HiperJobViz: Visualizing Resource Allocations in High-Performance Computing Center via Multivariate Health Metrics

Ngan Nguyen
Department of Computer Science
Texas Tech University
Lubbock, TX, USA
ngan.v.t.nguyen@ttu.edu

Yong Chen
Department of Computer Science
Texas Tech University
Lubbock, TX, USA
yong.chen@ttu.edu

Jon Hass
Dell, Inc.
Austin, TX, USA
jon.hass@dell.com

Tommy Dang
Department of Computer Science
Texas Tech University
Lubbock, TX, USA
tommy.dang@ttu.edu

***Abstract*—Scheduling, visualizing, and balancing resource allocations in High-Performance Computing Centers are complicated tasks due to a large amount of data and the dynamic natures of the job scheduling and resource allocation problem. This paper introduces *HiperJobViz*, a visual analytic tool for visualizing the resource allocations of data centers for jobs, users, and resource usage statistics. The goals of this tool are: 1) to provide an overview of the current resource usages, 2) to track changes of resource usages by users, jobs, and hosts, and 3) to provide a detailed view of the resource usage via multi-dimensional representation of health metrics, such as CPU temperatures, memory usage, and power consumption. To support these goals, our visual analytics tool provides a full range of interactive features, including details on demands, brushing and links, filtering, and ordering. The visualization tool is demonstrated on the HPC center of 467 computing nodes.**

***Index Terms*—Multivariate analysis, Parallel Coordinates, Customizable Radar Charts, Job Scheduling, Health Metrics, Power Consumption, High-Performance Computing Centers**

I. Introduction

Job scheduling and resource balancing are the fundamental problems of efficiently managing High-Performance Computing (HPC) Centers [31]. These are challenging problems due to several constraints: (1) Job scheduling needs maximize the number of jobs/users to be served and minimize the waiting time and (2) Resource allocations needs to be balanced to avoid over-usage on some hosts while others are halted. The requirements (1) and (2) are tightly connected. While the first requirement comes from the users, the second requirement tight to the system. Due to their complexities and multiple constraints, there is no globally optimal solution for this problem. Usually, heuristics are involved in trading-off user expectations for simplicity. In other words, we can impose a list of hard and soft constraints with priorities so that the system can make the decisions of job scheduling and resource allocation within a given amount of time [4]. This process involves little human interference and usually not adaptive to the dynamic nature of the problem [30].

This paper aims to make this process more transparent to the users, especially to the system administrator who might be able to interfere and modify the scheduling algorithm to adapt to new requirements/changes. The contributions of this paper are three-fold:

- We introduce a scalable prototype, called *HiperJobViz*, for visualizing and monitoring resource allocations in HPC centers. The visualization provides summary views with respect to resource usage by users, jobs, and hosts which can be expanded into detailed view via user interactions with the system.
- We propose an approach to investigate and represent multi-dimensional status of the computers in HPC centers. Using this characterization approach, we group computing nodes into typical clusters to provide a system status overview and track changes (switching from one group to another) of computing node over its observed intervals.
- We demonstrate our interactive interface on a medium-size HPC center at a university. However, the work can be scaled to larger systems as we provide summary views of the major situation/system health status.

This paper is organized as follows: The next Section present related research in job scheduling and resource allocations in HPC centers. Section III presents our motivation, design choices, system overview, and details on major components of our system. A use case of *HiperJobViz* is demonstrated in Section IV. Lastly, Section V concludes the paper and present future direction for this work.

II. Related Work

A. Job scheduling

Job scheduling plays an important role in managing HPC centers, especially in heterogeneous distributed computing systems [11]. There are many algorithm/efforts in this direction [26]. A batch scheduler allows system users to use resources without worrying about the node availability and the interference of other jobs [6]. While there is no best scheduling algorithm, each has different goals/priorities, such as maximize the allocated users, minimize the average waiting time (before the allocation), minimize the completion time [30], or consider energy-efficient hardware [20].

978-1-7281-5991-1/19/$31.00 ©2019 IEEE
DOI 10.1109/DAAC49578.2019.00009

Zeno [28] apply machine learning methods to identify and diagnose stragglers for jobs. Lu et al. [19] performed a deep analysis of different types of imbalance in the Alibaba Cloud: spatial imbalance, temporal imbalance, proportional imbalance, and resource demand imbalance. Jiang et al. [16] explored the characteristics of co-allocated online services and batch jobs, revealing that half of the failed tasks are halted, leading to waste of time and computing resources. In distributed computing systems, Liu and Yu [18] found that co-located online services and batch jobs are usually managed by distinct schedulers which might have different scheduling algorithms and priorities.

B. Resource monitoring

Nagios [3] is an industry tool for HPC infrastructure monitoring, including hosts, associated hardware components, networks, storages, and services. Nagios can have two modes: active and passive. In an active mode, Nagios execute a plugin and pass the node IP address. The plugin will then check the operational state of the node or service and report the results back to the Nagios daemon. In a passive mode, passive checks are initiated and performed by external processes, and results are submitted to Nagios for processing. CHReME [22] is another a web-based interface for monitoring HPC resources, focusing on basic visualizations similar to Nagios web interface. Amazon CloudWatch [15] visualize log data and allow users to define alarms on different metrics. Similarly, Splunk [7] investigates log data in multiple formats (e.g., csv, json) in real-time. However, Splunk experiences slow performance on large datasets [14]. Grafana [12] standard charts which can be easily adapted to visualize HPC data. However, Grafana does not support more complicated high-dimensional representations, such as parallel coordinates [25] and scatterplot matrices [10].

In this paper, we apply the high-dimensional representations to represent health metrics of computing nodes and abstract them as different visual situations of the HPC centers [8]. This provides system administrator a high-level of system status and keeps track of jobs/nodes/system changes over time.

III. The *HiperJobViz* Approach

The primary goal of *HiperJobViz* is to provide a high-level view of current jobs and resource allocations in HPC centers. The work aims for the following visualization tasks [1]:

- **T1:** to provide a comprehensive overview of the current jobs, users, and their resource usages [29]
- **T2:** to track changes of resource usages by users, jobs, and hosts [2]
- **T3:** to project the resource usage into system health metrics using multidimensional representation [17]. These health metrics include CPU temperatures, memory usage, fan speeds, and power consumption.

The data for our visual analytics tool comes from two different sources:

- Job scheduling data: This data contains information on who are the current users of the system, what are the jobs associated with each user, and which are the computing nodes allocated for each job.
- Multivariate status data: This data contains health metrics of computing nodes in the HPC center.

In the next section, we explain in details how these data have been collected and provide some visual examples of the data. Data for this paper was obtained from a 467-node cluster at a university.

A. Job scheduling data

This data contains information about users, allocated hosts, and job scheduling collected via Univa Grid Engine (UGE). A user may have submitted multiple job requests. Each job requires multiple computing nodes (each has 36 cores). This formulates a hierarchical structure of three levels. Besides the hierarchical scheduling data, we also have the temporal data on each job: when is the job submitted and when is it started. Therefore, the hierarchical network of users, jobs, and hosts changes overtime. Capturing the dynamic structure is a daunting task [9], especially for large and complicated networks.

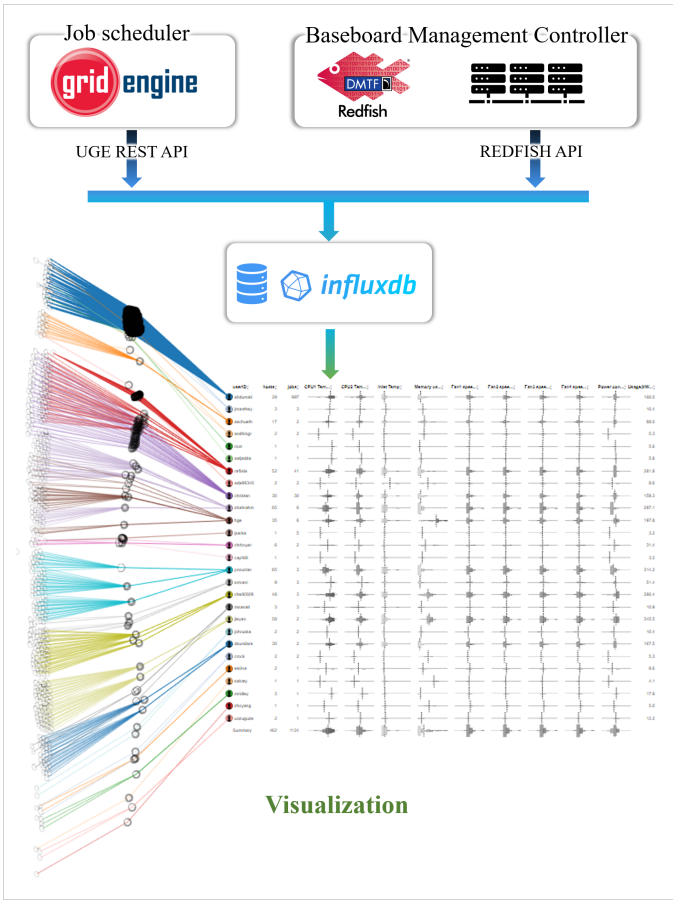

Fig. 1. Schematic overview of our *HiperJobViz* visualization.

B. Multivariate status data

These node metrics are collected through Baseboard management controller (BMC) via Redfish API [24]. The data

collection component performs three tasks: 1) queries and collects data across the entire cluster being monitored by leveraging multi-threading and parallelism; 2) builds metrics after receiving the monitoring data, and 3) stores health metrics in *InfluxDB*. The system currently supports a total of nine measurements, including CPU1 temperatures, CPU2 temperatures, Inlet temperature, memory usage, fan1 speed, fan2 speed, fan3 speed, fan4 speed, and power consumption. While these readings on some computing nodes are pretty stable, the others are highly dynamic. We want to be able to capture this temporal behavior and explain the fluctuations in the time series.

Figure 1 shows a conceptual overview of *HiperJobViz* system. In particular, the first procedure scatters the monitoring tasks evenly across the available CPU cores as a multi-threaded code and gathers the responses. Each thread makes use of "Fetch Metric Data" procedure to query and collect monitoring data through Redfish-enabled BMC and UGE job scheduler. After monitoring data is collected, metrics are generated and written to *InfluxDB* for persistent storage. The visualization component displays both job scheduling data and health metrics of 467 computing nodes in the system retrieved from *InfluxDB* via a REST API.

C. The HiperJobViz Visual Components

The visual design of *HiperJobViz* consists of two main components for two types of data collected from the previous section.

Job scheduling visualization

The job scheduling component allows a system administrator to visualize when a job is submitted (orange mark) and when it is started (green mark). Figure 2 shows an example snapshot of the system. The gray bars indicate waiting periods while green bars represent the running jobs. We can notice that for each user, the number of hosts is usually higher than the number of jobs. In other words, multiple computing nodes can be allocated for performing a multi-threaded process in parallel. However, this is not always true. For example, *abdumali* user (highlighted in the red box of Figure 2) has 997 jobs, sharing 29 computing nodes. That is, most of his jobs use a single core. The schedule on the right also indicates that the 997 jobs have been submitted and started at the same time without any queuing time.

While the left-to-right layout in Figure 2 can clearly represent the scheduling information of a large number of computing nodes in a limited display, it can show neither the hierarchical data nor multi-dimensional status of the computing nodes. We can alleviate this problem by converting the linear layout to spiral charts and embedding them directly into the hierarchical representation. The middle column of Figure 3 shows the same scheduling data in Figure 2 where the spirals grow outward, and a completed circle is one day. Additionally, the radar charts representing the major multi-dimensional health metrics of the allocated computing nodes in the system are displayed on the left. Instead of displaying 467 radar charts, we select the 11 representative radars (or groups)

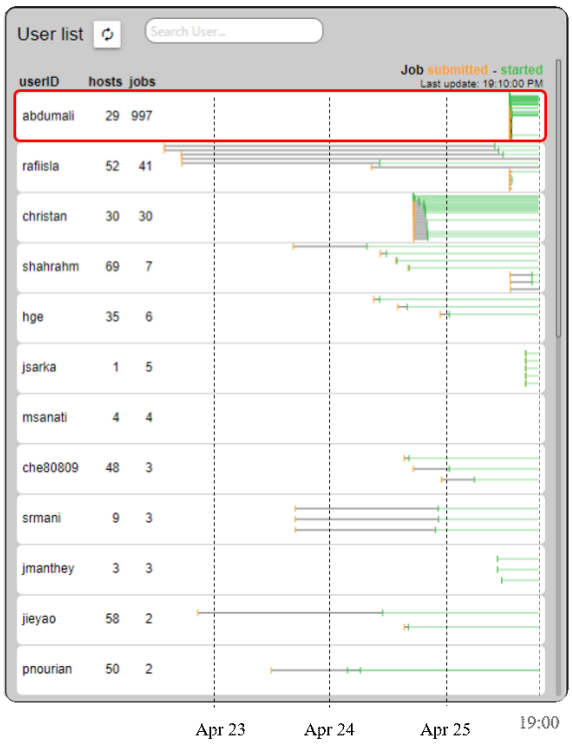

Fig. 2. Time series visualization for HPC job scheduling data: Gray horizontal bars represent waiting periods while green bars are currently active jobs.

on the left using k-means clustering algorithm [13]. The links between jobs and radars indicate the multi-dimensional node status of these jobs. The link thickness is encoded by the number of nodes in this status. The top-left radar is the most popular status as there are many jobs using nodes imposing these measurements. The same user **abdumali** is highlighted in red in both figures. The multi-dimensional radar representation will be discussed next.

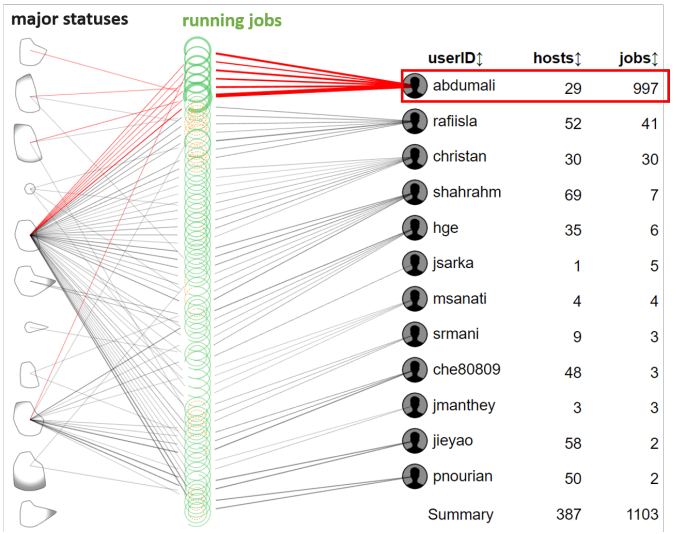

Fig. 3. Hierarchical structure of the job scheduling in the HPC center for the same users in Figure 2. The left radars are the 11 major statuses of nodes.

Multivariate health status visualization

There are different approaches to visualize high-dimensional data. Parallel coordinates [25], scatterplot matrices [33], and Radar charts [23] are typical examples in this class. Due to a large number of data entries (or computing nodes, which is 467 in this example data set), parallel coordinates and scatterplot matrices are unsuitable choices since they require more screen displays. The radar chart organizes health metrics in a circular manner, and a closed curve travels through associated values on each dimension representing the "morphology" of a data profile for intuitive visual comparisons [27]. Similar to parallel coordinates, the order of dimensions is significant in perceiving data profiles in radar layouts [21]. Therefore in this system, we allow users to customize the radar configuration, such as selecting, ordering, and positioning the data dimensions on the circular layout. Figure 4 shows two examples of the same summary data (representing min-max band on each dimension) on two different user configurations. As depicted, the shapes (or morphology) of two configurations (of the same data) are very different and may carry distinct visual values to the viewers.

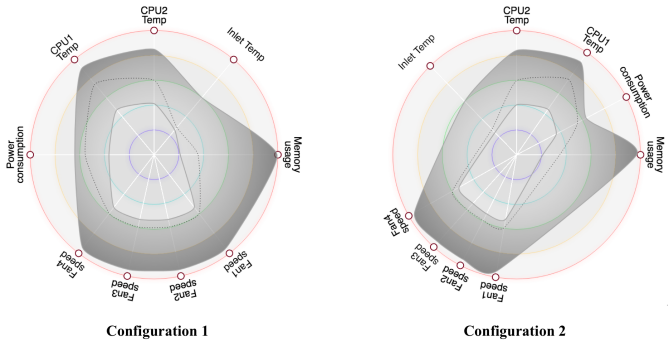

Fig. 4. The same example summary data on two different radar configuration: Users can drag the red circles to re-position variables on the radar layout.

Users customize the radar layout by directly dragging the associate dimensions on the summary radar chart manually selecting the angle in the table below. The second column in Figure 5 shows the distribution summary of node measurements on each dimension from the lower to higher thresholds. For example, on the CPU temperature dimensions, the lower threshold is 32°F, and the higher threshold is 98°F. The red dots on each dimension are outliers determined by the Box plot rule [32]. The last column in Figure 5 allows users enable/disable the corresponding variables. This allows users to focus the analysis on a subset of variables of interest. The configured layout affects all radar representations in the main view, as well as the k-means clustering results.

D. Multivariate clustering of computing nodes

To characterize the major health status of the system, we perform a modified k-means clustering on the nine measurements of computing nodes [13]. We modify the original k-means algorithm by (1) starting at a random entry in the high-dimensional space instead of a random position and

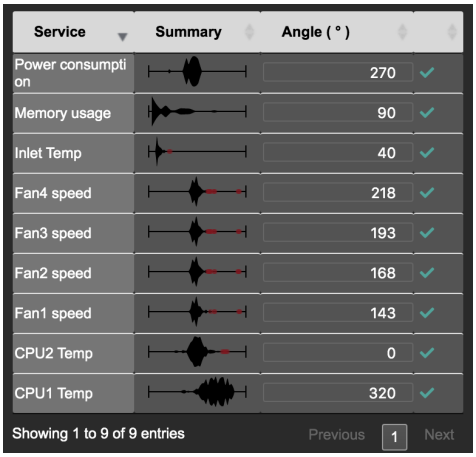

Fig. 5. The configuration table for nine measurements of the radar layout. Users can use the green ticks to enable/disables variables.

(2) passing the whole collections of computing nodes only one time (no refinement) to reduce the clustering time. This modified clustering algorithm is similar to the Leader binning discussed in Hartigan's book [13], where each selective entry is a leader for its group.

Since there are 467 nodes and 60 time steps, we have totally 28,020 multivariate entries to perform clustering. Figure 6 shows eleven typical clusters for April 25, 2019, which have distinct morphology. The orientation of variables on the circular layout is displayed in Configuration 1 of Figure 4. The first (red) group in the second row is unreachable nodes (or undetermined status) at various timestamps which has **null** values on all dimensions. There are 354 members (or temporal multivariate statuses) in this group. The first cluster is the most popular group while the last cluster only has 4 members of high *memory usage*.

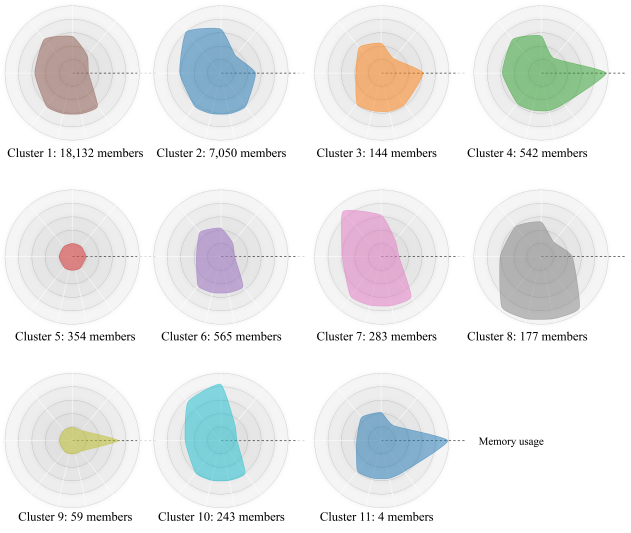

Fig. 6. The eleven clusters summarizing 28,020 multivariate entries of 60 time steps for the HPC center on April 25, 2019. Cluster sizes are displayed underneath each radar group.

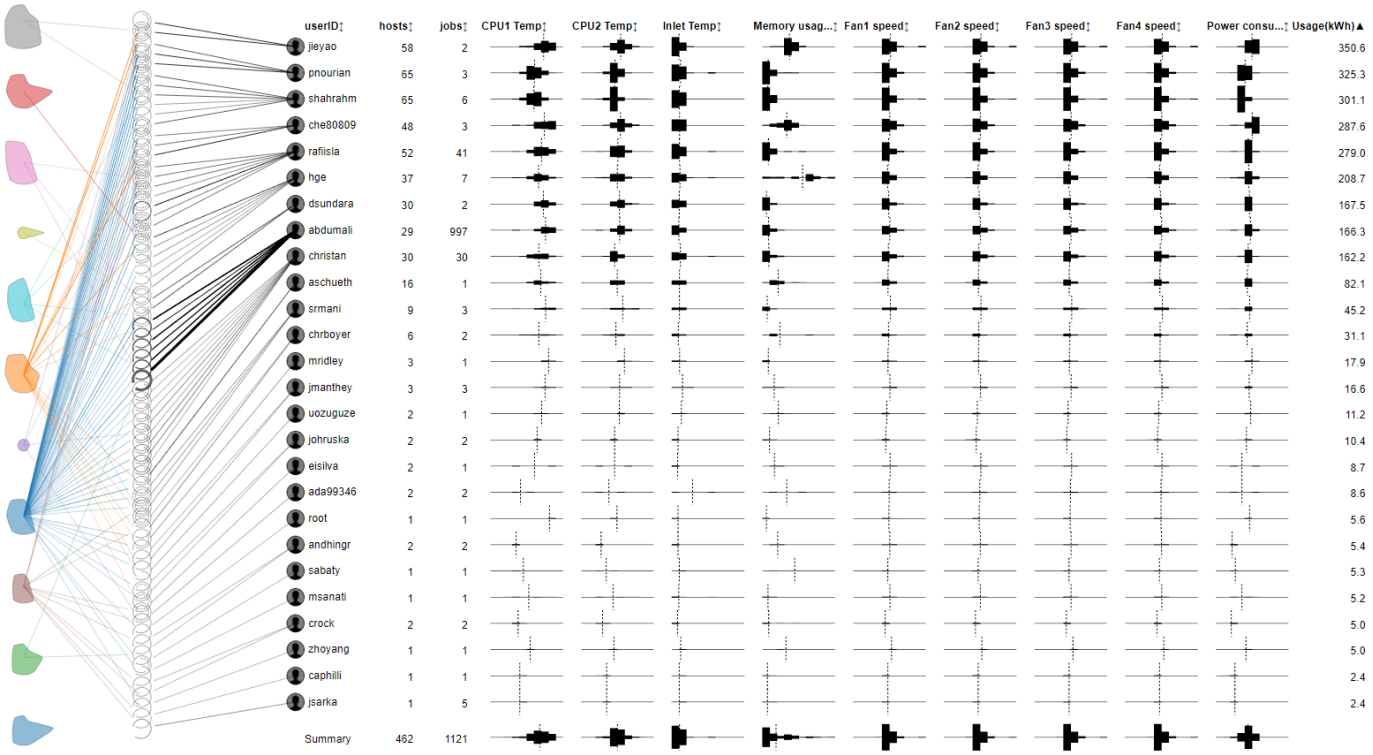

Fig. 7. Visualizing job scheduling and multivariate health status at the HPC center at a university in the last 5 hours of April 25, 2019. The eleven radar clusters are color-encoded and listed vertically on the left. The 26 HPC users are ordered by the total power usage on the right.

IV. Use Cases

Figure 7 shows a use case of our *HiperJobViz* for the HPC center at a university on April 25, 2019. From this overview, users can easily identify the users who use the most resources or consume the most electric power in the system. *HiperJobViz* supports a full range of interactive operations such as details in demands, filtering, and ordering. For example, by ordering the last column in Figure 7 descendingly, users can bring the users consuming the most power to the top of the list. The user **yieyao** consumes the most power on April 25, 2019. Similarly, by ordering memory usage, users can easily identify the jobs/users consuming the most memory of the HPC system. The symmetric histogram for each user on each dimension summarizes the variance of the associated readings overtime. This provides a visual summary for comparing the resource usage by different users. The radar charts on the left provide another abstraction level of the visual comparisons by compressing multiple dimension into a single chart.

User interactions: Users can request for more details by simply clicking on the summary histogram. Figure 8 shows examples when users explore this option. In particular, the top panel shows *CPU1 temperature* readings of 65 computing nodes by the user **shahrahm** from 7:00 pm to 11:55 pm on Thursday, April 25, 2019. The *CPU1 temperatures* are pretty stable except one sudden drop and pump around 11:30 pm of the *compute-8-17* (in red). The bottom panel shows *memory usages* of 35 computing nodes by the user **hge** during the same time interval. The *memory usage* readings vary a lot within a few hours. The two prominent examples are *compute-1-28* and *compute-1-30*. This might due to the allocated jobs for these computing nodes have finished and stopped running around 21:30. We can confirm this fact by linking them back to the scheduling charts in Figure 2.

Implementation: *HiperJobViz* is developed using JavaScript and in particular the D3.js library [5]. The online *HiperJobViz* prototype, source code, and more examples are available on our Github repository at https://git.io/Je3Qa.

V. Conclusion and Future Work

This paper presents a prototype *HiperJobViz* for visualizing and analyzing the multivariate dynamic behaviors and job scheduling information of HPC centers. The visualization has two main components for visualizing the hierarchical job data and multivariate health status of computing nodes using a radar layout. The input radar layout and data dimensions can be customized by the users, while the output representation can also be organized based on user interests for system overview and debugging purposes. The developed prototype and its interactive features are demonstrated on a medium-scale HPC center of 467 computing nodes at a university. Leader clustering allows *HiperJobViz* to scale with the number of data entries in a larger HPC centers. For future work, we will investigate the correlation or causal relationships between resource allocations/usages and job scheduling to maximize the performance of HPC centers.

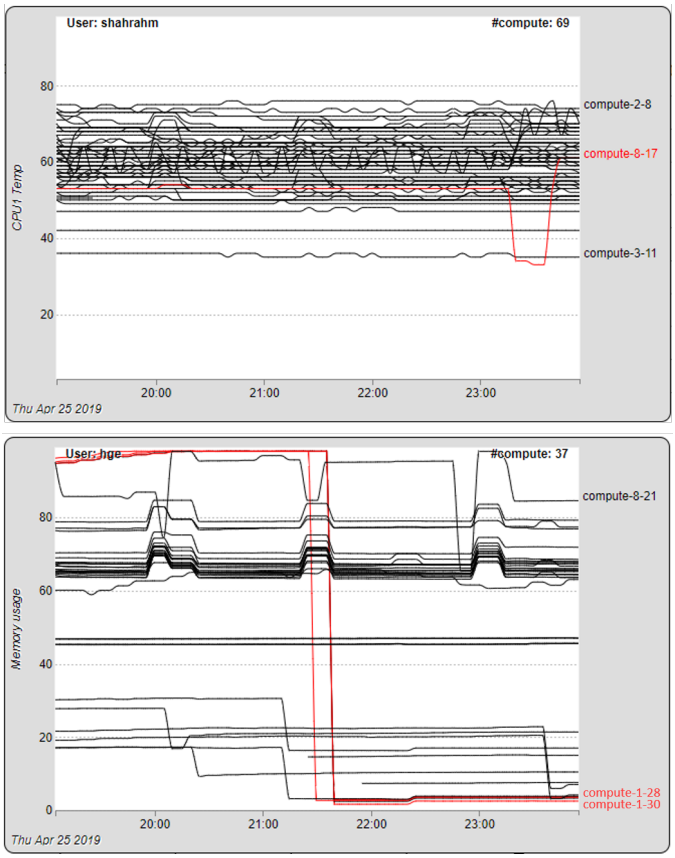

Fig. 8. Details on mouse clicks: (top) *CPU1 temperature* readings of 65 computing nodes by **shahrahm** and (bottom) *memory usages* of 35 computing nodes by **hge**. Each line chart represents a computing node.

REFERENCES

[1] 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*, pp. 15–24, 2005.

[2] N. Andrienko, G. Andrienko, and P. Gatalsky. Exploratory spatio-temporal visualization: an analytical review. *Journal of Visual Languages & Computing*, 14(6):503–541, 2003.

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

[4] F. Berman. High-performance schedulers. *The grid: blueprint for a new computing infrastructure*, 67:279–309, 1999.

[5] M. Bostock, V. Ogievetsky, and J. Heer. D3 data-driven documents. *IEEE Trans. Vis. Comput. Graph.*, 17(12):2301–2309, 2011.

[6] N. Capit, G. Da Costa, Y. Georgiou, G. Huard, C. Martin, G. Mounie, P. Neyron, and O. Richard. A batch scheduler with high level components. In *CCGrid 2005. IEEE International Symposium on Cluster Computing and the Grid, 2005.*, vol. 2, pp. 776–783 Vol. 2, May 2005. doi: 10.1109/CCGRID.2005.1558641

[7] D. Carasso. *Exploring splunk*. CITO Research New York, USA, 2012.

[8] T. Dang. Visualizing multidimensional health status of data centers. In A. Bhatele, D. Boehme, J. A. Levine, A. D. Malony, and M. Schulz, eds., *Programming and Performance Visualization Tools*, pp. 273–283. Springer International Publishing, Cham, 2019.

[9] T. N. Dang, N. Pendar, and A. G. Forbes. TimeArcs: Visualizing Fluctuations in Dynamic Networks. *Computer Graphics Forum*, 2016. doi: 10.1111/cgf.12882

[10] T. N. Dang and L. Wilkinson. Scagexplorer: Exploring scatterplots by their scagnostics. In *2014 IEEE Pacific Visualization Symposium*, pp. 73–80, March 2014. doi: 10.1109/PacificVis.2014.42

[11] M. I. Daoud and N. Kharma. A high performance algorithm for static task scheduling in heterogeneous distributed computing systems. *Journal of Parallel and Distributed Computing*, 68(4):399 – 409, 2008. doi: 10.1016/j.jpdc.2007.05.015

[12] Grafana. The open platform for beautiful analytics and monitoring, 2019. https://grafana.com/.

[13] J. Hartigan. *Clustering Algorithms*. John Wiley & Sons, New York, 1975.

[14] N. D. Hugh Greenberg. Tivan: A scalable data collection and analytics cluster. 2018. The 2nd Industry/University Joint International Workshop on Data Center Automation, Analytics, and Control (DAAC).

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

[16] C. Jiang, G. Han, J. Lin, G. Jia, W. Shi, and J. Wan. Characteristics of co-allocated online services and batch jobs in internet data centers: A case study from alibaba cloud. *IEEE Access*, 7:22495–22508, 2019.

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

[18] Q. Liu and Z. Yu. The elasticity and plasticity in semi-containerized co-locating cloud workload: A view from alibaba trace. In *Proceedings of the ACM Symposium on Cloud Computing*, pp. 347–360. ACM, 2018.

[19] C. Lu, K. Ye, G. Xu, C.-Z. Xu, and T. Bai. Imbalance in the cloud: An analysis on alibaba cluster trace. In *2017 IEEE International Conference on Big Data (Big Data)*, pp. 2884–2892. IEEE, 2017.

[20] O. Mämmelä, M. Majanen, R. Basmadjian, H. De Meer, A. Giesler, and W. Homberg. Energy-aware job scheduler for high-performance computing. *Computer Science - Research and Development*, 27(4):265–275, Nov 2012. doi: 10.1007/s00450-011-0189-6

[21] M. Meyer, T. Munzner, and H. Pfister. Mizbee: A multiscale synteny browser. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):897–904, Nov. 2009. doi: 10.1109/TVCG.2009.167

[22] G. Misra, S. Agrawal, N. Kurkure, S. Pawar, and K. Mathur. Chreme: A web based application execution tool for using hpc resources. In *International Conference on High Performance Computing*, pp. 12–14, 2011.

[23] N. Nguyen and T. Dang. Hiperviz: Interactive visualization of cpu temperatures in high performance computing centers. In *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (Learning)*, PEARC '19, pp. 129:1–129:4. ACM, New York, NY, USA, 2019. doi: 10.1145/3332186.3337959

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

[25] G. Palmas, M. Bachynskyi, A. Oulasvirta, H. P. Seidel, and T. Weinkauf. An edge-bundling layout for interactive parallel coordinates. In *2014 IEEE Pacific Visualization Symposium*, pp. 57–64. IEEE, 2014.

[26] A. Reuther, C. Byun, W. Arcand, D. Bestor, B. Bergeron, M. Hubbell, M. Jones, P. Michaleas, A. Prout, A. Rosa, et al. Scheduler technologies in support of high performance data analysis. In *2016 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–6. IEEE, 2016.

[27] M. J. Saary. Radar plots: a useful way for presenting multivariate health care data. *Journal of clinical epidemiology*, 61(4):311–317, 2008.

[28] H. Shen and C. Li. Zeno: A straggler diagnosis system for distributed computing using machine learning. In *International Conference on High Performance Computing*, pp. 144–162. Springer, 2018.

[29] 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*, VL '96, pp. 336–. IEEE Computer Society, Washington, DC, USA, 1996.

[30] T. S. Somasundaram and K. Govindarajan. Cloudrb: A framework for scheduling and managing high-performance computing (hpc) applications in science cloud. *Future Generation Computer Systems*, 34:47 – 65, 2014. Special Section: Distributed Solutions for Ubiquitous Computing and Ambient Intelligence. doi: 10.1016/j.future.2013.12.024

[31] R. L. Warrender, J. Tindle, and D. Nelson. Job scheduling in a high performance computing environment. In *2013 International Conference on High Performance Computing Simulation (HPCS)*, pp. 592–598, July 2013. doi: 10.1109/HPCSim.2013.6641474

[32] L. Wilkinson. Visualizing big data outliers through distributed aggregation. *IEEE transactions on visualization and computer graphics*, 2017.

[33] L. Wilkinson, A. Anand, and R. Grossman. High-dimensional visual analytics: Interactive exploration guided by pairwise views of point distributions. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1363–1372, 2006.