EuroVis Workshop on Visual Analytics (2014)
M. Pohl and J. Roberts (Editors)

Ribbons: Enabling the Effective Use of HPC Utilization Data for System Support Staff

Robert Sisneros[†1], Joshi Fullop[1], B. D. Semeraro[1], and Gregory H. Bauer[1]

[1]The National Center for Supercomputing Applications & The University of Illinois at Urbana-Champaign, Urbana, IL, USA

Abstract

Beyond raw computational power, a supercomputer offers the capability of generating and logging a significant amount of diagnostic data. While adding to the burden of maintenance, this data nevertheless represents compelling opportunities for development directed toward improved evaluations, diagnostics, analytics, etc. We have developed such a utility, a visual analytics tool for the support staff of the Blue Waters supercomputer. Our initial goal was broad: provide an informative illustration of current running jobs on the machine for the purpose of system monitoring. Additionally, we were able to collect diverse utilization data to the extent that both minimizing exclusion of as well as intuitively coordinating information were equally challenging. Our primary visual element is an extension of a stacked bar chart to increase horizontal continuity; resulting visualizations show system utilization as a series of concurrent job "ribbons". The remaining elements are common visual/interactive techniques offering expansive functionality. Together these components were deployed as a web application, which is referred to as the "ribbon viewer" by its regular users. In this paper we will highlight the design nuances and development complexities that are belied by the ribbon viewer's apparent simplicity. We will also discuss use-case scenarios in terms of both typical usage and specific examples.

Categories and Subject Descriptors (according to ACM CCS): I.3.8 [Computer Graphics]: Applications—

1. Introduction

Modern HPC (high performance computing) environments produce an overwhelming amount of simulation data. A less common focus is the large volume of performance and system state data also produced. The system generates a variety of data in the form of load factors, temperature levels, network data flows, and other similar data. The job scheduler and operating system also log data such as job launch times, run times, and exit codes. It is possible to mine this "non-scientific" data to determine particulars of job performance. In addition the data can be used to diagnose system problems that may be affecting the performance of the system as a whole.

This data has the potential to provide valuable insights to system administrators and application consultants, but pouring over log output is an unacceptable approach. Visualization and analytical techniques are a natural fit to make sense of this large volume of data. Indeed, much of the system performance data is amenable to standard information visualization techniques, i.e. charts and graphs of nodes in use and load factors are acceptable representations. However, some system data is better represented by alternative display techniques. This is the case with our primary data: system-wide information related to job residence on the communications fabric.

Blue Waters support staff conveyed the essential nature of job related data in even our earliest meetings. Machine issues requiring staff intervention, particularly those reported by users, are routinely directly related to a job or set of jobs. Initially, the only deployed utility was a large table displaying the information associated with the set of running jobs. Through continued correspondence we learned of many potential uses of this data: verifying the system scheduler, system monitoring, diagnostics and reporting, anomaly detection, etc. Implementing this collection of functionality required our approach to cover the full machine and impart relative job sizes, run times, and node layouts.

To this end we developed a visualization technique that

† sisneros@illinois.edu

DOI: 10.2312/eurova.20141139

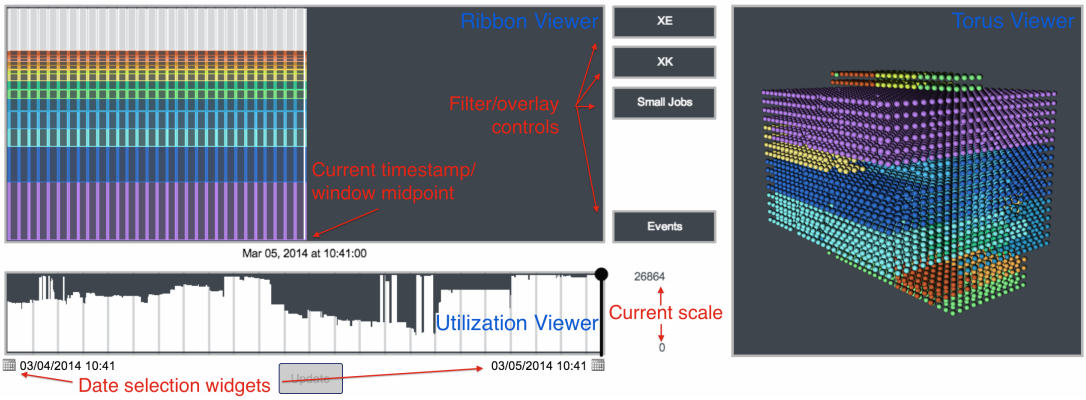


Figure 1: An annotated screenshot of our utility upon startup. During this 24 hour period, a total of 7485 jobs ran on the 26864 nodes of Blue Waters.

displays individual job flow through the system over time. In this display jobs are represented as two dimensional "ribbons" that encode start and end times as well as relative size. Ribbons are stacked and color coded corresponding to job ID. The ID links each ribbon to a set of spheres in the three dimensional topological representation of the torus network found on the Blue Waters system. These visual metaphors combine to provide a powerful tool that has gained use in both monitoring as well as exploring the time history of jobs on the system.

2. Background and Related Works

In their work [ZSB*12], Zhang et al. provide both a survey of the state-of-the art in commercial visual analytics as well as the provide the components common among them: managing, analyzing, and visualizing data. This paper is the result of a collaboration of those with expertise in exactly these areas. The data we manage, analyze, and visualize in addition to obvious temporal components, contains some notion of a spatial component as well. Von Landesberger et al. survey techniques common for this type of data [vLBA*12]. Our approach utilizes the same principles highlighted in that work, namely having an overview while providing zoom and filter and details-on-demand capabilities.

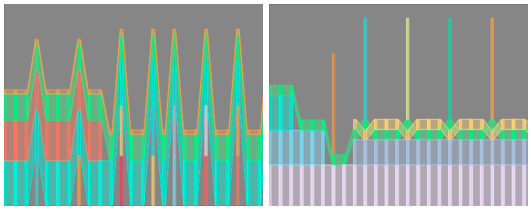
Our primary visual component is the ribbon viewer. The ribbon viewer is a stacked bar chart enhanced to highlight flow across horizontal axis. It is possible to use stacked bar charts alone in a similar manner [HHN00], and a similar technique has also been proposed to link sections [VWD04]. This type of linkage frequently occurs in methods for visualizing categorical data. Categorical data plots similar to parallel coordinates [ID87] link parallel sets [KBH06]. This technique was also extended to interactive object groups over time [BAA*11]. Our data contains multiple such categories that these approaches are directly applicable to, but none which maintain a single job as an identifiable unit in the resulting plot. Equating a job to a category also nullifies the exact component that the visual linkage in these approaches is intended to highlight, i.e. the variations in time dependent categories.

Maximizing throughput is critical given the cost of operation of HPC machines. Monitoring job flow through the system can benefit this effort in a number of ways, and there are many systems in place to do so [MCC04, Bri, Nag, Cac]. However, we find each missing at least one of what we feel are crucial components: the concept of jobs, workflow visualization, or the ability to scale to a machine the size of Blue Waters. There are utilities with heavier visual components such as torus viewing [ILG*12] or parallel components of single jobs [FS10], but to our knowledge none address the specific requests from our user base.

3. The Data

The storage engine utilized is a MySQL database with tables to facilitate the following types of data. The main data objects are jobs. Typical job info consists of an identification number, name, an associated user and group, a status, a queue in which it resides, various times such as the time it was queued, started and ended, and resource requests/requirements by type and number. While it is waiting to be launched, the scheduler tracks estimated start time, queue priority and afterwards, an exit code, actual runtime, and other technical metrics.

Hosts are the logical, OS-based units that execute the codes submitted by the user. Conceptually, one can think of the host as a server that is available to run jobs. The hosts (or nodes) have a node ID (nid), their name, physical cabinet location, coordinates (x,y,z) in the network torus, and class of machine. This is static configuration data that only changes when if the network architecture is modified. The more static host data is collected once by virtue of some script parsing



(a) (b)

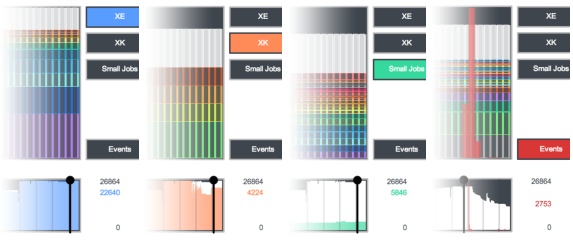
Figure 2: The prototype system displayed problems such as distracting zig-zag patterns (a) and jobs crossing over one another (b). These were removed by enhancing the sorting algorithm to find jobs that start and end within the ribbon window and to increase cross-time awareness for tasks such as careful tie-breaking.

of command-line management tools from the hardware vendor. Formatted once, the data is loaded into the database table. Job data on the other hand is highly dynamic. It is also sourced from multiple scheduling components, each with its own log streams. The asynchronous nature of this data leaves us reliant on ingesting logs and parsing out the relevant log lines and the data contained therein. The job record is in varied stages of completeness during its life cycle and it is necessary to contend with timing delays and out-of-order operations. Furthermore, scheduler codes may generate incorrect data, such as zero Unix timestamps or orphaned jobs that get queued and started but are never assigned a closing record. Therefore we must periodically perform active comparison of database vs. scheduler states to maintain an accurate description of the machine state.

4. The Web Application

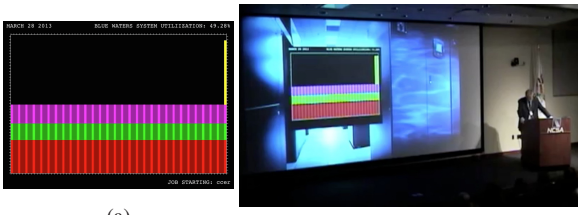
Our utility is primarily javascript code that acts as glue among the various components of the application: the ribbon viewer (Processing), torus view (three.js), data reading (PHP), and controls (jQuery).

With a design focusing on the temporal element, we were



(a) (b) (c) (d)

Figure 3: The interface buttons may restrict the view to only XE jobs (a), XK jobs (b), jobs too small at the current scale (c), or add an overlay of significant event counts (d).



(a) (b)

Figure 4: We were asked to create a ribbon viewer simulation (a) that was used in a presentation (b) during the official launching of Blue Waters.

forced to make one of two concessions: jobs would lack a certain "flow" across time, or the vertical scale of our visualization would not equal the number of nodes. To control the layout we coded our own stacked bar chart in Processing, stacking independently across time and based on size. That is, for every minute in the ribbon window we collect a list of jobs running at that time, scale by node size, assign colors, and plot. The result, as is common with stacked bar charts, contained many visually distracting discontinuities. We then rounded all start/end times to the nearest minute and incorporated a similar notion in the graph itself, i.e. force negative space between temporal units. Now, taking care to avoid issues such as those in Figure 2 and connecting multiple temporal units of a job with quads, we avoided such harsh discontinuities and had a greatly increased flow for each job.

Representation of job placement on the communication fabric of the system is done by displaying spheres colored by job association at nodes in a three dimensional topological representation of the torus network found on the Blue Waters system. Grid nodes represent communication nodes. Each communication node contains two compute nodes so there are half as many nodes represented in the visualization as there are compute nodes in the system. In other words, the torus is represented by a Cartesian grid with the understanding that the boundaries are periodic. This, coupled with inherent visual occlusion may seem prohibitively problematic, but the torus viewer has proven to be valuable in some situations. For example, the likelihood of the network traffic of a job interfering with that of another may be instantly gauged via a glance at their relative layouts.

At startup, the database is queried for all job info for the past 24 hours as well as specific placement info for all currently running jobs. Figure 1 is an annotated screen capture of the utility as it is accessible on the web. The main window is the ribbon viewer which displays individual job flow through the system over time. The window directly below the ribbons covers a much larger portion of time, and is a basic area plot of machine utilization.

This length of time may be changed by the user with the date widget directly under this graph. The faint gray bars in

this graph correspond to each one hour window of time (the width of the ribbon viewer). These windows are linked interactively; the slider widget on the utilization graph updates the ribbon viewer which may also be "grabbed" a la mobile applications and "dragged" for much finer tuning. For the windows to stay visually consistent, the utilization graph is averaged in an overlapping series of one hour averages per single pixel.

The buttons between the ribbon viewer and the torus view allow for further narrowing of which jobs are shown in the ribbon view. These correspond to jobs on the XE (CPU only) and XK (CPU+GPU) partitions, as well as allowing for a zoom-in on small jobs. "Small jobs" refers to the white ribbon at the top of every Figure showing a ribbon view in action. This is the cumulative size of each running job that is too small to map to a single pixel on its own. Figure 3 shows each of these filters in place. That Figure also shows the results of the event button that will overlay on any view the count of logged "significant events", a designation provided by system admins.

Jobs are selectable in the ribbon viewer which highlights the selected job and provides the remainder of the loaded job info. This selection is also shown in the torus view (see Figure 5(b). This is coordinated with a series of accessor functions implemented in Processing that the torus viewer calls to get the correct selected job ID and corresponding color. Each change in the torus represents a new database query, and this is the only factor that detracts from the interactivity of the system.

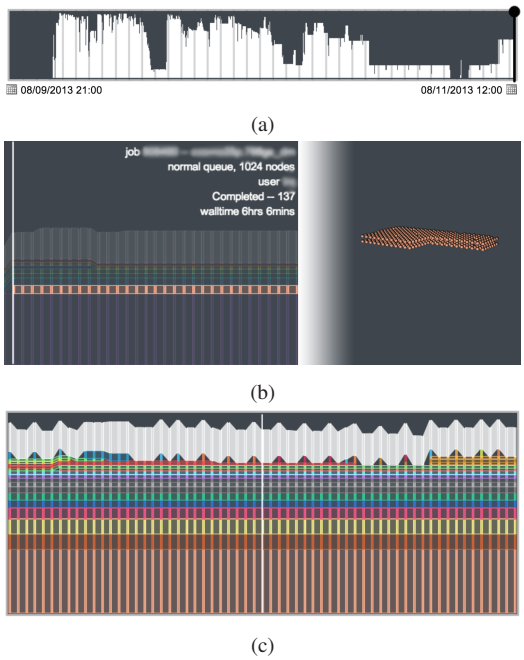


Figure 5: Examples used in practice: an average utilization chart (a), the beginning of a period of poor machine performance that coincides with one job starting (b), and several jobs starting and immediately ending (c).

5. Results

We have maintained communications with users and have received much feedback. This has led to a small collection of feature requests; several features presented in this work are attributable to these. Other responses have been largely positive and a simple ribbon simulation was even used during a presentation for Blue Waters' launch day (Figure 4). The actual web application is routinely used for documenting system events. A recent upgrade of the system had the possibility to adversely affect job scheduling. After the upgrade, support staff used the ribbon viewer to better understand the extent to which this was happening. Follow-up reports on utilization issues included Figure 5(a).

Users are not allocated unlimited time on a HPC resource; maximizing the efficiency of a user's time is essential. This encompasses the necessity of a supercomputer center to sustain expected performance. There is therefore active research, both theoretical and hands-on, toward understanding the root causes of many system conditions, especially periods of degraded performance. On a machine the scale and complexity of Blue Waters, there are many factors that may contribute to these periods. It is for this reason that a common first step is to load the period in question into the ribbon viewer. There have been cases where single jobs, sometimes through unexpected behavior, have crippled the system. Discovering such a job is a simple task with the ribbon viewer. Figure 5(b) is from a diagnostic report where the onset of a system-wide degradation of performance corresponds with the beginning of one job.

Finally, our utility provides a discerning view of the full system state and is arguably most valuable as a tool for monitoring. Figure 5 (c) shows a one hour ribbon window with a conspicuous set of job failures. These all belong to one user and are the result of being over a file system quota. Sudden failure events are observable in real-time which is an important feature as they sometimes represent issues requiring corrective actions. It is also possible to distinguish events that exhibit similar behavior. For example, a drop in utilization may signify the scheduling and imminent run of a large job or the beginning of a catastrophic event. Considering the relative high frequency of occurrence of the former, a drop in utilization itself does not constitute a matter of interest. However, a drop in utilization immediately after a series of logged significant events, as in Figure 3 (d), is cause for concern.

References

[BAA*11] BREMM S., ANDRIENKO G., ANDRIENKO N., SCHRECK T., VON LANDESBERGER T.: *Interactive analysis of object group changes over time*. Bibliothek der Universität Konstanz, 2011. 2

[Bri] Bright-computing cluster manager. URL: http://www.brightcomputing.com/Bright-Cluster-Manager.php. 2

[Cac] Cacti. URL: http://www.cacti.net/. 2

[FS10] FÜRLINGER K., SKINNER D.: Capturing and visualizing event flow graphs of mpi applications. In *Euro-Par 2009–Parallel Processing Workshops* (2010), Springer, pp. 218–227. 2

[HHN00] HAVRE S., HETZLER B., NOWELL L.: Themeriver: Visualizing theme changes over time. In *Information Visualization, 2000. InfoVis 2000. IEEE Symposium on* (2000), IEEE, pp. 115–123. 2

[ID87] INSELBERG A., DIMSDALE B.: *Parallel coordinates for visualizing multi-dimensional geometry*. Springer, 1987. 2

[ILG*12] ISAACS K. E., LANDGE A. G., GAMBLIN T., BREMER P.-T., PASCUCCI V., HAMANN B.: Exploring performance data with boxfish. In *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:* (2012), IEEE, pp. 1380–1381. 2

[KBH06] KOSARA R., BENDIX F., HAUSER H.: Parallel sets: Interactive exploration and visual analysis of categorical data. *Visualization and Computer Graphics, IEEE Transactions on 12*, 4 (2006), 558–568. 2

[MCC04] MASSIE M. L., CHUN B. N., CULLER D. E.: The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Computing 30*, 7 (2004), 817–840. 2

[Nag] Nagios. URL: http://www.nagios.com/. 2

[vLBA*12] VON LANDESBERGER T., BREMM S., ANDRIENKO N., ANDRIENKO G., TEKUSOVA M.: Visual analytics methods for categoric spatio-temporal data. In *Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on* (2012), IEEE, pp. 183–192. 2

[VWD04] VIÉGAS F. B., WATTENBERG M., DAVE K.: Studying cooperation and conflict between authors with history flow visualizations. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (2004), ACM, pp. 575–582. 2

[ZSB*12] ZHANG L., STOFFEL A., BEHRISCH M., MITTELSTADT S., SCHRECK T., POMPL R., WEBER S., LAST H., KEIM D.: Visual analytics for the big data era-a comparative review of state-of-the-art commercial systems. In *Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on* (2012), IEEE, pp. 173–182. 2