Report

Callum Simpson

```
getwd()
```

```
## [1] "D:/Masters/Cloud/Terapixel/Terapixel_v2/Terapixel_2"
```

This is my report of my CSC8634: Cloud computing with Project.

In this report I will be discussing how I under took a Performance evaluation project on a set of data relating to the creation of a Terapixel image rendered in Cloud (Super)computing.

A background

A Terapixel image offers a new and intuitive way to present information sets to stakeholders that is also extremely accessible to all, allowing viewers to interactively browse big data across multiple scales. Typically there made up of over one trillion pixels and provide a fluid experience where the viewer can see an overview of the whole image or zoom into incredible detail.

The importance of Terapixel images is that viewing it only depends on the image display capabilities of the web browser that can display it, making terapixel images accessible on a wide range of thin clients. The client would not need any expensive software or hardware to view the image, only a web browser and a link. Terrapixels can also be very detailed allowing for impressive designs that a client might require.

This project will be a performance evaluation looking at processes behind creating the supposed first terapixel visualization of IoT data within a 3D urban environment by Members of Newcastle university in the report Petascale Cloud Super computing for Terapixel Visualization of a Digital Twin (https://arxiv.org/ftp/arxiv/papers/1902/1902.04820.pdf)

The image created was a realistic 3D visualization of the city of Newcastle upon Tyne to display psychical locations of a set of sensors that are located around the university. One of the big parts of this project was that it was created by using a scalable architecture for cloud-based visualization meaning the user can control what they deploy and pay for.

The project had three key goals to achieve this

- 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.

In this project the Terapixel visualization using a path tracing rendered the image in under an hour using 1024 public IaaS cloud GPU nodes.

Iaas (Infrastructure as a Service) if a form of cloud computing that provides virtualized computing resources over the internet from a third party provider in order to create the IT environment that best fits business requirements. The benefit of IaaS is that they significantly reduce the cost of IT infrastructure because it allows the use of access of cloud resources without having to purchase your own data center hardware/maintain your own equipment. The responsibility for the maintenance of the infrastructure to the IaaS provider whose job it is to ensure its availability.

What is the reasons for this project

The aim of this project is to undertake a Performance evaluation on the creation of this TeraPixel image using cloud computing to find keys areas in which we could improve. We have data relating to the different tasks, recording of GPU usage and the coordinates of each tile (so what part of the map is being worked on).

There are two main areas that we can focus in this project that any findings could be used to help improve the creation of Terapixel projects in the future.

Firstly We know that Terapixels two main costs are time needed to generate the image and the Money. As IaaS recourse are basically rented, if we can find ways to reduce the amount of time needed to Generate a Terapixel image we would be able to rent servers for less amounts of time (therefore saving money). An increase in speed in image generation would mean that more images could be made within a given time frame, which would be beneficial for clients.

It took the terrapixel 43 minuets to create image that we will be investigating in this report. So finding areas to shave of a few seconds here and there will help reduce the time making the image.

Secondly, As we have the data about the clouds 1024 nodes that where used to create this image we can perform an analysis on the cloud side of the project to see how the processes effected the GPUS that we used. Hardware issues could slow down the the speed at which the image is created or may lead to other problems. If we can highlight potential issues in the hardware then we could go fix the GPUS/Hardware, hopefully resulting in future terrapixel images being smoother to run.

My responce

For this project I mainly performed Exploratory Data Analysis to discover patterns, to spot anomalies and to check assumptions via summary statistics and graphical representations.

There where many questions I could ask and try to answer with this data. Because of this I decided that I would follow a Crisp-DM model when under taking my analysis. From the start I knew I had a few question that I wanted to ask and I fully expected that what I may discover from one question may help me gain a better understanding of a question that I previously asked and could possibly suggest a new question/ route of analysis that I should go explore. The crisp DM model would be perfect for this line of questioning as it clearly gave me steps that I could follow in order to ensure that I wouldn't get confused or lost in my analysis.

Each cycle played out like this, I would want to answer a question so I modify the data to get the bits of information that I wanted. After I got what I wanted I visualized what I had made just made to see what new insights I could gain from it. Once I had done this I used this new insight to loop through the questions I had asked to see if it helped add anything to previous questions or opened up a new route of investigation.

When looking into cost of power and time how they effected things like heat and utilization I used a mixture of univariate analysis to analyses each features and Bivariate analysis to work out the how some elements effected another element (primarily how X effects time).

When looking into The GPUS to try and find any issues I mainly preformed univariate analysis to try and discover the medians that each GPU produced, That way I could look for the GPUS that had a higher median than the other GPUS suggesting that something may be up with that parictualr GPU. For example if I worked out the median powerusage for all GPUs and found that most had a median of 40 but one had a median of 50 then we know something might be up with that GPU. This didn't go 100% to plan so I

I used the following works as inspiration

Still working on this

https://www.researchgate.net/publication/224221699_Performance_Analysis_of_Cloud_Computing_Services_for_Many-Tasks_Scientific_Computing

https://www2.seas.gwu.edu/~howie/publications/GPU-CNN-ICPP16.pdf

https://www.sciencedirect.com/science/article/pii/S2352864816301456

What I did and what I discovered.

For this project I mainly performed Exploratory Data Analysis to discover patterns, to spot anomalies and to check assumptions via summary statistics and graphical representations.

The data that I received where

- application-checkpoints : The application checkpoint events throughout the execution of the render job. Data in this was primarily Polynomial Variable (i.e events and tasks names) with the Continuous Variable of time.
- gpu : metrics that were output regarding the status of the GPU on the virtual machine. Key data in this was Continuous (i.e amount of power used)
- task-x-y : the x,y co-ordinates of which part the image was being rendered for each task. Mainly made up of Discrete Variables.

Before I really started anything I went through each variable in each data set to really understand what I was working with. Through checks I discovered that there didn't seem to be any issues with any of the data (apart from a few duplication) as it was really clean.

The following will be some of my analysis into the questions that I asked and what I discovered.

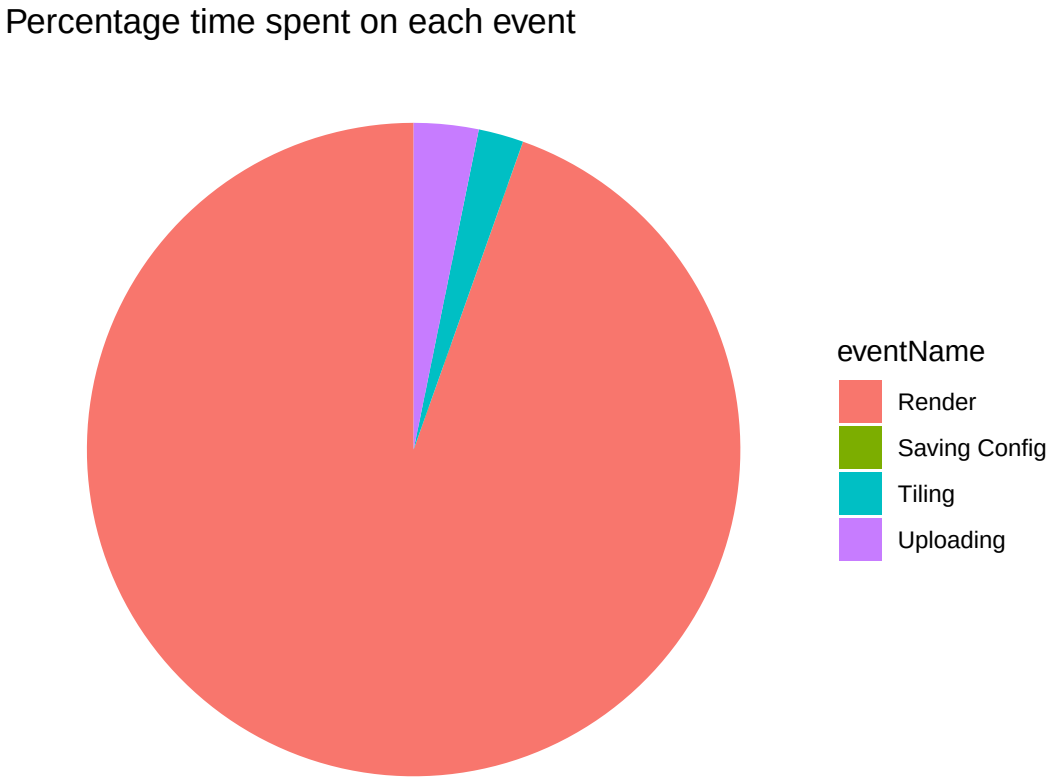
A TLDR would be, After discovering that rendering event was the part of each task I that took the most time to complete I decided looked over how rendering effected heat and power to discover that an increase in rendering time caused an increase in both of them. This lead me to creating a K means system called colour Quantization to work out what was in each tile to see what was being rendered had any effect on the rendering task. After that some investigation was done to see if I could find errorus GPUS.

Which event types dominate task runtimes

For more detail for the section Please see "Events_Explore" report

The First question I asked was "Which event types dominate task runtimes". I felt that this was a good first question to answer as if I want to reduce the amount of time needed to make an image then seeing what event took the most amount of time it would tell me what I needed to inverstergate. After working out the time that had passed between an event starting and ending I worked out the median for each and then worked out the percentage of how much of the total task time was spent on that event.

```
ggplot(data=time_sumary_omit_totalrender )+
  geom_bar(aes(x="", y=percenatge, fill=eventName), stat="identity", width = 1)+
  coord_polar("y", start=0)+
  theme_void() + ggtitle("Percentage time spent on each event")
```



Using the Pie chart we see that Rendering is the dominating event (TotalRender was removed due to it being the combination of all other events combined) and none of the other events came close. A further check on the different jobs had any effect on events found out they genuinely didn't par from uploading in job 8 did have a higher percent.

This told me that if we want to speed up the processes of the creation of the image we would need to reduce the amount of time spent rendering. Also the higher median uploading in job 8 suggest that something had gone wrong in uploading in this job.

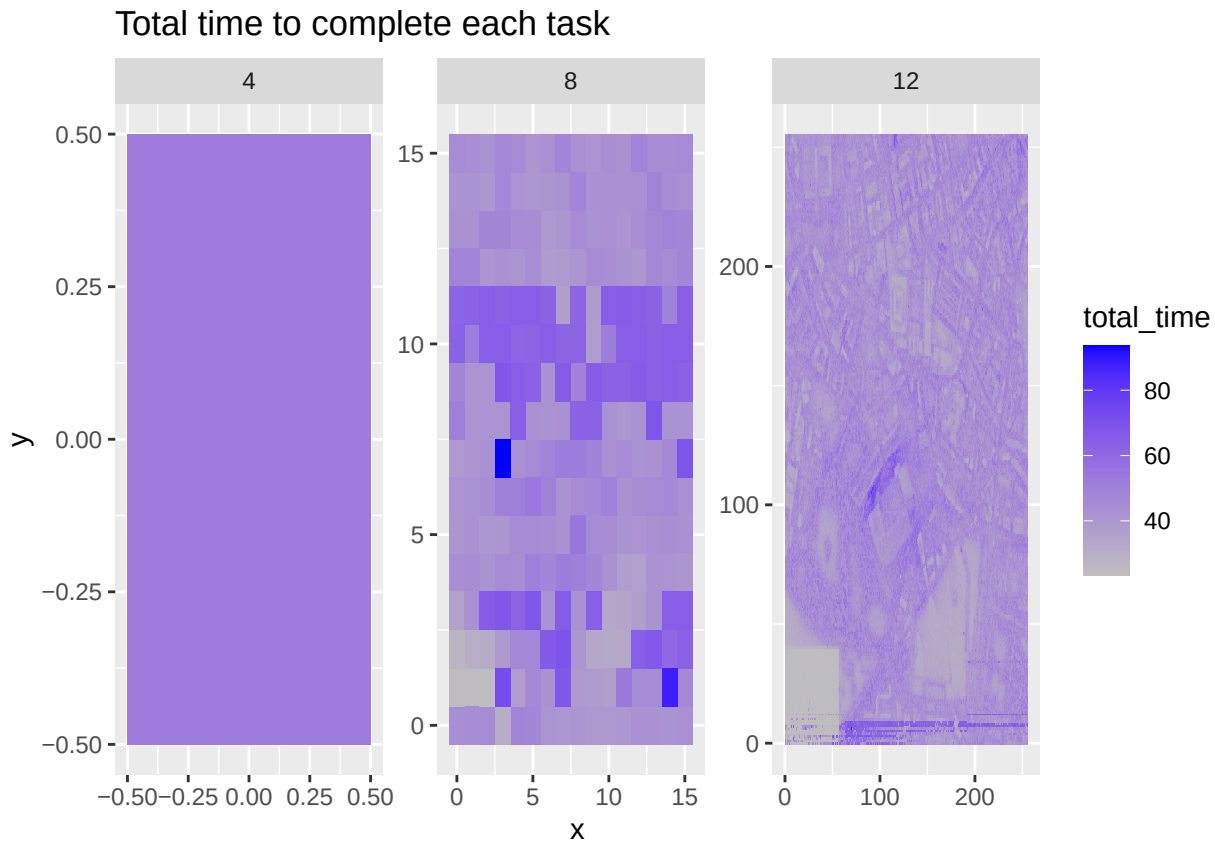
Whilst investigating this I discovered that a task takes about 43 seconds but that the time taken does change based on the job.

By combining the data frame which has the time taken to complete a task mixed with the x and y cords we could use it to visualize a heat map of xy cords and how long it took to render that tile.

```
last_tasks_xy <- left_join(last_tasks, xy_df ,by = c("jobId","taskId") )

last_tasks_xy <- last_tasks_xy %>% mutate(timestamp = as.POSIXct(timestamp,format="%Y-%m-%d %H:%M:%OS")

ggplot(last_tasks_xy, aes(x, y, fill = total_time)) + geom_tile()+
  facet_wrap(~ level,scales = "free")+   scale_fill_gradient(low="grey", high="blue") + ggtitle("Total
```



Looking at the lvl 8 and 12 graphs we see that there is are distinct areas in which the render time was really low (bottom left). This suggest that what is actually is in the tile that is being rendered does seem to effect the time taken to complete the task.

We also notice that there seems to be a distinct band in the low Y cords in the lvl 12 graph in which rows of x cords seem to have the same or similar high completion times. We also see a similar band appear in the y = 10 ish in lvl 8. This seems really off and was the subject of later investigation.

What is the interplay between GPU temperature and performance

For a more detailed overview please see GPU_heat

Next I wanted to look into GPU temperature and performance. High temperature will have two negative effects on a GPU.

- A high temperatures will shorten the life of hardware by damaging the hardware
- GPU typically have a fail safe that will try and combat high temperature by ether slowing down in order to cool down or fully shut down. Both would increase the time needed to run a program.

So we need to investigate what effect using cloud computers to make Terapixel has an effect on a GPUs heat and if they are being pushed to the point at which they may be damaged/cause issues.

The median heat per tick is 40 degrees and that the data is normally distributed.

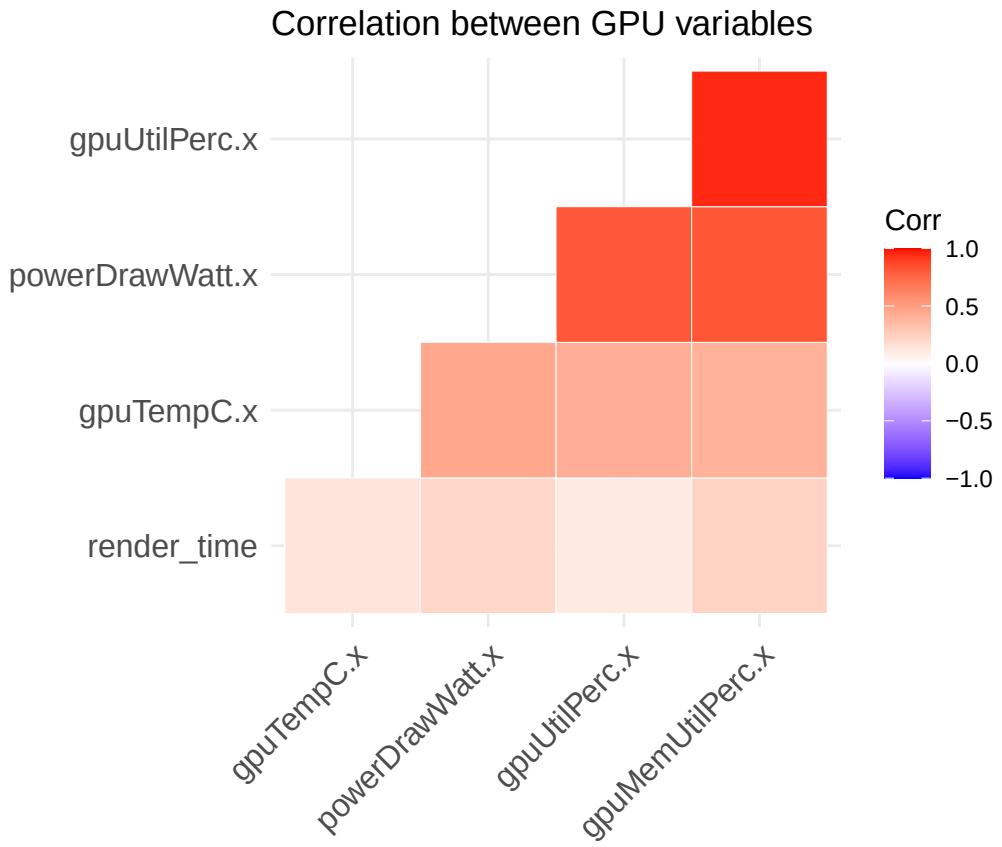
One statistic that can be used to calculated the performance of a GPU is fill rate, the number of pixels that can be rasterize by the card and write to memory per second. We know that each task is basically the filling in of 4096 * 4096 pixels so we can say that tasks with low total rendertimes could be said to have a

better performance. On the flip side of that we can say that task that took longer to render had a worse performance.

By selecting the rendering events in checkpoints and linking via host name and time to the appropriate GPU in the GPU data frame I was able to create a summary of power usage depending on the length on the rendering processes.

Calculating the correlation between GPU variables ...

```
ggcorrplot(cor(gpu_render_highlight %>% filter(render == 1) %>% select(powerDrawWatt.x, gpuTempC.x, gpuU
, hc.order = TRUE, type = "lower",
     outline.col = "white") + ggtitle("Correlation between GPU variables ")
```



We see that there is a slight correlation between render time and gpu temptaure meaning that an increase in render time does have an effect (though not major) on GPU heat. We also see that as GPU utilization increase that temperature increases.

This suggest that If can work out a way to reduce render time we will also be able to reduce the amount of heat produced per GPU. If we aren't able to find ways to reduce render times than a new cooling system may need to be put in place.

What is the interplay between increased power draw and render time

For a more detailed breakdown please see GPU_Power_Usage report

Through my investigation I found that the median gpu power usage per tick is 45897.71 watts.

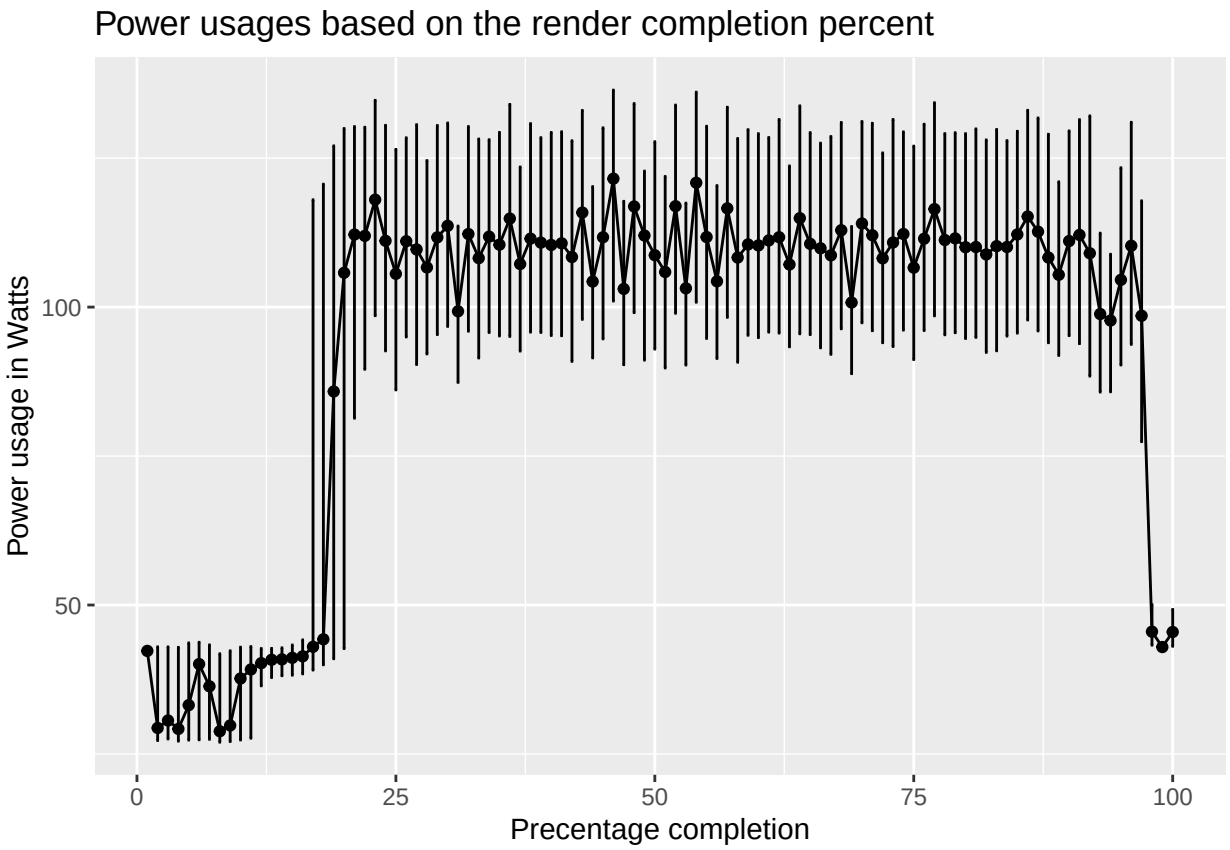
Using the application-checkpoints data frame to see what time rendering took place and combined it to the gpu df by host name I was able to work out which ticks in the GPU data frame where happing during a task render event and which where happening when another event was occurring.

A quick view over how power usage changed in rendering and none rendering events in one of the GPUS we saw that rendering a tasks started off at lower power, spiked up to high power then near the end dropped back to lower power. We also saw that the longer the render time the longer and higher this high power line reached.

This gave me the idea to work out how far through the rendering task each tick was and assign it a percentage.

Using the medians power usage in watts for all percentages we get.

```
ggplot(Power_used_at, aes(rendering_percenatge,median)) + geom_line()  + geom_point() + geom_errorbar(a
```



We see that typically the first 20ish percent of the rendering processes the power usage is around 30 watts per tick. Once we reach 20% percent of the way through the rendering processes power usage dramatically jumps to around 110 watts per tick. We see that power usage continues to stay around 110 watts until we reach roughly 95% when power watts jump back down to roughly 45 watts per tick. This tells us that during the rendering processes it take a few moments before the processes to actually start rendering and once its done there is a few seconds where the render "cools off".

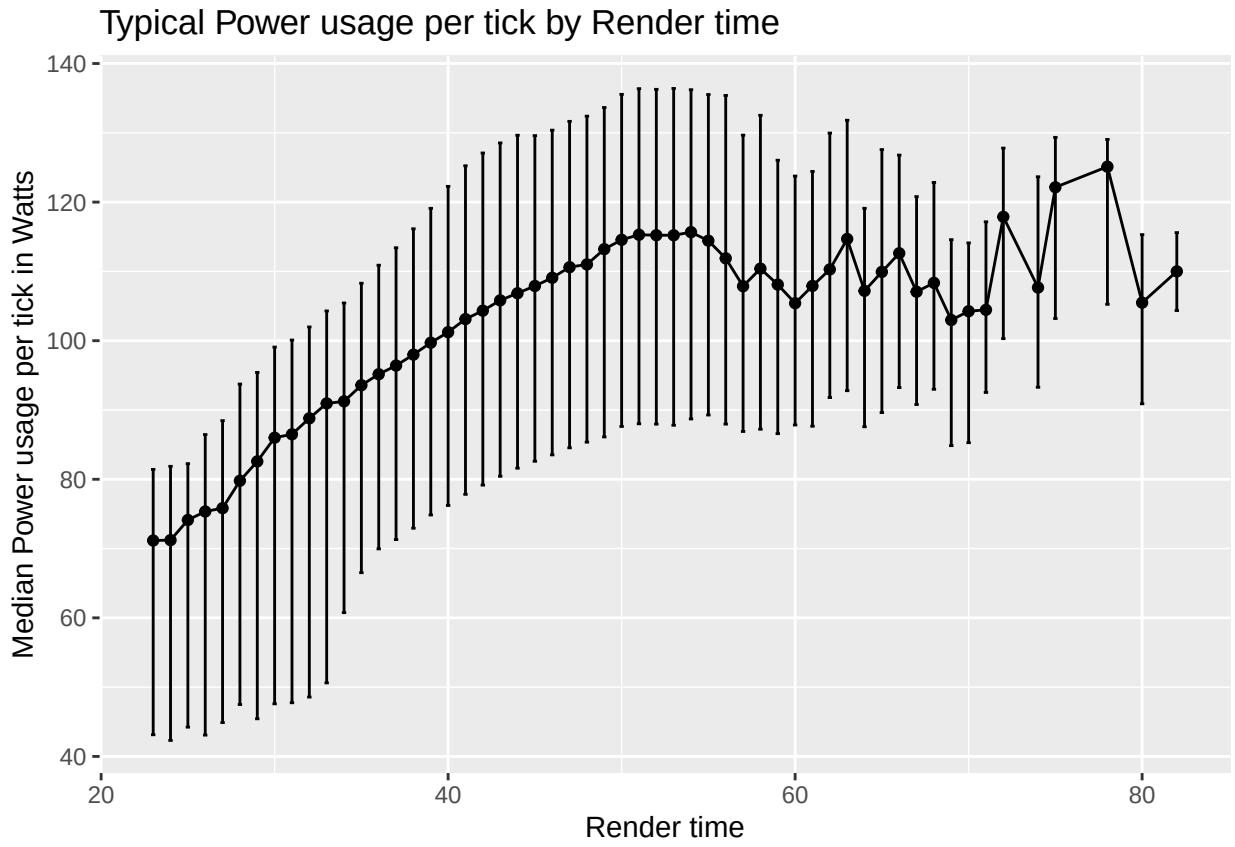
By plotting out the amount of time needed to render a tile and plotting each as line graph of power used by percentage completion we saw that the longer the render time the median power tick increased until we reached a point where the power per tick gets capped at around 110 watts (further investigation found out that this was due to the GPU utilization being maxed out/ limited).

Doing this produced allot of graphs all with the same pattern, all had what seems like await period of around 7ish second where power was low before power and utilization jumped high. This suggests that the rendering needs about 7 seconds to perform some task/calculation, no matter what it is going to render.

Seeing that as render time typical power increase I plotted the following.

```
#gpu_render_highlight_table

ggplot(gpu_render_highlight_table, aes(render_time, median)) + geom_line() + geom_point() + geom_errorba
```

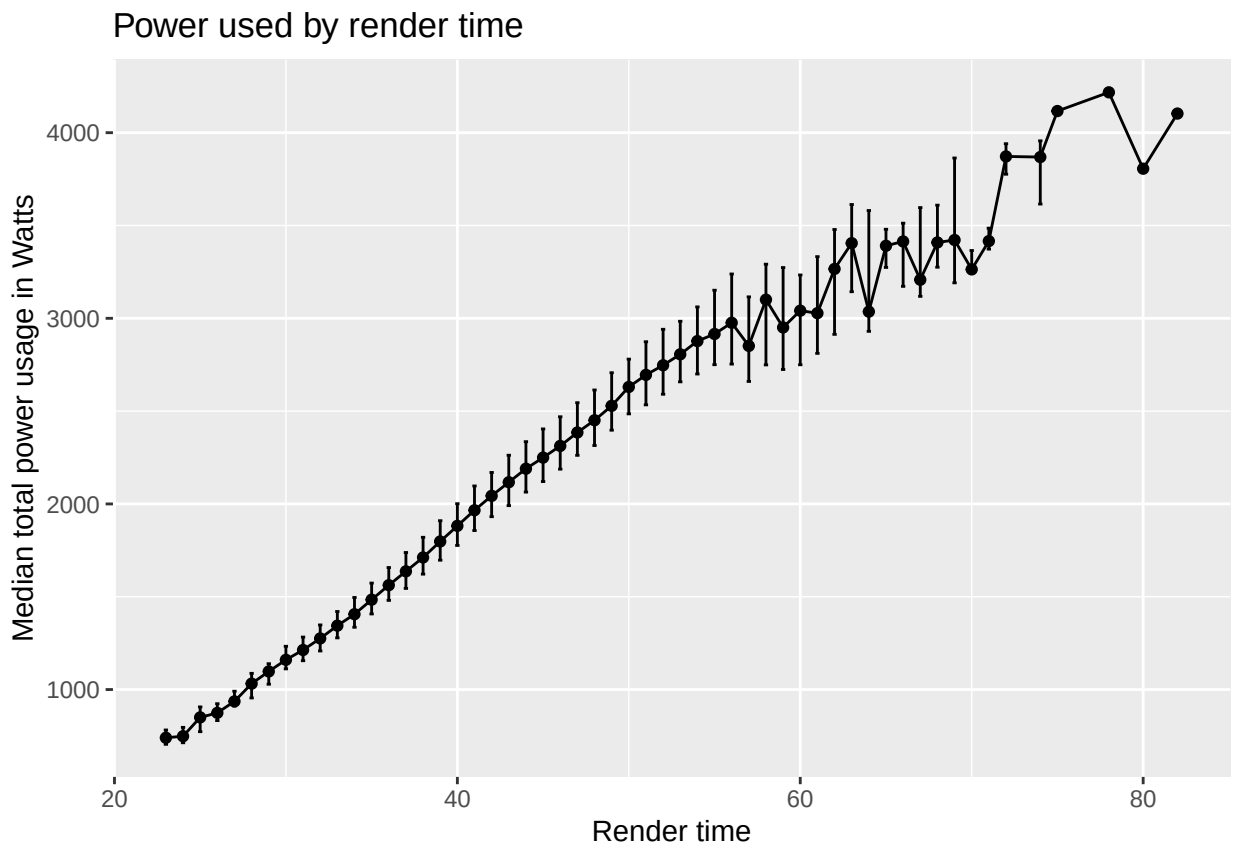



```
#ggplot(gpu_render_highlight_table, aes(render_time, mean)) + geom_line() + geom_point() + geom_errorba
```

We see that once the time needed to render reached around 50 seconds we see time to render stabilizes at around 110 watts per seconds.

Finally, Plotting the amount of power used in a render task we see that as render time increases the amount power increases in an almost linear fashion

```
ggplot(gpu_render_highlight_sum, aes(render_time, median)) + geom_line() + geom_point() + geom_errorbar
```




```
#ggplot(gpu_render_highlight_sum, aes(render_time, mean)) + geom_line() + geom_point() + geom_errorbar(
```

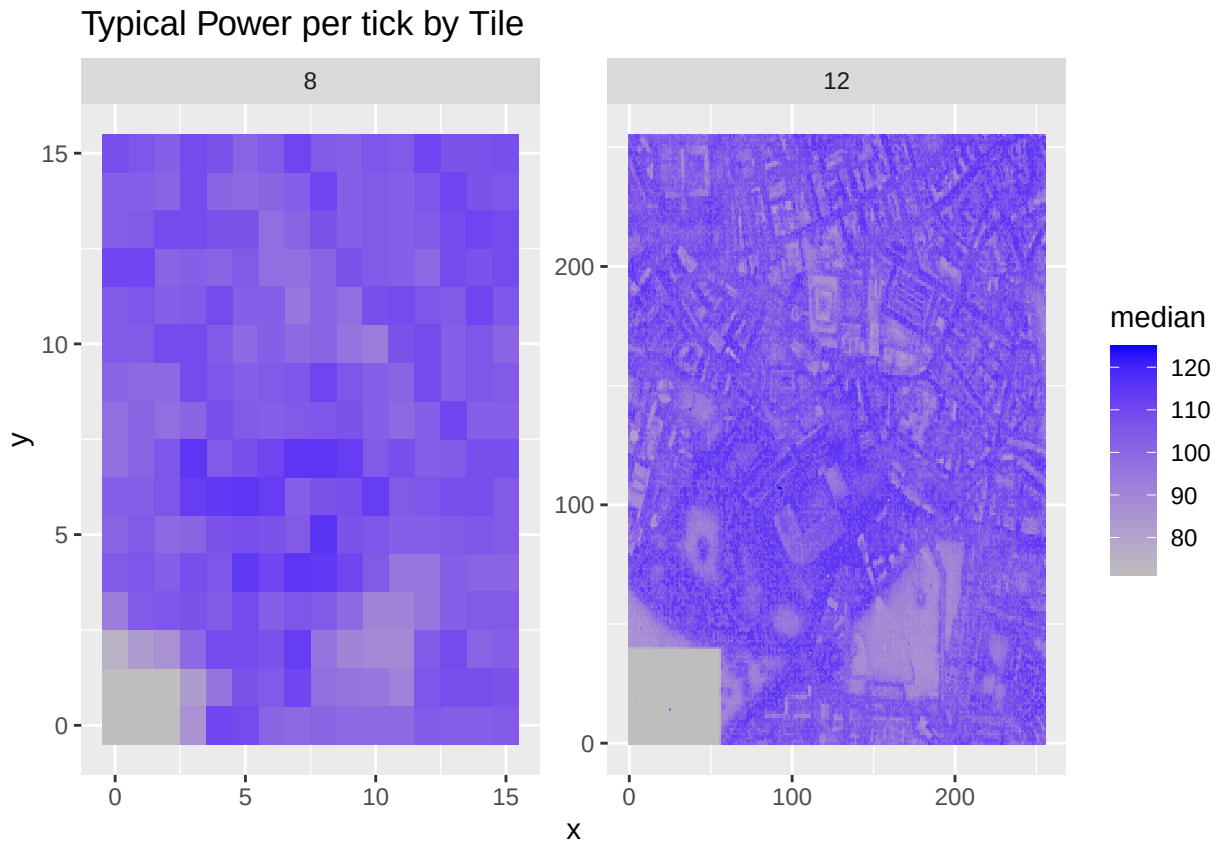
Can we quantify the variation in computation requirements for particular tiles?

For a more detailed breakdown please see GPU_Power_Usage report

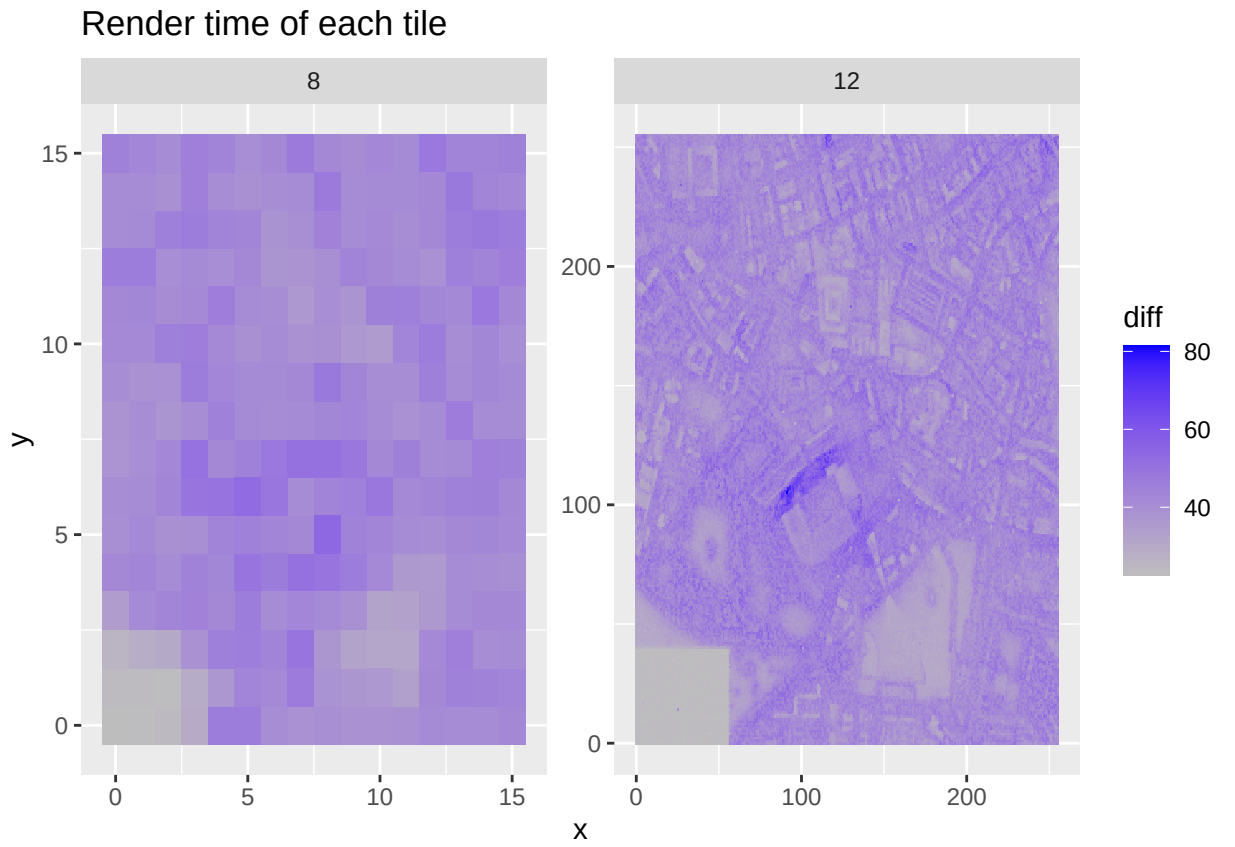
Using the coordinates tied to each task I was able to create a heat map of how much power was typically used in the rendering of each tile. I also created a heat map of how long it took to render each tile.

```
rendertasks <- times %>% filter(eventName == "Render")
rendertasks<- na.omit(rendertasks)
render_with_cords <- inner_join(rendertasks , xy_df , by = "taskId")

 ggplot(gpu_renders_xy_table %>% filter(level != 4) , aes(x, y, fill= median)) + geom_tile()+ facet_wra
```




```
ggplot(render_with_cords %>% filter(level != 4) , aes(x=x, y=y, fill= diff)) +
geom_tile() + facet_wrap(~ level,scales = "free") + scale_fill_gradient(low="grey", high="blue")  + gg
```



Looking at the heatmap, tiles that seem to have rendered a large flat area like a grassy patch/pond used the least amount of energy. Roads and roofs that use the mid amount of power to render. It seems like the tiles that use the most power to render are tiles that contain trees. We also see that this trend seems to exists in the Render time heat map suggesting that what is actually being rendered has an effect on how long a tile takes to render and because of that it also increases the amount of power used to render that tile.

This suggests that different tiles will require different computation requirements for particular tiles

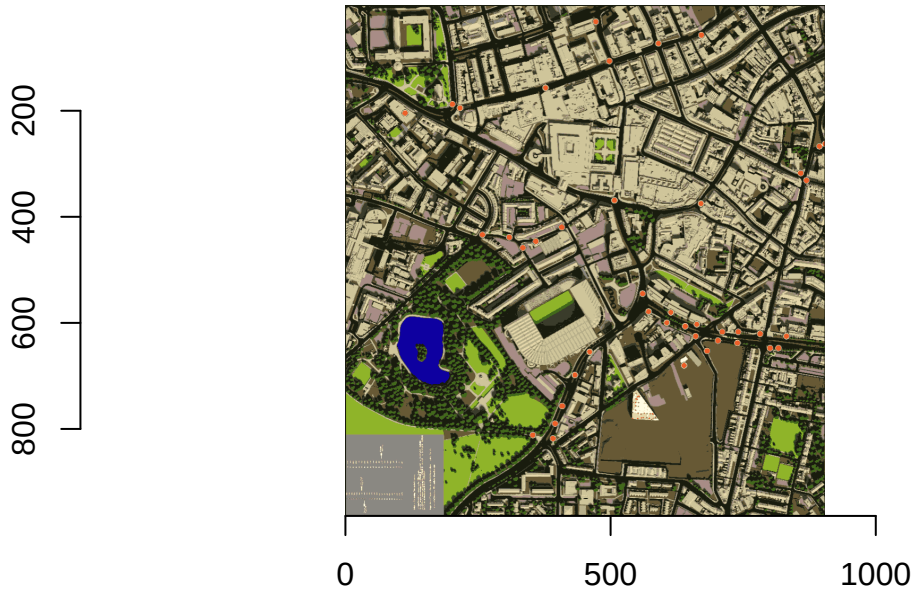
Is whats is actually being rendered effect how long it takes to render the tile.

Alot of my previous analysis showed me that what is keeping the terrapixel image back is the amount of time it takes to render a tile. Increased render time increases power used, heat and GPU ultisation. I wanted to find out if what we are rendering in each tile effects the rendering time. I felt this was important as if we could work out what objects where causing an increase in render time then work could be done to change how the render deals with these object hopefully saving time.

Originally I created a way that allowed me to split an image of the full terapixel image into the tiles that where being rendered. This allowed me see what was in certain tiles as I could basically filter tiles by render times. However I felt I could create a better way. I ended up using some python code that would allow me to apply a K-means clustering algorithm to an image so that I could perform Color Quantization. Color Quantization is the process of reducing number of colors in an image, I wanted to do this to make classifying what is in a tile easier. Doing this I was able to split an image of the terrapixel map into 15 distinct colours with each colour potentially relating to a distinct object.

Here is that Color Quantization image.

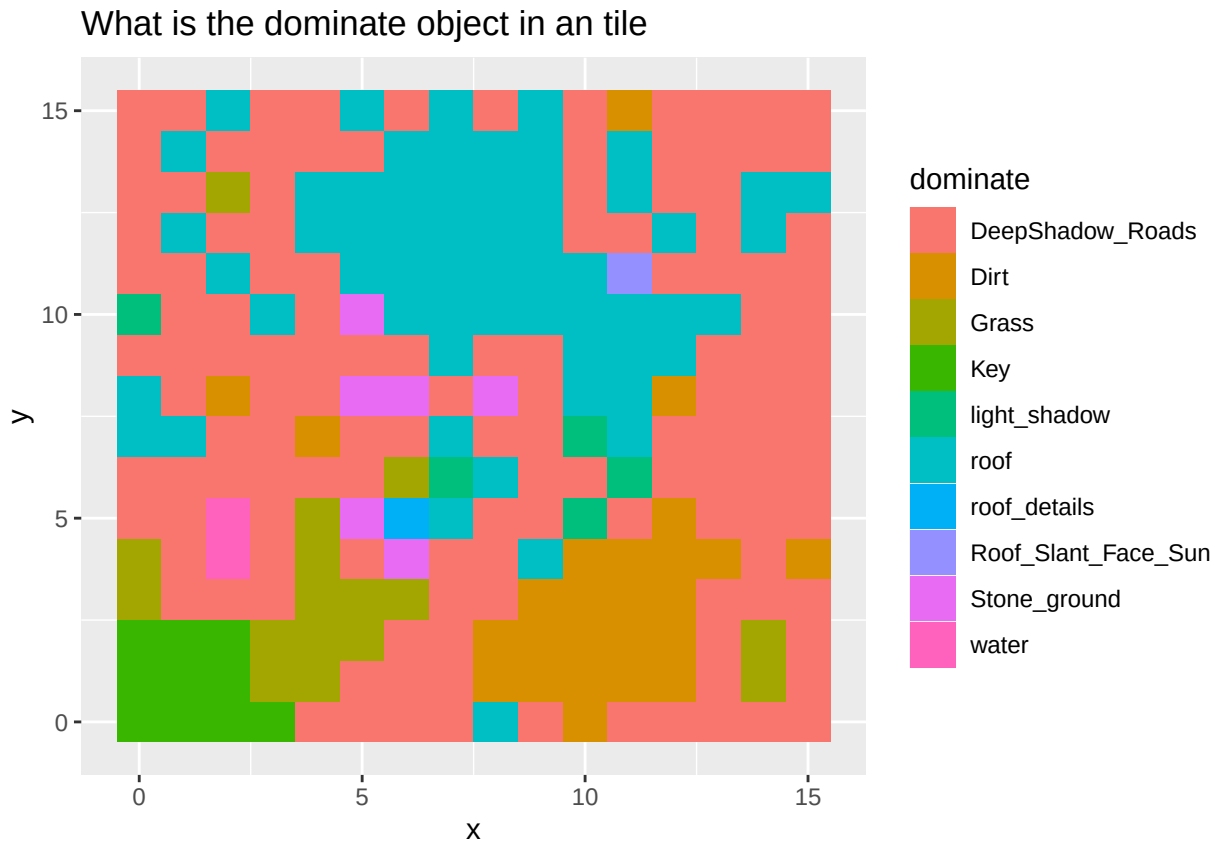
```
plot(K15_map)
```



I will admit the there are a few mistakes with this image. It has grouped the dark shadows and roads together due to the fact they where both really dark hues in the original image. Its has also turned the light blue in the original image (which I believe was flat ground) into roofs but as this blue only appeared in a small part of the graph I don't think It would cause any issues.

After working out what each color meant I was able to determine the most dominate object in a tile.

```
ggplot(job_8_Tiles_dom, aes(x, y, fill= dominate)) +
 geom_tile() + ggtitle("What is the dominate object in an tile")
```



Looking at the majority color in a tile we see that we have clumps of grass, dirt and roofs in a sea of tiles that are primary made up of roads/shadows. We also notice that out of 15 objects only 10 are dominate in a title and some are only dominate in one suggest that the colors can be split into two groups.

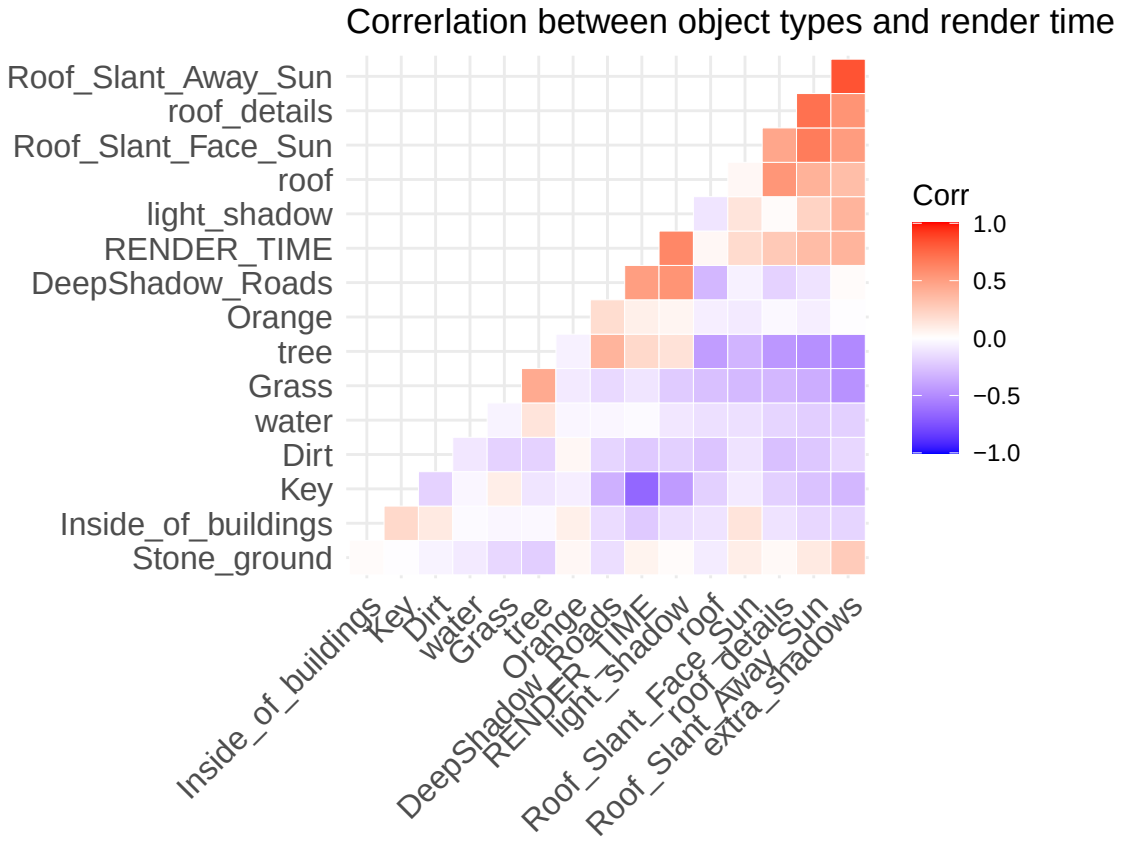
- Used to represent something, i.e water and roofs
- used to represent details or shadows.

My next idea was to work out the correlation between each of the different objects and the render time. By doing this It would distinctly show if and how each of the different objects effected the amount of time needed to render each tile. The following is that correlation plot.

```
task_join_col <- inner_join(render_with_cords %>% filter(level == 8) , job_8_Tiles_dom , by = c("x","y")

task_join_col <- task_join_col %>% rename( RENDER_TIME = diff)

ggcorrplot(cor(task_join_col %>% select(RENDER_TIME, roof,roof_details,Roof_Slant_Away_Sun,Grass,Dirt,In
, hc.order = TRUE, type = "lower",  insig = "blank",
     outline.col = "white") + ggtitle("Correrlation between object types and render time")
```



We see that the more of flat objects that are not effected shadows and are fully in daylight (so Stone_ground, insides of building, the Key, grass and dirt) that appear in a tile then the quicker the that tile is to render.

We also see that the more that shadows appear (so Deep shadows, light shadows and extra shadows) in the tile then the longer that the tile takes to render (with quite a strong correlation).

Anything to do with roofs has caused an increase in render time.

Other bits of information that we can take away from this is that

Trees seem to one of the main reasons for the shadows

Extra shadows seem to effect mainly roofs.

Overall this tell us that what is being rendered does have an effect on the render time and that some objects types (mainly those used in details) have a greater impact on the render time.

GPU errors

We have alot of GPUs that have been used to generate this images. The generation of this image depends on all theses GPUS working together. If a GPU is having any issues then we will need to repair/replace them as a single issue may slow down the overall creation of a terrapixel image.

Originally I wanted to used the fact that the median time to complete a task was 43 seconds. This didn't work as I wanted it, there are way to many additional factors that effect how long it takes to complete a task. So saying "oh this GPU must have flaws because took on average a few seconds longer to complete each its of its task" didn't feel a good way of highlighting errors. Its possible that a GPU with no errors was just given a bunch of tasks that just took a while to processes.

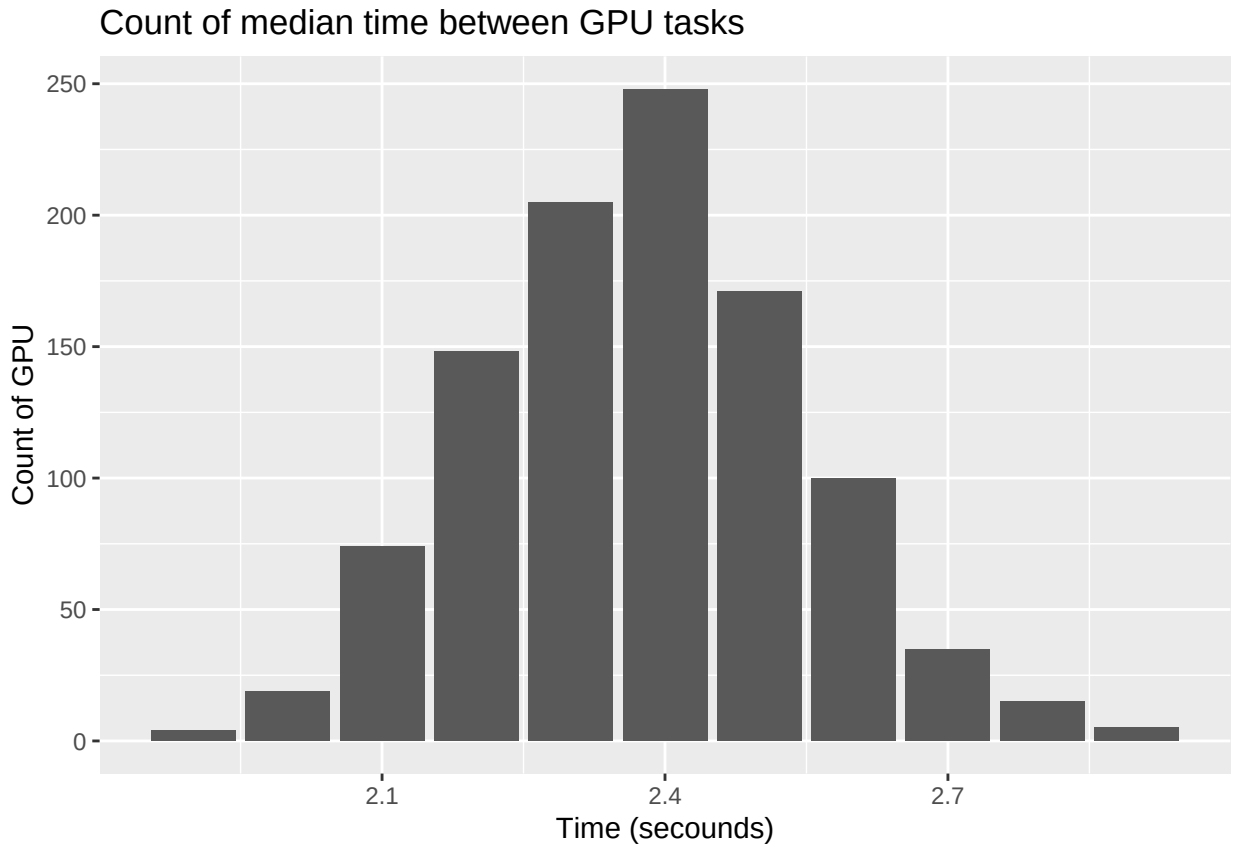
I instead decided to look at the median amount of time between each task on the GPU (so time between the GPU last task ending and its next task starting). I feel like this would be a better way that to gauge which GPUs may have errors as its something that cant be effected by the rendering processes. We find that the median amount of time between each task is 2.4 seconds.

Creating a barchart of the time typical time between GPU task we get the following.

```
idle_time <- link_gpu_task %>% group_by(gpuSerial) %>% mutate(idletime = lagtime - lag(timestamp.x)) %>%


ggplot(data=idle_time %>% group_by(gpuSerial) %>% summarize(median =  round(median(idletime), 1)) %>% co
  geom_bar(stat="identity") + ggtitle("Count of median time between GPU tasks") + ylab("Count of GPU") +
```

```
## Don't know how to automatically pick scale for object of type difftime. Defaulting to continuous.
```



Looking at this we see that we have around 20 GPUS that median time between tasks is 2.8 seconds or longer. I suggest that further investigation should be done into these GPUS as the fact that they are taken longer to start new tasks suggests that they may be some issues with them.

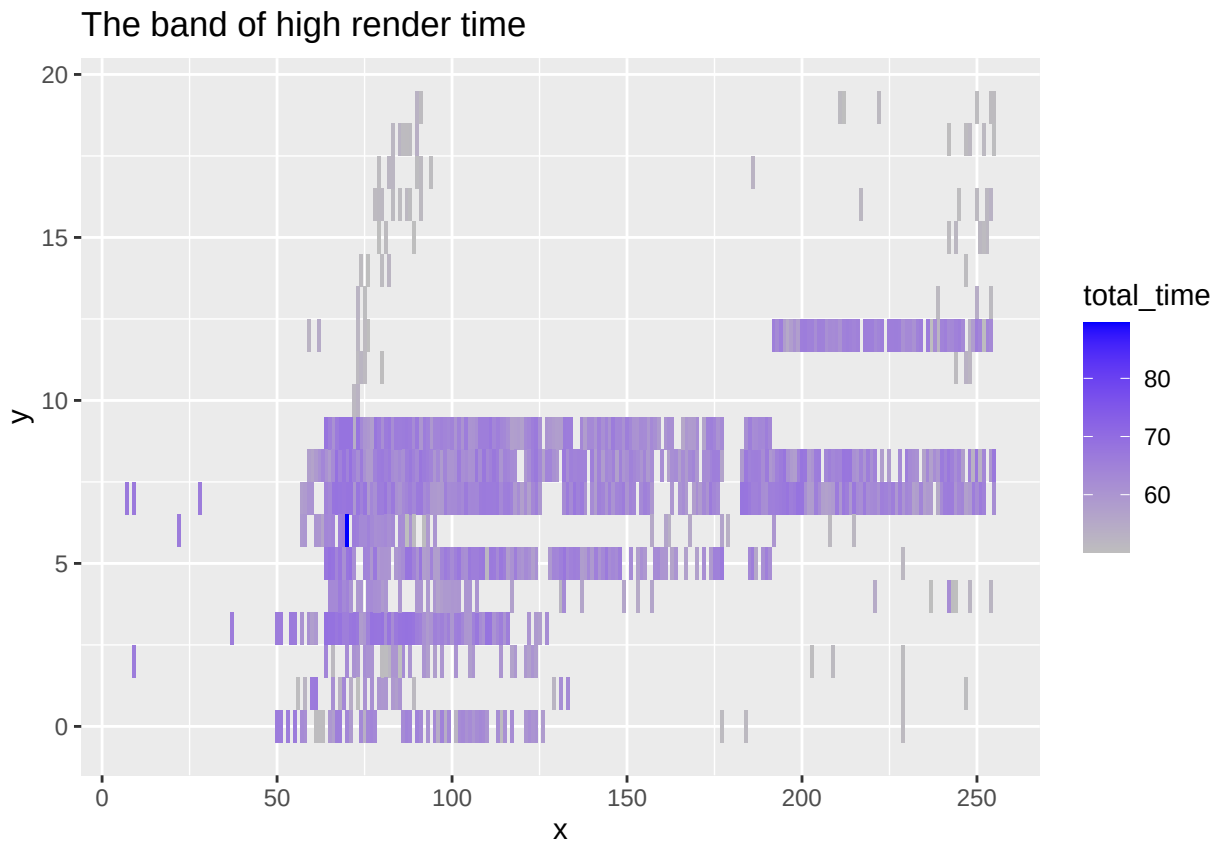
GPU errors - The upload issues

We know that trying to work out which individual GPUS had errors may be difficult so I decided to look into if there where any issues effecting a wide range of GPUS instead.

Mentioned earlier, When we plotted out the x y coordinates and the time taken to fully complete each task we noticed that there are a few tasks that took a while to complete (almost double the expected amount), especially in the low Y cord in the job 12 graph.

```
link_gpu_task_xy <- left_join(link_gpu_task, xy_df ,by = c("jobId","taskId") )

ggplot(link_gpu_task_xy %>% filter(level == 12 , y < 20, total_time > 50), aes(x, y, fill= total_time))
  geom_tile() + scale_fill_gradient(low="grey", high="blue") + ggtitle("The band of high render time")
```

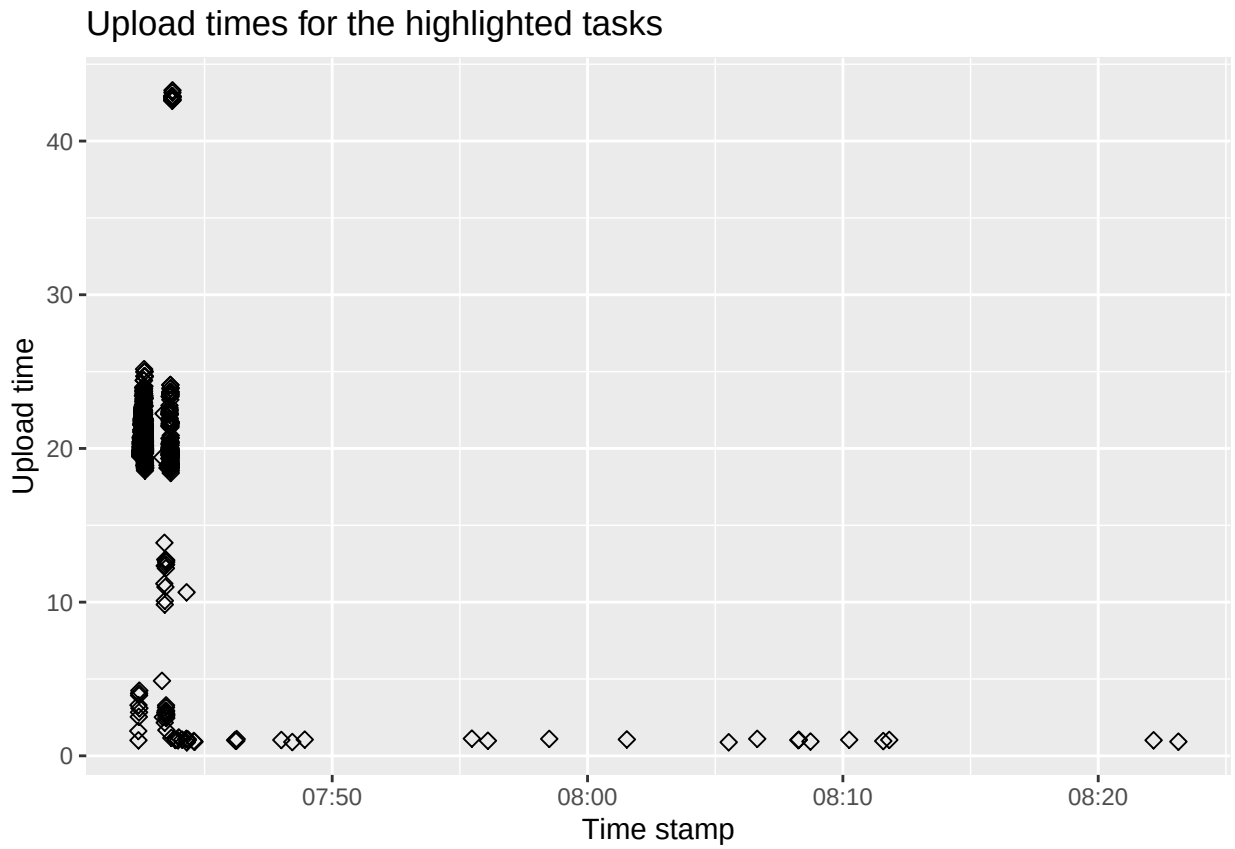


Doing some further investigation into this I found out that this was due to the upload times for these tasks being extremely high.

```
y_lessthan_15 <- link_gpu_task_xy %>% filter(level == 12 , y < 15, total_time > 50)

Over50 <- y_lessthan_15$taskId

ggplot(subset(times %>% na.omit(times) %>% filter(jobId == "1024-lvl12-7e026be3-5fd0-48ee-b7d1-abd61f74
, aes(x=timestamp, y=diff)) +
  geom_point(size=2, shape=23)  + ggtitle("Upload times for the highlighted tasks")+ ylab("Upload time"
```



These high upload times seemed to only really appear between 07:41:00 and 07:44:00. It also seemed to effect alot more tasks than the ones highlighted. This "event" only occurs between this time suggesting that something happened at this point which caused the the uploads to increase by roughly 20 fold. Originally I though that this might be caused by the GPUS switching from doing task from job 8 to doing tasks from job 12 but this dost seem to be the case. I honestly couldn't figure out what caused this so more investigation should be done on this in order to find any errors.

The success and weaknesses of my project

As one of goals was to find potential ways that to reduce the amount of the time taken to produce a terapixel images, I feel like I have been very successful in completing this task. This was important as the less time needed to create an image then the less money needed to rent the IaaS. Through out my analysis I have discovered that the event in which most of the improvements can be made is the rendering. Rendering takes up the most of the time needed to complete each task so and we see that as correlation between time needed to render an image and increases in power draw per tick, heat per tick and GPU utilization. I feel like my investigation into these areas was successfully and methods that I used (mainly comparisons of median) allowed me to get evidence that shows me a clear picture of what was going on and allowed me to answer the question I had asked.

I also went quite in depth with my analysis meaning I was able to find that there is some sort of wait period at the start of each rendering tasks which might be slowing down the whole processes. This could be really important as finding out if this wait is needed or not could take almost 7 seconds off each event run, If this is possible then this could be massive.

I feel like my investigation into if whats in a tile is a massive successes as the evidence I received from that really answered the question and could be really beneficial to future works. Using a K-means clustering algorithm to Color Quantization I was able to work out what exactly was causing some tiles to have a really

long render times via correlation. The correlation graph clearly shows that more shadows and details in an tile the longer to it takes to render. I feel like this is every important to achieving my goals as it tells us what exactly is causing rendering to take so much time. However there are a few weakness with this. Firstly the image I used to do this was a screen shot of the terrapixel so its not the same as the real image (screen shot has a lot less pixels than real image). This means the image I used to perform Color Quantization is not 100% perfect as it has compressed some details down. Secondly the K I used was somewhat random. I chose a K of 15 so that my process would get 15 get different objects(colors). This may not be the perfect number however it did give me enough to work with. To little K and I wouldn't have enough to work with (objects would get blend together) to Big of a K and I would have too much to work with (meaning that It wouldn't be able to classify objects easily). Another issue was that I was only able to see use tiles in job lvl 8. This is because my PC had an issue try to split the image into the 65536 tiles needed for job lvl 12. If i had better hardware I would like to try this.

I do feel like my project has a few weaknesses. I found it difficult trying to find GPUS that where problematic. Originally I wanted to check the median time it took each GPU to complete each rendering tasks against one another, if we could work out GPUs that took the most time per tick the we might be able to find errors. However I had allot of difficulty with this. We know that the rendering event makes up most of the time in a task so its possible that a GPU that is perfectly fine but just had a few tiles that took a while would be labeled as defective. Whilst I did fine some GPUs that where effected by a long time between jobs, I dont think I answered this question to well. I could only find an event that effected most GPUs once (which is important to look into) but couldnt really lable out spefic GPU as having errors.

Another weakness of this project is that it was based around only one run of the creation of the terrapixel image. It would be interesting to see what would change/hold if we ran another investigation into another terapixel image.

Future implications for work from this project

From my project I believe that work should be undertaken on reviewing the rendering processes. We have clearly seen that rendering takes up the majority of the time to complete a tile. We have see that the more detailed a title is the longer the render time, we also see that areas that contain shadows/shade seem to increase the render time and that areas with a high density of trees seem to cause an issue.

However, as the whole point of the terapixel image is to be detailed removing detail to decrease the render time would against the whole point of this project.

Instead I believe that work should be done on the shadows / shading processes. It seems that all the tiles that visibly contain something casting a shadow or being in the shade cause the render time to increase. This might be due to the algorithm needed to create the shadows having to spend calculating how the shadow should fall onto the ground and how it should interact with whatever it touched.

The fact that the shadows are causing an increase in render time tells us if we rendered an image where the sun was at sunset position (so causing longer shadows) then this would cause a increase in render time. On the flip side of this it suggest that the if the "sun" was positioned at mid day (so in the center of the graph) then we should have shorter render time.

Even if the algorithm needs to calculate shadows can be improved by a second or two then in the long run it would shave off a few minuets processing the fully rendered image.

If we are unable to improve the rendering of the terrapixel image its possible that we could use this information to try and predict how long it would take to render a new image. This could then be used to calculate how much power would be used in generating the image meaning that you could possibly work out the cost of the image generating the image / renting IaaS resources.

An investigation should be launched into what the GPU is doing at the start of each rendering Processes. We see that is that it seems to be a 7 second period at the start of every rendering task where its doing something thats not increasing the power drawn in (so I don't think its rendering anything). If this is an error then its possible the we could shave and extra 7ish seconds off each rendering tasks.

Finally a further investigation should be launch into what happened between 07:41:00 and 07:44:00 that caused such a spike in upload times. As this increase in upload time caused some tasks to take twice as long as expected. We need to figure out what had happened and why to prevent this happening in the future as it slows down efficiency.

Reflection

I enjoyed the project, The topic was really interesting and the data that we where given allowed me to try and answer so many different questions. This ment I could try alot of diffrent methods however I felt like I stuck onto using comparision via medians a lot as I felt that was the best method to use to answere the question I was asking. There was 1024 diffrent GPUS doing so many diffrent things that trying to anlaysis them could be a bit overwhelming at times. I feel like going foward I Should try and brach into to diffrent methods more to make my report more visually intresting and it may help me discover new things.

Another issues I had was the I feel like I may have gone a bit overboard in this project and tried to answere alot of questions. Whilist this helped me complete my goals, looking back through my project I think It would of been better if INstead of investergating a new question I went back through what I had previously done and investergate further to try to uncover some more spefic insights / refine my investergation. I think in the future I will try and narrow down my scope so that I focus on one section more to uncover a better answer to a question instead of trying to breifly skim over a bunch of question. This would help make a more detailed report in which I can be more confident in my answers whilst also making the report easier to write.

What I found the most difficult was the fact trying to investergate which GPU where poetenially defective, I feel like this was down to a mixture of my metholody being wrong and there being some many "moving parts" in this data that could effect results that would of made tracking down issues difficult. I feel like I need to inverstergate more strategies to deal with issue like this in the future.

I am really happy with the work I did on the K means colour Quantization as I feel like it adds alot to the project. I had the idea of working out if what was in each tile near the start of the project so I'm really happy its somthing that I ended up touching. Learning about colour quantization was actually really fun to do and inversgate. Know that I sort of understand how it work Im thinking of of ways to use it in the future for somthing.

One issue I had was trying find improvements to the actual technology / cloud structure used in this report. Whilist I read the report that this project was and got a grip with how it works it was really difficulty to get an understaning of how it could be improved. Clearly alot of effort has gone into the methods used to create the terapixel image and I like I dont have the know how to solve this image.

I normally find writting reports difficult but this one came pretty straight foward for me which was nice. I knew what I wanted to sayand how I would structure it. Also I beleive Im starting to really get a hang of R projects, it makes doing a project like this really easly to make make reproducable.