A Modular, Responsive, and Accessible HPC Dashboard Built upon Open OnDemand

Richie Tan
Rosen Center for Advanced Computing
Purdue University
West Lafayette, Indiana, USA
tan447@purdue.edu

Guangzhen Jin
Rosen Center for Advanced Computing
Purdue University
West Lafayette, Indiana, USA
jin456@purdue.edu

Abstract

High-performance computing (HPC) has become a critical enabler of scientific advancement across many research domains. As the HPC user community continues to expand, there is an increasing need to reduce barriers to entry and improve accessibility for researchers with varying levels of computational expertise. This paper presents a modular, responsive, and user-friendly web-based dashboard built upon the Open OnDemand framework. The system integrates several redesigned functional pages, providing streamlined access to announcements, system status, job information, and resource usage. By consolidating these capabilities into an intuitive interface with a modern design, the dashboard enhances user experience, reduces reliance on command-line tools, and facilitates more efficient interaction with HPC resources.

CCS Concepts

• **Human-centered computing** → **Web-based interaction**; • **Software and its engineering** → **3-tier architectures**.

Keywords

Open OnDemand, dashboard, Slurm, HPC, Ruby on Rails

ACM Reference Format:
Richie Tan and Guangzhen Jin. 2025. A Modular, Responsive, and Accessible HPC Dashboard Built upon Open OnDemand. In *Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC Workshops '25), November 16–21, 2025, St Louis, MO, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3731599.3767434

1 Introduction

1.1 Background

High-performance computing (HPC) has many applications throughout the research landscape, accelerating discoveries across every field. Despite these capabilities, HPC-enabled research presents notable barriers to entry, including the ability to interact with a terminal interface, set up a secure network connection to an HPC cluster, and submit batch jobs, which can be non-trivial for researchers whose primary expertise lies outside computational science. To alleviate some of these barriers, some web-based platforms and tools have been developed. XD Metrics on Demand (XDMoD) [8] can provide system-wide performance metrics, usage reports, and user-level analytics. While powerful, its focus is more on performance analysis and administrative reporting rather than live user interaction with the HPC system. SUPReMM [1] further provides an extension of XDMoD that supports per-user and per-job performance monitoring and integrates deeper job-level metrics. Some HPC centers have implemented custom dashboards to display resource usage or allocation summaries, such as the Texas Advanced Computing Center's (TACC) *TACC User Portal* [2].

Among these web-based tools, Open OnDemand [5] has been the most popular and frequently used HPC dashboard solution. Open OnDemand allows users to seamlessly run common scientific applications, including Jupyter Notebook, RStudio, VS Code, and many others, on a cluster within a web portal, eliminating the need for extensive command-line interaction. While this adequately addresses the barrier of entry to running certain applications on a supercomputer, several critical user activities—such as monitoring job status, tracking allocation usage, and analyzing job performance—still often require command-line expertise.

At the Rosen Center for Advanced Computing (RCAC) at Purdue University, we extended the Open OnDemand framework to incorporate these additional capabilities directly into the web interface. By integrating job monitoring, allocation tracking, and performance insights into a unified, browser-accessible dashboard, our enhancements further reduce barriers to entry and improve the accessibility of system information, thereby empowering a broader range of researchers to fully leverage HPC resources.

1.2 Motivation and Objectives

Although Open OnDemand offers several highly useful features for cluster users, it lacks some features that would make the web interface a comprehensive solution for interacting with cluster resources. In particular, Open OnDemand does not show information about partitions, allocations, and job history from Slurm, so users still resort to terminal access with specific terminal commands frequently in order to access that important information. Additionally, there are several pages on Open OnDemand, including the homepage, that are not very informative to the user.

As the high-performance computing community grows along with the rapidly increasing need for large computing power to conduct modern research, the number of users who use clusters, as well as the number of users with little experience using command-line interface, is also growing. To address this challenge, a simple, all-inclusive dashboard is essential to make it easy for any user, regardless of computing background, to start using HPC resources for their research.

The main objective of this project is to create a one-stop dashboard that provides HPC system information in an accessible manner that does not require advanced HPC knowledge. There are three major goals: 1) redesigning the dashboard with additional user-friendly information; 2) improving the existing Open OnDemand functional pages; and 3) sharing resource usage and job efficiency data with users and groups.

The paper is organized as follows: section 2 gives an overview of the entire system. From section 3 to section 8, details for each major component in the system are provided along with their underlying techniques. Section 9 concludes the paper and discusses future work.

2 System Overview

Open OnDemand has been serving as the main web user portal for HPC resources at RCAC, so this work will be carried out based on it. We deprecated many of the original Open OnDemand dashboard pages and redesigned them with more informative and user-friendly components. The design flowchart is shown in Figure 1.

2.1 Open OnDemand

Open OnDemand is an open-source HPC web portal developed by the Ohio Supercomputing Center and funded by the National Science Foundation. It provides a simple interface for users to access cluster resources and run common applications with just a few clicks. It is used all across the globe in over 2100 locations [7], including hundreds of HPC centers. One of its main features is interactive apps, which allow users to run popular HPC applications, including Jupyter Notebook, RStudio, and MATLAB on clusters by simply submitting a form on the Open OnDemand dashboard. The user is then able to access and interact with the application through their web browser. However, Open OnDemand is still limited in how much can be done through the web interface, requiring users to still use the terminal for more advanced tools and detailed information about their jobs and allocations. Our improvements to the Open OnDemand dashboard on HPC clusters at our HPC center allow users to view various information about jobs, allocations, partitions, nodes, and other cluster details in a modern, user-friendly interface.

2.2 System Architecture

2.2.1 Frontend. The frontend is written using standard HTML, CSS, and JavaScript, using Embedded Ruby (ERB) templates to pre-render certain pieces of server-side data, such as the username. The CSS framework used is Bootstrap 4, a popular CSS framework with commonly-used components for building and designing websites. The design of the improved dashboard focuses on clearly conveying the most important information to most users in a simple, accessible manner. To reduce the loading times of certain components that fetch data from the server and the Slurm daemons, client-side caching is implemented using IndexedDB—a fast structured data store—and server-side caching is implemented with Ruby on Rails in-memory caching mechanisms.

2.2.2 Backend. The backend is written in Ruby using the Ruby on Rails web framework, which is the same as what Open OnDemand uses. The majority of backend routes are API routes, meaning their responses are in JavaScript Object Notation (JSON) and are fetched via the JavaScript frontend code. Since all HPC clusters at RCAC use Slurm [6] for job scheduling and job accounting, most of the backend routes run Slurm commands to gather job details, allocation information, and system statuses directly from Slurm's database, with only a couple of routes calling other helper scripts and commands. In order to reduce Slurm daemon traffic, Slurm and other helper script query results are cached with various expiration times ranging from a few seconds to a few hours, providing near real-time data with quicker responses and more efficient Slurm usage on average. More details will be given in the next section.

2.3 Code Structure

Each dashboard feature consists of a frontend ERB template file paired with one or more backend API routes. This structure—in contrast to providing the Slurm data upfront through the ERB template—allows components in the frontend to refresh their data from the server without requiring a full refresh of the page. Also, it allows the dashboard to load instantly and display a loading animation if the data requires some time to load. Otherwise, waiting for the data to load before showing any part of the dashboard would leave users wondering whether the dashboard is offline or broken. This approach provides a more modern, seamless experience and reduces navigational frustration.

Because most components are fully contained within their respective template and API route files, they can be moved to other Open OnDemand dashboards in isolation by simply copying the template and API route files and setting up a route in the Ruby on Rails routing configuration to link to the component. Both full pages (such as the My Jobs and Cluster Status apps) and homepage widgets share this common structure, so most components of the dashboard are easily transferable to other dashboards. However, it is recommended to include the IndexedDB cache logic file in order to retain the client-side caching features.

2.4 Design Considerations

Modularity. An important design aspect of the new dashboard is the modularity of the system, both on the backend and the frontend. We wanted the dashboard to be modular in order to allow developers and other HPC centers to easily migrate this system or some subset of features to their own Open OnDemand dashboard. Also, the modular design ensures that if one widget or component stops working, it does not break the entire dashboard, reducing major user experience issues. In order to make the system modular, the backend is designed so that each component in the frontend is paired with one API route in the backend. This ensures that components can be easily moved and modified as isolated parts rather than a bunch of interlinked components and API routes.

Performance. One of the key improvements of the new dashboard is the focus on speed and scalability. Because there could be many users using Slurm and the Open OnDemand dashboard simultaneously, we designed the dashboard to frequently cache data from Slurm queries and other data sources in order to speed up responses on the frontend. In order to balance quick response times with up-to-date information, we selected different cache expiration times for each data source depending on the use case so that stale

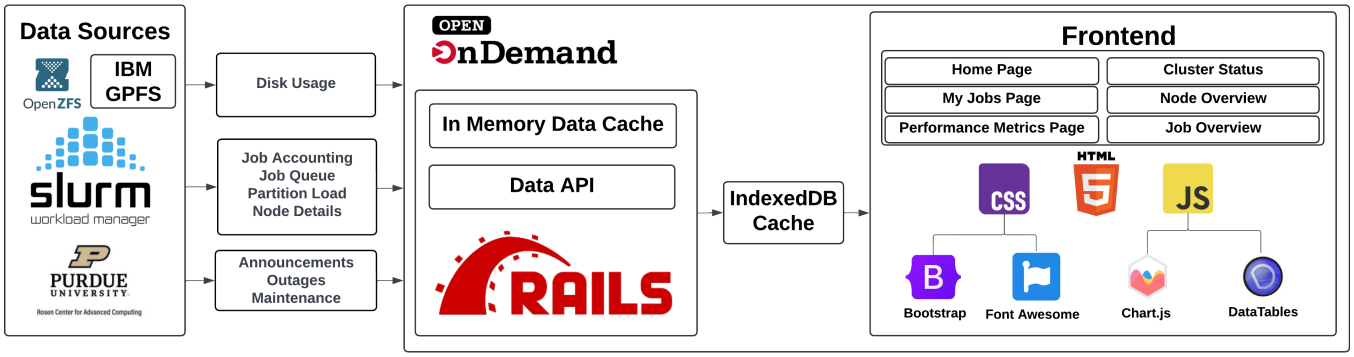
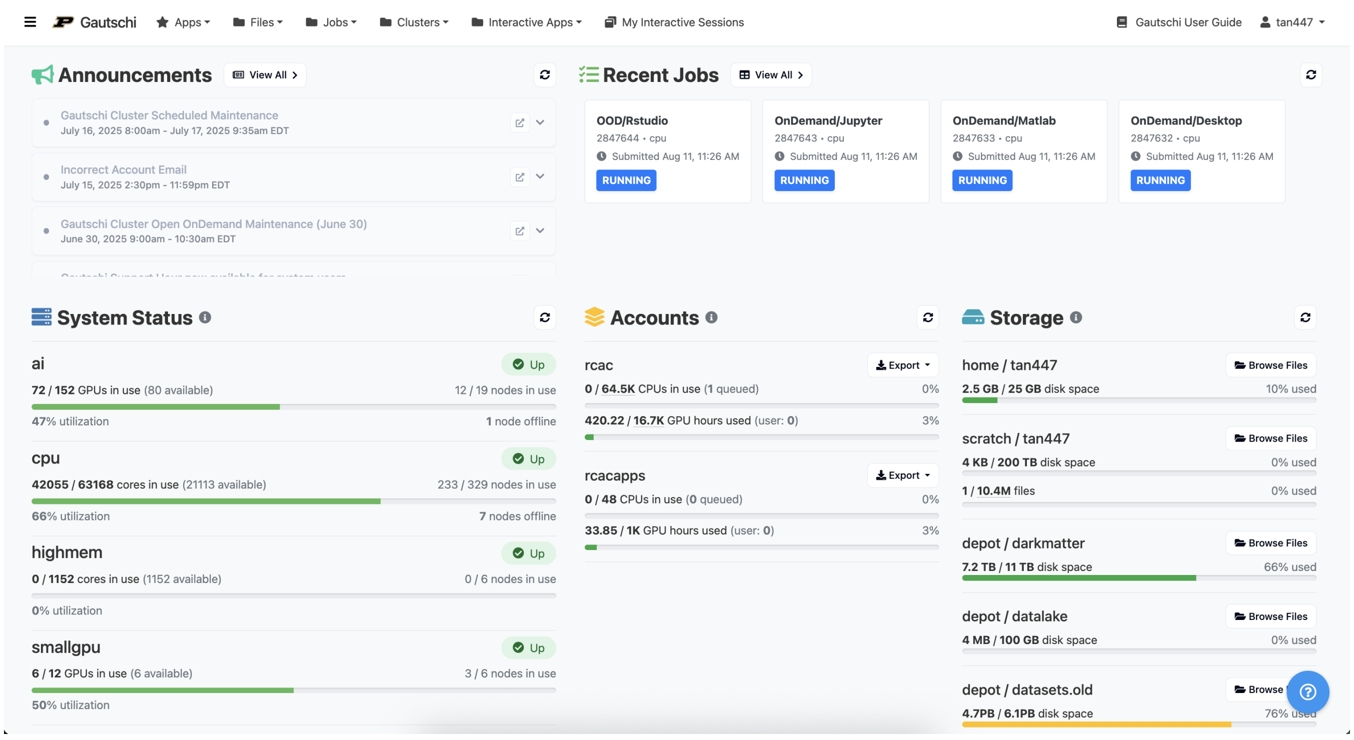


Figure 1: System architecture and data flow. Logos are trademarks of their respective owners; see Section 10.



Announcements View All

Gautschi Cluster Scheduled Maintenance
July 16, 2025 8:00am - July 17, 2025 9:35am EDT

Incorrect Account Email
July 15, 2025 2:30pm - 11:59pm EDT

Gautschi Cluster Open OnDemand Maintenance (June 30)
June 30, 2025 9:00am - 10:30am EDT

Recent Jobs View All

OOD/Rstudio
2847644 • cpu
Submitted Aug 11, 11:26 AM
RUNNING

OnDemand/Jupyter
2847643 • cpu
Submitted Aug 11, 11:26 AM
RUNNING

OnDemand/Matlab
2847633 • cpu
Submitted Aug 11, 11:26 AM
RUNNING

OnDemand/Desktop
2847632 • cpu
Submitted Aug 11, 11:26 AM
RUNNING

System Status

ai Up
72 / 152 GPUs in use (80 available) 12 / 19 nodes in use
47% utilization 1 node offline

cpu Up
42055 / 63168 cores in use (21113 available) 233 / 329 nodes in use
66% utilization 7 nodes offline

highmem Up
0 / 1152 cores in use (1152 available) 0 / 6 nodes in use
0% utilization

smallgpu Up
6 / 12 GPUs in use (6 available) 3 / 6 nodes in use
50% utilization

Accounts

rcac Export
0 / 64.5K CPUs in use (1 queued) 0%
420.22 / 16.7K GPU hours used (user: 0) 3%

rcacapps Export
0 / 48 CPUs in use (0 queued) 0%
33.85 / 1K GPU hours used (user: 0) 3%

Storage

home / tan447 Browse Files
2.5 GB / 25 GB disk space 10% used

scratch / tan447 Browse Files
4 KB / 200 TB disk space 0% used
1 / 10.4M files 0% used

depot / darkmatter Browse Files
7.2 TB / 11 TB disk space 66% used

depot / datalake Browse Files
4 MB / 100 GB disk space 0% used

depot / datasets.old Browse
4.7PB / 6.1PB disk space 76% used

Figure 2: Homepage of the dashboard.

information is not cached for too long. For example, cluster announcements and maintenance alerts are not being updated every few seconds, so we are able to cache the articles in the announcements widget for 30 minutes to an hour without worrying about the displayed articles being too outdated. On the other hand, the recent jobs widget queries squeue, which changes quite frequently and users likely want to see their new jobs without waiting for a long time, so we set the cache expiration time to around 30 seconds. This way, users can quickly see any updates to their recent jobs without overloading the Slurm `slurmctld` daemon if they decide to refresh the page frequently.

These caching mechanisms are implemented in two parts: the frontend cache and the backend cache. On the frontend, we use IndexedDB [3], an efficient client-side structured data storage, to store responses to API calls we make to the backend. This way, the user almost always instantly sees the full component showing near real-time data upon opening the dashboard rather than watching a loading screen. Meanwhile, the backend uses Ruby on Rails in-memory caching to store the responses to all Slurm commands and external API calls, refreshing their values periodically. By using this dual-caching structure, we can ensure that users get a seamless experience when using the dashboard while protecting the backend API routes from repeated queries in close succession.

Portability. One of the primary design goals was portability, ensuring that the system could be deployed on other Open OnDemand installations with minimal modification. The bulk of the system relies on Slurm commands, which can be run on any Open OnDemand server that has access to their cluster's Slurm workload manager. This simplifies porting this project to other clusters which are also using Slurm. Also, each feature is largely isolated so that other HPC centers can choose to implement only a portion of the features that they are interested in from this project. Additionally, the fact that this project was built upon the existing Open OnDemand framework opens it up for future open source contributions and easy integration into existing Open OnDemand dashboards. We have shared the source code for this project via our GitHub repository [4] for readers who are interested.

Privacy. The dashboard is built with users' privacy in mind, so we make sure that users do not see information that is not related to them. On the homepage, we only show allocations and disks that each user has access to, preventing them from looking at other users' information and usage habits easily. In the My Jobs app, we limit the job table to only show jobs that the user submitted themselves or jobs submitted under an allocation/group the user is a part of. This ensures that users are not looking at unrelated jobs that they have no association with. Additionally, logs in the Job Overview page are only viewable to the user who submitted the job. This keeps users' directories and files secure and prevents random users from looking at other people's jobs and research. By keeping the dashboard personal to the user, we make it easy for the user to find the information they are looking for without scrolling through other users' jobs and allocation information.

Table 1: Dashboard features with associated data sources

Feature	Data Source(s)
Announcements widget	`API call to RCAC news page`
Recent Jobs widget	`squeue (Slurm)`
System Status widget	`sinfo (Slurm)`
Accounts widget	`scontrol show assoc (Slurm)`
Storage widget	`ZFS and GPFS storage database`
My Jobs	`sacct (Slurm)`
Job Performance Metrics	`sacct (Slurm)`
Cluster Status	`scontrol show node (Slurm)`
Job Overview	`scontrol show job (Slurm)`
Node Overview	`scontrol show node (Slurm)`

3 Dashboard Homepage

The dashboard homepage (shown in Figure 2) was redesigned in order to show key information HPC users might want to know (announcements, quota usage, accounting usage, system status, etc.) at a glance upon opening Open OnDemand in a clean, easy-to-read layout.

3.1 Announcements

While it is crucial for HPC centers to communicate with users about system outages, upcoming maintenance, and other announcements in a timely manner, we try to introduce different levels of communications. In addition to regular emails, the announcements widget on the homepage provides an integrated, less intrusive way of sharing system announcements and updates. It gathers the latest news from the news API on our center's website and displays important recent news, including outages, maintenance periods, new features, and more. This widget allows users to anticipate when the cluster will not be available so they can plan their job scheduling accordingly. The ability to view all news at the click of a button also lets users navigate to a list of all cluster-related articles, where they can learn more about recent events regarding the cluster. The list of announcements is designed in an accordion layout, where users can see the title, date, and time of each recent article in a collapsed view while being able to click on any specific article to expand the contents of the article. The most important announcements are also color-coded to indicate the urgency of an announcement, with outages being red, maintenance periods being yellow, and everything else being gray. In addition to color-coding, current and future announcements are given an active style showing their current relevance while past announcements are styled in a faint gray, indicating that they are no longer relevant and only shown for informative purposes.

3.2 Recent Jobs

Another key widget on the new homepage is the recent jobs widget, which shows an overview of the latest jobs run by the user on the cluster. This widget serves to inform users of the most recent jobs they have run either through the terminal or through Open OnDemand and provides them a quick way to view all their running and queued jobs without scrolling through the Active Jobs page. The job name, job ID, current status, and the time at which the job was submitted, started, or ended are shown in a compact card view, with a description of the status and reason behind the status visible via a hoverable tooltip. This saves users from having to run the `squeue` command in a terminal to see the jobs they just submitted, increasing the accessibility of this information.

Because the `squeue` command queries Slurm's central management daemon (slurmctld)—which also handles all job allocation—rather than Slurm's database daemon (slurmdbd), querying `squeue` too frequently could slow down slurmctld, causing delayed responses when running job allocation commands like `salloc` and other `squeue` commands. In order to limit the amount of load we add to the slurmctld daemon, the results returned from `squeue` are cached in both the Ruby on Rails backend and the frontend, as previously discussed in Section 2.4.

3.3 System Status

The System Status widget displays an overview of the partitions on the cluster. This is useful for determining how busy the resources on the cluster are, such as the CPUs, GPUs, and nodes currently in use. The key components of this widget include each partition's name, current status, and CPU/GPU traffic. The information is represented in both text form and a color-coded progress bar, with green representing less than 70% utilization, yellow representing between 70% and 90% utilization, and red representing over 90% utilization. There is also a link in the widget header with more details about the cluster's partitions and configured resources.

3.4 Accounts

The Accounts widget shows the various allocations (also called accounts) the user has access to and how much of their resource limit they have used. Each allocation has a limit on the number of CPUs they can have allocated at once as well as a limit on the hours of GPU usage. Similar to the System Status widget, the Accounts widget shows each account with its name, CPUs in use and queued, and GPU hours used. In addition to the web view of this information, there is a dropdown for each account to allow users to export the breakdown of account usage by user into an Excel or CSV file. This is especially helpful for managing group allocation usage and identifying users using more resources than others. This widget is also paired with a link to the cluster's user guide with more information about accounting on the cluster.

3.5 Storage

The Storage widget details disks the user has access to on the cluster and their usage in terms of file count and storage size. Every user has access to their home directory—where their personal files are stored—as well as their scratch directory—where they can store temporary files. In addition to these two directories, users have access to various directories associated with their allocations/groups. For each directory, the directory path, disk usage, and file count are shown, along with a color-coded progress bar for clarity. Each directory also has a link to the directory in the built-in Open OnDemand files app, which is a simple web-based file management app to browse and move files around.

4 My Jobs

In order to improve some of the limitations we identified with Open OnDemand's Active Jobs app, we implemented a new job accounting app called My Jobs. The main motivations for this app were to show more information than what is available in the original Open OnDemand Active Jobs app, more job types than just queued jobs, and to provide better filtering methods. As shown in Figure 3, this app includes a table with detailed information about each of the user's jobs as well as a couple of charts visualizing data broken down by user and their associated group(s).

4.1 Job Table

The main component of the My Jobs app is the table listing all of the recent jobs of the user and their associated group(s). This includes jobs that are queued, running, completed, failed, and every other job status. In addition to the columns in Active Jobs, the My Jobs table also shows columns for Quality of Service (QoS), start and end times, wait time, and job efficiency, allowing glanceable information about each job in an organized layout. When expanding a job in the list, more details are shown, including requested memory, GPU hours used, allocated CPUs, session ID, nodes, and other job-specific information. This essentially provides the user with any information they may need to know about their jobs without the need to use additional Slurm terminal commands such as `sacct` or `scontrol`. Another improvement on this widget is the more user-friendly messages for job reasons, which can be obscure to understand for beginners and have not been included in any Open OnDemand apps. For example, a message "It means this job's association has reached its aggregate group CPU limit." will be shown next to the reason "AssocGrpCpuLimit" for a pending job so users will know their job is pending because their group CPU limit has been reached.

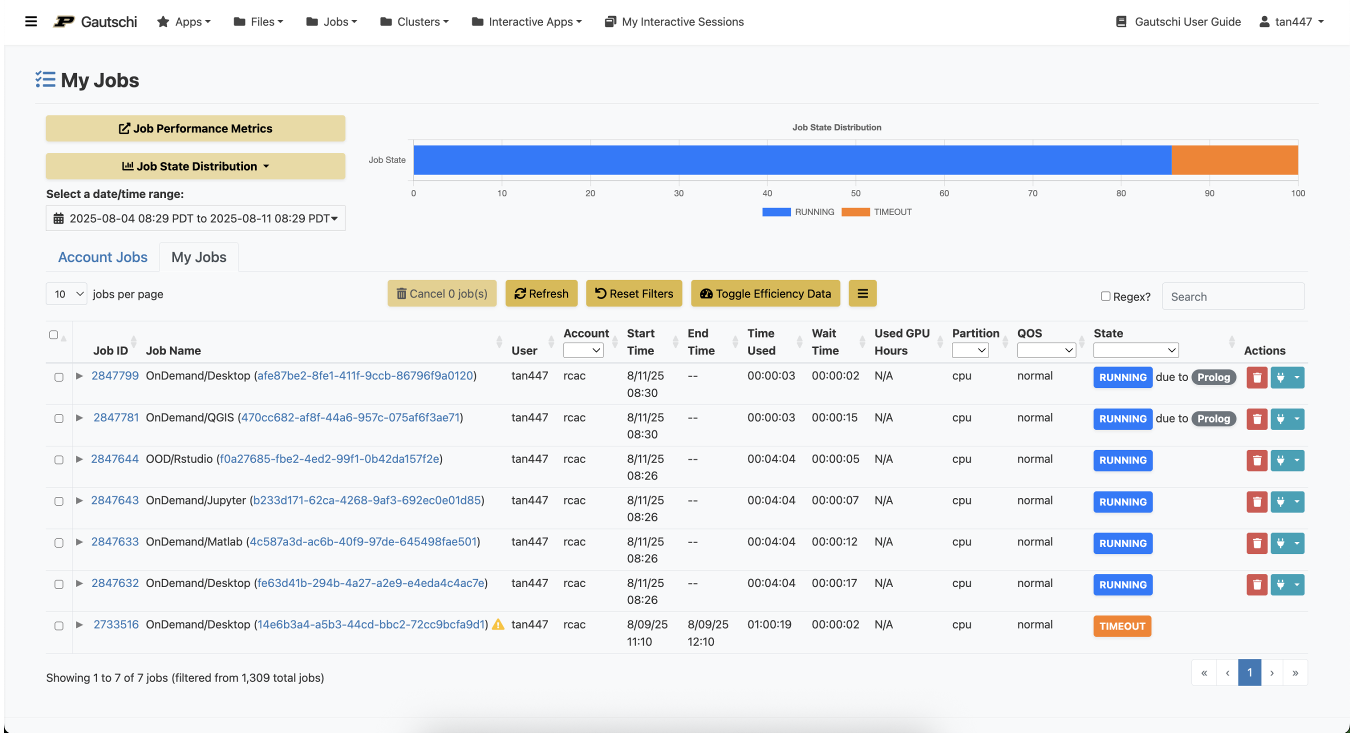
A key feature of the job table is the efficiency warnings, which inform users of how efficiently they are requesting and using resources on the cluster. When a job uses significantly less resources than what was requested and allocated, there are alerts that tell the user that they are only using a certain percentage of what they requested and that requesting less resources in the future will reduce their queue wait times and leave more resources for others. This is helpful because many users will request an unreasonably high number of CPUs and/or memory with long time limits, not realizing that they are barely using what they are requesting. With these alerts, users can become more aware of their resource usage and work towards requesting a more reasonable amount of resources. As additional tools are necessary to collect job-level GPU efficiency, this work only includes efficiency warnings for CPU and memory. The implementation of GPU efficiency is currently underway.

4.2 Job Status and Usage Charts

In addition to the job table, the My Jobs app contains charts that help users visualize the distribution of jobs in the job list. Currently, the two charts that have been implemented are the job state distribution chart and the GPU hour distribution chart. The job state distribution chart displays the percentages of job states in the job list grouped by user using a stacked bar chart with the Chart.js library. This can be useful information for users who may have a batch of jobs that failed or want to filter by just completed jobs by clicking on the completed section of the chart. The GPU hour distribution chart shows the distribution of GPU hours used by all the jobs in the job list grouped by user. This information lets users and group managers monitor their GPU usage within their own account or allocations/groups and keep track of GPU resource usage among groups.

4.3 Job Efficiency

Some new HPC users might frequently request significantly more resources—such as CPUs, memory, and wall time—than they realistically need, which usually brings utilization concerns to the system administrator. The job efficiency metric lets people know how efficiently they have been utilizing their requested resources and how they can reduce their request size to reduce their own queue wait times. The Toggle Efficiency Data button toggles three columns in the job table: time efficiency, CPU efficiency, and memory efficiency. Time efficiency measures the percentage of the requested time that was used, CPU efficiency measures the percentage of the requested CPU time that was used, and memory efficiency measures how much memory was used compared to how much was requested. It is common to see low efficiency on interactive app jobs such as Jupyter Notebook jobs where users will request many CPUs and a long time limit and only use it for a short period of time with minimal load. These efficiency columns along with the efficiency warnings keep users aware of their resource usage and remind them to request only what they need.



Select a date/time range:
2025-08-04 08:29 PDT to 2025-08-11 08:29 PDT

10 jobs per page

Job ID	Job Name	User	Account	Start Time	End Time	Time Used	Wait Time	Used GPU Hours	Partition	QOS	State	Actions
2847799	OnDemand/Desktop (afe87be2-8fe1-411f-9ccb-86796f9a0120)	tan447	rcac	8/11/25 08:30	--	00:00:03	00:00:02	N/A	cpu	normal	RUNNING due to Prolog	
2847781	OnDemand/QGIS (470cc682-af8f-44a6-957c-075af6f3ae71)	tan447	rcac	8/11/25 08:30	--	00:00:03	00:00:15	N/A	cpu	normal	RUNNING due to Prolog	
2847644	OOD/Rstudio (f0a27685-fbe2-4ed2-99f1-0b42da157f2e)	tan447	rcac	8/11/25 08:26	--	00:04:04	00:00:05	N/A	cpu	normal	RUNNING	
2847643	OnDemand/Jupyter (b233d171-62ca-4268-9af3-692ec0e01d85)	tan447	rcac	8/11/25 08:26	--	00:04:04	00:00:07	N/A	cpu	normal	RUNNING	
2847633	OnDemand/Matlab (4c587a3d-ac6b-40f9-97de-645498fae501)	tan447	rcac	8/11/25 08:26	--	00:04:04	00:00:12	N/A	cpu	normal	RUNNING	
2847632	OnDemand/Desktop (fe63d41b-294b-4a27-a2e9-e4eda4c4ac7e)	tan447	rcac	8/11/25 08:26	--	00:04:04	00:00:17	N/A	cpu	normal	RUNNING	
2733516	OnDemand/Desktop (14e6b3a4-a5b3-44cd-bbc2-72cc9bcfa9d1) ⚠	tan447	rcac	8/09/25 11:10	8/09/25 12:10	01:00:19	00:00:02	N/A	cpu	normal	TIMEOUT	

Showing 1 to 7 of 7 jobs (filtered from 1,309 total jobs)

« ‹ 1 › »

Figure 3: My Jobs page.

5 Job Performance Metrics

The Job Performance Metrics app (Figure 4a) is a summary of users' aggregate job metrics, including total job count, average queue wait time, mean job duration, total wall time, and various efficiency metrics. Users have the option to select a time range to analyze, ranging from the last 24 hours to all time, as well as a custom time range option to select any date range to analyze. This tool is useful for users checking how they have been using the cluster over a certain time period and knowing how much of their requested resources they actually used.

6 Cluster Status

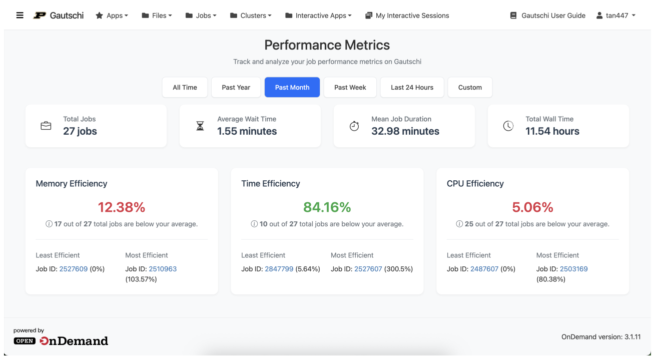
In order to view information about nodes on a cluster, users have to run the `scontrol show node` command in the terminal and interpret the resulting output to find the information they are looking for. The Cluster Status app (Figure 4b) was created to simplify this process by providing an interactive web interface to view the status and information about every node on the cluster. This app supports two modes: grid view—which shows each node as a cell in a grid—and list view—which shows the nodes in a standard table layout.

In grid view, each node is represented as a color-coded square labeled with the name of the node. At a glance, users can tell the overall distribution of node statuses based on the cell colors, with green representing an online node in use, faded green representing an online node not in use, yellow representing a drained node, orange representing a node in maintenance, and red representing an offline node. Users can also hover over any node to view more information about the node, including the number of CPUs and memory it is configured with and the amounts currently being used as well as the partitions it is included in. Clicking on any node will navigate to the node overview page for that node (see Section 6.1 for details).

In list view, the nodes are represented in a table, where each row contains one node's information. This table shows each node's name, state, partitions it is part of, CPU load, and memory load. This view also contains a search bar at the top to filter the nodes by keyword, which lets users view a certain group of nodes, such as a certain partition or a certain node status. Additionally, users can sort any column to find the nodes with the highest or lowest CPU or memory load and/or view the nodes in alphabetical order. Clicking on a node in this list will also link to the node overview page (see Section 6.1 for details) for that node.

6.1 Node Overview

The Node Overview page (Figure 4c) provides a comprehensive look at a node on the cluster and its status, configuration, and the jobs running on it. The top section includes two cards: a status card on the left and a resource usage card on the right. The status card shows the node status (i.e. online, mixed, drained, etc.) and the timestamp when it was last active. The resource usage card shows the CPU usage, GPU usage (if applicable), and memory usage, with both the numeric format and a color-coded progress bar to make it easy to read the information.



Performance Metrics

Track and analyze your job performance metrics on Gautschi

Total Jobs 27 jobs

Average Wait Time 1.55 minutes

Mean Job Duration 32.98 minutes

Total Wall Time 11.54 hours

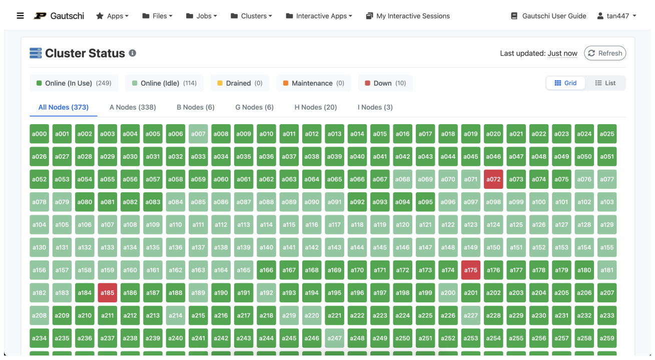
Memory Efficiency
12.38%
17 out of 27 total jobs are below your average.
Least Efficient Job ID: 2527609 (0%)
Most Efficient Job ID: 2510963 (103.57%)

Time Efficiency
84.16%
10 out of 27 total jobs are below your average.
Least Efficient Job ID: 2847799 (5.64%)
Most Efficient Job ID: 2527607 (300.5%)

CPU Efficiency
5.06%
25 out of 27 total jobs are below your average.
Least Efficient Job ID: 2487607 (0%)
Most Efficient Job ID: 2503169 (80.38%)

OnDemand version: 3.1.11

(a) Performance Metrics



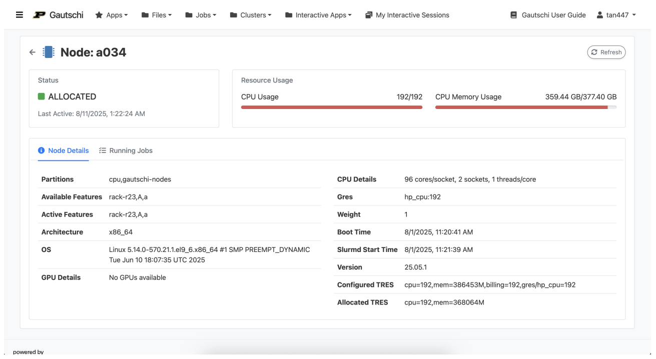
Cluster Status

Last updated: Just now

Online (In Use) (249) · Online (Idle) (114) · Drained (0) · Maintenance (0) · Down (10)

All Nodes (373) · A Nodes (338) · B Nodes (6) · G Nodes (6) · H Nodes (20) · I Nodes (3)

(b) Cluster Status



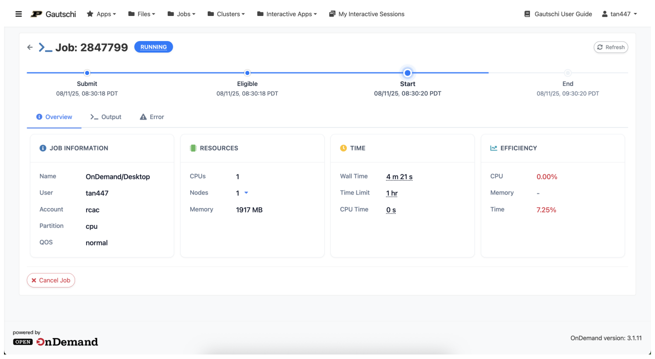
Node: a034

Status: ALLOCATED
Last Active: 8/11/2025, 1:22:24 AM

Resource Usage
CPU Usage 192/192
CPU Memory Usage 359.44 GB/377.40 GB

Partitions	cpu,gautschi-nodes	CPU Details	96 cores/socket, 2 sockets, 1 threads/core
Available Features	rack-r23,A,a	Gres	hp_cpu:192
Active Features	rack-r23,A,a	Weight	1
Architecture	x86_64	Boot Time	8/1/2025, 11:20:41 AM
OS	Linux 5.14.0-570.21.1.el9_6.x86_64 #1 SMP PREEMPT_DYNAMIC Tue Jun 10 18:07:35 UTC 2025	Slurmd Start Time	8/1/2025, 11:21:39 AM
GPU Details	No GPUs available	Version	25.05.1
		Configured TRES	cpu=192,mem=386453M,billing=192,gres/hp_cpu=192
		Allocated TRES	cpu=192,mem=368064M

(c) Node Overview



Job: 2847799 RUNNING

Submit 08/11/25, 08:30:18 PDT
Eligible 08/11/25, 08:30:18 PDT
Start 08/11/25, 08:30:20 PDT
End 08/11/25, 09:30:20 PDT

JOB INFORMATION
Name OnDemand/Desktop
User tan447
Account rcac
Partition cpu
QOS normal

RESOURCES
CPUs 1
Nodes 1
Memory 1917 MB

TIME
Wall Time 4 m 21 s
Time Limit 1 hr
CPU Time 0 s

EFFICIENCY
CPU 0.00%
Memory -
Time 7.25%

OnDemand version: 3.1.11

(d) Job Overview

Figure 4: Other dashboard pages.

The bottom section contains two tabs: a node details tab and a running jobs tab. The node details tab provides a list of information regarding the configuration of the node pulled directly from Slurm's `scontrol show node` command. This includes fields such as available features, operating system, GPU specifications, and other node-related fields. Putting this information here allows the user to easily learn any information they might want to know about a node without spending extra time or effort to open a terminal and run `scontrol` manually. The running jobs tab contains a table listing all the jobs currently running on the node, detailing each job's name, ID, user, partition, state, allocated memory, and other job allocation details. This provides a streamlined way for users and admins to see what jobs are running on a certain node and to identify potentially problematic nodes contributing to job failure.

7 Job Overview

The Job Overview page (Figure 4d) allows users to analyze a single job in detail through a visual interface. At the top of the page, the job ID, name, and color-coded state and reason are shown in the header in a large font size. Below that, there is a timeline of when the job was submitted, became eligible to start, started, and ended, with the color of the timeline matching the color of the job state. The dates and times are adjusted for the user's local timezone. The bottom section contains several tabs with detailed information about the job. This section is separated into tabs, with the overview tab showing general job information, the session tab showing interactive session details, the output and error tabs showing logs from the running job, and the job array tab showing other jobs in the job array if applicable. Each job ID from the My Jobs (Section 4) and Cluster Status (Section 6) pages link to the Job Overview page for that job.

The overview tab contains four cards: Job Information, Resources, Time, and Efficiency. The Job Information card shows details like the job name, user, allocation, partition, and QoS. This information generally describes the job being submitted and identifies the values of several arguments of the original job request. The Resources card includes the number of CPUs, nodes, and memory allocated, as well as links to the Node Overview pages for specific nodes allocated to the job (see Section 6.1 for details). The Time card shows the wall time, time limit, and CPU time of the job, informing the user of how much time they have left for their job, which is especially important when running interactive jobs through the terminal or using Open OnDemand interactive apps. Finally, the Efficiency card details the job's CPU efficiency, memory efficiency, and time efficiency, which tell the user when they have requested an unnecessarily high number of CPUs or time limits for their job.

The session tab is specific to jobs which are run via one of Open OnDemand's interactive apps, such as JupyterLab or RStudio. This tab details the interactive app's name, linking to the form to start another instance of the same interactive app. It also contains the session ID to internally identify the specific interactive job running the app and a link to the session's working directory in the Open OnDemand files app. Below that are the buttons and controls to launch the interactive app once the job starts, identical to what is in the My Interactive Sessions page within Open OnDemand. This makes it quicker for the user to open their interactive app since they can just go directly to this tab rather than scrolling through the list of interactive apps they have launched before on the My Interactive Sessions page.

The output and error tabs are for viewing the respective output and error logs for the job. The contents of the log files are shown in a scrollable read-only text view that automatically scrolls to the bottom of the log to show the most recently logged lines. Because this function is directly linked to the file system, it inherits file permissions from the file system so users cannot check job output and error logs from other users. This feature is extremely helpful for Open OnDemand users to debug their interactive app jobs directly without trying to go inside the session folder and search in the output log file. In order to make the log files easier to read, the line numbers are included on the left side to identify where the user is looking in the log file. Also, the interface will only show the most recent 1000 lines in the log files so the file loads quickly and does not force the user to scroll through too much text to find what they are looking for. If the user decides they want to view the entire file, they can choose to click the link in the top right of the frame to navigate to the Open OnDemand files app page for that file.

Although not all jobs are part of a job array, it can be useful for users to know the statuses of other jobs in the array when they are running jobs in bulk. The Job Array tab only shows up for jobs that are part of a job array, and it lists all the jobs in a job array along with their state, timestamps, node list, and other information.

8 Migration to Other Sites

This dashboard is built upon the Open OnDemand framework, which facilitates easy migration of these features to Open OnDemand dashboards on other HPC clusters. Configuring this dashboard consists of two main steps: downloading the source code and modifying cluster-specific components. To download the source code for this project, either use the `git clone` command or download a ZIP file of the source code from GitHub. The correct location for the files can be cluster-dependent, but the usual location is under the file path `/var/www/ood/apps/sys` in a directory named `dashboard`. After downloading the source code to the correct location, certain components of the dashboard may require cluster-specific modifications. Common areas that need modifications include partition names, the cluster name, and cluster-specific file system paths and scripts. After these modifications are completed, the dashboard should be available at your cluster's Open OnDemand domain.

9 Conclusion and Discussion

Developed on top of the existing Open OnDemand framework, this work redesigns the user-facing dashboard and the key components connected with it. It is designed in a modular manner with minimal dependencies on cluster-specific features and tools, which makes it easier to migrate to other centers with similar HPC configurations. We have shared the source code for this work via our GitHub repository [4] for readers who are interested. The caching mechanisms and minimal webpage design improve the overall performance, enhance the user experience, and reduce system load. The functions for each widget and page, as well as the internal connections between the dashboard and cluster, provide seamless and informative access for HPC users.

The dashboard has been deployed across multiple HPC clusters at RCAC and has become a widely used and effective tool for both end users and system administrators, enhancing their interaction with our HPC resources. Continuous engagement from users and center staff through feedback, feature requests, and bug reports has played a critical role in refining the system and guiding its ongoing development.

Ongoing and future work includes implementing GPU utilization metrics, real-time job monitoring, and AI-powered analysis of users' jobs. Permission-based job accounting, such as administrator-only content, is another feature under development. Additionally, we are continuing to improve the modularity, portability, and performance of the system to create a better experience for all users.

10 Image Credits

Logos used in Figure 1 are trademarks of their respective owners:

- OpenZFS, Public domain, via Wikimedia Commons
- Slurm, GPL, via Wikimedia Commons
- Purdue RCAC, Purdue University, via RCAC website
- Open OnDemand, CC BY 4.0, via Open OnDemand website
- Ruby on Rails, CC0, via Wikimedia Commons
- Bootstrap, Public domain, via Wikimedia Commons
- Font Awesome, CC BY 4.0, via Wikimedia Commons
- Chart.js, MIT, via Wikimedia Commons
- DataTables, CC BY-SA 4.0, via Wikimedia Commons
- CSS, CC0, via Wikimedia Commons
- HTML5, CC BY 3.0, via Wikimedia Commons
- JavaScript, Public domain, via Wikimedia Commons

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 2005632. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] 2025. https://github.com/ubccr/supremm. Accessed: August 11, 2025.
[2] 2025. https://tacc.utexas.edu/portal/login?next=/portal/. Accessed: August 11, 2025.
[3] 2025. https://www.w3.org/TR/IndexedDB/. Accessed: August 11, 2025.
[4] 2025. https://github.com/richtan/Anvil-OOD-Dashboard. Accessed: August 11, 2025.
[5] Alan Chalker, Robert Deleon, David Hudak, Douglas Johnson, Julie Ma, Jeff Ohrstrom, Hazel Randquist, Travis Ravert, Joseph Patrick White, Matt Walton, Emily Moffat Sadeghi, and Ronald Lee Liming. 2024. Open OnDemand: Connecting Computing Power With Powerful Minds. In *Practice and Experience in Advanced Research Computing 2024: Human Powered Computing* (Providence, RI, USA) *(PEARC '24)*. Association for Computing Machinery, New York, NY, USA, Article 16, 8 pages. doi:10.1145/3626203.3670538
[6] Morris A. Jette and Tim Wickberg. 2023. Architecture of the Slurm Workload Manager. In *Job Scheduling Strategies for Parallel Processing*, Dalibor Klusáček,

Julita Corbalán, and Gonzalo P. Rodrigo (Eds.). Springer Nature Switzerland, Cham, 3–23.
[7] Open OnDemand. 2025. About Us. https://openondemand.org/about-us. Accessed: 2025-08-11.
[8] Jeffrey T Palmer, Steven M Gallo, Thomas R Furlani, Matthew D Jones, Robert L DeLeon, Joseph P White, Nikolay Simakov, Abani K Patra, Jeanette Sperhac, Thomas Yearke, et al. 2015. Open XDMoD: A tool for the comprehensive management of high-performance computing resources. *Computing in Science & Engineering* 17, 4 (2015), 52–62.