report

Yang Cong

2022/1/20

Introduction

Background

Terapixel images offer an intuitive, accessible way to present information sets to stakeholders, allowing viewers to interactively browse big data across multiple scales. The challenge we addressed here is how to deliver the supercomputer scale resources needed to compute a realistic terapixel visualization of the city of Newcastle upon Tyne and its environmental data as captured by the Newcastle Urban Observatory.

Our solution is a scalable architecture for cloud-based visualization that we can deploy and pay for only as needed. 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.

We demonstrate that it is feasible to produce a high quality terapixel visualization using a path tracing renderer in under a day using public IaaS cloud GPU nodes. Once generated the terapixel image supports interactive browsing of the city and its data at a range of sensing scales from the whole city to a single desk in a room, accessible across a wide range of thin client devices.

Need for the project

The dataset was created from application checkpoint and system metric output from the production of a terapixel image. There are varieties of problems that can be addressed using the TeraScope dataset. What we need is to use what we learned to explore and analyse the Data.

Explore and analyse

Data Process

First, I decide to clean and deduplicate the data. Then look at the first question, that is, 'Which event types dominate task runtimes?'. To solve this question, I count the hostname for each data set and check whether both data set the hostname is the same using the setequal function of dplyr.

To get which event dominate task runtimes, we first should create a new data frame to compute the task runtimes and change time to hms format. I also create a new feature called duration which features the running time for each event. The new data frame we finally get is bellow.

```
## # A tibble: 6 x 8
##   hostname    jobId     taskId   Tiling             `Saving Config`      Render
##   <chr>       <chr>     <chr>    <Duration>         <Duration>           <Durat>
## 1 0d56a73007~ 1024-lvl~ b47f026~ 0.720999999999997s 0.00199999999999889s 23.501s
## 2 0d56a73007~ 1024-lvl~ 20fb9fc~ 0.99s              0.00300000000000011s 41.12s
## 3 0d56a73007~ 1024-lvl~ 3dd4840~ 0.976999999999997s 0.00199999999999889s 39.733s
## 4 0d56a73007~ 1024-lvl~ c9e249d~ 0.977s             0.00200000000000244s 33.156s
## 5 0d56a73007~ 1024-lvl~ c5a7a2d~ 0.911s             0.00199999999999889s 40.571s
## 6 0d56a73007~ 1024-lvl~ f600a58~ 0.927999999999997s 0.00199999999999978s 32.781s
## # ... with 2 more variables: TotalRender <Duration>, Uploading <Duration>
```

Then, for the next question: 'What is the interplay between GPU temperature and performance?' We only need the powerDrawWatt gpuTempC gpuUtilPerc and gpuMemUtilPerc. Here we creat three data frames to find out there interplay.

```
##   c_temp power_avg
## 1     26  25.85500
## 2     27  26.11038
## 3     28  26.54370
## 4     29  26.84733
## 5     30  27.22727
## 6     31  28.46230
```

```
##   c_temp gpuUtilPerc_avg
## 1     26      0.00000000
## 2     27      0.01000000
## 3     28      0.06463116
## 4     29      0.26281530
## 5     30      0.66648561
## 6     31      1.84438776
```

```
##   c_temp gpuMemUtilPerc_avg
## 1     26         0.00000000
## 2     27         0.00000000
## 3     28         0.01582349
## 4     29         0.15482971
## 5     30         0.38488399
## 6     31         0.99986211
```

For question: 'What is the interplay between increased power draw and render time?' We need to find the relationship between power and render time. So I firstly calculate the avg render for each hostname. Then I calculate the avg power for each hostname. Now we can combine the data as bellow.

```
##                                c_hostname  avg_render_list avg_power_list
## 1 0d56a730076643d585f77e00d2d8521a00000Q 38.4030579710145       80.51031
## 2 0745914f4de046078517041d70b22fe700000E 39.5478636363636       80.74267
## 3 b9a1fa7ae2f74eb68f25f607980f97d7000011 38.4058823529412       81.08708
```

```
## 4 0745914f4de046078517041d70b22fe7000008 38.9108805970149       81.22676
## 5 b9a1fa7ae2f74eb68f25f607980f97d700000X 39.3390447761194       81.24218
## 6 e7adc42d28814e518e9601ac2329c51300000J 38.9289264705882       81.27640
```

For quesion: 'Can we identify particular GPU cards (based on their serial numbers) whose performance differs to other cards? (i.e. perpetually slow cards)' We should calculate average Task Duration for Each GPU Seria and draw the total render time avg for each host. During this we also need to find out the relation between these differnt data frame to find a correct way to combine them. By checking it is obviously each hostname is a gpu card. So we can continue our calculating and combining. Here I finally draw 10 data so that it can be easier to analysis.

```
##                                c_hostname Total_Render_time_avg       c_card
## 1 0d56a730076643d585f77e00d2d8521a000013               2785.27 325117172504
## 2 e7adc42d28814e518e9601ac2329c513000010              2781.344 325117063047
## 3 db871cd77a544e13bc791a64a0c8ed5000000U              2780.764 323617021202
## 4 35bd84d72aca403b8129a7d652cc275000000M              2780.156 323617021014
## 5 0d56a730076643d585f77e00d2d8521a000009              2780.127 325117171253
## 6 4a79b6d2616049edbf06c6aa58ab426a00001C              2780.036 325117172005
```

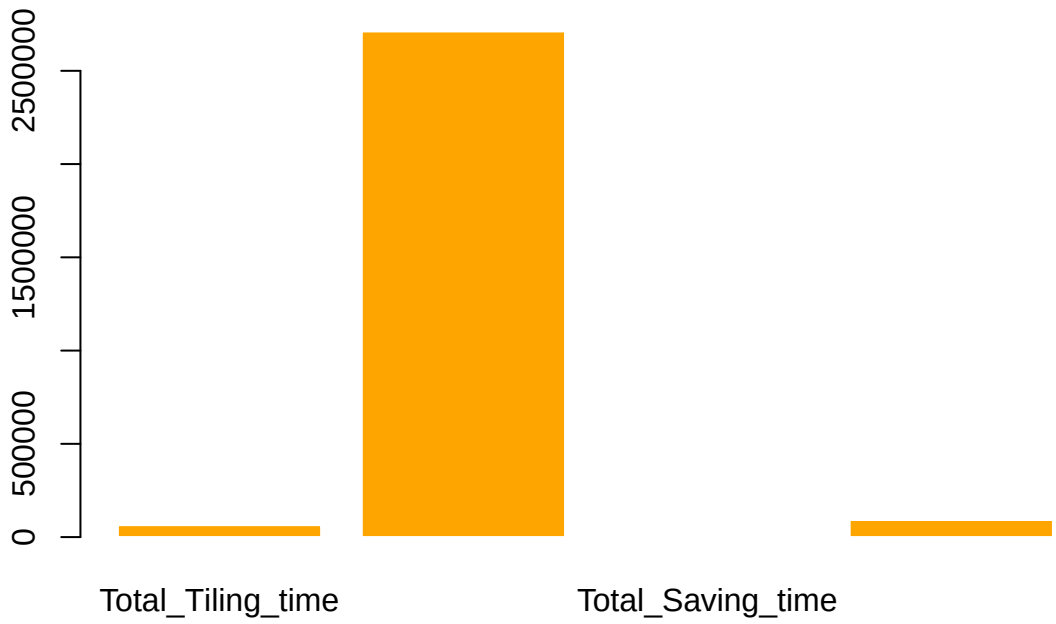
Finally the last quesion: 'What can we learn about the efficiency of the task scheduling process?' First we should find out which can be uesd to measure the efficiency. I think the efficiency would be measured by the amount of idle time between tasks. For this analysis we would assess each hostname separately and we would record the total idle time for each hostname. The data would be used to analyse is below.

```
##                                c_hostname duration_time_list
## 1 0d56a730076643d585f77e00d2d8521a00000G            2911.27
## 2 a77ef58b13ad4c01b769dac8409af3f800000O            2871.26
## 3 2ecb9d8d51bc457aac88073f6da0546100000L           2900.228
## 4 a77ef58b13ad4c01b769dac8409af3f8000018           2879.901
## 5 0745914f4de046078517041d70b22fe7000013            2869.55
## 6 83ea61ac1ef54f27a3bf7bd0f41ecaa700000K           2895.302
##   Total_Render_time_avg efficiency_duration
## 1              2733.432           0.9389139
## 2              2697.437           0.9394611
## 3              2725.208           0.9396530
## 4               2706.66           0.9398448
## 5              2697.341           0.9399875
## 6              2722.004           0.9401451
```

Data Analysis

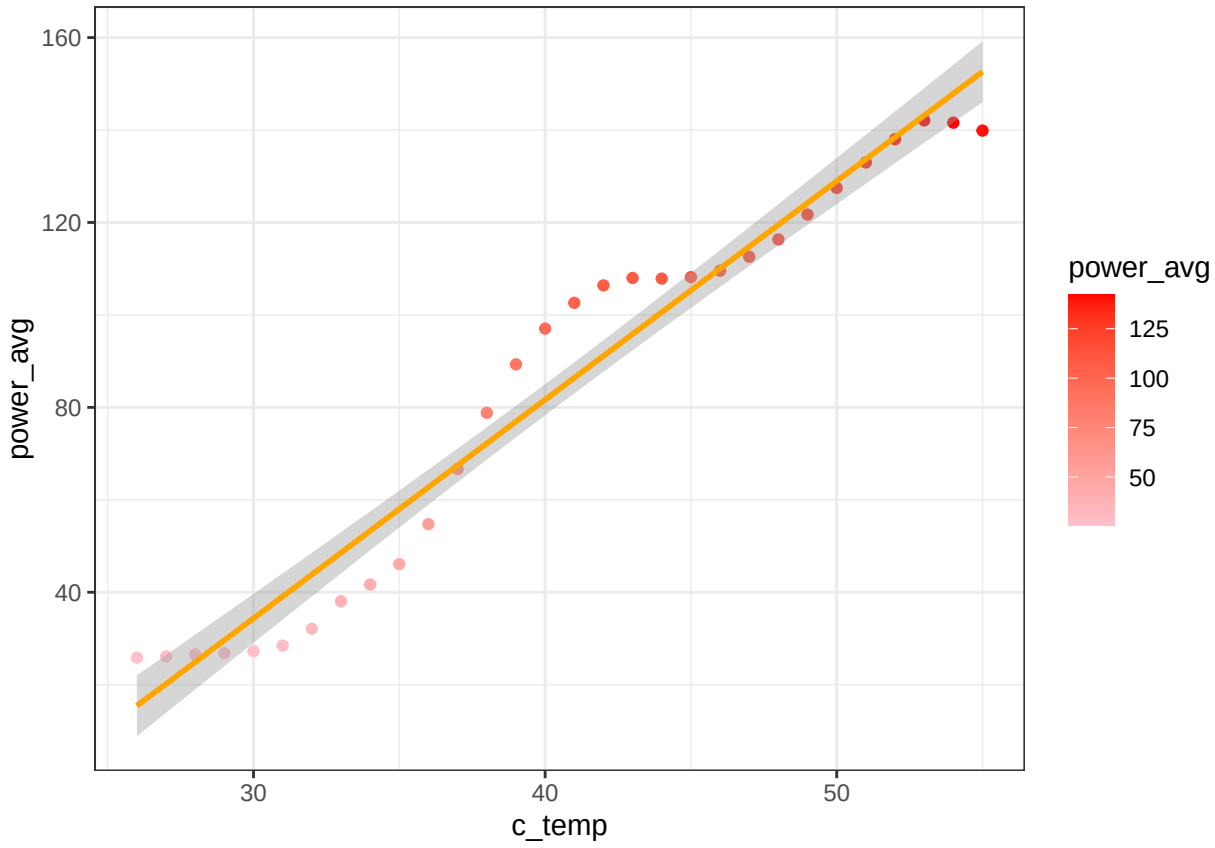
For the first question, I use a bar chart to display the running time for each event. It is obviously render dominate task runtimes

```
## # A tibble: 1 x 8
##   hostname    jobId     taskId   Tiling             'Saving Config'      Render
##   <chr>       <chr>     <chr>    <Duration>         <Duration>           <Durat>
## 1 0d56a73007~ 1024-lvl~ b47f026~ 0.720999999999997s 0.00199999999999889s 23.501s
## # ... with 2 more variables: TotalRender <Duration>, Uploading <Duration>
```

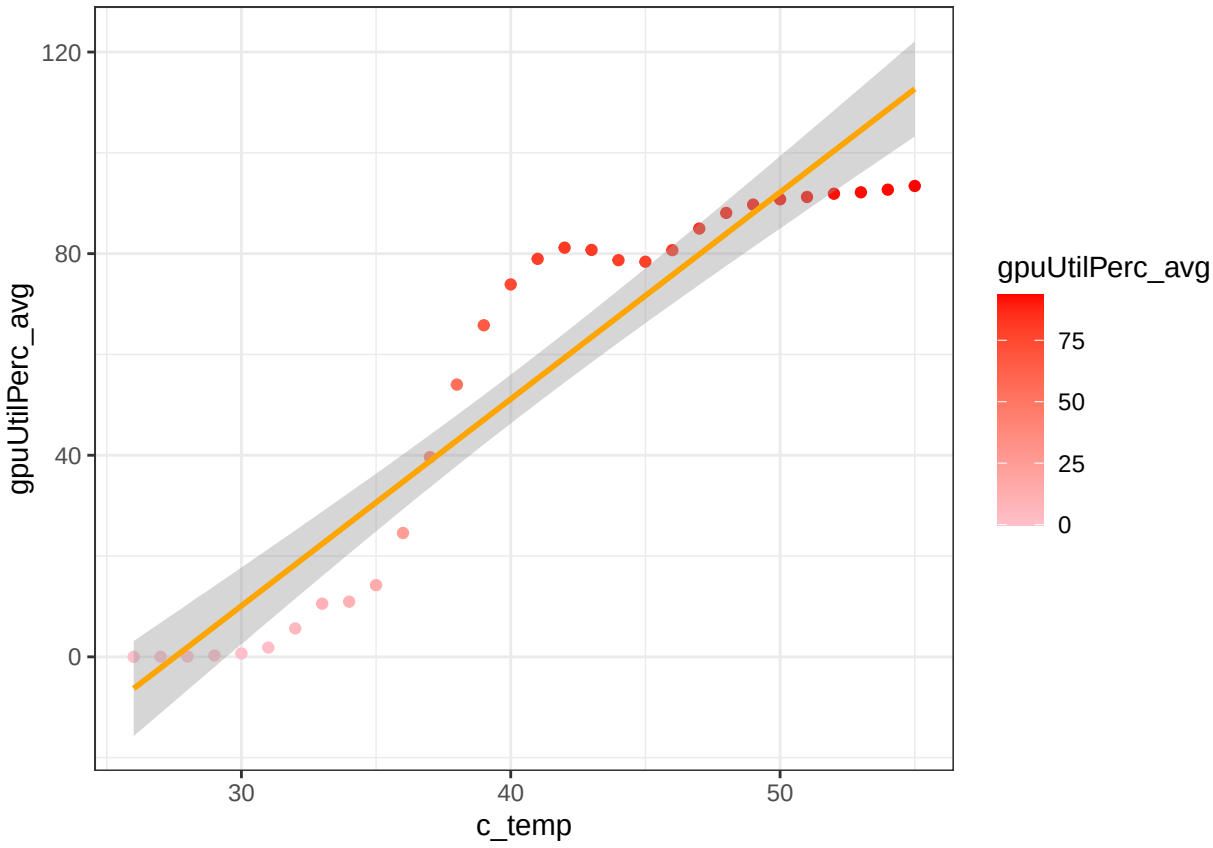


For the second question, I use three scatter plots to display the relation ship between these variables and a fit line to make the relationship more visible. So the GPU temperature is proportional to performance.

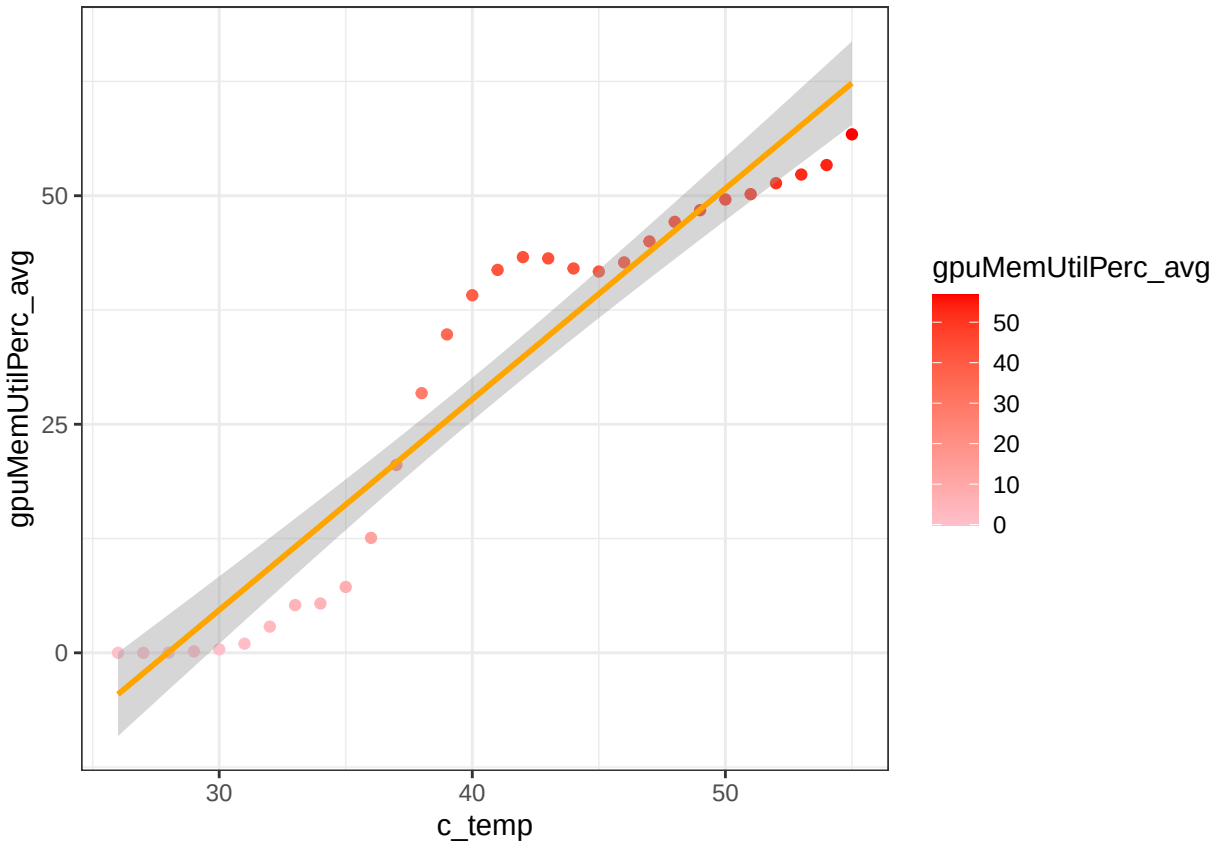
```
## `geom_smooth()` using formula 'y ~ x'
```




```
## `geom_smooth()` using formula 'y ~ x'
```

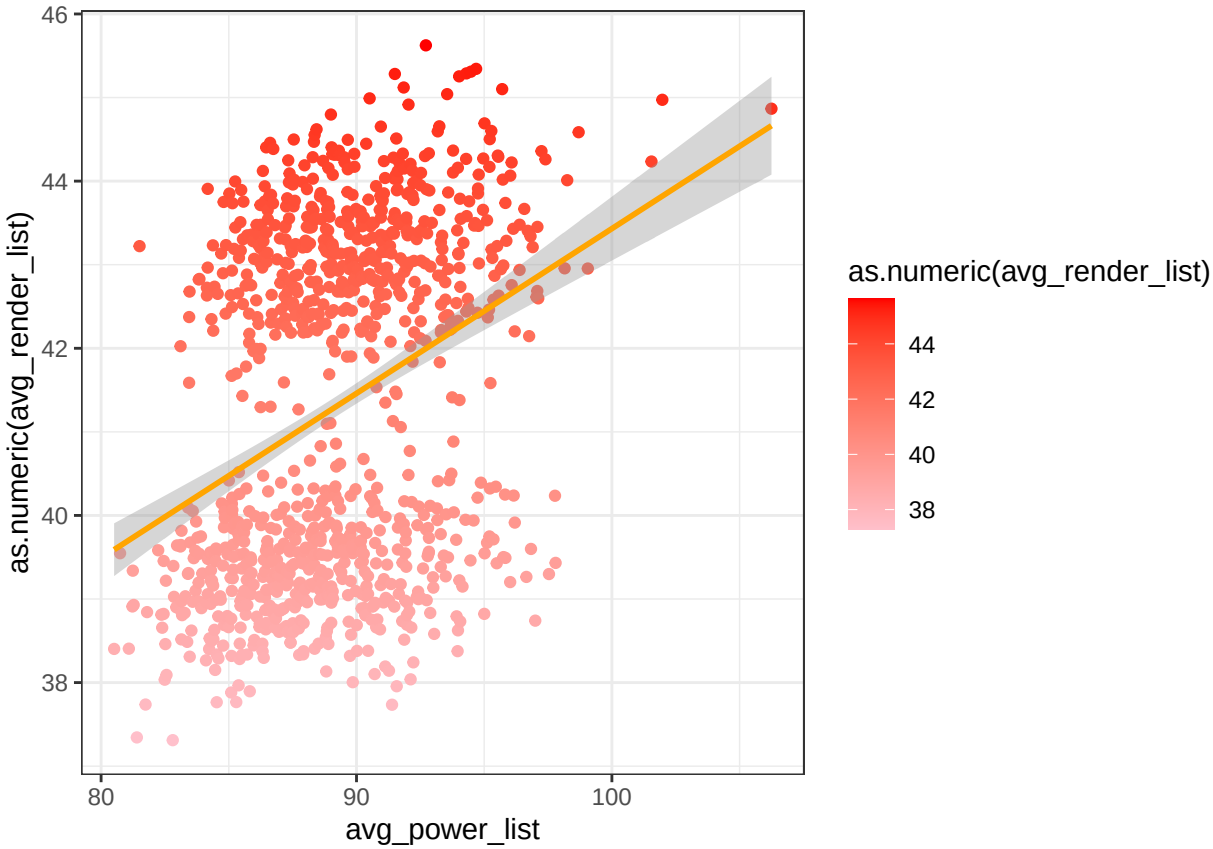



```
## `geom_smooth()` using formula 'y ~ x'
```

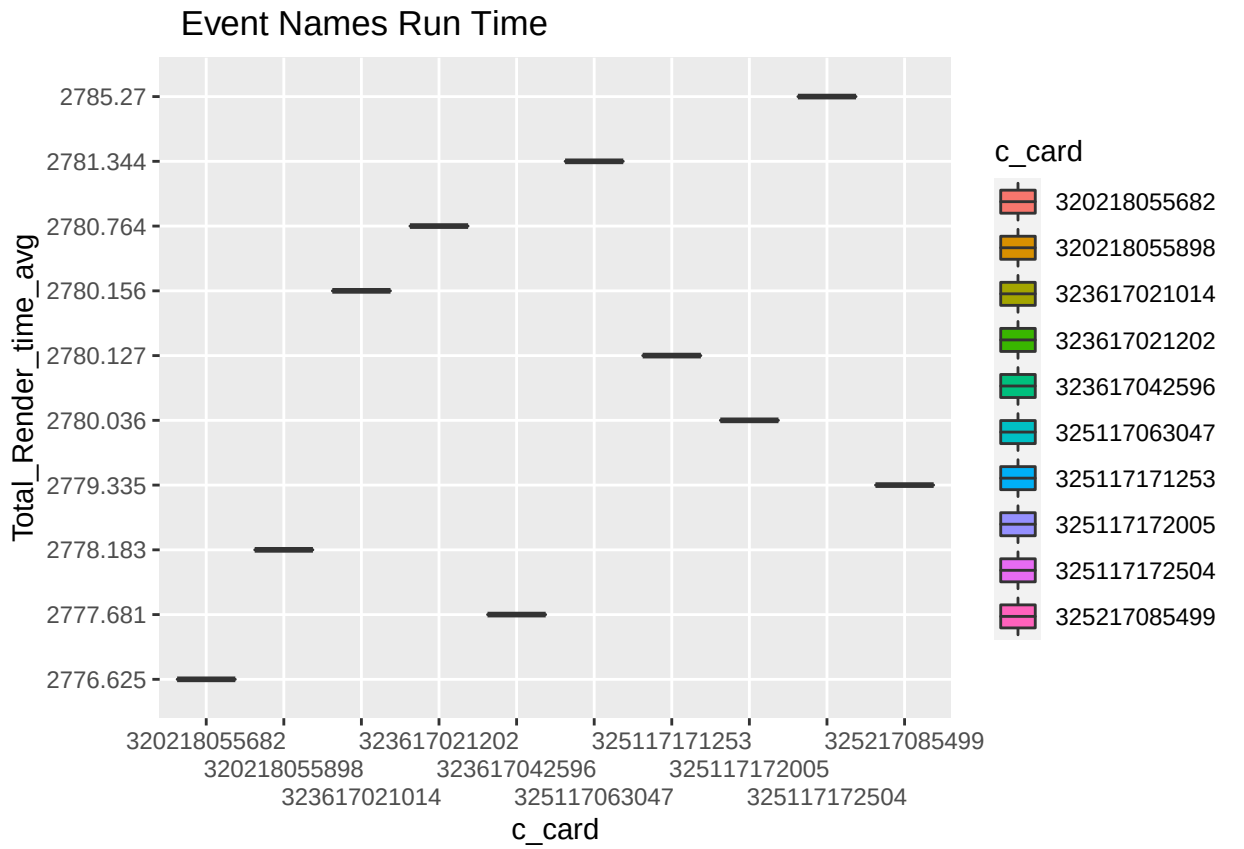


For the third question. I decide to use the hostname as a distinction and compute the avg of the power and render.It also use a scatter plot to display the relationship. It seems that increased power draw means to a longer render.

```
## `geom_smooth()` using formula 'y ~ x'
```



For the forth question, I first check the hostname for each gpu card, otherwise I can not connect the two data frame and so that I can not find out the question. Luckily I managed to find it. Then we can plot the box plot. Here I list the slowest 10 gpu cards.



The last question. It is about the amount of idle time between tasks. So I calculate the running time and render time for each GPU card and list them. Then I divide the total run time by the render time, got the efficiency.

```
##                                c_hostname duration_time_list
## 1  0d56a730076643d585f77e00d2d8521a00000G            2911.27
## 2  a77ef58b13ad4c01b769dac8409af3f8000000            2871.26
## 3  2ecb9d8d51bc457aac88073f6da0546100000L           2900.228
## 4  a77ef58b13ad4c01b769dac8409af3f8000018           2879.901
## 5  0745914f4de046078517041d70b22fe7000013            2869.55
## 6  83ea61ac1ef54f27a3bf7bd0f41ecaa700000K           2895.302
## 7  b9a1fa7ae2f74eb68f25f607980f97d700000D           2874.336
## 8  4c72fae95b9147189a0559269a6953ff000013           2873.465
## 9  dcc19f48bb3445a28338db3a8f002e9c000007            2870.11
## 10 2ecb9d8d51bc457aac88073f6da0546100001D           2878.175
##    Total_Render_time_avg efficiency_duration
## 1                2733.432           0.9389139
## 2                2697.437           0.9394611
## 3                2725.208           0.9396530
## 4                 2706.66           0.9398448
## 5                2697.341           0.9399875
## 6                2722.004           0.9401451
## 7                2702.451           0.9402001
## 8                2703.341           0.9407948
## 9                2700.329           0.9408451
```

```
## 10              2708.414          0.9410178
```