Comparative Case Study Between D3 & Highcharts on Lustre Metadata Visualization

Omar ElTayeby*
Clark Atlanta University

Dwayne John†
National Institute for Computational Sciences

Pragnesh Patel
National Institute for Computational Sciences

Scott Simmerman
National Institute for Computational Sciences

ABSTRACT

One of the challenging tasks in visual analytics is to target clustered time-series data sets, since it is important for data analysts to discover patterns changing over time while keeping their focus on particular subsets. A web-based application to monitor the Lustre file system for system administrators and operation teams has been developed using D3 and Highcharts. This application is a use case to compare those JavaScript libraries demonstrating the differences in capabilities of using them. The goal of this application is to provide time-series visuals of the Remote Procedure Calls (RPCs) and storage patterns of users on Kraken, a University of Tennessee High Performance Computing (HPC) resource in Oak Ridge National Laboratory (ORNL).

Index Terms: D3, Highcharts, web-based visualization, Lustre file system, High Performance Computers

1 INTRODUCTION

In web development, JavaScript libraries are the best platforms which provide built in functions to be utilized. Specifically for visualizing on the web, D3 and Highcharts libraries have been evolving through the implementation of more complex functions which manipulates data, either to visualize statistical or scientific data. The essences of a high quality visualization are the simultaneous features represented and the interaction tools which manipulate these features upon event capturing [1]. These manipulations should give the user the ability to focus on specific subset of data and hover around other subsets. At ORNL, the user assistance team of the NICS department has developed a tool that helps the system administrators to monitor the Lustre file system which is responsible for managing the storage and the input/output (I/O) of the files. The objective of this paper is to compare between the D3 and the Highcharts libraries as being used to build up an application that monitors the users usage behaviour in correlation with the Lustre file systems performance.

2 APPLICATION PURPOSE

In ORNL, the Lustre file system is one of the shared file systems which manage the RPC stream including the Object Storage Target (OST) block I/O stream of the HPCs. The architecture of the Lustre system mainly consists of the Metadata services, Clients and Object Storage Services[5]. Visualizing the metadata for system administrators on portals as web application is portable and easy to use from any device. The purpose of the Kraken's storage visualization is researching the correlation of the users behavior with the OST performance in real time, where the system administrators can use to scroll through time and observe any outliers during system failures. While the Krakens I/O requests monitor helps the system administrators to define the correlation between the system failures and the number of I/O requests.

*e-mail: omar.eltayeby@students.cau.edu
†e-mail:djohn@utk.edu

IEEE Symposium on Large Data Analysis and Visualization 2013
October 13 - 14, Atlanta, Georgia, USA
978-1-4799-1658-0/13/$31.00 ©2013 IEEE

3 MAPPING ATTRIBUTES INTO FEATURES

Before starting any visualization development, the attributes should be mapped into the intended features. In the Krakens storage visualization, the number of files and the total sizes of the files are the basic attributes presented on the x and y axes respectively of a scatter plot, which is constructed using circle objects, and the data is sampled frequently in JSON format. In order to hover around different days, a forward and a backward buttons were implemented to make transitions through these samples. Each circle is tagged by the user name, in order to specify users with outlier activities. The interface at the beginning of the application gives the ability to set up a threshold for both attributes, where the stroke of the circles turns red to be used as an alert, and specify the number of the past samples needed to be hovered upon. It also gives the ability to selecting either the top five users or all of them to show. Fig.1 shows a sample from the visualization of the top five users. To see the actual visualization, please follow this link: http://www.nics.tennessee.edu/ otayeby/paper/example1. For the Krakens I/O requests, three features represent three attributes simultaneously. The number of requests and the number of jobs are presented on the x and y axes respectively, while the diameter of the circles is directly proportion with the number of cores. Fig.2 shows one sample of users with various diameters. To see the actual visualization, please follow this link: http://www.nics.tennessee.edu/ otayeby/paper/example2.

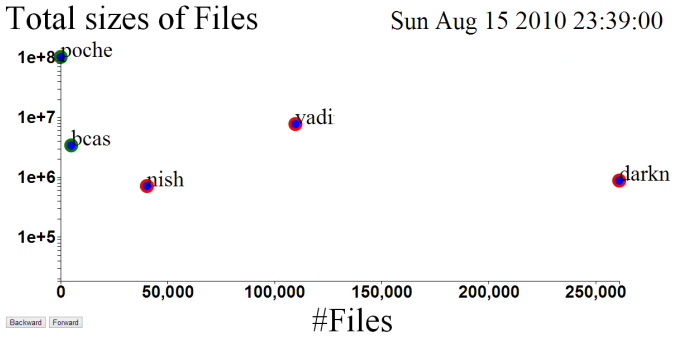

Figure 1: A sample from the Kraken's storage visualization, which shows the top five users, when 3 users have crossed the thresholds and their circles' strokes turned red.

4 IMPLEMENTATIONS IN D3 & HIGHCHARTS

In Highcharts the developer would just need to enable or disable embedded features in a chart fields to use them. The most two helpful embedded features which improved the Krakens storage visualization are the automatic scalability according to the

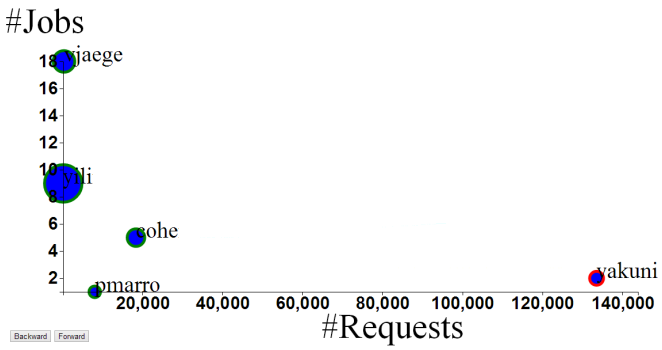

Figure 2: A sample from Kraken's I/O requests monitor, showing users with various diameters, when a user has crossed the thresholds that have been set up for the number of requests and number of cores.

current sample and the enabled zooming option. Even when transitioning to other samples, the adequate scaling will take place. To see the actual visualization, please follow this link: http://www.nics.tennessee.edu/ otayeby/paper/example3. The process of development using Highcharts requires the creation of the container first then render then chart to the container. While in order to bind the chart to data from an external file, the get function of the specific format is called then the variable where the chart is saved is passed as an argument to the data function [4]. The first initial snapshot is drawn using the Chart function. Setting the transition with animations in Highcharts are much more difficult than of that in D3. The only option given to change the object referring to the current sample is setData function, which gives no animation between samples. Thus linear frames in between the current and next samples are calculated first to be stored, and then each frame is drawn through a loop till the actual next sample. All frame calculations and drawings are executed upon clicking the button. On the other hand in D3, all what needs to be done is binding the data to the circle objects then change the index that is passed to the transition function [3]. The circles smoothly transits between samples.

4.1 Implementation challenges

The other disadvantage when using Highcharts for transitions is the speed which becomes an issue to calculate the frames for all users. It takes almost a minute to transit from a sample to another. Initially for drawing the chart there has also a slight delay at the beginning. We used Firebug to profile the JavaScript execution time. The profile when visualizing all users, who are around 3800 in total, the top two functions in own time are in line 43 of the highchart.js. These functions are responsible for rendering the points on the chart, but when it comes to visualizing the top users they do not take that long compared to other functions. However, using D3 the transitions for all users take five seconds more and not smooth as the transitions for the top five users. One of the bottlenecks in visualizing large datasets is the clusters formed due to the high probability non-homogenous scales among the datasets. Fig.3 shows a sample of all users using D3 for the Krakens storage visualization. Thus the need of zooming is mandatory at the clustered regions. In Highcharts the zooming option solves this issue by just enabling the embedded zoom field specified in the chart variable. While instead of a zoom out option, a reset zoom button appears upon the first zoom event to reset to the original scale. The clusters become clearer when the mouse is dragged over their areas. Fig.4 shows a sample of all users dataset, which requires zooming in for the cluster at the corner, it is the overview while the mouse is being dragged to zoom into the cluster. On the other hand, the visualization in D3 needs another event capturer to zoom in the clusters[2].

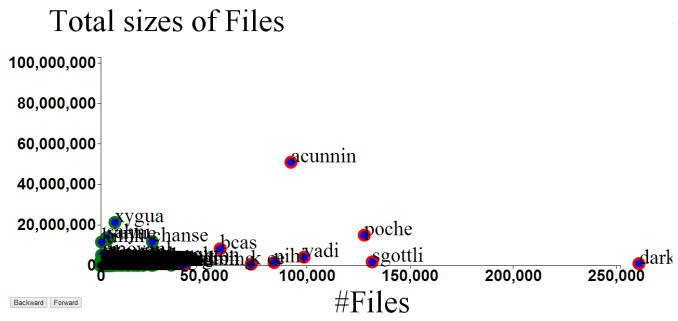

Figure 3: A sample of all users for the Kraken's storage visualization, using D3 showing the clusters formed due to uneven distribution across the dataset.

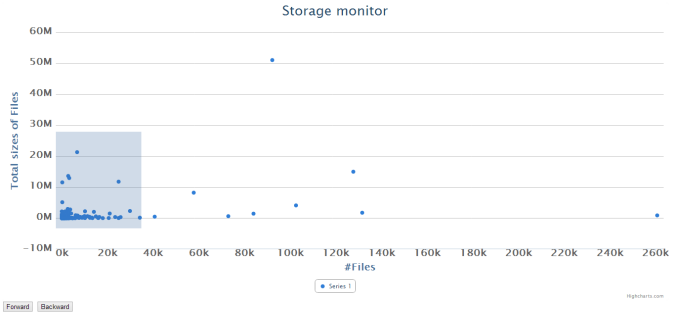

Figure 4: A sample of the Kraken's storage visualization using High charts, while zooming into a clustered dataset.

5 Conclusion

In order to view all comparison points when application was implemented using both libraries; table 1 was created to view the advantages and challenges. The ongoing research now is about multithreading the Highcharts function at line 43, to improve the speed of rendering for large data sets.

References

[1] *Readings in information visualization: using vision to think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.
[2] M. Bostock. Zoomable area chart.
[3] M. Bostock, V. Ogievetsky, and J. Heer. D3: Data-driven documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2011.
[4] Highcharts. *Preprocesssing data from a file*.
[5] Oracle. *Lustre Components*, lustre 1.8 operations manual edition, 2010.

Table 1: Advantages & challenges when using both libraries to implement the application

Aspect	D3	Highcharts
Chart creation	Create DOM	Specify the type of the chart
Transition function	Available	Frames inbetween instances
Rendering points	Objects bound to data	Embedded function
Zoom feature	Event capturer needed	Enabled