スーパーコンピュータ「富岳」における HPCクラスタ用Webポータル Open OnDemandの運用

中尾 昌広[1,a)]　金山 秀智[1]　長久 勝[2,3]　藤原 一毅[2]　竹房 あつ子[2]　三浦 信一[1]　山本 啓二[1]

概要：「富岳」などの HPC クラスタの問題点として，HPC クラスタを用いるための前提知識が多いため，初心者にとって利用するまでの学習コストが大きい点が挙げられる．そこで，我々は HPC クラスタの計算資源を簡易に利用可能にする Web ポータル Open OnDemand を「富岳」で運用している．ユーザの利便性をさらに向上させるため，Open OnDemand と HPCI（High Performance Computing Infrastructure）共用ストレージおよび GakuNin RDM（Research Data Management）との間でデータ共有を行うことができるアプリケーションを開発した．本稿では，我々が「富岳」の Open OnDemand に対して行っている様々な工夫およびそのデータ共有アプリケーションの開発について述べる．

1. 背景

理化学研究所 計算科学研究センター（R-CCS：RIKEN Center for Computational Science）は，日本におけるフラッグシップスーパーコンピュータとして「富岳」を運用している [1]．さらに，R-CCS は可視化やデータ変換等を行うためのプリポスト環境も提供している．「富岳」とプリポスト環境の概念図を**図 1** に示す．プリポスト環境は GPU を搭載したノード，大容量メモリを搭載したノード，ワークフローアプリケーション用のノードで構成される．「富岳」とプリポスト環境のログインノードは共通であるが，「富岳」のジョブスケジューラは Fujitsu TCS（Fujitsu Software Technical Computing Suite）[2] であるのに対し，プリポスト環境のジョブスケジューラは Slurm[3] である．

「富岳」などの HPC クラスタを利用するためには，Shell による CLI（Command Line Interface），SSH の鍵ペアの生成と公開鍵の登録，ジョブスケジューラなどの知識が必要であるため，初心者にとって学習コストが大きいという問題点がある．また，近年では，対話的操作を伴う GUI（Graphical User Interface）アプリケーションを HPC アプリケーションとして動作させることを望まれているが，そのようなアプリケーションを HPC クラスタの計算ノード上で動作させる手順は煩雑である．

以上の問題点を解決するため， 我々は HPC クラスタ用の Web ポータル Open OnDemand[4] を「富岳」で運用している．Open OnDemand を用いると，SSH の代わりに Web ブラウザから HPC クラスタの計算資源を利用できる．さらに，HPC クラスタの計算ノード上で動作する対話的アプリケーションも Web ブラウザから簡易に操作できる．Open OnDemand は Slurm などのジョブスケジューラに対応しているが，Fujitsu TCS には未対応であった．そこで，我々の過去の研究において，Fujitsu TCS を利用するために Open OnDemand の拡張を行った [5]．その拡張は GitHub で管理されている Open OnDemand のリポジトリ（`https://github.com/OSC/ood_core`）にマージされているため，Fujitsu TCS を利用している他の HPC クラスタにおいても Open OnDemand が利用可能である．

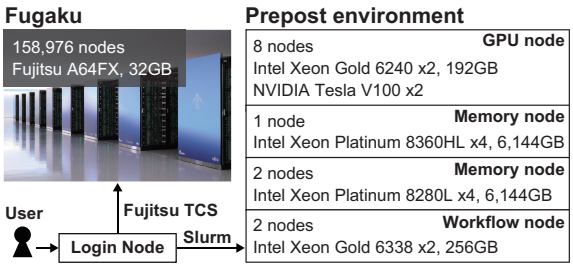


図 1: Overview of Fugaku and pre-post environment

我々は「富岳」において Open OnDemand を運用する上で，ユーザに対する利便性をさらに向上させる様々な工夫を行っている．その一環として，Open OnDemand と HPCI（High Performance Computing Infrastructure）共用ストレージ [6] および GakuNin RDM（Research Data Management）[7] との間でデータ共有を行うことができ

[1] 理化学研究所 計算科学研究センター 兵庫県神戸市中央区港島南町 7-1-26
[2] 国立情報学研究所 東京都千代田区一ツ橋 2-1-2
[3] Lifematics 株式会社 東京都千代田区神田神保町 3-5
[a)] masahiro.nakao@riken.jp

るアプリケーションを開発した．HPCI 共用ストレージと GakuNin RDM は，研究データを管理・共有するための研究データ管理サービスである．これらのアプリケーションを用いると，「富岳」から HPCI 共用ストレージもしくは GakuNin RDM に対するデータ転送を Open OnDemand 上で行うことができる．本稿では，我々が「富岳」の Open OnDemand（Fugaku OnDemand）に対して行っている工夫およびそのデータ共有アプリケーションの開発について述べる．

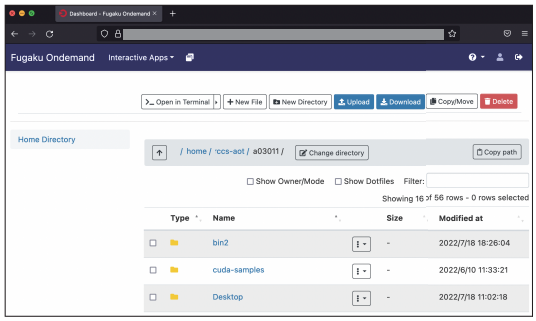
本稿の構成は次の通りである．2 章では Open OnDemand の概要について，3 章では HPCI 共用ストレージと GakuNin RDM の概要について述べる．4 章では Fugaku OnDemand に対して行っている工夫とデータ共有アプリケーションについて述べる．5 章ではデータ共有アプリケーションの性能について評価する．6 章では本稿のまとめと今後の課題について述べる．

2. Open OnDemand

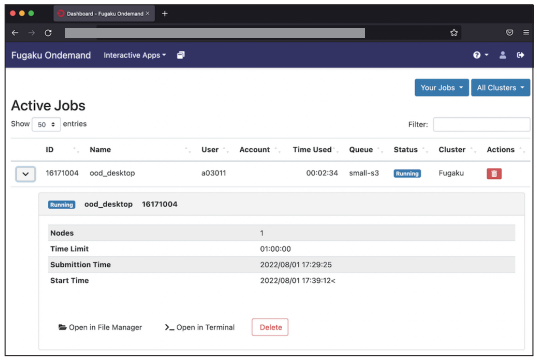
Open OnDemand は HPC クラスタの計算資源を簡易に利用可能にするためのオープンソースソフトウェアである [4]．Open OnDemand の目的の 1 つは，HPC クラスタの利用のための学習コストを小さくすることである．Settlage らは，ユーザの最初のログインからジョブを投入するまでの時間の中央値が従来の SSH を用いた方法では約 22 時間であるのに対し，Open OnDemand を用いた方法では約 2 時間であることを示している [8]．

Open OnDemand で利用可能なアプリケーションは，(A) Open OnDemand がインストールされたサーバで動作するアプリケーションと (B) HPC クラスタの計算ノードで動作するアプリケーションに分けることができる．それぞれについて紹介する．

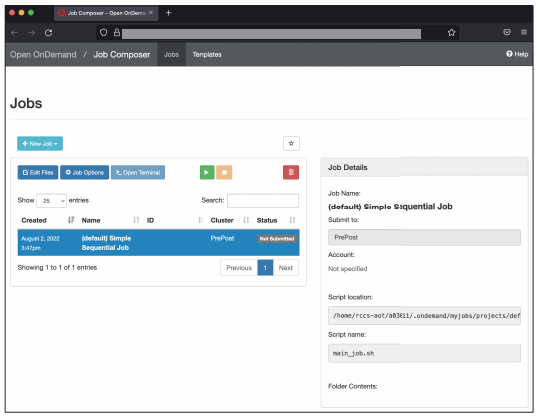
(A) について，Open OnDemand にプリインストールされているアプリケーションを**図 2** に示す．**図 2a** の Home Directory はファイルのアップロード・ダウンロード・編集を行う．**図 2b** の Active Jobs はジョブの監視を行う．**図 2c** の Job Composer はジョブの作成と投入を行う．**図 2d** の Shell は Web ベースのターミナルを提供する．なお，Open OnDemand が提供するフレームワークを用いることで，新しいアプリケーションの開発・導入も簡易に行うことができる [9]．4.5 節で述べる HPCI 共用ストレージと GakuNin RDM に対するデータ共有アプリケーションの開発は，そのフレームワークを用いている．Home Directory において，クラウドストレージ上のファイル管理を行うソフトウェアである rclone[10] を用いることで，Open OnDemand から Amazon S3 などのクラウドストレージとデータ転送を行うことも可能である．なお，R-CCS を含む日本の研究機関の多くは，国立情報学研究所（NII：National Institute of Informatics）が運営する学術情報ネットワーク SINET に



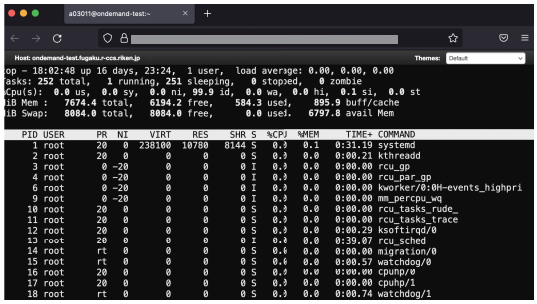
(a) Home Directory



(b) Active Jobs



(c) Job Composer



(d) Shell

図 2: Applications pre-installed in Open OnDemand

接続しており，SINET が提供するサービスの 1 つである SINET クラウド接続サービスを用いることで，自機関とクラウドストレージとの間で高速データ通信を行うことができる．ただし，rclone は HPCI 共用ストレージと GakuNin RDM には対応していない．

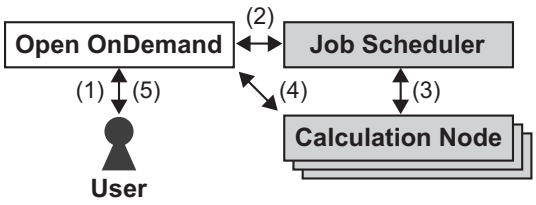


図 3: Operation flow of interactive application

(B) では，Open OnDemand は HPC クラスタのジョブスケジューラと連携し，計算ノードにインストールされたアプリケーションを呼び出す．そのアプリケーションは対話的アプリケーションが想定されているが，通常のバッチジョブも可能である．対話的アプリケーションの場合の動作フローを**図 3** と次に示す．(1) ユーザは Open OnDemand 上でアプリケーションの実行命令を発行する．(2) Open OnDemand はジョブスケジューラにジョブを登録する．(3) ジョブが計算ノード上で実行する．(4) Open OnDemand は確保された計算ノードの情報や GUI アプリケーションが利用しているポート番号などを受け取り，その計算ノードに接続するためのリバースプロキシの設定を行う．(5) ユーザはリバースプロキシの URL を用いて，Web ブラウザから HPC クラスタ内部の計算ノードに接続する．

3. 研究データ管理サービス

3.1 HPCI 共用ストレージ

HPCI とは日本の研究組織の計算資源を SINET で繋いだ共用計算環境基盤であり，R-CCS が運用する「富岳」も HPCI の一部である．そして，HPCI 共用ストレージとは，地理的に分散している HPCI の各組織で研究データを高速かつセキュアに共有するための大規模データ共有基盤である．HPCI 共用ストレージは R-CCS（兵庫県神戸市）と東京大学情報基盤センター（千葉県柏市）が共同で運用しており，各拠点にファイルサーバが設置されている．HPCI 共用ストレージのファイルシステムには Gfarm[11] が採用されており，各拠点にファイル複製を 1 個ずつ配置する完全ミラー化が行われている．

HPCI 共用ストレージへの接続方法について説明する．(1) HPCI 共用ストレージへは GSI（Grid Security Infrastructure）認証を用いてアクセスするため，HPCI 証明書発行システム（`https://portal.hpci.nii.ac.jp`）で電子証明書と代理証明書を発行する．(2) Gfarm クライアントがインストールされた環境にログインし，`myproxy-logon` コマンドで代理証明書のダウンロードを行う．その際は，代理証明書の発行時に入力したパスフレーズが必要である．(3) `mount.hpci` コマンドを実行すると，その内部で `fusermount` が利用されることで，HPCI 共用ストレージにマウントできる．コマンド例は次の通りである．

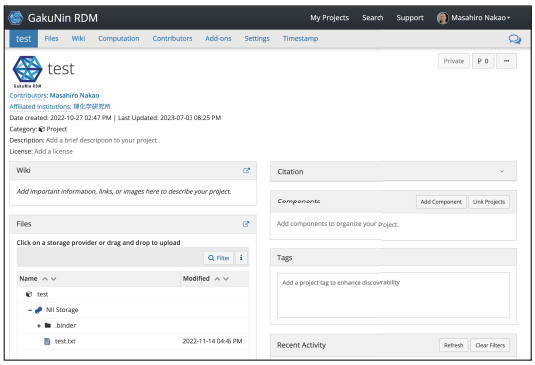


図 4: Project page in GakuNin RDM

```
$ myproxy-logon -s portal.hpci.nii.ac.jp ...
Enter MyProxy pass phrase:
$ mount.hpci
```

3.2 GakuNin RDM

GakuNin RDM は研究データや関連資料を保存または共同研究者と共有するための研究データ管理サービスであり，NII が開発と運営を行っている．GakuNin RDM は米国の非営利団体 Center for Open Science が提供している Open Science Framework をベースとしている．GakuNin RDM は Open OnDemand と同様に Web ブラウザを介したデータ転送やデータ分析を行うためのアプリケーション（JupyterLab など）との連携機能を提供している．

GakuNin RDM の利用方法を説明する．(1) ユーザは GakuNin RDM 上で研究プロジェクトを作成する．(2) 共同研究者の登録を行う．(3) 研究プロジェクト毎に確保されるストレージに対して，研究データの保存・共有・バージョン管理などを行う．GakuNin RDM のプロジェクトのページを**図 4** に示す．Web ブラウザ以外で GakuNin RDM のプロジェクト毎のストレージに接続するには，NII が公開しているスクリプト（`https://github.com/RCOSDP/CS-rdmfs`）を用いてマウントする方法がある．このスクリプトには Python3 用の FUSE (Filesystem in Userspace) ライブラリである pyfuse3 が利用されている．マウントの際は，GakuNin RDM の自身のプロジェクトのページから入手できるプロジェクト ID とパーソナルアクセストークンが必要である．

4. 「富岳」における Open OnDemand の運用

4.1 概要

本章では，Fugaku OnDemand の設定と工夫点について述べる．本章で説明するアプリケーションの設定などは `https://github.com/RIKEN-RCCS/ondemand_fugaku` で公開している．

表 1: Interactive Applications in Fugaku Open OnDemand

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

表 2: Applications for Batch Job in Fugaku Open OnDemand

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

4.2 利用可能なアプリケーション

Fugaku OnDemand では，2 章で述べた (B) HPC クラスタの計算ノードで動作するアプリケーションを「対話的アプリケーション」と「バッチジョブ」に分けている．それぞれで利用可能なアプリケーションを**表 1** と**表 2** に示す．表中でアスタリスク (*) がついてあるアプリケーションは商用であることを意味している．

表 1 の対話的アプリケーションについては，商用アプリケーション以外の多くは「富岳」とプリポスト環境にインストールされていない．そこで，HPC 向けコンテナ環境である SingularityPro[12] を用いて，該当のアプリケーションをインストールしたコンテナイメージを Open OnDemand から呼び出している．図 1 に示した「富岳」の CPU は ARM アーキテクチャに基づく Fujitsu A64FX[13] であるのに対し，プリポスト環境の CPU は一般的な x86_64 アーキテクチャであるため，アーキテクチャ毎のコンテナイメージを用意している．プリポスト環境のコンテナイメージには，表 1 の商用アプリケーション以外のすべてのアプリケーションをインストールしている．「富岳」のコンテナイメージには，管理コストの削減のため，「富岳」において利用頻度が高いと考えられるソフトウェアのみをインストールしている．具体的には開発環境である Desktop, Jupyter, RStudio, VSCode と，複数プロセスを用いた並列処理が可能なビューアーである Paraview のみをインストールしている．

表 2 のバッチジョブのアプリケーションについては，「富岳」ではパッケージ管理システム spack[14] を用いた管理が行われている．そのため，Fugaku OnDemand から `spack load` コマンドを用いて該当のアプリケーションを呼び出している．

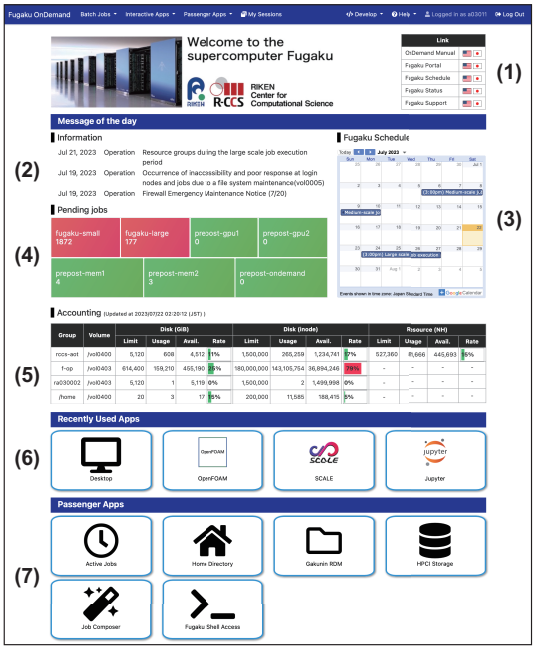


図 5: Dashboard on Fugaku OnDemand

4.3 ダッシュボード

ユーザが Open OnDemand にログインした時に最初に現れるページをダッシュボードと呼ぶ．Fugaku OnDemand のダッシュボードの画面を**図 5** に示す．管理者が直接 eRuby で記述することにより，ダッシュボードに表示する内容を自由に編集することができる．そこで，ユーザの利便性を向上させるため，「富岳」を利用する上で重要な次の情報を表示するようにしている．(1) マニュアルなどへの外部リンク，(2) 障害情報やオペレーション情報，(3) 全系利用やメンテナンスなどのカレンダー（Google カレンダーを利用），(4) ジョブスケジューラの各キューの待ちジョブ数（Grafana を利用），(5) ユーザのディスクとバジェット

表 3: Maximum number of items in Fugaku queue

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

表 4: Maximum number of items in pre-post queue

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	-

の利用率．ここで，ユーザのディスクとバジェットの利用率を得るためには Fujitsu TCS のコマンドを複数回実行する必要があり，すべての情報を取得するには数秒程度必要である．ダッシュボードを表示させる度に数秒の遅延が発生することを避けるため，cron を用いて 1 日 1 回の頻度で全ユーザの情報を保存し，その情報をダッシュボードから読み込んでいる．

一度利用したアプリケーションは再度利用されることが多いと考えられる．一度利用したアプリケーションをユーザがすぐに起動できるように，(6) Recently Used Apps に最近利用したアプリケーションを表示させている．これは Open OnDemand のデフォルトの機能であるが，デフォルトのままだと，(6)Recently Used Apps のアイコンをクリックすると，前回実行したパラメータを用いたジョブの投入まで自動的に行われてしまう．そこで，毎回パラメータを入力するように，デフォルトの設定が定義されている `/var/www/ood/apps/sys/dashboard/app/views/widgets/_recently_used_apps.html.erb` を編集した．(7) は 2 章で説明した (A)Open OnDemand がインストールされたサーバで動作するアプリケーションである．

4.4 Web フォーム

「富岳」とプリポスト環境におけるジョブスケジューラの各キューとユーザが指定できる主要な設定項目の最大値を**表 3** と**表 4** に示す．プリポスト環境では 1 ノードのみ利用可能であるが oversubscribe（1 ノードに複数のジョブを同時に実行できること）も可能であるため，ユーザは CPU コア数，メモリ量，GPU 数が指定可能である．`prepost-ondemand` キューはワークフローのためのキューであるため，長い時間を設定可能である．

表 1 と表 2 に示したアプリケーションの実行方法について説明する．ユーザは図 5 のダッシュボードの最上部にあるナビゲーションバーもしくは (6)Recently Used Apps から実行したいアプリケーションを選択すると，**図 6** のような `fugaku-small` キュー用の Web フォームが表示される．ここで，キューの項目を「prepost-gpu1」にすると，**図 7** のような `prepost-gpu1` キュー用の Web フォームに切り替わる．各 Web フォームの入力項目とその最大値は表 4 で示した通りに設定される．「Launch」をクリックすると，指定したキューにジョブが投入される．対話的アプリケーションの場合，そのジョブが計算ノード上で実行されると，**図 8** のようなその計算ノードに接続するためのリンクがユーザに表示される．バッチジョブの場合は，ジョブが実行中というメッセージのみが表示される．

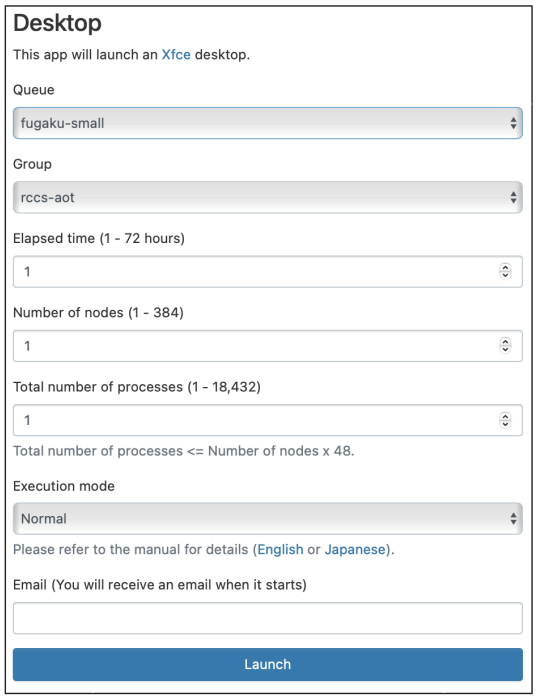


図 6: Web form for fugaku-small queue

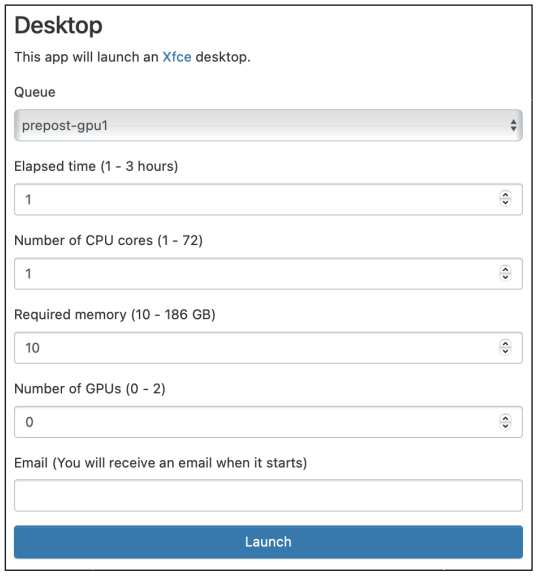


図 7: Web form for prepost-gpu1 queue

図 8: Link to connect to compute nodes

```
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
```

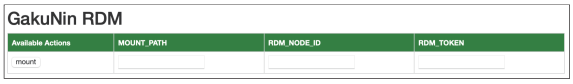
図 9: A part of setting for small queue in Fugaku

「富岳」では 1 人のユーザが複数のグループに属していることがある．バジェットはグループ毎に設定されているため，図 6 のグループの入力項目において，どのグループのバジェットを利用するのかを選択する．なお，プリポスト環境はバジェットを消費しないため，グループの選択肢はない．また，表 3 では省略しているが，「富岳」ではバジェットの使用率が 95%を越えたグループのみが利用できる，バジェットは消費しないが低優先度の特別なキューがある．その場合，図 6 のキューの入力項目に特別なキューも表示させるようにしている．このように，ユーザによって異なる Web フォームを表示させるために，Open OnDemand は Web フォームを動的に生成する仕組みを持っている．この機能の問題点として，ダッシュボードを表示させるときに，すべてのアプリケーションの Web フォームが生成されるため，ダッシュボードの表示が遅くなるという点が挙げられる[*1]．そこで，4.3 節と同様に，cron を使ってユーザ毎やグループ毎の情報を事前に取得しておくことで，Web フォームの動的生成を高速化する工夫を行っている．

Web フォームは YAML 形式の設定ファイルで定義する．`fugaku-small` キューの設定の一部を**図 9** に示す．このような設定ファイルをアプリケーション毎に用意する必要があるが，すべてのアプリケーションの設定項目はほぼ同じである．そこで，コードの管理を簡易化するため，eRuby を用いた設定ファイルの自動生成を行っている．具体的には，図 9 をそのまま出力する関数を定義し，すべてのアプリケーションの Web フォームの設定ファイルからその関数を呼び出している．この工夫を行うことで，1 つのアプリケーションあたりの設定ファイルの行数を 200〜300 から 20〜30 に短縮できた．また，本稿では省略するが，ジョブスケジューラからアプリケーションを起動させるシェルスクリプトにも同様の工夫を行っている．

[*1] `https://osc.github.io/ood-documentation/latest/issues/overview.html#dashboard-may-be-slow-due-to-logic-in-erb-templates`

(a) HPCI Shared Storage



(b) GakuNin RDM

図 10: Data communication applications

4.5 データ共有アプリケーション

我々が開発した HPCI 共有ストレージと GakuNin RDM とのデータ共有アプリケーションの画面を**図 10** に示す．**図 10a** において，HPCI 共有ストレージの入力項目は，HPCI ID，マウントする時間の長さ，代理証明書を発行したときに設定したパスフレーズである．「mount」ボタンをクリックすると，3.1 節で説明した `myproxy-logon` コマンドと `mount.hpci` コマンドが実行される．マウントされるローカルのパスは `mount.hpci` コマンドが自動決定し，そのパスが右端に出力される．そのパスは図 2a の Home Directory アプリケーションの起動リンクにしており，そのリンクをクリックすると該当のディレクトリが Home Directory アプリケーション上で開くようにしている．**図 10b** において，GakuNin RDM の入力項目は，マウントされるローカルのパス，プロジェクト ID，パーソナルアクセストークンである．「mount」ボタンをクリックすると，3.2 節で説明したスクリプトによるマウントが行われる．その後は，HPCI 共有ストレージと同様である．

本アプリケーションを用いると，「富岳」から HPCI 共有ストレージもしくは GakuNin RDM へのデータ転送を図 2a の Home Directory アプリケーションから行うことが可能になる．3 章で述べたようなコマンドラインによる煩雑な手順を省略できるため，ユーザは研究データをより簡易に管理できると考えられる．

5. データ共有アプリケーションの性能評価

本章では 4.5 節で説明した Open OnDemand 上で動作するデータ共有アプリケーションにおいて，Open OnDemand を用いる場合と用いない場合における「富岳」と外部ストレージとの間のデータ転送速度を比較することで，Open OnDemand のオーバヘッドを明らかにする．オーバヘッドの大きさは，外部ストレージに HPCI 共用ストレージと GakuNin RDM のどちらを用いても同じであると考えられるため，本実験では Fugaku OnDemand が動いているサー

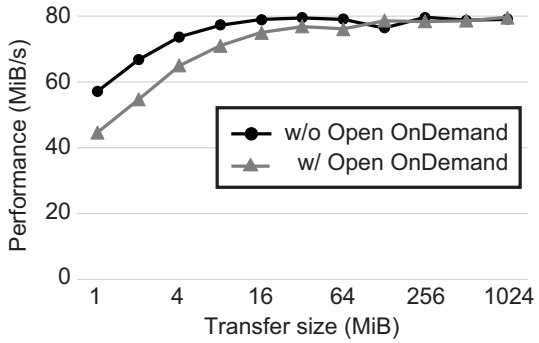


図 11: Data transfer speed evaluation

バ（Fugaku OnDemand Server）と同じ R-CCS 内にある HPCI 共用ストレージのみを用いる．

Open OnDemand を利用した場合の測定時間には Open OnDemand のログに出力される各命令の経過時間を用いる．なお，Home Directory アプリケーションで行われるファイルの転送には，システムの cp コマンドが用いられている．Open OnDemand を用いない場合では，SSH を使って Open OnDemand サーバに直接ログインし，cp コマンドを用いて同様の転送を行う．3.1 節で述べた通り，HPCI 共用ストレージは 2 つの拠点に分かれているため，本実験では R-CCS 内にあるファイルサーバにデータが転送されるように設定を行った．Fugaku OnDemand Server と「富岳」のファイルシステムとは InfiniBand EDR（100Gbps）で接続されており，HPCI 共用ストレージとは 100GbE で接続されている．Fugaku OnDemand Server のスペックは，図 1 中の「Workflow node」と同じである．実験に利用した Open OnDemand，Gfarm クライアント，gfarm2fs のバージョンは，それぞれ 3.0.1，2.7.24，1.2.17 である．

様々なサイズのファイルを「富岳」から HPCI 共用ストレージに対してデータ転送を行い，性能を評価した．10 試行中の最良値を**図 11** に示す．この結果より，小さいデータサイズではオーバヘッドの影響で Open OnDemand を用いない方が最大 28%早いが，データサイズが大きくなるにつれ性能差がなくなることがわかる．ここで，外部ストレージにデータ転送を行う場合は，複数の小さなファイルを 1 つの大きなデータに圧縮して送ることが多いため，Open OnDemand のオーバヘッドは実用上は問題ではないと考えられる．

6. まとめと今後の課題

本稿では，HPC クラスタが持つ計算資源を簡易に利用可能にする Open OnDemand を「富岳」に導入し，その運用のための工夫について述べた．具体的には，様々な対話的アプリケーションとバッチジョブのためのアプリケーションのインストールと SingularityPro の利用，ダッシュボードにおいてユーザにとって有益な情報の表示と高速化，Web フォームの動的生成および設定の簡易化，HPCI 共用ストレージと GakuNin RDM に対するデータ転送アプリケーションの開発などである．また，データ共有アプリケーションの性能評価を行い，Open OnDemand のオーバヘッドは十分に小さいことを確認した．

今後の課題は次の通りである．(1) HPCI 共用ストレージに複数のデータを転送する場合，gfpcopy などを用いた並列転送により短時間でデータ転送を完了させることができる．そこで，Open OnDemand の Home Directory アプリケーションにおいても，並列転送を選択できるオプションを追加する．(2) Fugaku OnDemand の導入の効果を定量的に明らかにするために，ユーザのログイン数やジョブの投入数などの統計情報を取得し，従来のユーザと Open OnDemand を利用するユーザとを比較する．(3) Open OnDemand を用いると，HPC クラスタに共通のフロントエンドを構築できるため，システム開発コストの低減とユーザに対する統一されたインタフェースの提供が可能になる．そのため，「富岳」における Open OnDemand の経験を今後も公開していくことで，その普及を促進する．

参考文献

[1] Mitsuhisa Sato et al. Co-Design for A64FX Manycore Processor and ”Fugaku”. In *International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 651–665, Los Alamitos, CA, USA, 2020. IEEE Computer Society.
[2] Fujitsu. Fujitsu software technical computing suite (in japanese), 2018. https://www.fujitsu.com/downloads/JP/jsuper/tcs-v4-datasheet.pdf.
[3] Slurm. https://slurm.schedmd.com/.
[4] Dave Hudak et al. Open ondemand: A web-based client portal for hpc centers. *Journal of Open Source Software*, Vol. 3, No. 25, p. 622, 2018.
[5] 中尾昌広 他. スーパーコンピュータ「富岳」における hpc クラスタ用 web ポータル open ondemand の導入. 研究報告ハイパフォーマンスコンピューティング（HPC）, No. 2022-HPC-186, 2022.
[6] HPCI 共用ストレージ. https://www.hpci-office.jp/info/display/cnt00011.
[7] GakuNin RDM. https://rdm.nii.ac.jp.
[8] Settlage, Robert et al. Open ondemand: Hpc for everyone. In *High Performance Computing: ISC High Performance 2019 International Workshops, Frankfurt, Germany, June 16-20, 2019, Revised Selected Papers*, pp. 504–513, 2019.
[9] Huan Chen and Chris Fietkiewicz. Version control graphical interface for open ondemand. In *Proceedings of the Practice and Experience on Advanced Research Computing*, PEARC ’18, 2018.
[10] Rclone Website. https://rclone.org.
[11] Osamu Tatebe, Kohei Hiraga, Noriyuki Soda. Gfarm Grid File System. *New Generation Computing, Ohmsha, Ltd. and Springer*, Vol. 28, No. 3, pp. 257–275, 2010.
[12] Sylabs. https://sylabs.io/.
[13] A64fx microarchitecture manual. https://github.com/fujitsu/A64FX/.
[14] Spack Website. https://spack.io.