Blue Waters Diagnostic Data: Analyzing the Design and Designing the Analysis

Investigator:
Yibo Jiang
2016 Summer SPIN Intern
Undergraduate Student
Electrical Engineering and Mathematics, Senior
Jeremy Enos, Blue Waters Supporting Hardware Systems Technical Program Manager
Blue Waters Project Office

Abstract

Massive diagnostic data are being extracted from Blue Waters every day. As opposed to the simulation data, the diagnostic data is a collection of independent events and/or measurements. Those data provide enormous opportunities for data analytic research, especially for engineers working on Blue Waters who can monitor the system through the data. However, naively scanning through the text file would not be much beneficial given the size of the data. An intuitive approach to address this problem would be data visualization. In this project, a new version of visualizer tools implemented on Blue Waters has been experimented and implemented. Further discussion on the heuristic behind the choices for the design will follow.

1. Introduction

The Blue Waters supercomputer produces an enormous amount of simulation data every day. In the meantime, a large volume of performance and system state data is also produced. Unlike the physically-structured data, the diagnostic data provides insight into the particulars of job performance and the Blue Water system as a whole.
Because of the nature of supercomputer, keeping log file for system administration would be extremely painful to maintain and unhelpful to provide any useful information. To solve this problem, visualization and analytical techniques should be a natural fit.
Therefore, the main of the project is to enhance the current version of the visualizers made by NCSA as well as research and evaluate current effort and tool in visualizing and analyzing those data and tune cost and benefits to something that is reasonable for this project.

2. Literature Review

My project mostly involves two visualizers, ribbon viewer and torus visualizer, which are the two main products made in NCSA to visualize the diagnostic data. In their work [1], Sisneros R, Fullop J, Semeraro B D, et al. provide the basic framework for the ribbon and torus viewer. Ribbon viewer tracks the nodes utilized by different jobs at any given moment. The Blue Waters system has a total of 26864 nodes, 22640 of which are XE nodes and the rest are XKnodes [2]. XE nodes are CPU-based and XK nodes are GPU-based. The ribbon viewer can separate them in different views or combine them in a single window, which makes it easier to monitor the ups and downs of the usage of Blue Water.
Torus visualizer, on the other hand, utilizes a direct mapping of torus coordinates to a 3D Cartesian space. And all the nodes working for the same job are also colored the same. The jobs tend to work better when the nodes are adjacent to each other. So visualizing nodes can certainly benefit the job scheduler research. However, this representation is not perfect. First,

the representation is not technically accurate because of the mapping. There are therefore instances where it is difficult to determine actual distances or bounding boxes of a job layout. Also, as with any 3D space, there is visual occlusion of inner nodes [3]. Alternative flat layout approach has been experimented by Sisneros & Chadalavada[4], but more costs and benefits need to be determined.

The two visualizers may work effectively on their own, but many problems occur when the two are combined together. Further discussion will be presented in the next section.

3. Methodology

Initially, the ribbon viewer was designed to display the Blue Waters usage in a specific time span whereas the torus visualizer was supposed to show the geometric information at every second. The simple utilization of the APIs provided by torus visualizer is not sufficient. One of the problem that arises with this implementation is the synchronization issue. There are many actions of ribbon viewer that may trigger a rendering of the torus. For instance, every time the current time stamp is changed, the torus viewer will query the data and render based on the time stamp. Also, when a single job is selected, the torus viewer will do the same but with only the job id information. The torus viewer is implemented in three.js, which turns out cannot handle the renderings simultaneously. To address this problem, a spinlock-like flag is used to abort the old actions on the torus when the new instructions come. In addition, the previous data from the database is also stored temporarily in order to prevent query overhead. However, further research still needs to be done to make the system more efficient.

Besides the synchronization issue, a lot of work has been contributed to make the ribbon and torus viewer work together seamlessly as opposed to be two separate modules in a single page. Previously, the ribbon would call the functions of torus visualizer without providing much information. But since they both query the same database, more communication can be made available between the two to reduce overhead. In particular, the two visualizers have different criterion as to what should be shown in the view. Since the basic workflow begins with ribbon query the database, the jobs displayed in the view can be recognized by their ids, which can be sent to the torus visualizer. Particularly, the filter options in the ribbon view can be applied to the torus as well.

4. Results

The graph shown below is the current production version of the torus and ribbon viewer. All the ribbons labeled with color are matched in the torus visualizer. With the default all jobs option on, each filter button can select jobs utilizing one type of the node or some small jobs that cannot fit in the normal all jobs view. At bottom of the ribbon viewer lies the summary view which will be rescaled according to the total node count. Different time span can be selected through the data pickers and an update button will appear to enable another query and rendering of both visualizers. Furthermore, a single job can be selected in the ribbon view window that can result in the torus viewer act accordingly by only showing the selected job locations.

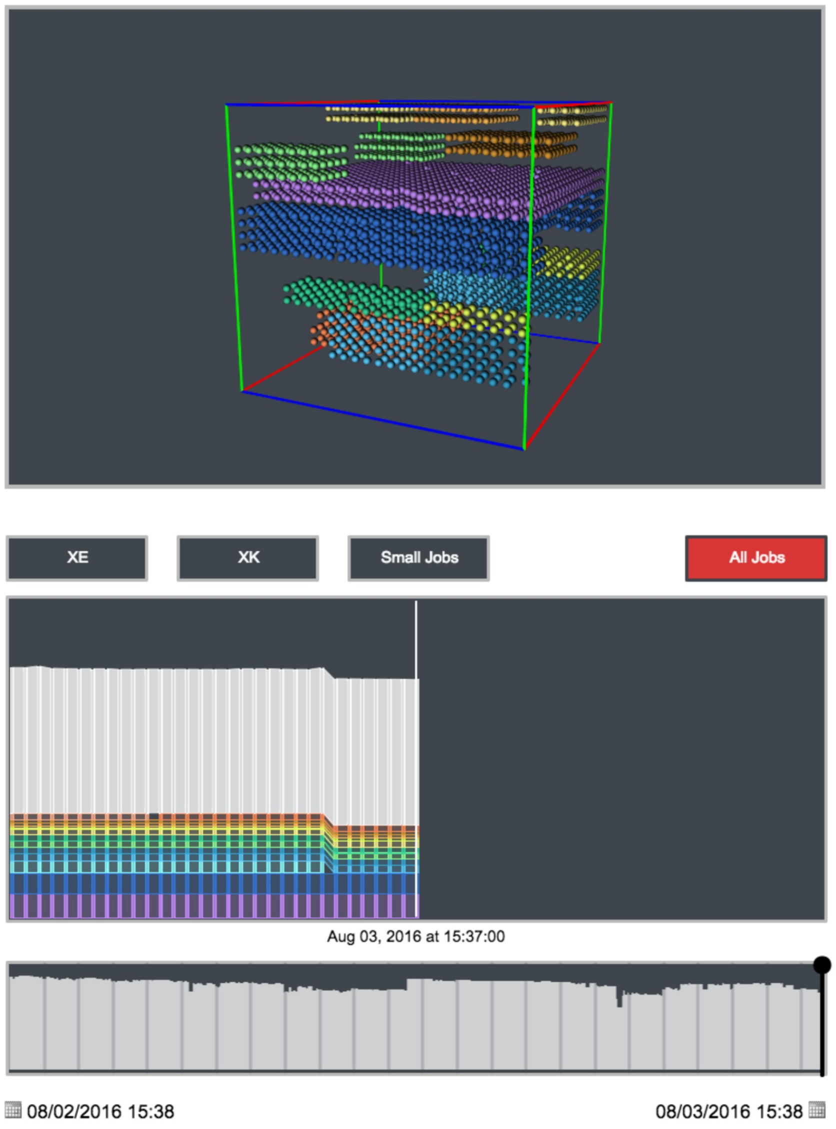

Figure 1: A screenshot of the torus viewer (top) and ribbon viewer (bottom)

5. Discussion

HPC data analytic and visualization can benefit researchers and administrators of the Blue Waters to monitor and maintain the system. The current progress has made it possible for administrators to view the usage of the Blue Waters from different angles. In addition, the HPC diagnostic data is a special set of big data in that it is generated every second and need to reflexed in real time. Future research may even include automatic diagnosis. Viewed in the fashion, the Blue Waters diagnostic data represents a great opportunity for big data research and the technique used in the project can be applied to other field as well.

6. Conclusion

The current version of the visualizers serves as great tool to monitor the Blue Waters. But more still need to be researched. In particular, the query time of the torus takes way too long to complete and is performed at each new timestamp which may not be necessary given the rather stable usage of the Blue Waters nodes. By reducing the query overhead, an animation option that shows all the changes of the usage through a given time period can be implemented. In addition, the viewers are made to maintain the system and further research can be contributed to automatic diagnosis to ease the process.

7. References

[1] Sisneros, Robert, et al. "Ribbons: Enabling the Effective Use of HPC Utilization Data for System Support Staff." *EuroVis Workshop on Visual Analytics, The Eurographics Association*. 2014.

[2] Mendes, Celso L., et al. "Deployment and testing of the sustained petascale Blue Waters system." *Journal of Computational Science* 10 (2015): 327-337.

[3] Sisneros, R. "Visualizing the Big (and Large) Data from an HPC Resource." *Astronomical Society of the Pacific Conference Series*. Vol. 498. 2015.

[4] Sisneros, R., & Chadalavada, K. 2014, in Proceedings of the Cray User Group meeting