obtains the final result[2]. In this project, Cloud (super)computing is used for high quality terapixel visualization. The visualization of trillions of pixels requires powerful computing resources. Therefore, cloud-based supercomputer resources provide greater flexibility for visualization.

## Methodology

The structure of the project is built by **ProjectTemplate**. As we all know, the correctness and reproducibility of data science code is very important. Thus, an appropriate project template can not only improve the efficiency of development, but also make the code and documents more concise and orderly.

Git is used as a version control tool for this project. All development activities are being recorded using Git and stored in a private repository. Due to the size and privacy of dataset, the raw data will not be uploaded to the GitHub repository.

Rstudio is the development tool of this project, The project process adopts the data exploratory analysis steps of **CRISP-DM** model. And, the literate programming framework is used to record the results of data analysis and generate the final project report.

README file will be used to record the analysis steps of the project and the reproduction process of the analysis.

## Business Understanding

The realistic terapixel visualization of the city of Newcastle upon Tyne is calculated by cloud-based supercomputer resources. In addition, the rendered 3D visual city also supports daily updates, hence it is necessary to evaluate the scalable architecture of cloud supercomputer for optimal performance.

The image rendering process is mainly completed by GPU. Therefore,

The project detail see link.

# Data Understanding

The dataset collected from the project link above. It contains the running information of ‘1024 1024 GPU nodes. The running task is divided into three parts, which are used to render the visualization output of level 4, 8 and 12 respectively. In this dataset we can get detailed information about the performance timings of the render application, the performance of the GPU card, and which part of the image is being rendered in each task. The size of the dataset, the number of fields and the specific explanations are shown in the following tables:

The application.checkpoints dataset contains 660400 data and 6 fields.

| | Fields | Info |
|---|---|---|
| 1 | timestamp | Timestamp |
| 2 | hostname | Hostname of the virtual machine auto-assigned by the Azure batch system. |
| 3 | eventName | Name of the event occuring within the rendering application. |
| 4 | eventType | Possible Values: "START", "STOP" |
| 5 | jobId | ID of the Azure batch job(corresponding to three job tasks). |
| 6 | taskId | ID of the Azure batch task. |

Table 1: dataset information of application.checkpoints

The explain of `eventName` values:

- **TotalRender** is the entire task(sum of other events running time).
- **Render** is when the image tile is is being rendered.
- **Saving Config** is simply a measure of configuration overhead.
- **Tiling** is where post processing of the rendered tile is taking place.
- **Uploading** is where the output from post processing is uploaded to Azure Blob Storage.

examples:

```
## # A tibble: 6 x 6
##   timestamp           hostname      eventName  eventType jobId        taskId
##   <dttm>              <chr>         <chr>      <chr>     <chr>        <chr>
## 1 2018-11-08 07:41:55 0d56a7300766~ Tiling     STOP      1024-lvl12-~ b47f0263-~
## 2 2018-11-08 07:42:29 0d56a7300766~ Saving Co~ START     1024-lvl12-~ 20fb9fcf-~
## 3 2018-11-08 07:42:29 0d56a7300766~ Saving Co~ STOP      1024-lvl12-~ 20fb9fcf-~
## 4 2018-11-08 07:42:29 0d56a7300766~ Render     START     1024-lvl12-~ 20fb9fcf-~
## 5 2018-11-08 07:43:13 0d56a7300766~ TotalRend~ STOP      1024-lvl12-~ 20fb9fcf-~
## 6 2018-11-08 07:43:56 0d56a7300766~ Render     STOP      1024-lvl12-~ 3dd4840c-~
```

The GPU dataset contains 1543681 data and 8 fields.

| | Fields | Info |
|---|---|---|
| 1 | timestamp | Timestamp |
| 2 | hostname | Hostname of the virtual machine auto-assigned by the Azure batch system. |
| 3 | gpuSerial | The serial number of the physical GPU card. |
| 4 | gpuUUID | The unique system id assigned by the Azure system to the GPU unit. |
| 5 | powerDrawWatt | Power draw of the GPU in watts. |
| 6 | gpuTempC | Temperature of the GPU in Celsius |
| 7 | gpuUtilPerc | Percent utilisation of the GPU Core(s). |
| 8 | gpuMemUtilPerc | Percent utilisation of the GPU memory. |

Table 2: dataset information of gpu.csv

examples:

```
## # A tibble: 6 x 8
##   timestamp           hostname      gpuSerial gpuUUID     powerDrawWatt gpuTempC
##   <dttm>              <chr>             <dbl> <chr>               <dbl>    <int>
## 1 2018-11-08 08:27:10 8b6a0eebc87b~   3.23e11 GPU-1d1602~         132.        48
## 2 2018-11-08 08:27:10 d8241877cd99~   3.24e11 GPU-04a2de~         117.        40
## 3 2018-11-08 08:27:10 db871cd77a54~   3.23e11 GPU-f45979~         122.        45
## 4 2018-11-08 08:27:10 b9a1fa7ae2f7~   3.25e11 GPU-ad773c~          50.2       38
## 5 2018-11-08 08:27:10 db871cd77a54~   3.23e11 GPU-2d4eed~         142.        41
## 6 2018-11-08 08:27:10 265232c5f681~   3.24e11 GPU-717654~         120.        43
## # ... with 2 more variables: gpuUtilPerc <int>, gpuMemUtilPerc <int>
```

---

The task dataset contains 65793 data and 5 fields and the specific information of **task-x-y.csv**:

- **jobId**: Id of the Azure batch job.
- **taskId**: ID of the Azure batch task.
- **x**: X co-ordinate of the image tile being rendered.
- **y**: Y co-ordinate of the image tile being rendered.
- **level**: The visualisation created is a zoomable "google maps style" map. In total we create 12 levels. Level 1 is zoomed right out and level 12 is zoomed right in. You will only see levels 4, 8 and 12 in the data as the intermediate level are derived in the tiling process.

examples:

```
## # A tibble: 6 x 6
##   taskId                               jobId           x     y level hostname
##   <chr>                                <chr>       <int> <int> <int> <chr>
## 1 00004e77-304c-4fbd-88a1-1346ef947567 1024-lvl12~   116   178    12 0745914f4d~
## 2 0002afb5-d05e-4da9-bd53-7b6dc19ea6d4 1024-lvl12~   142   190    12 83ea61ac1e~
## 3 0003c380-4db9-49fb-8e1c-6f8ae466ad85 1024-lvl12~   142    86    12 83ea61ac1e~
## 4 000993b6-fc88-489d-a4ca-0a44fd800bd3 1024-lvl12~   235    11    12 cd44f5819e~
## 5 000b158b-0ba3-4dca-bf5b-1b3bd5c28207 1024-lvl12~   171    53    12 8b6a0eebc8~
## 6 000d1def-1478-40d3-a5e3-4f848daee474 1024-lvl12~   179   226    12 b9a1fa7ae2~
```

## Data Preparation

```
table(duplicated(application.checkpoints))
table(duplicated(task.x.y))
table(duplicated(gpu))
```

At the First, We need to make sure that the dataset is not affected by duplicate data. We found that the dataset of application.checkpoints have 2470 duplicate data, and the gpu dataset have 9 duplicate data. After remove the duplicate data the following information can be summarized.

```
table(task.x.y$jobId)
```

1. The task.x.y dataset contains 65793 data.
    - These all 65793 tasks is undering the 3 jobs.
2. The application.checkpoints dataset contains 657930 data.
    - every task have ten events: "Tiling"-"START" and "STOP" "Saving Config"-"START" and "START" "Render"-"START" and "START" "TotalRender"-"START" and "START" "Uploading"-"START" and "START"
3. the gpu dataset contains 1543672 data.
    - every gpu have a unique hostname.

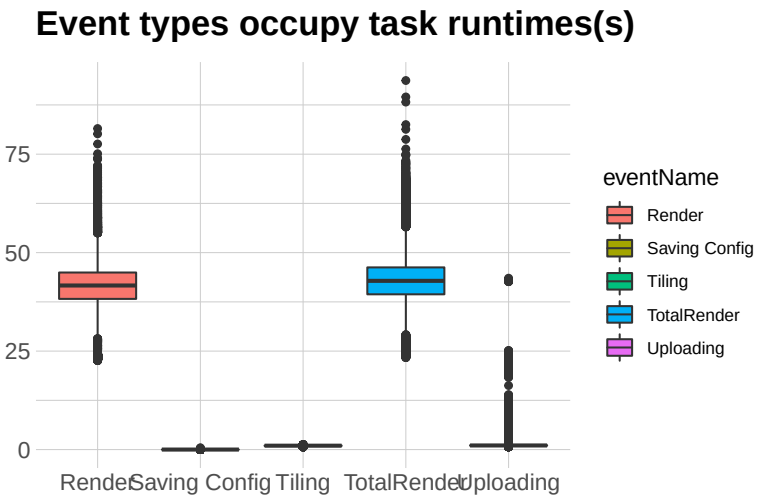
```
unique(task.x.y$jobId)
# find how many gpu render this level
length(unique(task.x.y[task.x.y$level==12,]$hostname))
length(unique(task.x.y[task.x.y$level==4,]$hostname))
length(unique(task.x.y[task.x.y$level==8,]$hostname))
```

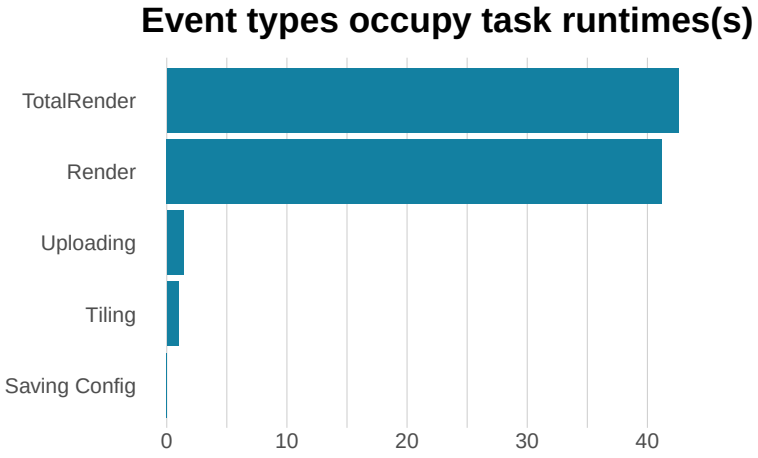
The 3 `jobId` corresponding to the rendering of 3 levels.

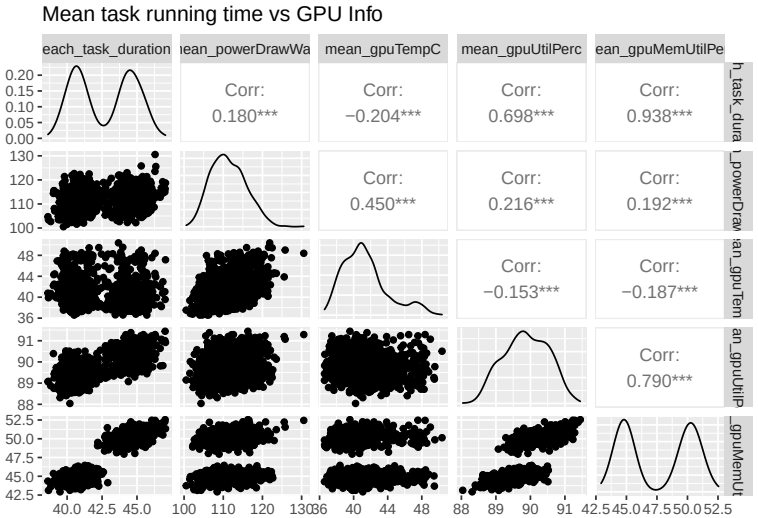
- "1024-lvl12-7e026be3-5fd0-48ee-b7d1-abd61f747705", the rendering of level 12, have 65536 task records, 1024 gpu rendering.
- "1024-lvl4-90b0c947-dcfc-4eea-a1ee-efe843b698df", the rendering of level 4, have 1 task records, 1 gpu rendering.
- "1024-lvl8-5ad819e1-fbf2-42e0-8f16-a3baca825a63", the rendering of level 8, have 256 task records, 256 gpu rendering.

# Data Analysis

## The event types dominate task runtimes

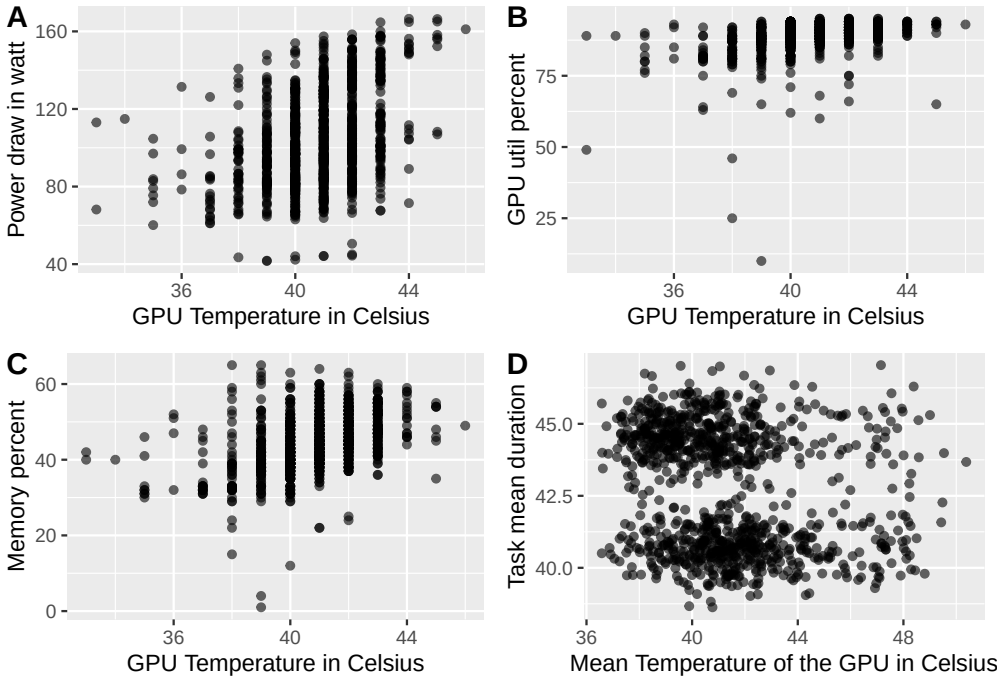


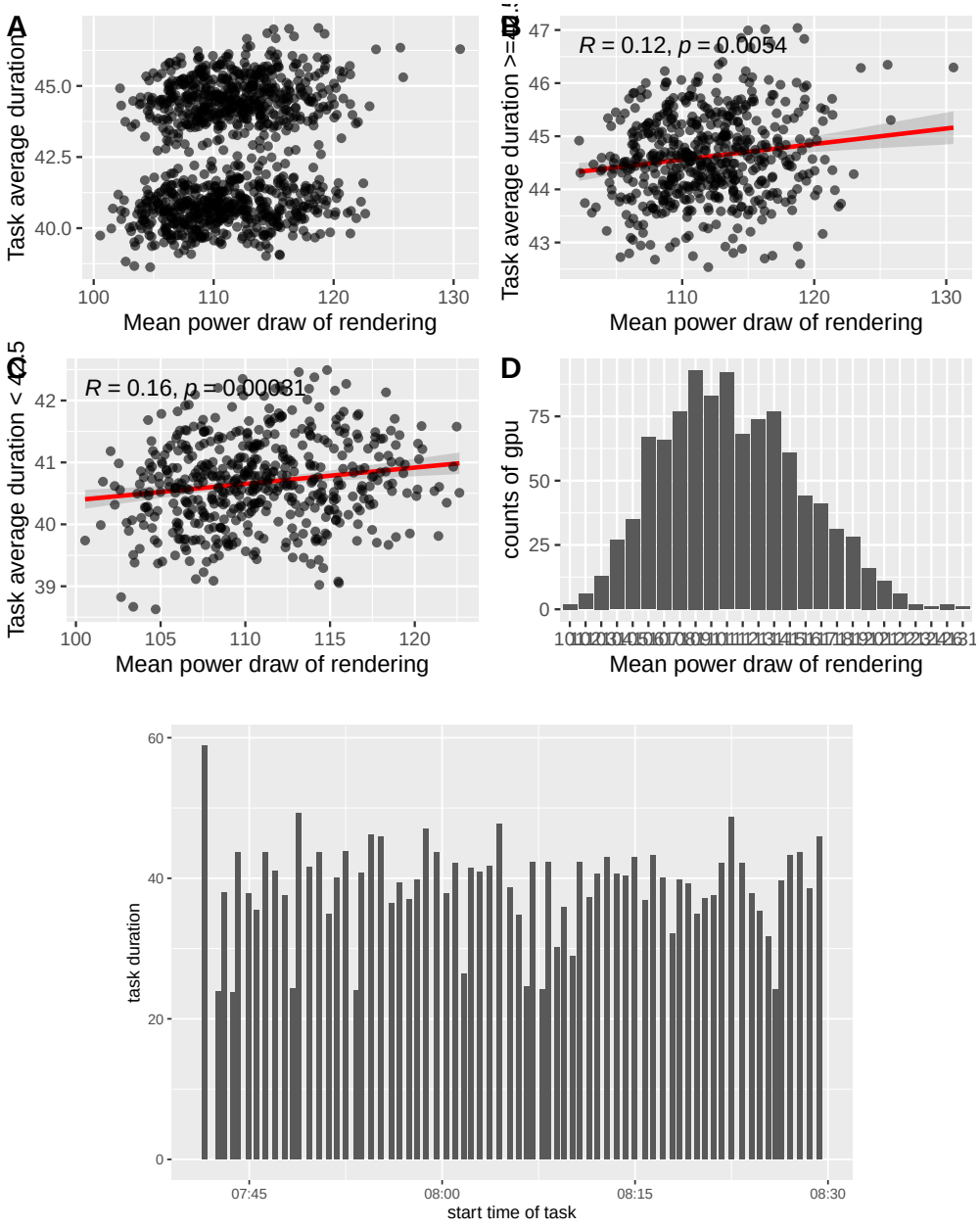
## Interplay between GPU temperature and performance

It is very difficult to measure the performance of GPU. Therefore, the power draw, percent utilisation of GPU and percent utilisation of GPU memory and the average task rendering duration of each GPU are used to represent GPU performance. The average task rendering duration comes from the number of tasks rendered by the GPU and the sum of the duration of all these tasks.

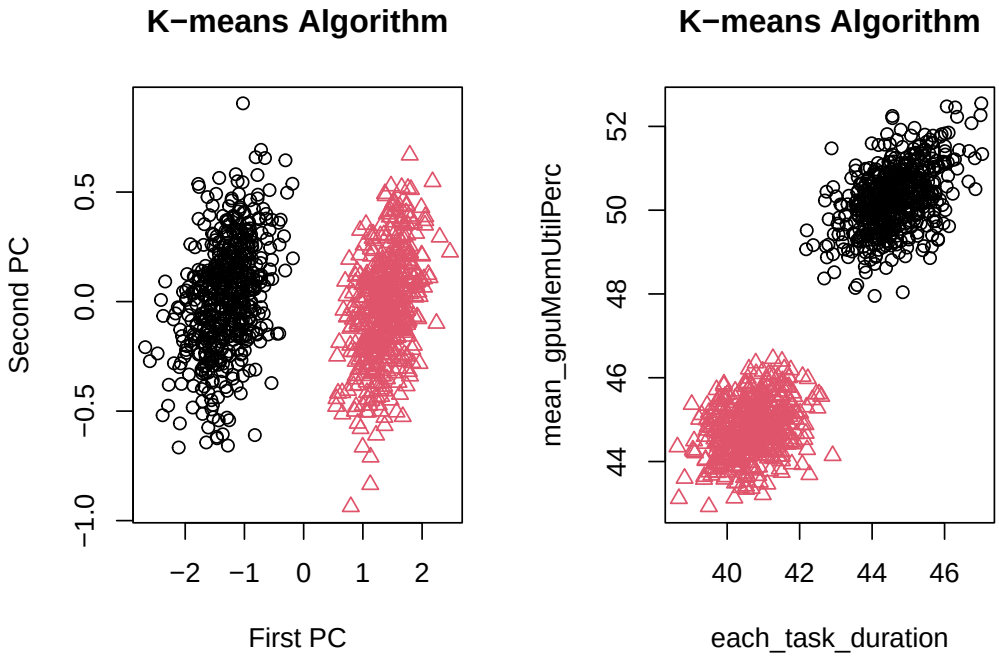



As a result, based on these four figures, there is no obvious correlation between GPU temperature and GPU performance.

## Interplay between increased power draw and render time



### Identify particular GPU cards




## Conclusoin

# References


[1] Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I. and Zaharia, M., 2010. A view of cloud computing. Communications of the ACM, 53(4), pp.50-58.

[2] Hayes, B., 2008. Cloud computing.

[3] Christensen, R., 2006. Log-linear models and logistic regression. Springer Science & Business Media.

[4] McLeod, A.I. and Xu, C., 2010. bestglm: Best subset GLM. URL http://CRAN. R-project. org/package= bestglm.

[5] McNeish, D.M., 2015. Using lasso for predictor selection and to assuage overfitting: A method long overlooked in behavioral sciences. Multivariate Behavioral Research, 50(5), pp.471-484.