HPCViz: Monitoring Health Status of High Performance Computing Systems

Tommy Dang
Texas Tech University
Lubbock, TX, USA
tommy.dang@ttu.edu

Vinh T. Nguyen
Texas Tech University
Lubbock, TX, USA
vinh.nguyen@ttu.edu

Vung Pham
Texas Tech University
Lubbock, TX, USA
vung.pham@ttu.edu

Ghazanfar Ali
Texas Tech University
Lubbock, TX, USA
ghazanfar.ali@ttu.edu

Yong Chen
Texas Tech University
Lubbock, TX, USA
yong.chen@ttu.edu

***Abstract*—This paper introduces *HPCViz*, a visual analytic tool for tracking and monitoring high-performance computing (HPC) system events through a RESTful interface. The goals of this tool are: 1) to monitor a set of system events from multiple hosts and racks in real-time statistics, 2) to support system administrators in alarming and detecting unusual signature-based patterns exhibited by health records of hosts in a complex system, and 3) to help in performing system troubleshooting and maintenance with a visual layout for both computing resource allocation and health monitoring map that represent the actual system. A case study was conducted on a medium-scale, *Redfish*-enabled production HPC system with a total of 10 racks and 467 hosts. The result of the case study shows that the visualization tool offers excellent support for system administrators and analysts to profile and observe system behavior and further identify the traces of issues occurred.**

***Index Terms*—HPC visualization, high-performance computing, RESTful API, Redfish, baseboard management controller**

I. Introduction

According to Customer Market Research Services [1], the high-performance computing (HPC) market size is expected to grow from 32.11 billion USD in 2017 to 44.98 billion USD by 2022. This potential growth is driven by an increasing need for HPC systems not only meet the requirements of reliable storage, enhanced scalability, and efficient computing but also quickly handle a large volume of data and accelerate data analysis. In response to this need, HPC systems are set up in many centers and organizations, which presents a new challenge of monitoring these increasingly large-scale distributed computing systems. The meeting of monitoring experts from nine large supercomputing centers [2] reported that hierarchical data management, better power monitoring and control, and standard interfaces for monitoring are among critical challenges to be addressed in the near future.

The existing, widely-used approach of monitoring HPC systems is through the Intelligent Platform Management Interface (IPMI), which is a robust protocol that gives administrators control over remotely deployed servers. It is widely supported by many standard model server hardware from major manufacturers such as Dell, HP, IBM, and Lenovo. However, the main drawback of IPMI is that is vulnerable to remote network attacking because of its control over the system and insecure protocols used in its design and implementation [3].

Recently, the Distributed Management Task Force (DMTF) released *Redfish* [4], an open industry standard specification, and schema for server configuration that aims to supersede IPMI over the network (IPMI over LAN). These embedded firmware web servers are expected to provide end users with simple, secure management of scalable platform hardware by enhancing security and reliability to Baseboard Management Controller (BMC).

Although *Redfish* capabilities are very promising in system management, the adoption of this embedded firmware web servers is still in the early stage of development. It is not feasible for the "data center" admins to monitor hosts using Redfish API monitoring capabilities. On the other hand, Redfish API can be integrated with monitoring frameworks. Monitoring framework can use Redfish API to fetch monitoring data from the host. In this regard, Nagios Core [5] is integrated with Redfish API to fetch monitoring status. Nagios has a web interface which provides a basic view of the hosts and services. However, an administrator cannot capture a holistic monitoring view of the entire data center by using the Nagios web interface. Instead, an administrator has to check textual data row by row among a large number of records. In this study, we propose and develop an entirely new interface to enable visual monitoring through the integration with Nagios and Redfish API.

II. Existing Approaches

There are several useful tools in the literature to support monitoring high-performance computing systems. Massie et al. [6] provided the Ganglia, an open source PHP-based web front-end interface that allows users to gather information via real-time dynamic web pages. This information, including CPU usage, memory usage, disk usage, network statistics, the number of running processes, is plotted on similar graphs. It leverages the widely used XML technologies for data representation. The advantage of real-time responsive on the web front-end, however, leads to high latency due to the size of the XML tree. Thus, Ganglia is not practical on the less powerful machines when the amount of data is large.

Nagios Core [5] is another open-source monitoring system that is capable of handling a variety of servers and operating systems with the industry standard. It provides a basic web interface for the core monitoring engine as depicted in Fig. 1.

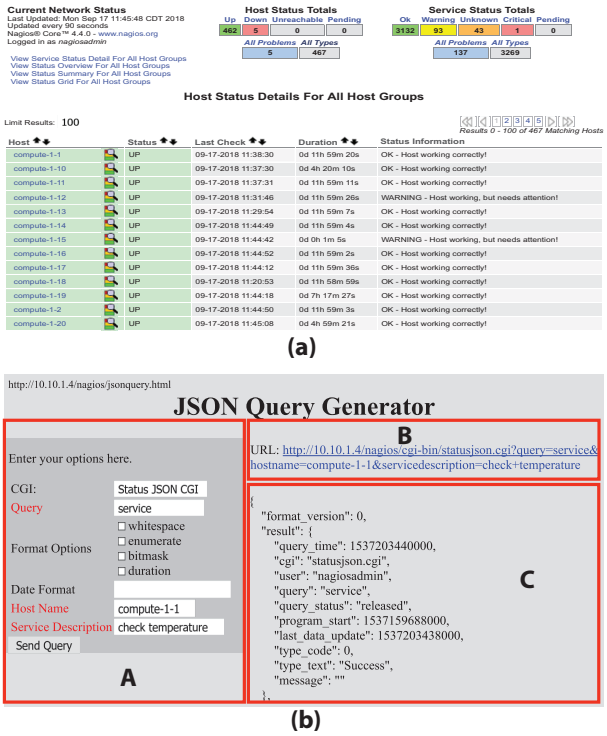

Fig. 1. Nagios Core [5]: (a) simple web interface and (b) RESTful API.

To trace a problem, however, it is challenging for system administrators to navigate through pages of reports on hosts, services, and status. Even though basic filtering operations are provided, system status overview, which is useful to correlate the isolated temporal/spatial issues, can be lost [7]. Users are also able to communicate with the core engine using Nagios Plugins. The commercial version of Nagios (Nagios XI), developed on the top of Nagios Core, has some essential features monitoring through a web configuration GUI.

Designed by Amazon Inc [8], Amazon CloudWatch is a web service that allows end users to collect, view, and analyze predefined metrics in the form of logs, metrics, and events. Users can set the threshold to alarms, visualize logs/metrics besides each other, and take automated actions. However, the main drawback of the tool is that it has a few metrics and is only applicable to Amazon cloud resources.

These aforementioned tools are useful to some extent, depending on the needs of a user. As pointed out by William Allcock [2], the management tools often provide way more functionality than users need, or even worse, it forces users to use the tool in a way it was not designed for.

III. Contribution

Visualization is a process of representing data visually. Allcock et al. [2] indicated that visualization of the data in HPC systems plays a vital role in the management of the system since it allows administrators to explore and understand what is happening without even knowing what they are looking for. Besides, new questions can also be formulated during the exploration process such as how resources are allocated to each job, which in turn allows administrators to make more efficient use of the system. Therefore, there is a need to have a visual analytic tool to help system administrators to manage their daunting tasks efficiently.

This paper introduces *HPCViz* that enables system administrators to monitor HPC systems, in particular, the health status of HPC systems, and thus they can report to end-users about the resource outages as well as expected vulnerability period of disruption. The contributions of this research therefore are:

- it provides a visualization tool for system administrators to monitor system health status;
- it allows system administrators to set up filters and alarms to detect and resolve resource outages promptly;
- we demonstrate its feasibility through a medium-scale, production HPC system equipped with Redfish servers with 10 racks, containing 467 hosts in total (the Quanah cluster at High Performance Computing Center (HPCC) of Texas Tech University).

The rest of this paper is organized as follows: Section IV describes an approach to the tool design. Section V explains the components of *HPCViz* architecture. Section VI introduces fundamental interactions of the tool. We conclude the current study and discuss possible future work in Section VII.

IV. The *HPCViz* approach

HPCViz[1] is developed using JavaScript, a web browser programming language, and in particular the D3.js library [9]. The primary goal of *HPCViz* is to create an interactive visual analytic tool that presents system administrators a high-level view of the health status of many hosts resided in racks. The requirements of the visualization tool are repeatedly refined by closely working with the HPC system administrators and users. The tool should enable system users to: 1) investigate how CPU temperatures change over time of multiple CPUs across hosts and racks, 2) the tool should be able to identify which users are performing jobs related to which hosts, 3) the tool should allow users to set temperature threshold for a CPU for the purpose of quick detection if a certain node needs to be taken care of, and 4) administrators are able to explore other related system services such as BMC health, CPU load, CPU health, fans health, memory health, and memory usage. To meet this goal, this paper proposes several visualization tasks that are implemented in *HPCViz*:

- **Overview Display (T1)**. Display an overview of temperature's distribution over time [10], and users consuming jobs on specific hosts.
- **Details-On-Demand (T2)** [11], including historical temperatures and resource usage.
- **Temperature critical detection (T3)**. Highlight or alarm critical CPU temperature on a host [12].
- **Explore other related information (T4)**. Summarize other system parameters such as BMC health, CPU load, CPU health, fans health, and memory usage.

[1]For a short demo video and web interface application of *HPCViz*, please visit https://goo.gl/DH6ea2

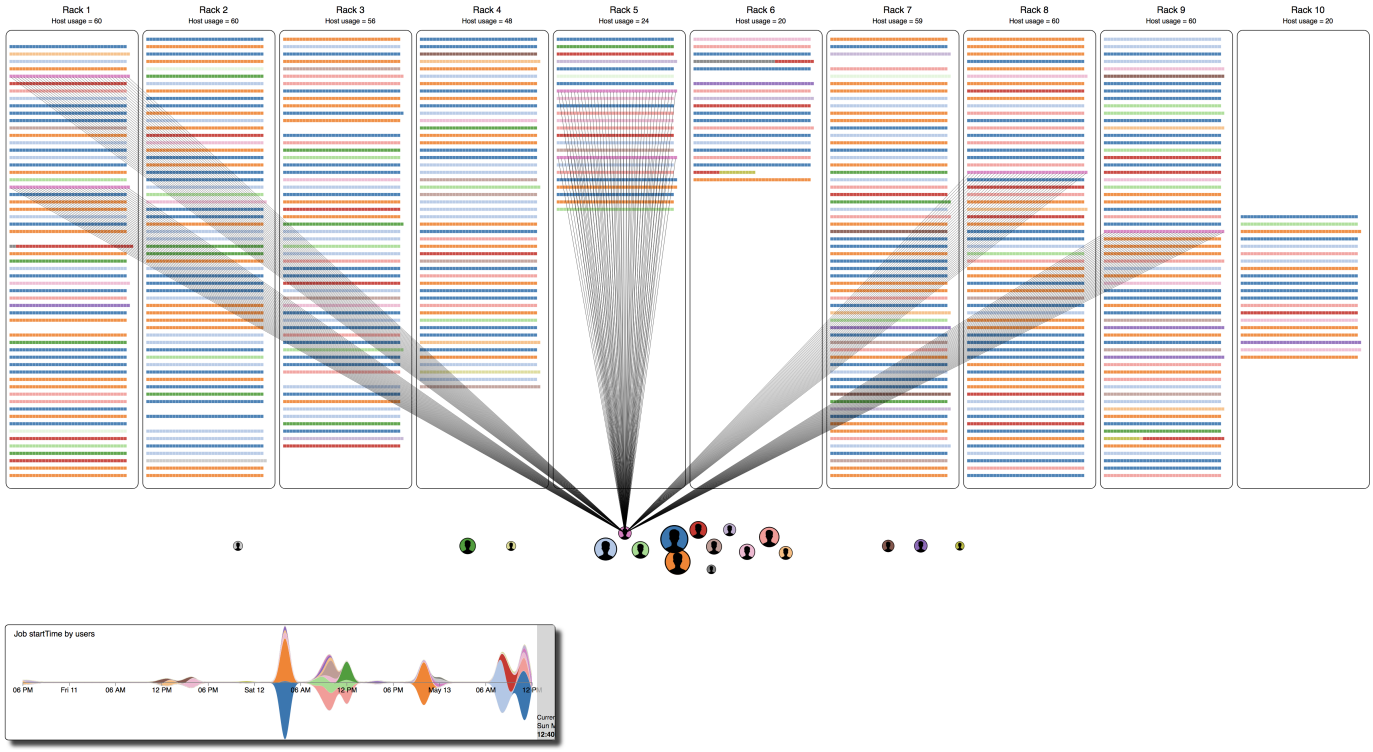

Fig. 2. Host allocations to HPCC users in 3 days from Thursday, May 10 to Sunday, May 13, 2018: (top) Racks and hosts colored coded by the assigned users, (middle) Network of users who can be connected to the allocated hosts, and (bottom) Allocation timeline for three days.

The *HPCViz* leverages the existing Nagios Core engine for data retrieval through a RESTful API web interface (as depicted in Fig. 1(b)). The instructions for available API queries are found in the HTML file (*jsonquery.html*) which is located at the root folder of the Nagios Core web directory. Basically, users choose the desired host or service list on the left panel of Fig. 1(b)-A, the corresponding response result in JSON format will display on the right side in Fig. 1(b)-C along with generated query link URL in Fig. 1(b)-B. Our visualization tool uses this generated link to retrieve data from the system on the fly.

V. The *HPCViz* Architecture

In this section, we first introduce the HPC system spatial layout and then our *HPCViz* visualization on historical and real-time data.

A. HPC system spatial structure and job scheduling

Fig. 2 shows a snapshot of the 467-node Quanah cluster at noon of Sunday, May 13, 2018. This visualization aims to answer the following questions: **Who**- who are the current users of HPC resources? **When** - What time were the resources allocated to these users? **Where** - What is the spatial distribution of these resources in the system? These components are useful for system administrators to keep track of resource allocation and current status of the HPC system. Within the visualization, these components (Who, When, and Where) are linked into a single view. In particular, the top panel displays the rack structure (ten racks from left to right). Within a rack, hosts are listed top down. Rectangles on each row are associated to jobs, which are colored by the owner of these accepted jobs. Notice that, one host can be assigned to multiple users (rows with many colors). The middle panel displays all users who are currently using the HPC computing resources. The node sizes are computed based on how much computing resources that they had requested (and allocated). User information and resource details can be linked (as depicted in Figure 2) and displayed in a popup window on mouseover. At the bottom, we show the actual timeline of when the resources are allocated for a specific user. As we use the same system layout which exhibits the Quanah HPC system spatial structure, this resource allocation map can be used to link the system health monitoring map in Fig. 3 for system troubleshooting, maintenance, etc.

B. HPCViz visualization for monitoring CPU temperature

As discussed above, current HPC standard interfaces heavily rely on text-based layouts and do not allow to present spatial structure and temporal information in a single display. Therefore, visual pattern detections are not intuitively supported in order to help system administrators to track down issues in the event of resource outages. In fact, this is usually a time-consuming process. To tackle this challenge, *HPCViz* introduces a spatial-temporal visualization as depicted in Figure 3.

The Overview Panel shows the spatial distribution of 467 hosts grouped into 10 racks (the visualization task **T1**), repre-

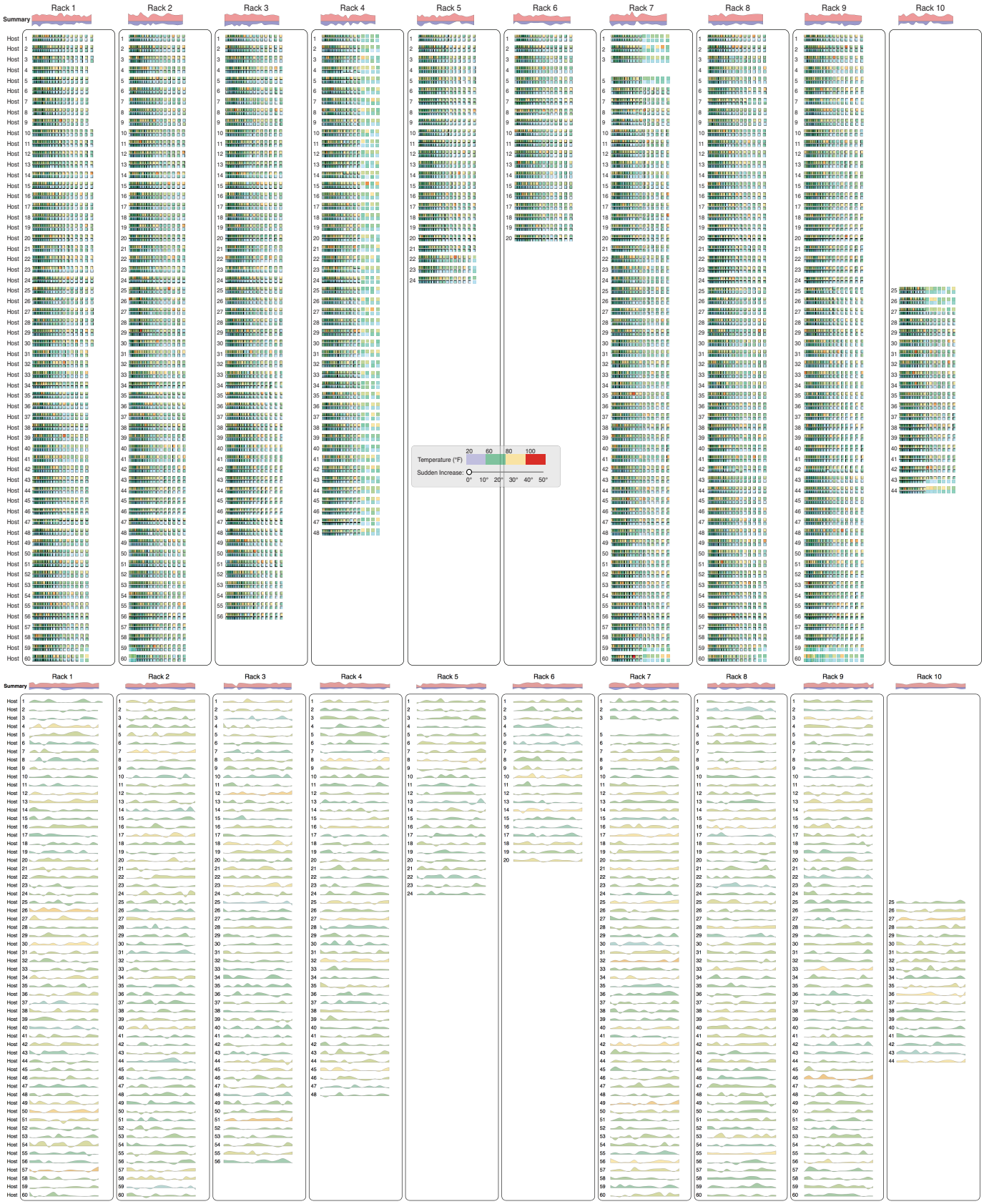

Fig. 3. *HPCViz* visualization for temperature readings using: (top) heatmaps and (bottom) area charts.

senting the physical location of hosts. Each host is located in one row. In the top panel of Figure 3, each color cell represents the CPU temperature returned from Nagios server (through the use of provided API). The temporal distant between two consecutive cells is 2 minutes (as this is also the updating period set up on the cluster). In other words, we scan through the list of 467 hosts iteratively within 2 minutes, make *http requests* on *temperature* readings, and create a new cell color-coded by the associated result (such as OK, warning, unknown, critical, pending). We advise the reader to view our submitted video at https://goo.gl/DH6ea2 for the real-time demo.

Users are provided with another option to switch to area charts as shown in the bottom panel in Figure 3. A graph area chart compared to a baseline (user input) is used to represent the CPU temperature of a host at a given time and it is color-encoded based on the level of heat at the current time. To assist system administrators, a horizontal line is drawn at a specific user input value (by default this value is set as 60).

The Detail Panel A pop-up window is introduced to further display the details of host information on demand (visualization task **T2**). While the overview panel shows general information on all CPU temperatures, the detail panel allows users to unveil more detailed temperature of each CPU on a given host by a set of line graphs. The vertical axis represents the CPU temperature (from 20 to 100 degree of Fahrenheit) while the horizontal axis represents the time.

Under the line chart is the *radar chart* (also known as spider or cobweb chart), which shows seven status information of the system (visualization task **T4**), including temperature, BMC health, CPU load, CPU health, fans health, memory health, and memory usage. The purpose of this chart is to enable users to detect similar patterns and outliers among multivariate observations. While the current visualization focuses on CPU temperature, we can easily adapt it to visualize other metrics such as CPU load, memory usage, or power consumption. This is a desired feature as suggested by domain experts in our informal interviews.

VI. User interactions

HPCViz enables users to interact with the visually represented components through mouse over and zooming. *HPCViz* also supports filtering (visualization task **T3**) on temperature readings. By monitoring and controlling temperature automatically, we can avoid overheats during rush hours and reduce wasting cooling power during downtime.

VII. Conclusion and future work

In this paper, we have introduced *HPCViz*, a graphical tool that allows system administrators to better monitor HPC system health status based on real-time data gathered through the state-of-the-art and industry-standard Redfish protocol and API (v.s. the aging IMPI protocol). To the best of our knowledge, this study is the first of its kind in the data center management, monitoring, and analytic domains. The system also supports integrated features to set alarms on temperature for automatically adjusting cooling units. In future work, this study can be extended for even more powerful features, such as utilizing historical data to predict the health status of the system ahead of a given time frame. Our long-term research goal in this space is to develop powerful, friendly, and highly-efficient tools and methodologies to automate HPC system management, monitoring, and control.

References

[1] C. M. R. Services, "High performance computing market," 2018.

[2] W. Allcock, E. Felix, M. Lowe, R. Rheinheimer, and J. Fullop, "Challenges of hpc monitoring," in *SC '11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov 2011, pp. 1–6.

[3] P. F. Roberts, *IPMI: The most dangerous protocol you've never heard of*, Aug. 2013, https://www.itworld.com/article/2708437/security/ipmi–the-most-dangerous-protocol-you-ve-never-heard-of.html.

[4] S. D. Organization, "Distributed management task force," 2013, https://www.dmtf.org/standards/redfish.

[5] W. Barth, *Nagios: System and network monitoring*. No Starch Press, 2008.

[6] M. L. Massie, B. N. Chun, and D. E. Culler, "The ganglia distributed monitoring system: design, implementation, and experience," *Parallel Computing*, vol. 30, no. 7, pp. 817–840, 2004.

[7] N. Andrienko, G. Andrienko, and P. Gatalsky, "Exploratory spatio-temporal visualization: an analytical review," *Journal of Visual Languages & Computing*, vol. 14, no. 6, pp. 503–541, 2003.

[8] A. Inc, "Amazon cloudwatch," 2012, http://aws.amazon.com/cloudwatch/. .

[9] M. Bostock, V. Ogievetsky, and J. Heer, "D3 data-driven documents," *IEEE Trans. Vis. Comput. Graph.*, vol. 17, no. 12, pp. 2301–2309, 2011.

[10] D. A. Keim, C. Panse, and M. Sips, "Information visualization : Scope, techniques and opportunities for geovisualization," in *Exploring Geovisualization*, J. Dykes, Ed. Oxford: Elsevier, 2004, pp. 1–17.

[11] B. Shneiderman, "The eyes have it: A task by data type taxonomy for information visualizations," in *Proceedings of the 1996 IEEE Symposium on Visual Languages*, ser. VL '96. Washington, DC, USA: IEEE Computer Society, 1996, pp. 336–. [Online]. Available: http://dl.acm.org/citation.cfm?id=832277.834354

[12] R. Amar, J. Eagan, and J. Stasko, "Low-level components of analytic activity in information visualization," in *Proc. of the IEEE Symposium on Information Visualization*, 2005, pp. 15–24.