Toward Understanding Congestion Protection Events on Blue Waters Via Visual Analytics

Robert Sisneros and Kalyana Chadalavada
National Center for Supercomputing Applications
University of Illinois at Urbana-Champaign
Urbana, IL USA
Email: {sisneros, kalyan}@illinois.edu

***Abstract*—For a system the scale of Blue Waters it is of primary importance to minimize high-speed network (HSN) congestion. We hypothesize that the ability to analyze the HSN in a system-wide manner will aid in the detection of network traffic patterns thereby providing a clearer picture of HSN congestion. The benefit of this is obvious we want to eliminate, or at lest minimize HSN congestion and have a better chance of doing so with a more complete understanding. To this end we have developed a visual analytics tool for viewing system-wide traffic patterns. Specifically, we employ a simple representation of Blue Waters' torus network to visually show congested areas of the network. In this work we will describe the development of this tool and demonstrate its potential uses.**

***Keywords*-File System**

I. Introduction

For a system the scale of Blue Waters it is of primary importance to minimize high-speed network (HSN) congestion. This is evidenced by examining currently deployed preventative measures: congestion protection events. When the system detects a series of packets high transit times across the HSN, a congestion protection event throttles the HSN thereby reducing system-wide injection bandwidth. During such periods all running jobs experience severe performance degradation. There is therefore a clear benefit to gaining more complete understanding of HSN congestion in that such knowledge has the potential to be applicable in future efforts to minimize HSN congestion.

Less ambitious than minimizing HSN congestion we are also interested in simply pinpointing causes of congestion protection events. The difficulty in doing this on Blue Waters is magnified; the architecture is unique in that routing happens on the Gemini HSN for both message passing as well as I/O. There are tools available to make suggestions as to where events are initiated but these are not perfect. In fact, it is common that identifying these locations requires much further analysis of system experts. It is not uncommon for a single job to trigger a congestion protection event. This, in combination with the fact that there are ways to modify a code to balance its network injection provides intuition as to the direct value of locating causes of congestion protection events.

We believe that addressing these requires the ability to analyze the HSN across the entire system. To this end we have developed a visual analytics tool for viewing system-wide traffic patterns. Specifically, we employ a simple representation of Blue Waters' torus network to visually show congested areas of the network. In this paper we will describe the development of this tool as well discuss necessary inputs in terms of I/O and message passing traffic. Such inputs may require a heavy level of code profiling. That is, for each packet transmission we need both a starting location and destination; historically this data does not exist and may be costly to acquire. For these reasons we will perform a cost-benefit analysis for this tool with regard to increased I/O profiling. We will also take the first step in determining the potential for useful heuristics with historical system data.

A noteworthy recent development on Blue Waters is the collection of OVIS data. OVIS [1], [2] is a suite of HPC monitoring tools, under active development at Sandia National Laboratories and Open Grid Computing. OVIS provides a lightweight collection of data of interest from HPC platform components, a variety of analysis and visualization tools to operate on stored data, the ability to evaluate data during collection, and the ability to provide notification to system administrators and/or feedback to platform components. On Blue Waters, a variety of metrics are collected including network bandwidth, packet size, Lustre file system counters and CPU load averages. We feel this signifies that we are on the cusp of a change in the state of the art for the type of analysis discussed in this paper. However, the design focus for this approach to ensure compatibility across these different generations of data collection. We therefore see incorporating and viewing new data, such as OVIS, as a relatively simple task.

In the remainder of this paper we first give an overview of relevant background work in Section II. In Section III we detail the design and development of our visual analytics tool for analyzing Blue Waters' HSN. We finish with a cost-benefit analysis and a case study in Section IV.

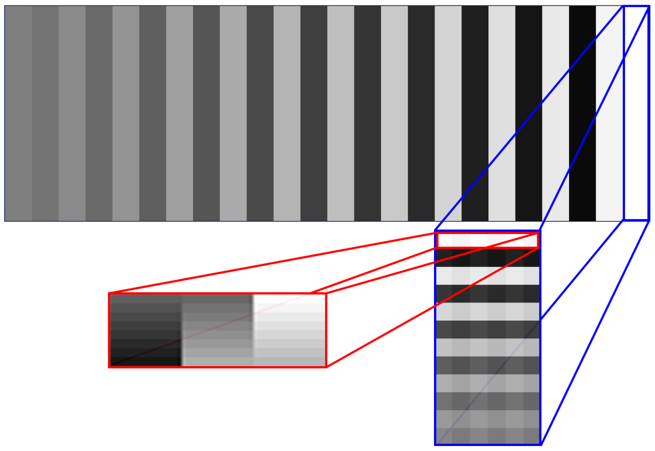


Figure 1: Three images corresponding to the three dimensions of the torus. In the top image, the full torus is colored by its X component. The first pop-out (in blue) is column zero of the full torus colored by its Y component. The final pop-out (in red) shows column zero, row zero and is colored by Z.

II. Congestion Protection Events

Network congestion is a condition that occurs when the volume of the traffic on the high-speed network (HSN) exceeds the network's capacity to handle it. This causes traffic in the network to stall and make slow forward progress. While network congestion is not a unique or serious problem by itself, in the rare case of extreme congestion, serious problems can arise that may affect system performance. On Cray systems that use the Gemini network, periods of intense communication between nodes can cause network congestion.

Congestion on the Gemini network may result from a variety of causes. Congestion occurs most often when application programmers use one-sided programming models such as PGAS (Partitioned Global Address Space) and Cray SHMEM with applications that perform many- or all-to-one communication, and these issues can be magnified by unfortunate job placement. Missing or compromised lanes, routes, links and channels on the HSN can also cause congestion. It is possible that the Lustre Network Driver (kgnilnd) or any of the higher level software that relies on it (e.g. Lustre and Data Virtualization Services) can cause congestion [3]. In extreme cases this congestion degrades system performance, and can even cause code to abort or cause the system to act to protect itself.

To manage congestion on the Gemini network, software that runs on the HSS (Hardware Supervisory System) monitors and throttles traffic injected into the network when necessary. Throttling limits the aggregate injection bandwidth across all compute nodes to less than the ejection bandwidth of a single node. This alleviates congestion in the system, which in turn has the effect of reducing network latency. This trade-off is acceptable because throttling is active only in cases of extreme congestion. The throttle remains active until congestion subsides and the number of congested tiles and nodes drops below their respective low-water mark thresholds. If an application (or combination of applications) persistently congests the network, throttling is reapplied and released periodically until the application terminates. Upon the termination of a throttled application, a message appears in stderr showing the number of seconds the nodes used by the application were throttled. All users running applications while the system is throttled will receive a message upon application completion indicating this throttling event, thus receipt of this message alone does not necessarily indicate that a user application has caused the throttling event. However, all applications running on the system are affected when throttling occurs. Users can modify their applications to avoid causing network congestion and subsequent throttling by addressing conditions leading to network throttling [4].

Cray provides the system operators/Admin with list of nodes that are suspected of injecting high number of packets in to the network. This list can be cross referenced with ALPS and APIDs to determine the application that may have caused a network congestion state and corresponding network throttling event. In most cases this helps in figuring out the culprit application especially if the same application shows up across multiple network throttling events. However, since network congestion is a result of total number of packets in the network, it is impossible to accurately determine a specific set of applications as the root cause for any given congestion protection event.

III. Analyzing Blue Waters' HSN

We believe the ability to study the entire HSN in a single setting is fundamental in understanding or identifying network traffic patterns. Our first step therefore is to find a reasonable representation for the HSN of Blue Waters. The Gemini nodes of Blue Waters are connected so as to create a $24 \times 24 \times 24$ 3D torus. The obvious difficulty in representing a 3D torus is that it is a shape existing in four spatial dimensions. In our experience, a one-to-one mapping of torus nodes to nodes of a 3D mesh provides a fairly

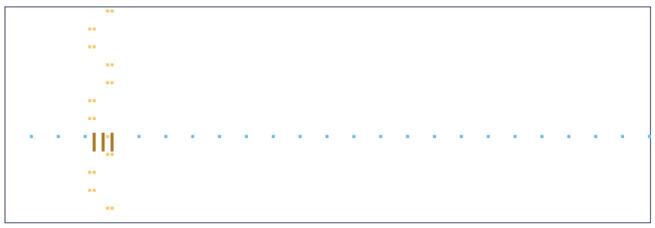


Figure 2: Torus mode. Clicking in the torus shows hops in all three dimensions to make a trip around the torus.

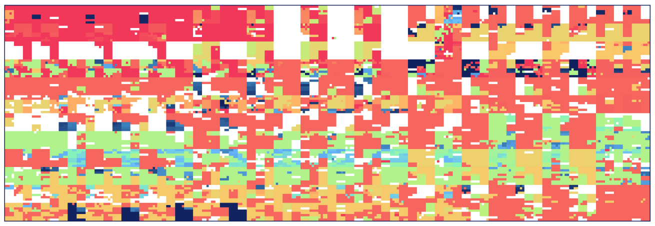


Figure 3: Blue Waters torus as utilized by 353 jobs. Color scale is a variant of typical rainbow colormap.

successful representation of the torus. This success is easily measured in feedback of such a system deployed for support staff of Blue Waters. However, such a mapping has inherent problems such as the visual occlusion of inner nodes and the misrepresentation of actual distances between torus nodes. These limit maximal utility to cases of viewing single, or possibly a few, jobs as they occupy the network. While it would be possible to view traffic patterns, this again is likely limited to a small piece of the network. Furthermore, the issue of cross-boundary connections are much more of a liability in this scenario.

To clarify one point for the following discussion, we will denote the shape of the network as a *practical 3D torus*, i.e. a 3D torus, but only after some simplifying physical assumptions. To our knowledge this is the case or the machine room utilizes four spatial dimensions. Either way a simple photograph of Blue Waters as it sits is an excellent approximation of a 3D torus; this is the inspiration for our layout. That is, our layout is 2D with each "cabinet" occupying an equal number of pixels. Each cabinet is located at a position corresponding to the actual cabinets location on the machine room floor. For the X and Y directions we maintain a realistic ratio of distances between theoretical nodes on a torus and nodes as they exist in the machine. Each cabinet is also flattened, and its space is subdivided by all nodes in the Z direction.

We find this layout amenable to several possible uses. In the remainder of this section we will describe three of these: further understanding the torus itself, viewing relative system-wide job layouts, and viewing traffic directly.

A. *Understanding the Torus*

In development of the visualization aspect of our approach we found ourselves in the rare position to be the target users of the approach. We took the opportunity to to create some simple visualizations and visual modes to for the purpose of simply better understanding the functioning HSN. Figure 1 shows our 2D layout of the Blue Waters supercomputer, with each column colored in a grayscale ranging from $[0, 23]$, i.e. each cabinet is colored by its X component in the torus. There is also a zoom-in on the Y and Z components. The translations from torus space to physical space is revealed in this image. The banding in the zoom-in colored by Y is due to the fact that each gemini node has two torus locations distinct in Y components.

We also found other aspects of physical layout to be unintuitive. For this reason we created the "torus" mode. While in this mode, any area of the window is clickable, and clicking an area will highlight that node and show a trip around the torus in all three dimensions. This is shown in Figure 2.

B. *Viewing Running Jobs*

For every running job on Blue Waters there is a significant amount of corresponding logged data stored in an accessible database. In addition to typical information such as ID, start and end times, exit status, etc., a job's full layout on the torus is also logged. For any point in time, we may therefore query for all running jobs and provide a system-wide layout. Figure 3 shows such a layout when 353 jobs were running. For color selections, we implemented a simple routine that takes an arbitrary number of colors representing a color scale and performs a linear interpolation to create the necessary number of colors to provide a different color for each running job.

C. *Viewing Network Traffic*

To show HSN traffic, we employ a rather simple technique. Connections between nodes are ignored, and bandwidth is instead accumulated at nodes packets pass through. Figure 4 shows an example of passing data from node A to node B. Since there is only one "size" of message in this example, there would be only one color. Therefore, we colored all hops according to torus component, as in

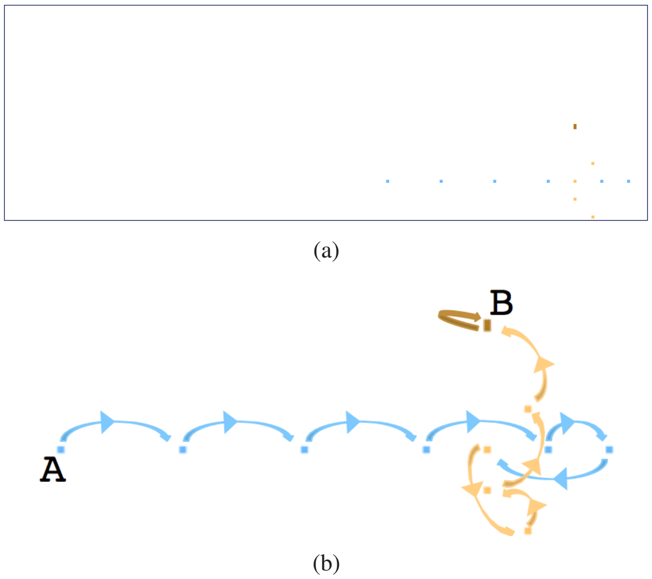
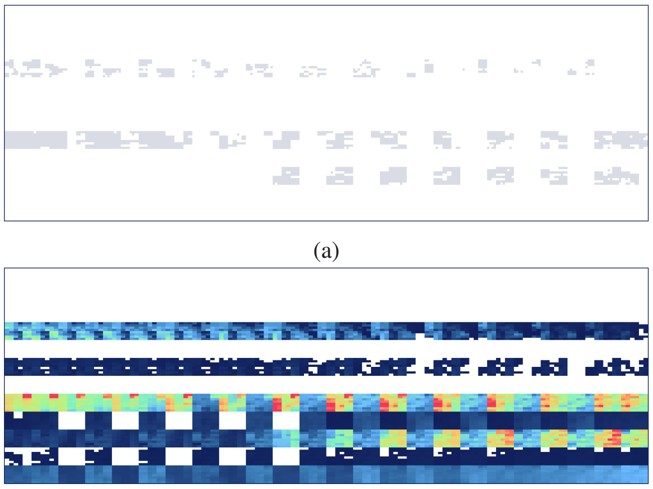


Figure 4: (a) Hops to send a message from node A, torus location $(5, 15, 6)$, and node B, at $(23, 6, 7)$. (b) Annotated closeup of the path in (a).



(a)

(b)

Figure 5: (a) The largest job running in Figure 3. (b) The pattern resulting from a simulated all-to-all communication.

Figure 2. When all torus nodes have accumulated different levels of traffic, we use the same interpolation scheme as in Section III-B.

IV. Results

After development of a full analytic system, we realized that we had no data with which to utilize it. That is, it is not useful without what is possibly costly advanced profiling. Furthermore, it is not possible to use this approach on data that we do have available relating to both jobs and congestion protection events. Therefore in this section we provide details of an alternate use to the proposed system that is applicable to historical data.

A. Case Study: Historical Data During Known Events

We now explore the possibility of analyzing historical data to gain understanding of known congestion protection events. Regardless of the cost of advanced profiling it is simply unavailable for use to analyze past events. Exhaustive information regarding every job ran on Blue Waters has been stored in an accessible database and reports have been logged for each triggered congestion protection event. The compute nodes with the highest incoming traffic are mapped to the applications running on those nodes as this process is likely a good indicator of which applications are causing the congestion. However, since only compute nodes are traceable back to applications all service nodes are ignored. While a practical decision, it is unclear how valuable this ignored information is. All issues are further complicated by the fact that all traffic is shared among all nodes. That is, I/O traffic is routed through compute nodes and similarly with MPI traffic through service nodes.

We have devised a simple test to verify that steps should be taken to analyze service nodes even if compute nodes are the only targets of interest. First we developed a rough heuristic for all-to-all communications. For links between each pair of nodes for a job we create an instance of the data structure described in Section III-C. As in that section, we aggregate the traffic for a single image. Figure 5 shows the layout of a single job on the torus as well as the result of the all-to-all heuristic. Such a communication pattern utilizes a significant portion of torus, at least twice that of the job's allocated nodes.

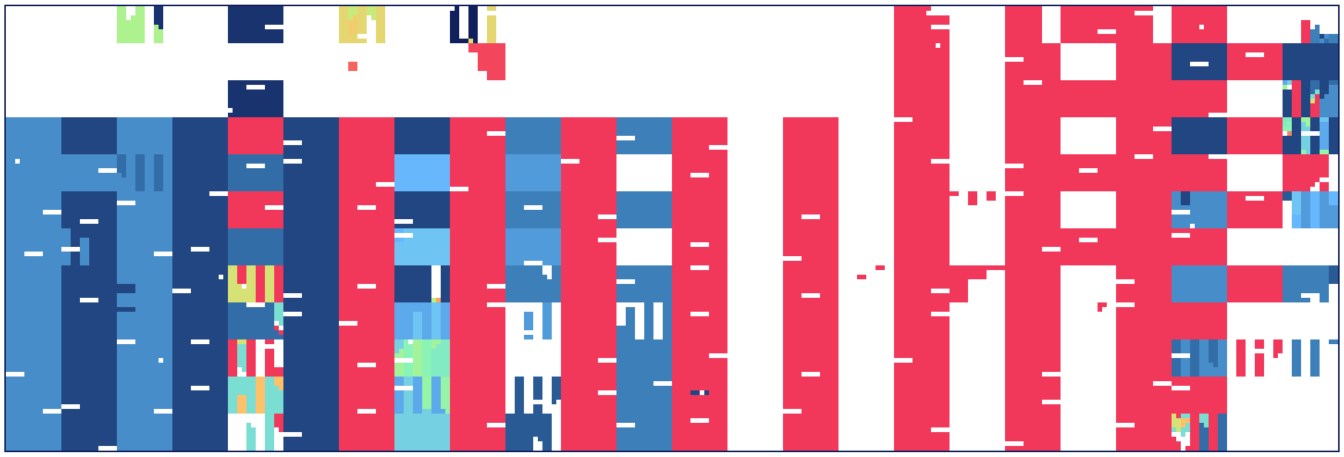
Secondly we identified an Blue Waters' congestion protection event that had the characteristic outlined above, heavy traffic reported on service nodes. This event occurred on March 25, 2014 when 32 jobs were running on the machine. See Figure 6(a) for their torus layouts. The white areas are compute nodes not utilized or service nodes. The latter is likely in areas where the white are surrounded by allocated compute nodes. Two such areas are annotated as "1" and "2" in Figure 6(a). Figure 6(b) is the result of a system-wide all-to-all heuristic corresponding to the time of this congestion protection event. Unsurprisingly, nearly the entire network shows effects of this operation. However, while the service nodes highlighted in area 2 are still clearly visible in Figure 6 (b) as processing less traffic than compute nodes, those in area 1 are indistinguishable from compute nodes. Were this not the case, this worst scenario would provide sufficient evidence that ignoring service nodes is appropriate. Our findings suggest there is at least one case where this is not true.

We have found a period of time where several events occurred. For that same period, we have accumulated the list of all running jobs.

V. Conclusion and Future Work

In this paper we presented a technique for visualizing a high speed network with 3D torus connectivity for the purpose of better understanding congestion protection events on Blue Waters. We developed an experiment to provide a proof of concept that analyzing general HSN traffic is unfortunately difficult. We believe this experiment has several potential areas for future refinement: new heuristics, heuristics based on actual implementations, and code-specific heuristics. Forward progress in this direction has the bonus of applicability to historical data.

We would also like to investigate the possibility of deploying a real-time system for viewing network traffic. The collection of OVIS data on Blue Waters should not only enable this, but eliminate the need for advanced profiling. This will not, however, alleviate the issue of pin-pointing nodes actually responsible for the traffic on any given node. We hope to use advanced profiling as a gateway to a probabilistic system model that can help identify these nodes.


(a)

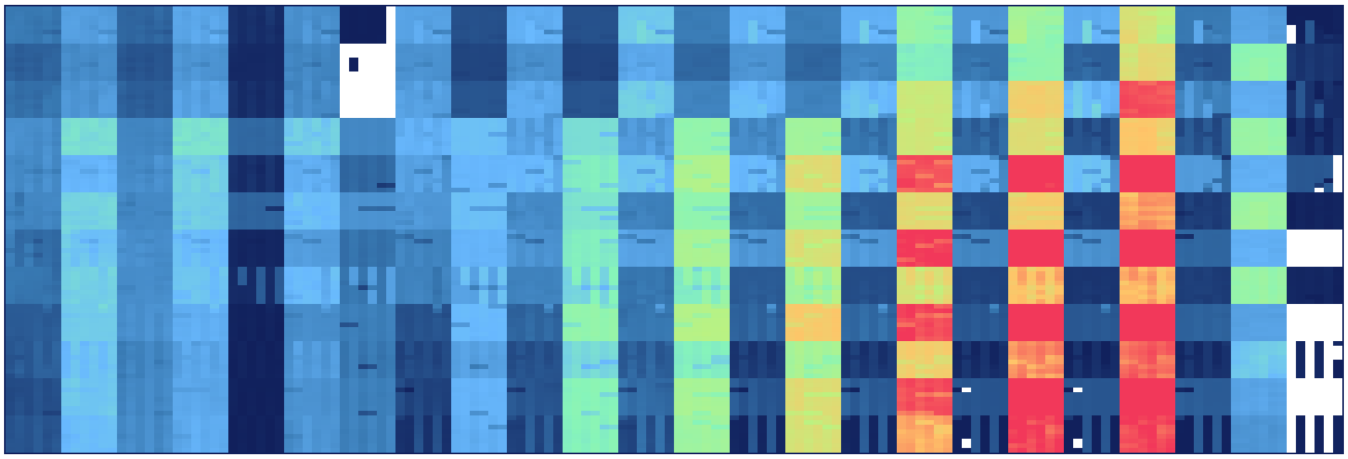

(b)

Figure 6

References

[1] Ovis. [Online]. Available: http://ovis.ca.sandia.gov

[2] J. Brandt, T. Tucker, A. Gentile, D. Thompson, V. Kuhns, and J. Repik, "High fidelity data collection and transport service applied to the cray xe6/xk6," 2013.

[3] *Managing Network Congestion in Cray XE^{TM} Systems*, Cray Inc, cray Doc S00343101a.

[4] *Modifying Your Application to Avoid Gemini^{TM} Network Congestion Errors*, Cray Inc, cray Doc S00313101a.