CPSC 4660

Program Report

Shaun Cullen

November 29, 2017

**Testing the Practical Performance of Different Raster Representations in a Standard Web Mapping Environment**

1. **Introduction**

This report will show the practical performance of a web mapping library when trying to visualize large raster datasets. Visualizing large rasters puts stress on full desktop GIS applications such as ESRI ArcMap, so the purpose of this program will be to investigate the benefits and trade-offs of different raster representations that are possible in a modern web mapping library that uses Javascript. The methods used to display the raster will be raw, mipmapped, and tiled., We will see that tiled is the only option for large raster datasets that will have a large amount of scrolling at a large scale (zoomed in close to the dataset).

1. **Methods**

The test program was built with React for the front-end, and for the mapping interface the open source Mapbox-gl.js library was used (https://www.mapbox.com/mapbox-gl-js/api/). This library was chosen for its ability to display raw images, such as satellite imagery, with correct georeferencing. The satellite image itself are usually rotated from true North to face the path of the satellite as it orbits around the Earth, therefore having the ability to display raw satellite imagery correctly on the map was a benefit. In addition, the library supports tiled data sources, such as from a WMS or TMS server.

The raw data was used from the Sentinel-2 satellite, accessed through the Copernicus data hub (<https://scihub.copernicus.eu/>). The image in question is a 10980 x 10980 jpeg2000 true color composite image, at 10m resolution, covering a geographical extent from long: -5.9342 lat: 52.3140 to long: -4.29 lat: 51.356.

Using gdal command line tools (gdal\_translate) the raw image was converted from jpeg2000 to png in order for it it to be displayed on the map. The mipmapped resolution scaled images were created using gdal\_translate as well, creating images at 75% resolution, 50% resolution, and 25% resolution, for the different viewing scales, (zoomed out, mid zoom, and close zoom). In order to generate the tiles for the tile data source, the gdal2tiles.py tool was used, which generated tiles in a TMS format at various zoom levels. These tiles are hosted using a simple http server on the same machine that is hosting the map.

After the images were converted, downsampled, and tiled, they were added to the map as data sources. These data sources can then be added as layers on demand, depending on the current data representation being tested. Using the built in program, the current raster representation can be toggled. Once the raster representation is selected, the user can move the map in various ways (pan or zoom) using buttons on the web page. Once a move is initiated, a timestamp is taken to signal the start of map changes. The mapping library emits a render callback that signals when the map is changing. After waiting a preset amount of time, the program will determine that no more render calls are incoming, after which a timestamp is taken to indicate the end of map updating. After subtracting the preset wait time, the total time for map changing is calculated by subtracting the first timestamp from the last timestamp. This time (in milliseconds) is displayed after each map move. Using this interface, data can be collected to determine the relative time taken to change the map when using each different data source. The time is dependent on browser and computer configuration, but the relative difference between methods should be constant.

1. **Results**

Raw image was tested first, with 2 sets of panning (right 200px, down 200px, left 200px, up 200px) (all times in milliseconds):

Initial Load Time: 11117

**Pan zoomed out:**

Pan Avg Load Time: 886.75

Pan Load Times:

Pan 0: 869

Pan 1: 891

Pan 2: 892

Pan 3: 895

Pan Avg Load Time: 920

Pan Load Times:

Pan 0: 978

Pan 1: 899

Pan 2: 889

Pan 3: 914

**Pan, zoomed mid:**

Pan Avg Load Time: 886.75

Pan Load Times:

Pan 0: 888

Pan 1: 888

Pan 2: 884

Pan 3: 887

Pan Avg Load Time: 901.75

Pan Load Times:

Pan 0: 905

Pan 1: 900

Pan 2: 909

Pan 3: 893

**Pan, zoomed in:**

Pan Avg Load Time: 898.75

Pan Load Times:

Pan 0: 892

Pan 1: 889

Pan 2: 904

Pan 3: 910

Pan Avg Load Time: 890.25

Pan Load Times:

Pan 0: 892

Pan 1: 889

Pan 2: 890

Pan 3: 890

**On average a pan with the raw image takes 897.375 ms.**

Zooming performed using the raw raster image (Zoom mid, zoom close, zoom mid, zoom out, zoom mid, zoom close)

**Zoom Avg Load Time: 899.8333333333334**

Zoom Load Times:

Zoom 0: 862

Zoom 1: 891

Zoom 2: 882

Zoom 3: 890

Zoom 4: 930

Zoom 5: 944

While ~ 900 ms does not seem to bad, from a user interaction perspective this is quite delayed, and this problem will only get worse as the amount of satellite imagery increases. Keep in mind, this is ONE satellite tile only. And we will see from the results for mipmapping and tiled, there is a lot of room for improvement.

Mipmapping only, displaying different resolution at different zoom level, two runs:

**Zoomed Out:**

Initial Load Time: 12659

Pan Avg Load Time: 52.25

Pan 0: 54

Pan 1: 48

Pan 2: 54

Pan 3: 53

Pan Avg Load Time: 55.5

Pan Load Times:

Pan 0: 62

Pan 1: 50

Pan 2: 56

Pan 3: 54

**Zoom Mid:**

Pan Avg Load Time: 217.25

Pan Load Times:

Pan 0: 217

Pan 1: 196

Pan 2: 228

Pan 3: 228

Pan Avg Load Time: 223

Pan Load Times:

Pan 0: 227

Pan 1: 203

Pan 2: 235

Pan 3: 227

**Mipmapped, zoomed in:**

Pan Avg Load Time: 507.25

Pan Load Times:

Pan 0: 505

Pan 1: 512

Pan 2: 507

Pan 3: 505

Pan Avg Load Time: 509

Pan 0: 508

Pan 1: 512

Pan 2: 509

Pan 3: 507

**Zoom changes:**

Zoom Avg Load Time: 773.5

Zoom Load Times:

Zoom 0: 718

Zoom 1: 1231

Zoom 2: 671

Zoom 3: 378

Zoom 4: 672

Zoom 5: 971

Mipmapping, across all zoom levels and operation has an average load time of 333.964 ms. Here we see time reduction of 63% compared to raw raster. This difference becomes even more drastic at far zoom levels. As we zoom closer and closer to the data set, the advantage of mipmapping becomes decreased.

Tiled:

**Zoomed out pan:**

Pan Avg Load Time: 492.5

Pan Load Times:

Pan 0: 645

Pan 1: 676

Pan 2: 645

Pan 3: 4

Pan Avg Load Time: 8.75

Pan Load Times:

Pan 0: 9

Pan 1: 9

Pan 2: 7

Pan 3: 10

**Zoomed mid pan:**

Pan Avg Load Time: 684

Pan Load Times:

Pan 0: 1110

Pan 1: 841

Pan 2: 775

Pan 3: 10

Pan Avg Load Time: 6.5

Pan Load Times:

Pan 0: 8

Pan 1: 5

Pan 2: 7

Pan 3: 6

**Zoomed in pan:**

Pan Avg Load Time: 502.5

Pan Load Times:

Pan 0: 667

Pan 1: 661

Pan 2: 675

Pan 3: 7

Pan Avg Load Time: 6.75

Pan Load Times:

Pan 0: 7

Pan 1: 6

Pan 2: 6

Pan 3: 8

**Tiled zoom times:**

**Zoom Avg Load Time: 243.83333333333334**

Zoom Load Times:

Zoom 0: 752

Zoom 1: 692

Zoom 2: 5

Zoom 3: 6

Zoom 4: 4

Zoom 5: 4

Using tiles we have an average of 277.833 ms, which is a 69% time reduction compared to raw, and a 17% time reduction compared to mipmapping alone. Important to note here, the significant outliers on the second runs of the panning and zooming. Here we are seeing intelligent caching that the mapping library can do with tiling data sources. So when we are seeing the same tiles again, the speed up is significant. Raw and mipmapping times are relatively constant, so the time will never get faster no matter how often we view the same data. Tiling however becomes significantly faster after the first viewing. This means that if we revisit the same data often, the tiling advantage will increase significantly. However, if the map is largely static or is viewed very zoomed out, then the difference between tiling and mipmapping is negligible.

Also of note is the tradeoffs of each representation. Raw has a significant advantage in disk space usage, and amount of files (which relates to storage and transfer costs):

raw png: 140 MB

mipmapped, 3 resampled files + original: 135 MB + 140 MB

tiled-low quality (8 zoom levels): 154 MB, 2039 files

tiled-high quality (12 zoom levels): 996 MB, 69154 files

The results were calculated for tiled using the low-quality tiles, but the high quality tile set was generated to show a worse case. Not only does the high-quality tile representation take up more disk space, it also is comprised of many more files, which affects transfer speed and storage requirements. In addition, the time cost to generate both mipmapped files and tiles must be considered. Mipmapping takes negligible time cost to generate, while generating the tiles in both high and low quality takes significant amount of time.

The setup cost of tiled representations should not be ignored. A TMS server must be created, and knowledge of tiling utilities (such as Mapnik or gdal2tiles.py) is required. This means that some extra knowledge is required on top of the requirements for the mapping library itself. Mipmapping may be good enough when considering the technical knowledge required to set up your own tile server. Overall using a basic javascript web server and gdal2tiles.py was fairly straight-forward, however, other implementations can get significantly more complex.

Finally, the importance of lazy loading is highlighted by the initial load time for each test. When loading all the source types at the start of the program, it takes an average of 12 seconds. The significant portion of loading is the raw image, so when working with the raw image, the initial load time must be considered. The mipmapped image at zoomed out scales and the tiled representation at all scales is very quick to load initially. The best way to load data would be to only load the data when it is required, especially for raw data, instead of loading it at the start of the program.

1. **Conclusion**

The results show that tiled representations are absolutely necessary for all high resolution rasters being displayed using web mapping libraries. The exceptions are mostly static maps, or maps where the scale remains relatively small (zoomed out). All other cases benefit from massive speed up by using a tiled representation. The trade offs of such a performance increase is the required technical knowledge to set up a tile server, and the extra disk space and time required to store and generate the tile datasets. In cases where the map is mostly static or zoomed out, mipmapping may be good enough, although these scenarios do not seem likely to be common in a web mapping context. Finally, raw rasters should not be used at all, as the performance for even a single satellite imagery tile is very laggy, averaging close to a second from user input to map update.