Advancing ODA standardization through an open source dashboard

Tim Osborne, Rachel Palumbo, Leah Huk, Ryan Adamson, Rob Jones, Corwin Lester
Oak Ridge National Laboratory, Oak Ridge, TN
Email: osbornetd@ornl.gov, palumborl@ornl.gov, hukln@ornl.gov, adamsonrm@ornl.gov, jonesjr@ornl.gov, lestercp@ornl.gov

***Abstract*—Supercomputers produce big data and operational data analytics (ODA) makes this information useful to users and administrators. ODA data, properly used, can impact energy use, operational costs, high performance computing (HPC) performance, and ultimately science. Each HPC site globally has to wrestle ODA into submission with each new system. As a new system is deployed, dashboards can be highly valuable to system administrators. If the telemetry data schema changes, dashboards must be recreated using different sources, query languages, and metric names. When dashboards need to be reworked, this keeps people from using them during this valuable time. In order to ease this burden across HPC sites, we have created a system dashboard to share between sites as a way to move toward a standard for telemetry data. This paper describes the dashboard and outlines how to use it at your site.**

***Index Terms*—Operational Data Analytics, HPC Post-Exascale Challenges, Monitoring, Telemetry, Data Analytics, Visual Analytics**

I. Introduction

HPC systems are capable of delivering more information than we as systems administrators typically ask for. Are we listening to what our systems are trying to tell us? A number of challenging problems emerge when we first start to decipher the language of telemetry; we must first identify important pieces and then transform the data, normalize the vendor-specific naming conventions and units of measure, and finally monitor data quality to prevent missing and incorrect data. It's in this early phase that most of us spend our time before useful observations can be made. The right design, implemented early on, can greatly speed up the implementation of future methods of analysis.

The major contributions of this paper are threefold: We first present a view of an Oak Ridge Leadership Computing Facility (OLCF) comprehensive system status dashboard. The second contribution is a generic representation of this dashboard that we are sharing for others to adapt to use at their facilities. Third, we envision this dashboard to be a target for vendors to design to as they develop new hardware and software, exposing telemetry from those system components. We hope that a de-facto standard will emerge in the coming years that will lower the barriers of entry to truly understanding what our systems are trying to tell us.

II. Background

During deployment of the first exascale HPC system, there were delays in providing visual ODA tools due to the need to obtain *a priori* understanding of the produced telemetry data. Differences in data format, density, and component granularity between vendors and solutions made it difficult to adapt reporting from earlier systems in a way that provided relevant comparisons. This led to a need for standardization across our own systems. OLCF now pushes all of its data through a Kafka instance into an Elasticsearch data lake [1]. At the same time, there was a growing understanding of the demand for a consistent framework for telemetry and other data feeds [2]. On the operational end, the earliest priorities expressed by system personnel and users revolved around temperature management, power attribution at the job level, and a synthesis of node and job level event information allowing diagnosis of issues caused by hardware across allocated jobs. This called for dashboards providing diagnostic and descriptive information across the pillars of building infrastructure, system hardware, system software, and applications as they apply to a job-oriented HPC model [3]. Since these issues are by no means unique to the system in question, we wished to establish the basis for a generic design-to target for HPC vendors similar to benchmark standardization.

III. Motivation

As Frontier [4] moved from acceptance to production, we created a system status dashboard for system administrators. Other HPC sites create similar dashboards for their sites. Our goal is to establish a standard dashboard for telemetry data that adequately represents HPC systems and jobs in a meaningful way. We would reduce the effort required for each site to understand and use this data effectively, for both system administrators and HPC users. Real time monitoring allows system administrators to manage systems and applications on the fly and mitigate inefficiencies or address problems that could escalate into systematic issues. HPC users can access real-time job performance metrics such as power and energy consumption [5]. This gives them the power and knowledge to make adjustments to their jobs if energy use is unusual or higher than necessary, helping reduce unnecessary energy and resource usage. Secondly, by creating a generic representation of this dashboard, we will be able to make it open source for HPC sites and the ODA community to use and address shared challenges. We want to reduce duplicated efforts and wasted resources in establishing monitoring. Lastly, it can serve as a resource for vendors as they develop telemetry solutions for future systems. We aim to help combat delays and inefficiencies during acceptance due to insufficient, incomplete, or nonexistent telemetry data.

IV. Visualization

A. Input Data

The dashboard is driven by three data sources: two custom slurm scrapers and High Performance Cluster Manager (HPCM) [6]. Slurm is a workload manager that is widely used by HPC systems and computing clusters across the globe; the slurm scrapers provide job related metrics. HPCM is an integrated system management solution built by Hewlett Packard Enterprise (HPE) that provides provisioning, management, and monitoring for HPC systems; we use Kafka Connect to pull telemetry data from it. As HPCM data flows through Kafka, we add the hostname of each node into that data to make merging with Slurm data easier later on.

OLCF's ODA data flows through our Kafka platform, STREAM [1]. Logstash transfers data from Kafka to Elasticsearch. Finally, Grafana reads the data from Elasticsearch to populate the dashboard.

B. Dashboard Layout and Description

There is a minimum intersection of various sets of data that are useful to users, operators, and administrators of HPC systems. Regardless of vendor or application specifics, information such as power consumption, telemetry measurements of system components, network congestion, system status, and workload scheduling are common to all HPC resources and generally architecture agnostic. The ubiquitous nature of these categories of data form the basis for our choices in design and content of the standard dashboard.

1) Node Info: Nodes are the smallest basic unit of allocatable resources in this HPC system and scheduler configuration. At the OLCF, all user jobs are defined by the number of nodes assigned, and individual resources on a node can then be further divided and assigned depending on the number of tasks planned for each. Status of a node is of particular interest to operators and administrators. The status of a node informs the health of the overall system, determining how often and whether a node requires replacement or maintenance. Telemetry metrics gathered by a node inform users and administrators about code performance and efficiency, and whether it is utilizing resources in an optimized manner.

The top segment of the node section displays the selected node's hostname and corresponding "xname" denoting its geographic location among the cabinets on the server floor, its status flags, the job number if it is currently allocated, which partitions the node is available in, and the date and time of its last reboot. The bottom segment of the node section displays several telemetry metrics of the selected node: node power consumption, CPU (Central Processing Unit) temperature, and GPU (Graphical Processing Unit) temperatures. Each metric is plotted as a time series over the selected time range of the dashboard, and to the right of each as a single most current value within a general maxima and minima of expected values

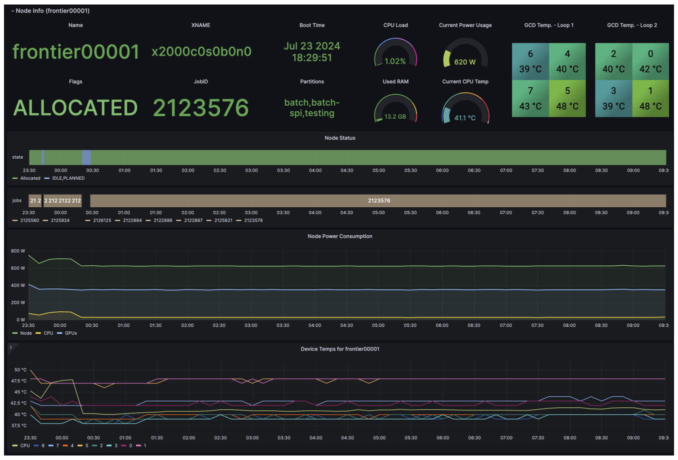

Node Info (frontier00001)
Name: frontier00001
XNAME: x2000c0s0b0n0
Boot Time: Jul 23 2024 18:29:51
Flags: ALLOCATED
JobID: 2123576
Partitions: batch,batch-spi,testing
CPU Load: 1.02%
Current Power Usage: 620 W
Used RAM: 13.2 GB
Current CPU Temp: 41.1 °C
GCD Temp. - Loop 1: 6 39 °C; 4 40 °C; 7 43 °C; 5 48 °C
GCD Temp. - Loop 2: 2 40 °C; 0 42 °C; 3 39 °C; 1 48 °C

Fig. 1. The layout of the Node Info section of the dashboard

for the metric. Load induced patterns incurred on a node by individual jobs is evident in the timeseries.

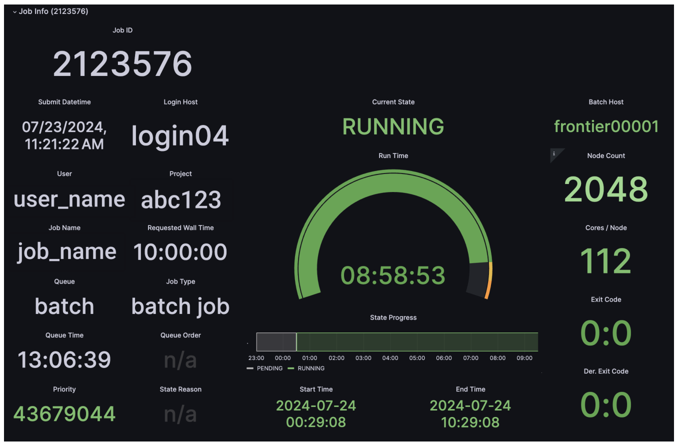

Job Info (2123576)
Job ID: 2123576
Submit Datetime: 07/23/2024, 11:21:22 AM
Login Host: login04
User: user_name
Project: abc123
Job Name: job_name
Requested Wall Time: 10:00:00
Queue: batch
Job Type: batch job
Queue Time: 13:06:39
Queue Order: n/a
Priority: 43679044
State Reason: n/a
Current State: RUNNING
Run Time: 08:58:53
Start Time: 2024-07-24 00:29:08
End Time: 2024-07-24 10:29:08
Batch Host: frontier00001
Node Count: 2048
Cores / Node: 112
Exit Code: 0:0
Der. Exit Code: 0:0

Fig. 2. The layout of the Job Info section of the dashboard.

2) Job Info: Users are frequently interested in monitoring the status of the jobs they've submitted to the system, and this information is often also the first stop for administrators to diagnose issues on behalf of users reaching out for assistance with problematic jobs. Job information generally falls into one of two kinds: relatively static accounting information originating from the submission of the user's batch script, and job status information that is subject to change over time.

The values in white in the top left display accounting information as set in the user's batch script, and include: the job identification number, the user's username, the allocation project charged for node hours, the job type (batch submission, or interactive), the queue name (in our case, "batch" is the normal queue, vs "debug" which is reserved for smaller testing jobs), the job name, the login node from which the job was submitted, the requested walltime, the date and time job was submitted to the scheduler, and finally the elapsed time the job has spent waiting in the queue. While the "queue time" is not static, we include it in this section as it is determined by the submission time and does not change after the job begins. We include metrics associated with queued jobs in ‘PENDING‘ status in this area for the same reason, but they are displayed in green as they are not dependent solely on submission time. These include the state reason, priority level, queue order, and the elapsed times over which the job is eligible to run and accruing priority.

The center of the job info section contains information about the state of the job, specifically job status. If the job has initialized, this includes start time, predicted end time, current elapsed time, state progress bar, and the head batch node from which the job was launched by the scheduler. To the right are the total number of nodes and number of cores per node requested, and the current value of the exit code, which is updated upon job completion.

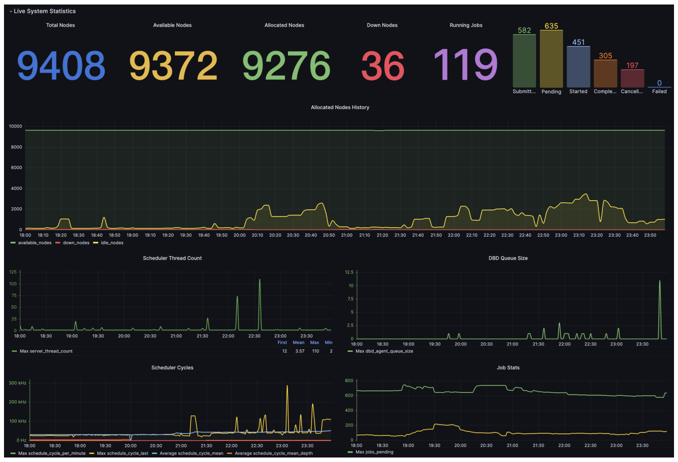


Fig. 3. The layout of the System Stats section of the dashboard

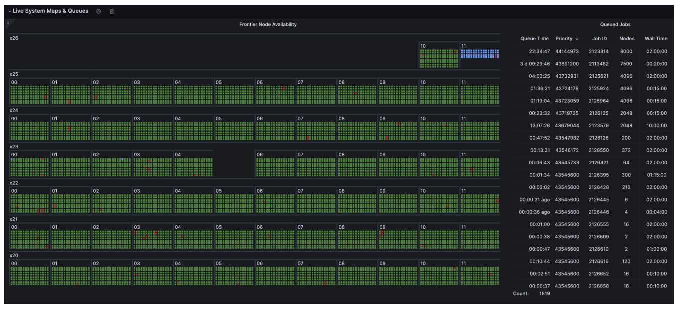
3) Live System Stats: This section of the dashboard contains aggregate statistics about the total number of jobs and nodes in the system, and recent trends in scheduler workload. A holistic overview of the current state of the system is necessary for operators and system administrators to monitor usage, identify issues with hardware, and respond quickly to adverse events with minimal latency.

The primary statistics displayed are the number of nodes in the following categories: total, available, allocated, and down; total currently running jobs is also included. Additionally, a histogram summarizes jobs that are not in a running state: total submitted, completed, cancelled, or failed in the active time range, and jobs currently pending.

Quantities that characterize scheduler behavior are plotted as time series over the selected time range. These include scheduler thread counts and cycles, pending vs running jobs, and backfill depth and cycles.

4) Live System Map: Having a visual companion to the Live System Stats in the form of a choropleth map is helpful for identifying large scale patterns regarding node states, node failures, and scheduler behavior including job placement. We use the HPE clusterview Grafana plugin for this [7]. The maps are a pseudo-accurate representation of the locations of cabinets/nodes on the server room floor: each colored square represents an individual node and is grouped by cabinet location and placement within the cabinet. The locations aren’t perfectly accurate in order to allow the viewer to see the entire map at once on a landscape monitor.

The primary panel in this section displays node status via color. Green nodes are ”allocated”, blue are ”idle”, and red are ”down”. The clusterview visualization is accompanied by a table of the currently queued jobs with information about their priority, queue time, jobid, number of requested resources, and requested wall time.



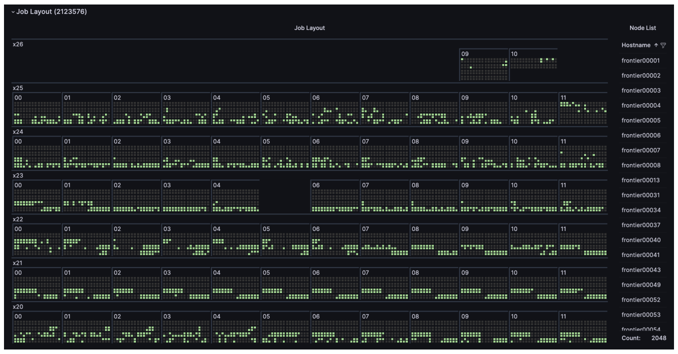
Queued Jobs

Queue Time	Priority ↓	Job ID	Nodes	Wall Time
22:34:47	44144973	2123314	8000	02:00:00
3 d 09:29:46	43891200	2113482	7500	00:20:00
04:03:25	43732931	2125621	4096	02:00:00
01:36:21	43724179	2125924	4096	00:15:00
01:19:04	43723059	2125964	4096	00:15:00
00:23:32	43719725	2126125	2048	00:15:00
13:07:26	43679044	2123576	2048	10:00:00
00:47:52	43547982	2126126	200	02:00:00
00:13:31	43546172	2126550	372	02:00:00
00:06:43	43545733	2126421	64	02:00:00
00:01:34	43545600	2126395	300	01:15:00
00:02:02	43545600	2126428	216	02:00:00
00:00:31 ago	43545600	2126445	6	02:00:00
00:00:36 ago	43545600	2126446	4	00:04:00
00:01:00	43545600	2126555	16	02:00:00
00:00:38	43545600	2126609	2	02:00:00
00:00:47	43545600	2126610	2	01:00:00
00:10:44	43545600	2126616	120	02:00:00
00:02:51	43545600	2126652	16	00:10:00
00:00:37	43545600	2126658	16	00:10:00

Count: 1519

Fig. 4. The layout of the System Map section of the dashboard

5) Job Layout: Similar to the Live System Map, we present a cluster visualization of the selected job from in the Job Info section of the dashboard. We use the same clusterview map as used previously where the nodes highlighted correspond to the selected job. The accompanying table lists the assigned nodes by hostname. Node metrics and their associated color scales can be toggled on and off within the table; these include current CPU and GPU temperatures, power consumption, and CPU load.



Node List

Hostname ↑
frontier00001
frontier00002
frontier00003
frontier00004
frontier00005
frontier00006
frontier00007
frontier00008
frontier00013
frontier00031
frontier00034
frontier00037
frontier00040
frontier00041
frontier00043
frontier00049
frontier00052
frontier00053
frontier00054

Count: 2048

Fig. 5. The layout of the Job Layout section of the dashboard

V. Open Source Effort

The dashboard we described would be useful to any HPC site and gives users and system administrators insight into job performance and energy use, leading to both scientific and environmental impact. Open source software projects have consistently shown that community input improves the final product. Additionally, since users switch between HPC sites, giving them a common job-centric view amongst different

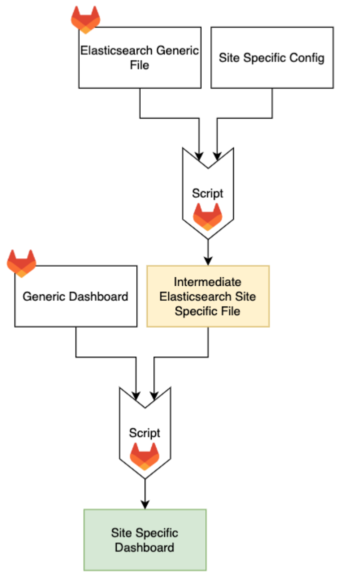


Fig. 6. This graphic represents the workflow for building a site-specific dashboard. Steps indicated with the gitlab symbol indicate provided opensource files or scripts.

hardware can enable better understanding of job performance and energy usage between sites. Since the backend of a Grafana dashboard is a json object, sharing across sites seems trivial at first glance. However, there are two hurdles that must be overcome: different implementations of data sources for telemetry data storage and vendor specific naming conventions.

A. Method

To make the dashboard portable, we've developed scripts that read the dashboard JSON model into a Python dictionary which can then be used for removing, transforming, and adding back in normalized site-specific information in a standardized way. Our repository contains the necessary script, a generic dashboard model, and a generic elasticsearch data source configuration. The generic model contains all visualization related configuration including layout, color schemes, etc. The generic elasticsearch data source config file contains the necessary tags for reading from an elasticsearch datasource. The script combines these files with a site-specific config file to insert your telemetry naming conventions into a site-specific dashboard for you to use (see figure 6).

B. Files

1) Script: The script is simple. To create a site-specific file, it reads through a configuration file and a generic file and replaces values in the generic file that match nested keys in the configuration file. There is one special key, ID_KEY that identifies what nested-key to match on when the object is part of a list (instead of a directly nested dictionary).

2) Site Specific Config: This file is directly related to your site's configuration. Each site will create and maintain their own site specific config file outside of the main repository. Everything inside it is specific to your site. For example:

```
"templating": {
  "list": {
     "node_info":{
       "current":{
         "text":"frontier.node-info",
         "value":"frontier.node-info"
       },
       "regex": "/^frontier\\.node-info$/"
     }
  }
}
```

3) Elasticsearch Generic File: This file contains a nested dictionary of elasticsearch data source specific configurations for the dashboard. This file is maintained in the repository for everyone. Everything inside it is generic. For example:

```
"templating": {
  "list": {
    "ID_KEY": "name",
    "node_info": {
      "query": "elasticsearch",
    },
  }
}
```

4) Elasticsearch Site Specific File: This is an intermediate file produced by the script that does not need to live in any repository. It is the combination of nested-key-value pairs in the site-specific config and the Elasticsearch generic config files. For Example:

```
"templating": {
  "list": {
    "ID_KEY": "name",
    "node_info": {
      "query": "elasticsearch",
      "current":{
        "text":"frontier.node-info",
        "value":"frontier.node-info"
      },
      "regex": "/^frontier\\.node-info$/"
    },
  }
}
```

5) Generic dashboard: Finally, the Generic dashboard contains the full json model without anything datasource or site-specific. For example:

```
"templating": {
 "list": [
  {
    "current": {
      "selected": false
    },
    "hide": 2,
    "includeAll": false,
    "multi": false,
    "name": "node_info",
    "options": [],
    "refresh": 1,
    "skipUrlSync": false,
    "type": "datasource"
  }
 ]
}
```

6) Site Specific Dashboard: When the Generic dashboard is combined with the Intermediate Elasticsearch Site Specific File, it creates the final usable dashboard which can be uploaded to your Grafana server and used. For example, combining the above files gives you (Note, this is not a full dashboard, please see repository):

```
"templating": {
 "list": [
  {
    "current": {
      "selected": false,
      "text": "frontier.node-info",
      "value": "frontier.node-info"
    },
    "hide": 2,
    "includeAll": false,
    "multi": false,
    "name": "node_info",
    "options": [],
    "query": "elasticsearch",
    "refresh": 1,
    "regex": "/^frontier\\.node-info$/",
    "skipUrlSync": false,
    "type": "datasource"
  }
 ]
}
```

C. How to Use

Currently our dashboard and files are shared on https://code.ornl.gov/oda-dashboards/oda-shared-dashboards.git. If you'd like to collaborate, please reach out for access. Eventually we plan to move to Oak Ridge National Lab's github after more work has been done. If you have Elasticsearch data sources, using our dashboard is straightforward:

1) Create your own site config file
2) Call the generator script with the elasticsearch starter and your site's file

That will create a new dashboard that you can upload into Grafana. If you do not use Elasticsearch, we will need to collaborate to create the right queries for your database backend. We welcome this work to make the dashboard more accessible to HPC sites.

VI. Evaluation

Even though the repository is Elasticsearch focused at the moment, the script's capabilities and the dashboard's usefulness make the project worth investing in. This work eases the burden of creating monitoring dashboards from scratch and promotes a shared way to monitor HPC systems. The more this is used across HPC sites, the better the dashboard will be for everyone as they add their own insights and expand the capabilities. Additionally, it will increase incentive for vendors to document monitoring metadata and standardize schemas as much as possible. This work is open source and uses accessible libraries and languages so that it can be easily utilized across different sites. We used the Python programming language which includes established Grafana Python client libraries. HPC operators, administrators, and users; ODA researchers; vendors; and energy/power efficiency researchers will benefit greatly from the collaboration that this work facilitates.

VII. Future Work

Future work will be focused in four main areas: schema, testing, usability, and community contribution. We will work to expand a standardized HPC telemetry schema to one that is comprehensive, valuable to multiple HPC sites, and supported by vendors. Continued collaboration from vendors is important, but HPC facilities will still likely need to have methods in place to map telemetry data to a standard format on their own. We will also continue to test this approach with a wider range of HPC sites and backend databases, adding additional datasource support and thus expanding the applicability of this dashboard. Additionally, we will continue to improve the user experience: adding API (Application Programming Interface) access to more easily create and update dashboards with minimal code interaction. By Open sourcing the project we hope to impact the ODA and HPC community and improve the code base.

VIII. Conclusion

In this paper we shared an ODA Grafana dashboard and described an approach to share it across sites by manipulating the underlying json model with a Python script. In the exascale era, ODA data has shown its value for systems analysis and power reduction. Our open source dashboard for users and system administrators, shared across sites, could lead to a generic ODA framework, better user experiences across sites, and, ultimately, better science.

IX. Acknowledgements

Michael Ott and the Energy Efficiency HPC Working Group ODA team

References

[1] R. Adamson, T. Osborne, C. Lester, and R. Palumbo, "Stream: A scalable federated hpc telemetry platform," 5 2023. [Online]. Available: https://www.osti.gov/biblio/1995656

[2] M. Ott, W. Shin, N. Bourassa, T. Wilde, S. Ceballos, M. Romanus, and N. Bates, "Global experiences with hpc operational data measurement, collection and analysis," in *2020 IEEE International Conference on Cluster Computing (CLUSTER)*, 2020, pp. 499–508.

[3] A. Netti, W. Shin, M. Ott, T. Wilde, and N. Bates, "A conceptual framework for hpc operational data analytics," in *2021 IEEE International Conference on Cluster Computing (CLUSTER)*, 2021, pp. 596–603.

[4] S. Atchley, C. Zimmer, J. Lange, D. Bernholdt, V. Melesse Vergara, T. Beck, M. Brim, R. Budiardja, S. Chandrasekaran, M. Eisenbach, T. Evans, M. Ezell, N. Frontiere, A. Georgiadou, J. Glenski, P. Grete, S. Hamilton, J. Holmen, A. Huebl, D. Jacobson, W. Joubert, K. Mcmahon, E. Merzari, S. Moore, A. Myers, S. Nichols, S. Oral, T. Papatheodore, D. Perez, D. M. Rogers, E. Schneider, J.-L. Vay, and P. K. Yeung, "Frontier: Exploring exascale," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3581784.3607089

[5] W. Shin, V. Oles, A. M. Karimi, J. A. Ellis, and F. Wang, "Revealing power, energy and thermal dynamics of a 200pf pre-exascale supercomputer," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3458817.3476188

[6] H. P. Enterprise, "Quickspecs hpe performance cluster manager," Hewlett Packard Enterprise, Tech. Rep., 2024.

[7] B. Holldorf, "Hpe clusterview grafana plugin," https://github.com/HewlettPackard/hpe-grafana-clusterview-panel, 2024.