Introducing Open OnDemand to Supercomputer Fugaku

Masahiro Nakao
Hidetomo Kaneyama
masahiro.nakao@riken.jp
hidetomo.kaneyama@riken.jp
RIKEN Center for Computational Science
Chuo-ku, Kobe, Hyogo, Japan

Masaru Nagaku
Ikki Fujiwara
Atsuko Takefusa
mnagaku@nii.ac.jp
ikki@nii.ac.jp
takefusa@nii.ac.jp
National Institute of Informatics
Chiyoda-ku, Tokyo, Japan

Shin'ichi Miura
Keiji Yamamoto
shinichi.miura@riken.jp
keiji.yamamoto@riken.jp
RIKEN Center for Computational Science
Chuo-ku, Kobe, Hyogo, Japan

ABSTRACT

One of the issues with high-performance computing (HPC) clusters is that the prerequisite knowledge required to use them is large, making the learning cost high for novice users. Moreover, it is desirable to run graphical user interface applications with interactive operations on the compute nodes, but the procedure is complicated. This paper describes how we introduced Open OnDemand, a web portal that enables easy use of the computing resources of an HPC cluster, to Fugaku, a Japanese flagship supercomputer. To introduce the resources to new users, we developed an adapter that enables the job scheduler used in Fugaku to be used from Open OnDemand. In addition, to further improve user convenience, we developed applications that enable data sharing between Open OnDemand and external storage, HPC infrastructure shared storage, and GakuNin Research Data Management. This paper describes the various features we have given to Open OnDemand for Fugaku and the development of the data sharing applications.

CCS CONCEPTS

• **Software and its engineering** → **Integrated and visual development environments**; *Organizing principles for web applications*;
• **Human-centered computing** → Web-based interaction.

KEYWORDS

web platform, high performance computing, user experience

ACM Reference Format:
Masahiro Nakao, Hidetomo Kaneyama, Masaru Nagaku, Ikki Fujiwara, Atsuko Takefusa, Shin'ichi Miura, and Keiji Yamamoto. 2023. Introducing Open OnDemand to Supercomputer Fugaku. In *Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis (SC-W 2023), November 12–17, 2023, Denver, CO, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3624062.3624150

1 INTRODUCTION

The RIKEN Center for Computational Science (R-CCS) has operated Fugaku as the flagship supercomputer in Japan[7]. In addition,

SC-W 2023, November 12–17, 2023, Denver, CO, USA

ACM ISBN 979-8-4007-0785-8/23/11.
https://doi.org/10.1145/3624062.3624150

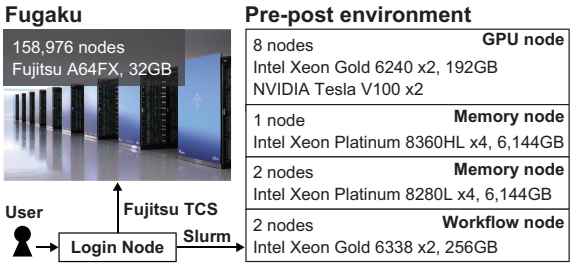


Figure 1: Overview of Fugaku and pre-post environment

R-CCS provides a pre-post environment for visualization and data conversion. Fig. 1 is a conceptual diagram of this environment. The pre-post environment consists of nodes with GPUs, some with a large memory capacity, and some for workflow applications. Fugaku and the pre-post environment share the login node. Fugaku's job scheduler is the Fujitsu Software Technical Computing Suite (Fujitsu TCS) [3], while the pre-post environment's job scheduler is Slurm[11]. In the conventional usage procedure, the user first logs in to the common login node using Secure Shell (SSH), and then submits jobs to each system using the corresponding job scheduler.

To use high-performance computing (HPC) clusters such as Fugaku, knowledge of the command line interface using Shell, SSH key pair generation and public key registration, and job scheduler is required, resulting in a high learning cost for novice users. In recent years, it has become desirable to run graphical user interface (GUI) applications with interactive operations as HPC applications. However, the procedure for running such applications on the compute nodes in an HPC cluster is complicated. For example, in the case of a web-based application such as JupyterLab, the user must perform the following steps each time the application is executed. (1) Log in to the login node via SSH. (2) Run JupyterLab on the compute node through the job scheduler. (3) Obtain the IP address of the compute node and the port number used by JupyterLab. (4) Connect to the local port by SSH tunneling to the obtained IP address and port number. (5) Open the local port with a web browser. As described above, not only do these tasks require a lot of time and effort, but they also require the aforementioned knowledge, which is a heavy burden for users.

To solve the above issues, we modified Open OnDemand[2], a web portal for HPC clusters, to run on Fugaku. Open OnDemand allows users to access the computing resources of the HPC cluster

from a web browser instead of SSH. Furthermore, interactive operation of GUI applications running on the compute nodes of the HPC cluster can be easily performed. Although Open OnDemand supports various schedulers (e.g., Slurm, PBS Pro, and Torque), it previously did not support Fujitsu TCS. Thus, this paper also describes the development of an adapter to use Fujitsu TCS in Open OnDemand. In addition, to further improve convenience for users, we developed applications on Open OnDemand that enable data sharing on HPC infrastructure (HPCI) shared storage[5] and GakuNin Research Data Management (RDM)[8]. HPCI shared storage and GakuNin RDM are research data management services for sharing research data in Japan. This paper describes our efforts for introducing Open OnDemand to Fugaku and the development of the adapter for Fujitsu TCS and the data sharing applications.

This paper is organized as follows. Section 2 gives an overview of Open OnDemand, and Section 3 gives an overview of HPCI shared storage and GakuNin RDM. Section 4 explains how we developed the adapter for Fujitsu TCS in Open OnDemand. Section 5 describes our efforts to extend Open OnDemand to Fugaku and the data sharing applications. Section 6 reports an overhead evaluation of the data sharing application for Open OnDemand. Section 7 summarizes this paper and discusses future work.

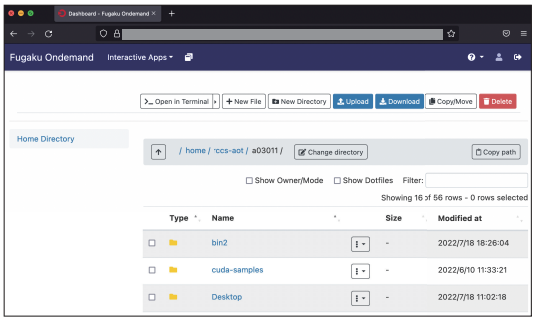
2 OVERVIEW OF OPEN ONDEMAND

Open OnDemand is open-source software that makes it easy to use computing resources on HPC clusters[2]. One of the purposes of Open OnDemand is to reduce the learning cost for using HPC clusters. Settlage[12] found that the median time from the user's first login to job submission was about 22 hours with the traditional SSH method, but about 2 hours with the Open OnDemand method. Applications on Open OnDemand can be divided into (A) those that run on a server where Open OnDemand is installed and (B) those that run on compute nodes of HPC clusters. Each is introduced below.

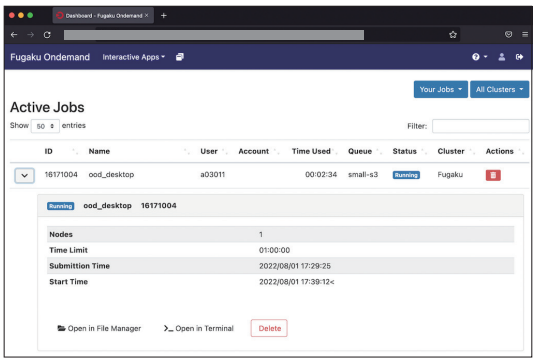
Fig. 2 shows the category (A) applications pre-installed in Open OnDemand. The Home Directory in Fig. 2a uploads, downloads, and edits files. The Active Jobs in Fig. 2b monitors jobs. The Job Composer in Fig. 2c creates and submits jobs to HPC clusters. The Shell in Fig. 2d provides a web-based terminal. The framework provided by Open OnDemand can be used to develop and introduce new applications[1]. The development of the data sharing applications for HPCI shared storage and GakuNin RDM described in Section 5.6 uses that framework.

When Home Directory works with rclone[9], which is software that manages files on cloud storage, it is possible to transfer data between Open OnDemand and cloud storage such as Amazon S3. Many Japanese research institutes, including R-CCS, are connected to the academic information network SINET operated by the National Institute of Informatics (NII). When using the cloud connection service provided by SINET, high-speed data communication between the institution and cloud storage can be performed. However, rclone does not support HPCI shared storage and GakuNin RDM.

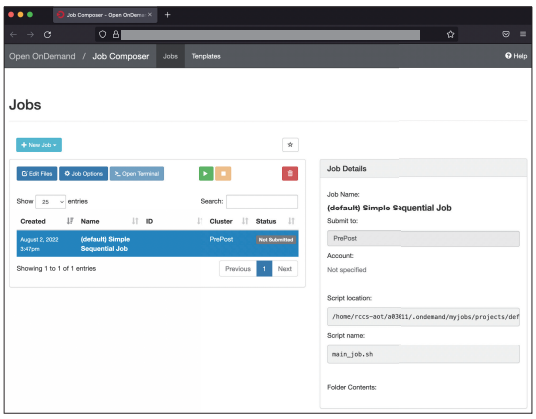
For category (B), Open OnDemand works with the job scheduler of the HPC cluster and calls applications installed on the compute



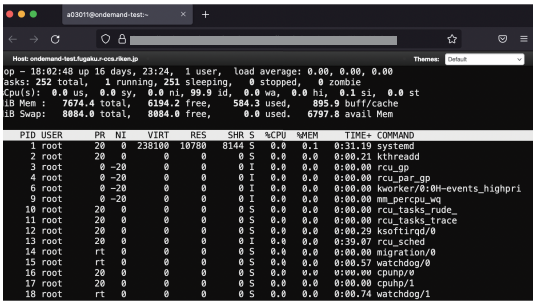
(a) Home Directory



(b) Active Jobs



(c) Job Composer



(d) Shell

Figure 2: Applications pre-installed in Open OnDemand

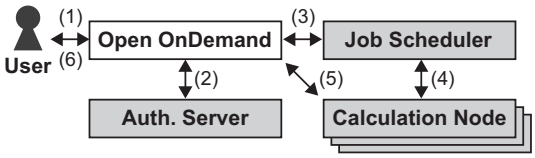


Figure 3: Operation flow of interactive applications

nodes. Although the application is intended to be interactive, a normal batch job submission is also possible. Fig. 3 shows the operation flow for an interactive application. In the figure, the authentication server, job scheduler, and compute nodes are services provided by the HPC cluster and are independent of Open OnDemand. (1) The user logs in to the web server running Open OnDemand using their own web browser. No software other than a web browser or plug-in for a web browser is required. (2) Open OnDemand performs authentication for the login. (3) Application execution commands are issued from the web browser. (4) The job scheduler allocates the compute nodes for the job. (5) When the job is executed, information about the compute node and the port number used by the GUI application are sent to Open OnDemand. Then, Open OnDemand sets up a reverse proxy to connect to the compute node. (6) The user connects to the compute node inside the HPC cluster from the web browser using the reverse proxy.

3 JAPANESE RESEARCH DATA SERVICE

3.1 HPCI shared storage

HPCI is a shared computing environment infrastructure that connects the computing resources of Japanese research organizations with SINET, and Fugaku is also a part of HPCI. HPCI shared storage is a large-scale data sharing infrastructure for sharing research data quickly and securely among geographically distributed HPCI organizations. HPCI shared storage is jointly operated by R-CCS (Kobe City, Hyogo Prefecture) and the Information Technology Center of the University of Tokyo (Kashiwa City, Chiba Prefecture), and file servers are located at each site. Gfarm[10] has been adopted as the file system for HPCI shared storage, and complete mirroring is performed by placing one file copy at each site.

The procedure for connecting to HPCI shared storage is as follows. (1) To access HPCI shared storage using Grid Security Infrastructure authentication, an electronic certificate and a proxy certificate are issued from the HPCI certificate issuing system[1]. (2) The user logs in to the environment where the Gfarm client is installed, and downloads the proxy certificate with the `myproxy-logon` command. The user must also enter the passphrase issued with the proxy certificate. (3) The user executes the `mount.hpci` command and the `fusermount` command to mount HPCI shared storage.

3.2 GakuNin RDM

GakuNin RDM is a research data management service for storing and sharing research data and related materials with collaborators, and it is maintained and operated by NII. GakuNin RDM is based on the Open Science Framework provided by the US nonprofit Center for Open Science. Like Open OnDemand, GakuNin RDM provides

[1]https://portal.hpci.nii.ac.jp

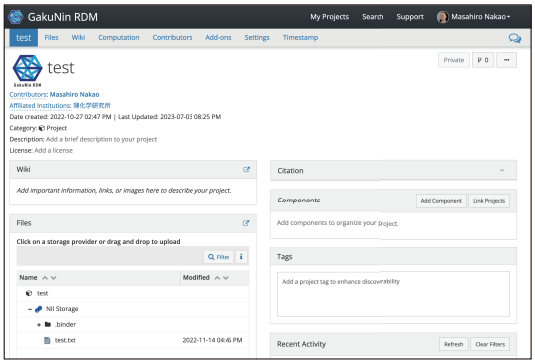


Figure 4: Project page in GakuNin RDM

functions for data transfer and data analysis using JupyterLab via web browsers.

A user takes the following steps to use GakuNin RDM. (1) Create a research project on GakuNin RDM. (2) Register joint researchers. (3) Save, share, and manage versions of the research data in the storage for each research project. Fig. 4 shows a GakuNin RDM project page. To connect to GakuNin RDM project storage using a method other than a web browser, the storage is mounted using a script published by NII[2]. This script uses pyfuse3, a filesystem in userspace library for Python3. When mounting storage, the user needs a project ID and a personal access token that can be obtained from the corresponding project page on GakuNin RDM.

4 ADAPTER FOR FUJITSU TCS

4.1 Overview

This section explains how we implemented an adapter for using Fujitsu TCS in Open OnDemand. In addition, we modified the existing code to display job details and job submission options related to Fujitsu TCS. The changes were merged into the master branch of the Open OnDemand GitHub repository[3]. Therefore, Open OnDemand can be used as is on other HPC clusters that have adopted Fujitsu TCS as the job scheduler.

4.2 Job scheduler adapter

Open OnDemand provides an adapter interface to support various job schedulers. Therefore, the methods defined in the parent class of the adapter are implemented with Ruby for Fujitsu TCS. Some of the methods are shown below.

submit Submit a job
delete Delete a job
status Get status of a job
hold Hold a job
release Release a held job
info Get information for a job
info_all Get information for all jobs
cluster_info Get system information for an HPC cluster
supports_job_arrays Bulk job support availability

[2]https://github.com/RCOSDP/CS-rdmfs
[3]https://github.com/OSC/ood_core

```
1  module Adapters
2   class Fujitsu_TCS < Adapter
3    class Batch
4     def delete_job(id)
5      call("pjdel", id.to_s)
6     end
7
8     def delete(id)
9      @fujitsu_tcs.delete_job(id.to_s)
10    end
```

Figure 5: Code for delete method

directive_prefix Prefix for a job scheduler (#PJM for Fujitsu TCS)

This section describes the implementation of the **delete** and **info_all** methods. To simplify the explanation, exception handling is omitted.

Fig. 5 shows the code for the **delete** method. In line 8, the overridden **delete** method is defined. In line 9, **@fujitsu_tcs** is an instance of the **Batch** class. The `pjdel` command called in line 5 is a Fujitsu TCS command that deletes a specified job, and the job ID is passed through as an argument.

Fig. 6 shows the code for the **info_all** method. In line 30, the overridden **info_all** method is defined. In the **get_jobs** method defined in line 1, the `pjstat` command called in line 6 is a Fujitsu TCS command that gets job information, and its options are defined in lines 2–4. In line 2, "-s" outputs detailed information, "--data" outputs data in the CSV format, and "--choose" selects the items to be output (from the left, the codes mean job ID, job name, resource group, status, standard output file path, standard error file path, submission time, start time, number of nodes, user name, time limit, and elapsed time, respectively). In lines 3–4, "--filter" is an option to filter the output of the `pjstat` command by job ID and user name. In lines 11–15, the hash array **jobs** is created by formatting the output of the `pjstat` command and options described above. In the **parse_job_info** method defined in line 21, a **Info** object is created for each job based on the hash array **jobs** obtained from the **get_jobs** method. In line 23, the **get_state** method converts the job state obtained from the `pjstat` command to one of the job states defined in Open OnDemand (undetermined, completed, queued_held, queued, running, or suspended). In lines 25–26, the **duration_in_seconds** method converts the duration to seconds (e.g., "00:01:23" changes to 83).

4.3 Detailed information

Clicking the button to the left of the ID number in the Active Jobs page shown in Fig. 2b displays the details of the job. However, in the case of unsupported schedulers, as was previously the case for Fujitsu TCS, nothing is displayed in the details. Therefore, we decided to display the number of nodes, time limit, submission time, and start time for the job, which is information obtained from the **pjstat** command. Fig. 7 shows the modified code. The variable **info** used in the code is the **Info** object created in lines 22–27 of Fig. 6.

```
1  def get_jobs(id: "", owner: nil)
2   args = ["-s", "--data", "--choose=jid,jnam,rscg,st,std,stde
          ,adt,sdt,nnumr,usr,elpl,elp"]
3   args.concat ["--filter", "jid=" + id.to_s] unless id.to_s.empty?
4   args.concat ["--filter", "usr=" + owner.to_s] unless owner.to_s.
          empty?
5
6   StringIO.open(call("pjstat", *args)) do |output|
7    output.gets() # Skip header
8    jobs = []
9    output.each_line do |line|
10    l = line.split(" ")
11    jobs << {:JOB_ID => l[1], :JOB_NAME => l[2],
12       :RSC_GRP => l[3].split[0], :ST => l[4], :STD => l[5],
13       :STDE => l[6], :ACCEPT => l[7], :START => l[8],
14       :NODES => l[9].split(":")[0], :USER => l[10],
15       :ELAPSE_LIM => l[11], :ELAPSE_TIM => l[12].split[0]}
16   end
17   jobs
18  end
19 end
20
21 def parse_job_info(v)
22  Info.new(id: v[:JOB_ID], job_name: v[:JOB_NAME],
23   queue_name: v[:RSC_GRP], status: get_state(v[:ST]),
24   submission_time: v[:ACCEPT], dispatch_time: v[:START],
25   wallclock_limit: duration_in_seconds(v[:ELAPSE_LIM]),
26   wallclock_time: duration_in_seconds(v[:ELAPSE_TIM]),
27   job_owner: v[:USER], native: v)
28 end
29
30 def info_all(attrs: nil)
31  @fujitsu_tcs.get_jobs().map do |v|
32   parse_job_info(v)
33 end
```

Figure 6: Code for info_all method

```
1  def extended_data_fujitsu_tcs(info)
2   return unless info.native
3   attributes = []
4   attributes.push Attribute.new "Nodes", info.native[:NODES]
5   attributes.push Attribute.new "Time␣Limit", pretty_time(info.
          wallclock_limit)
6   attributes.push Attribute.new "Submission␣Time", info.native[:
          ACCEPT]
7   attributes.push Attribute.new "Start␣Time", info.native[:START]
8   ...
```

Figure 7: Code for Active Jobs

4.4 Job submission option

If multiple compute nodes are used, an option to specify the number of nodes is required in the job submission command. For example, to use two compute nodes on Fujitsu TCS, the user enters "`pjsub -L node=2`." Fig. 8 shows the modified code to achieve this. Only lines 6–7 were added to the existing code.

```
1  def submit(fmt: nil)
2    slots = value.blank? ? 1 : value.to_i
3    case fmt
4    when "slurm"
5      native = ["-N", slots]
6    when "fujitsu_tcs"
7      native = ["-L", "node=#{slots}"]
8    ...
```

Figure 8: Code for job submission

5 OPEN ONDEMAND ON FUGAKU

5.1 Overview

This section describes the settings and innovations of Open OnDemand on Fugaku. The settings described in this section are available at https://github.com/RIKEN-RCCS/ondemand_fugaku.

5.2 Authentication

The conventional procedure to log in to Fugaku via SSH is as follows. (1) R-CCS sends a client certificate to the user. (2) The user installs the certificate in their web browser or OS. (3) The user creates an SSH key pair if the user does not have one. (4) The user logs in to the user portal of Fugaku and registers the SSH public key to the portal. (5) The user logs in to the login node via SSH.

As described above, Fugaku performs user authentication using a client certificate. As an authentication server for Open OnDemand in Fig. 3, we installed keycloak, an authentication software for single sign-on on the web, to perform authentication using a client certificate. When Open OnDemand works with keycloak, the users can log in to the Open OnDemand web portal by performing only steps (1) and (2) above. Thus, it is expected that the effort required to log in to Fugaku will be greatly reduced.

5.3 Available applications

Open OnDemand on Fugaku divides the applications that run on the compute nodes described in Section 2 (category B) into "interactive applications" and "batch jobs." Table 1 and Table 2 show the available interactive and batch job applications, respectively. Applications marked with an asterisk (*) in the table indicate that they are commercially available.

The commercial applications are installed on Fugaku or in the pre-post environment. Many of the interactive applications in Table 1, except for the commercial applications, are not installed on Fugaku or in the pre-post environment. Therefore, we prepared a SingularityPro[13] container image with the applications installed, which is called from Open OnDemand. The CPU used for the Fugaku nodes (Fig. 1) is the Fujitsu A64FX[4], which is based on the ARM architecture, while the CPU used for nodes in the pre-post environment has the general x86_64 architecture. Thus, we prepared container images for each architecture. In the container image for the pre-post environment, all applications other than the commercial applications in Table 1 are installed. In the container image of Fugaku, only software that is frequently used in Fugaku is installed to reduce management costs. Specifically, the development environments Desktop, Jupyter, RStudio, and VSCode are installed, as is

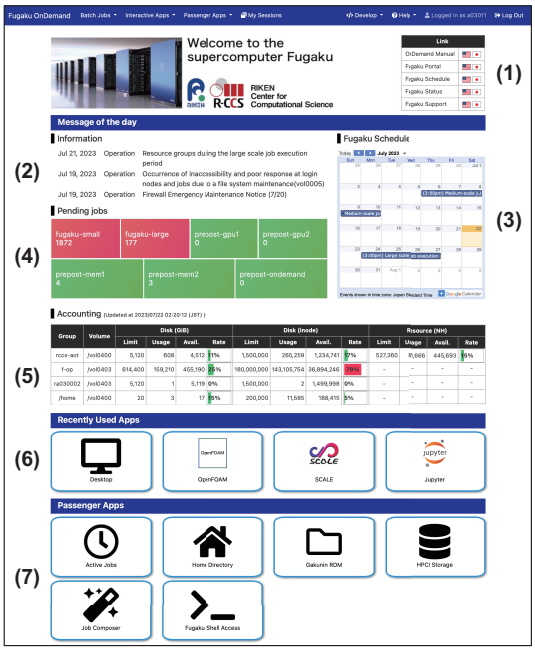


Figure 9: Dashboard of Open OnDemand on Fugaku

Paraview, a viewer that enables parallel processing using multiple processes. Although the definition files for the container image for each architecture are basically the same, some RPM packages for the ARM architecture do not exist (e.g., TurboVNC for Desktop). In such cases, the source code for the application is obtained using the `wget` command in the definition file, and then compiled and installed.

The applications for batch jobs in Table 2 are already installed in Fugaku and pre-post environment, and managed using the package management system Spack[6] on Fugaku. Therefore, the corresponding application is invoked from Open OnDemand using the `spack load` command.

5.4 Dashboard

The first page that appears when a user logs in to Open OnDemand is called the "dashboard." Fig. 9 shows the dashboard of Open OnDemand on Fugaku. The content displayed on the dashboard can be freely edited by the administrator by writing directly in eRuby. To improve user convenience, the following information that is important for the use of Fugaku is displayed. (1) External links to manuals and Fugaku resources. (2) System failure and operation information. (3) Operation calendar for large-scale job execution periods and maintenance (using Google Calendar). (4) The number of jobs waiting in each queue of the job scheduler (using Grafana). (5) The user's disk and budget utilization. To obtain the user's disk and budget utilization, it is necessary to execute a Fujitsu TCS command several times, and it takes several seconds to obtain all the information. To avoid a delay of several seconds each time the dashboard is displayed, the information for all users is saved once a day using `cron`, and the saved information is loaded to the dashboard.

Table 1: Interactive applications in Open OnDemand on Fugaku

Category	Application
Development	Desktop, Jupyter, MATLAB*, RStudio, VSCode
Profiler	NVIDIA Visual Profiler*, NVIDIA Nsight Compute*, NVIDIA Nsight Systems*, Vampir*
Viewer	AVS/Express*, C-Tools, GaussView*, ImageJ, OVITO, Paraview, PyMOL, SALMON view, Smokeview, VESTA, VMD, VisIt, XcrySDen
Workflow	WHEEL

Table 2: Applications for batch jobs in Open OnDemand on Fugaku

Category	Application
Climate	SCALE
Computer Aided Engineering	FDS, FrontFlow (blue/X), FrontISTR, OpenFOAM (Foundation/OpenCFD)
Condensed Matter Physics	ALAMODE, AkaiKKR, $\mathcal{H}\Phi$, mVMC, OpenMX, PHASE/0, Quantum Espresso, SALMON
Molecular Dynamics	GENESIS, GROMACS, LAMMPS, MODYLAS
Quantum Chemistry	ABINIT-MP, Gaussian*, NTChem, SMASH
Quantum Simulation	braket

Table 3: Items in Fugaku queues

Queue	Time (hours)	Nodes
fugaku-small	72	384
fugaku-large	24	12,288

Table 4: Items in pre-post environment queues

Queue	Time (hours)	CPU Cores	Memory (GB)	GPUs
prepost-gpu1	3	72	186	2
prepost-gpu2	24	36	93	2
prepost-mem1	3	224	6,045	-
prepost-mem2	24	56	1,511	-
prepost-ondemand	720	8	32	-

Applications that were used once are likely to be used again. To allow users to immediately launch such applications, "Recently Used Apps" (6) displays the applications that were used recently. This is the default function of Open OnDemand, but with the default setting, clicking an icon in (6) automatically submits a job using the parameters that were previously executed. To be able to deal with the case where the parameters need to be changed, we change the settings. The icons in (7) are applications that run on the server where Open OnDemand is installed, as described in Section 2 (category A).

5.5 Web form

Table 3 and Table 4 show each queue for the Fugaku and pre-post job schedulers, respectively, with the maximum values of the main items. Although only a single node is available in the pre-post environment, oversubscribing, the ability to run multiple jobs on a single node simultaneously, is also possible, which allows the user to specify the number of CPU cores, amount of memory, and number of GPUs. The `prepost-ondemand` queue is a workflow queue, so it can be configured to have a long duration.

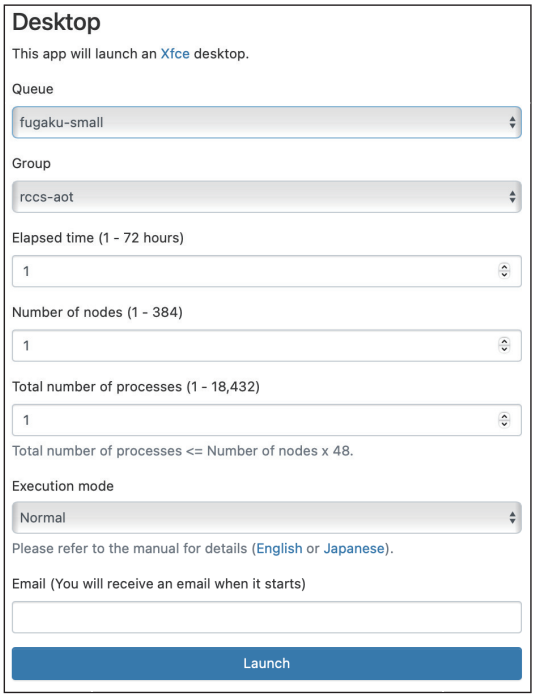


Figure 10: Web form for fugaku-small queue

The procedure for executing the applications shown in Table 1 and Table 2 is as follows. When a user selects an application icon from the navigation bar at the top or from Recently Used Apps on the dashboard in Fig. 9, the `fugaku-small` queue web form will be displayed as shown in Fig. 10. If the queue item is set to "prepost-gpu1," it switches to the web form for the `prepost-gpu1` queue, as shown in Fig. 11. The input items and their maximum values for each web form are shown in Table 4. When "Launch" is clicked, the job is submitted to the specified queue. For an interactive application, when the job is executed on a compute node, a link to connect

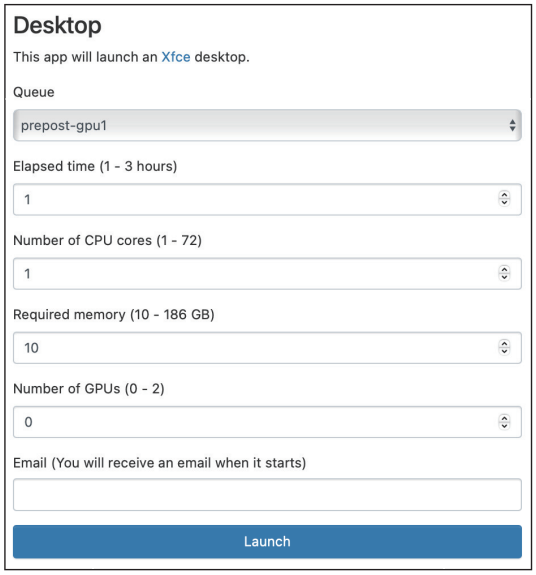


Figure 11: Web form for prepost-gpu1 queue

Figure 12: Link to connect to compute node

to that compute node is displayed to the user, as shown in Fig. 12. For a batch job, only a message indicating that the job is running is displayed.

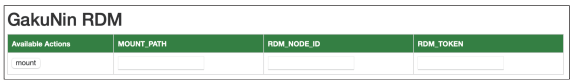
A user may belong to multiple Fugaku user groups. Because a budget is set for each group, it is necessary to select which group's budget is used in Fig. 10. Note that the budget is not consumed in the pre-post environment, so there are no group budget items in Fig. 11. Although omitted in Table 3, Fugaku has a special low-priority and no-consumption budget queue that can be used only by groups whose budget usage exceeds 95%. In such a case, this special queue is also displayed in the queue item in Fig. 10. In this way, Open OnDemand has a mechanism to dynamically generate different web forms for users with different circumstances. One problem with this mechanism is that, to display the dashboard, the web forms for all applications need to be generated, which slows down the display of the dashboard. Therefore, as described for the Fugaku status in Section 5.4, we use `cron` to obtain information for each user or group in advance to speed up the dynamic generation of web forms.

Configuration files for web forms are defined in the YAML format. Fig. 13 shows part of the configuration for the `fugaku-small` queue. Such a configuration file must be prepared for each application. However, the configuration items for all applications are almost the same. To simplify code management, eRuby is used to automatically generate configuration files. Specifically, we define a function that outputs Fig. 13 as it is, and generate the configuration files for the web forms of all applications. This function is called from the configuration files of all applications' web forms. By doing this, the number of lines in the configuration file per application is reduced from about 250 to 25. Although omitted from this paper, we have also made similar efforts in the shell scripts that invoke applications from the job scheduler.

```
1  fugaku_small_hours:
2    label: Elapsed time (1 – 72 hours)
3    widget: number_field
4    value: 1
5    min: 1
6    max: 72
7    step: 1
8    required: true
```

Figure 13: Some settings for fugaku-small queue

(a) HPCI Shared Storage



(b) GakuNin RDM

Figure 14: Data communication applications

5.6 Data sharing applications

Fig. 14 shows a screenshot of the data sharing applications for HPCI shared storage and GakuNin RDM. In Fig. 14a, the input items for HPCI shared storage are the HPCI ID, the elapsed time, and the passphrase for the proxy certificate. When the "mount" button is clicked, the `myproxy-logon` and `mount.hpci` commands described in Section 3.1 are executed. The local path to be mounted is automatically determined by the `mount.hpci` command, and the path is output on the right side of the display. The path is used as the startup link for the Home Directory application in Fig. 2a. Clicking the link opens the corresponding directory in the Home Directory application. In Fig. 14b, the input items for GakuNin RDM are the local path to be mounted, the project ID, and the personal access token. When the "mount" button is clicked, mounting is performed by the script described in Section 3.2. After that, the process is the same as for HPCI shared storage.

Using this application makes it possible to transfer data from Fugaku to HPCI shared storage or GakuNin RDM on the Home Directory application in Fig. 2a. This eliminates the cumbersome command-line procedure described in Section 3, and is expected to allow users to manage their research data more easily.

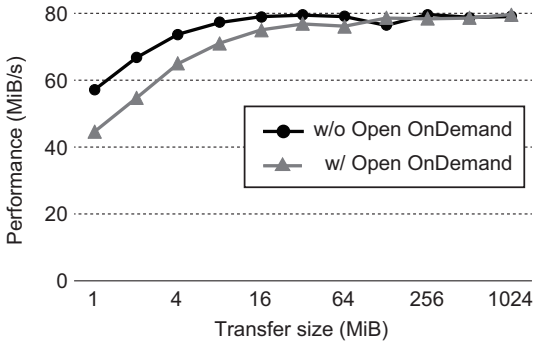


Figure 15: Data transfer speed evaluation

6 EVALUATION

This section clarifies the computational overhead of Open OnDemand by comparing the data transfer rate between Fugaku and external storage with and without the Open OnDemand functions described in Section 5.6. Because the overhead is expected to be the same regardless of whether HPCI shared storage or GakuNin RDM is used for external storage, only HPCI shared storage was used in this evaluation. Also, since the Open OnDemand server and HPCI shared storage are in the same building, we considered that there would be little disturbance in communication performance.

When Open OnDemand was used, the elapsed time for each command output to the Open OnDemand log was measured. The measurement accuracy was on the order of 10^{-5} s. The `cp` command was used for file transfers in the Home Directory application. When Open OnDemand was not used, login to the Open OnDemand server was performed directly using SSH and the same transfer was made using the `cp` command. As described in Section 3.1, HPCI shared storage is divided into two locations, so in this experiment, the data were configured to be transferred to a file server located in R-CCS. The server running Open OnDemand on Fugaku is connected to the Fugaku file system with InfiniBand EDR (100 Gbps), and the server is also connected to HPCI shared storage with 100 Gigabit Ethernet. The specifications of the server are the same as those of the "workflow node" in Fig. 1. The versions of Open OnDemand, Gfarm client and gfarm2fs used were 3.0.1, 2.7.24 and 1.2.17, respectively.

We evaluated the performance by transferring files of various sizes from Fugaku to HPCI shared storage. Fig. 15 shows the best values out of 10 trials. The results show that the data transfer without Open OnDemand is up to 28% faster for small data amounts due to its overhead, but the performance difference disappears as the amount of data increases. When transferring data to external storage, the overhead of Open OnDemand is not considered to be a problem in practical use, because multiple small files are often compressed into a single large data file and sent.

7 SUMMARY AND FUTURE WORK

This study extended Open OnDemand, which makes it possible to easily use the computing resources of an HPC cluster, to Fugaku, and this paper describes some of the modifications we made for its operation. Specifically, the following are described: support for Fujitsu TCS, use of SingularityPro for the installation of various interactive applications and applications for batch jobs, display of useful information for users on the dashboard and speeding up of the display, dynamic generation of web forms and simplification of configurations, and the development of data transfer applications for HPCI shared storage and GakuNin RDM. The performance of the data sharing application was evaluated, and it was confirmed that the overhead of Open OnDemand is sufficiently small.

Planned future work is as follows. (1) When multiple datasets are transferred to HPCI shared storage, data transfer can be completed in a short time by parallel transfer using the `gfpcopy` command. Therefore, we will add an option to select parallel transfer in the Home Directory application in Open OnDemand. (2) To quantitatively clarify the effect of introducing Open OnDemand to Fugaku, statistical information such as the number of user logins and the number of job submissions will be obtained and compared between conventional users and Open OnDemand users. (3) Since rclone described in Section 2 requires configuration through the CLI, it is difficult to set up for novice users that Open OnDemand assumes. Although rclone also has a web-based GUI for setting, it is necessary to launch a remote desktop on the compute node to use it from Open OnDemand. The data sharing application described in Section 5.6 can be configured entirely with a GUI on the Open OnDemand server. Thus, by extending the data sharing application to call rclone, it can be used for all cloud storage supported by rclone. (4) Open OnDemand enables the construction of a common front end for HPC clusters, thereby reducing system development costs and providing a unified interface for users. Therefore, we will continue to publicize our experience with Open OnDemand on Fugaku to promote its widespread use.

ACKNOWLEDGMENTS

We would like to thank Fujitsu engineers for their advice in implementing Open OnDemand.

REFERENCES

[1] Chen, Huan and Fietkiewicz, Chris. 2018. Version Control Graphical Interface for Open OnDemand. In *Proceedings of the Practice and Experience on Advanced Research Computing* (Pittsburgh, PA, USA) *(PEARC '18)*. Article 103, 4 pages. https://doi.org/10.1145/3219104.3229268
[2] Dave Hudak et al. 2018. Open OnDemand: A web-based client portal for HPC centers. *Journal of Open Source Software* 3, 25 (2018), 622. https://doi.org/10.21105/joss.00622
[3] Fujitsu Inc. 2018. FUJITSU Software Technical Computing Suite (in Japanese). https://www.fujitsu.com/downloads/JP/jsuper/tcs-v4-datasheet.pdf.
[4] Fujitsu Inc. 2023. A64FX Microarchitecture Manual. https://github.com/fujitsu/A64FX/.
[5] High Performance Computing Infrastructure. 2023. HPCI shared storage. https://www.hpci-office.jp/en.
[6] Lawrence Livermore National Laboratory. 2023. Spack Website. https://spack.io.
[7] Mitsuhisa Sato et al. 2020. Co-Design for A64FX Manycore Processor and "Fugaku". In *International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society, Los Alamitos, CA, USA, 651–665.
[8] National Institute of Informatics. 2023. GakuNin RDM. https://rdm.nii.ac.jp.
[9] Nick Craig-Wood. 2023. Rclone Website. https://rclone.org.
[10] Osamu Tatebe et al. 2010. Gfarm Grid File System. *New Generation Computing, Ohmsha, Ltd. and Springer* 28, 3 (2010), 257–275.
[11] SchedMD LLC. 2023. Slurm. https://slurm.schedmd.com/.
[12] Settlage, Robert et al. 2019. Open OnDemand: HPC for Everyone. In *High Performance Computing: ISC High Performance 2019 International Workshops, Frankfurt, Germany, June 16-20, 2019, Revised Selected Papers*. 504–513. https://doi.org/10.1007/978-3-030-34356-9_38
[13] Sylabs. 2023. SingularityPRO. https://sylabs.io/singularity-pro/.