HPCクラスタにおけるバッチジョブ投入のためのWebアプリケーションOpen Composerの開発

中尾 昌広[1,a)]　山本 啓二[1]

概要：HPC クラスタを利用するには，Linux コマンドやジョブスケジューラなどの前提知識が多いため，初心者にとって学習コストが大きいという課題がある．そこで，我々は HPC クラスタの主な利用用途であるバッチジョブの投入を簡易に行うための Web アプリケーション Open Composer の開発を行った．Open Composer は Web フォームを用いたジョブスクリプトの自動生成機能やジョブの投入などの機能を持つ．本論文では，Open Composer の設計，開発，利用方法などについて述べる．

1. 背景

HPC クラスタは幅広い分野で利用されているが，その利用には高い学習コストが伴うという課題がある．具体的には，SSH クライアントのインストール，鍵ペアの生成，公開鍵の登録といった準備に加え，CLI（Command Line Interface）による Linux コマンドやジョブスケジューラなどの知識が必要である．また，リモートデスクトップや JupyterLab などの GUI（Graphical User Interface）アプリケーションを計算ノード上で対話的に利用したい場合，ポートフォワーディングなどの設定も必要である．これらの知識や設定は HPC クラスタごとに異なるため，新しいクラスタを利用するたびに再度学習が必要である．

この課題を解決するため，Ohio Supercomputer Center は HPC クラスタを操作するための Web ポータル Open OnDemand を開発している [1]．Open OnDemand を用いると，ファイル管理，データ転送，ジョブの投入と管理，ターミナル接続，GUI アプリケーションを対話的に利用するための設定などを，ユーザが使い慣れた Web ブラウザから実行できる．Open OnDemand を用いることで，学習コストの削減と煩雑な準備や設定の省略を行うことができ，さらに統一されたインタフェースで様々な HPC クラスタを利用することができる．また，Open OnDemand はアプリケーションプラットフォームとしての機能も持っており，Open OnDemand には複数の便利な Web アプリケーションがプリインストールされている．また，他の Web アプリケーションの追加も行うことができる．

[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
[a)] masahiro.nakao@riken.jp

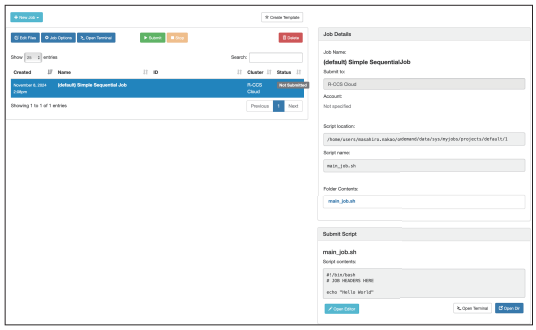

図 1　Job Composer on Open OnDemand

HPC クラスタの主な利用方法は，実アプリケーションをバッチジョブとして実行することである．HPC クラスタにおけるバッチジョブとは，ジョブスケジューラによって非対話的に実行される計算タスクのことを指す．一般的にバッチジョブはスクリプト形式で記述され，必要な計算リソース（ノード数や実行時間など）をジョブスケジューラ用のディレクティブ（例：`#SBATCH -N 4`）を用いて指定する．そして，バッチジョブがジョブスケジューラに投入されると，ジョブスケジューラは HPC クラスタの空き状況から判断した適切なノードを確保し，そのバッチジョブを実行する．Open OnDemand からバッチジョブをジョブスケジューラに投入したい場合，Open OnDemand にプリインストールされている Job Composer を利用できる（**図 1**）．Job Composer は Web ブラウザからジョブの投入と作成が可能であるが，ジョブスクリプトは手動で作成する必要がある．つまり，ユーザはアプリケーションの実行コマンドやジョブスケジューラのディレクティブを記述する必要があるため，その学習コストは大きく，また書き間違いなども発生すると考えられる．

表 1　Real-world applications on Fugaku Open OnDemand

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

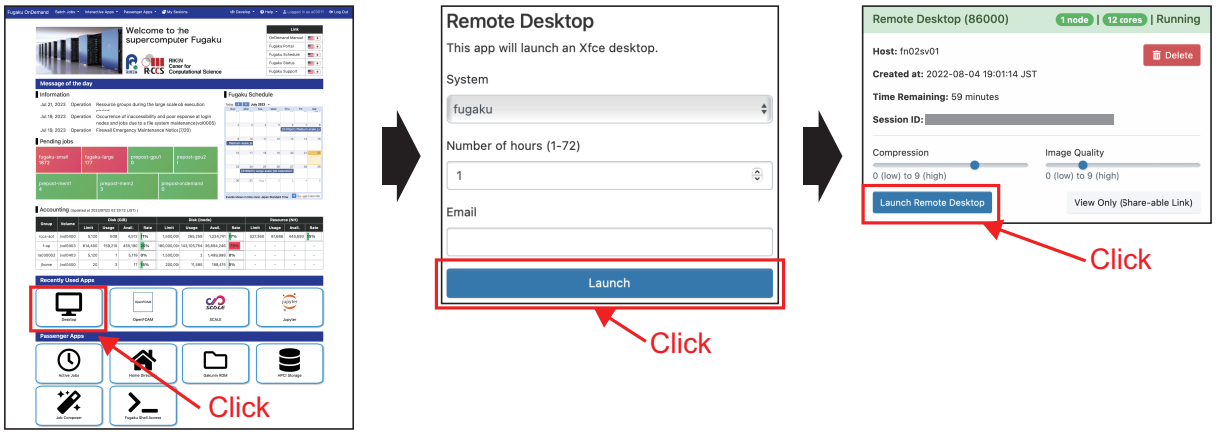

図 2　How to start a GUI application

そこで本稿では，ジョブスクリプトを自動生成できる Web アプリケーション Open Composer の開発を行う．Open Composer では，実アプリケーション毎に用意された Web フォームにパラメータを入力することでジョブスクリプトを生成するため，ジョブスケジューラに対する学習コストを下げ，また書き間違いの頻度を低減できると考えられる．また，Open Composer はジョブ投入や削除などの管理を行う機能も提供している．Open Composer は Open OnDemand 上で動作するアプリケーションであり，Open OnDemand のいくつかの機能と連携させている．

本稿の構成は次の通りである．2 章ではジョブスクリプトの自動生成に関連する研究についてまとめる．3 章と 4 章では，Open Composer の開発と利用方法について述べる．5 章では本稿のまとめを行う．

2. 関連研究

2.1 Open OnDemand の Web フォームの利用

我々は「富岳」[2] 上で Open OnDemand の運用を行っている [3], [4], [5]．「富岳」で実アプリケーションのバッチジョブの投入を簡易に行うために，1 章で述べた Job Composer の代わりに Open OnDemand の GUI アプリケーションを実行する Web フォームを利用している．「富岳」の Open OnDemand で運用している実アプリケーションの一覧を**表 1** に示す．まず，ユーザが Open OnDemand 上で GUI アプリケーションを実行する手順を**図 2** を用いて説明する．「富岳」の Open OnDemand のトップページ（図 2 左）から，実行したい GUI アプリケーションのアイコンをクリックすると，Web フォーム（図 2 中）が表示される．ジョブに対するパラメータを入力した後に「Launch」ボタンをクリックすると，Open OnDemand 内部で自動生成されたジョブスクリプトがジョブスケジューラに投入される．計算ノード上で GUI アプリケーションが実行されると有効になるボタンをクリックすると（図 2 右），Open OnDemand のリバースプロキシの機能により，計算ノード上で動作している GUI アプリケーションがユーザの Web ブラウザに表示される．

我々が Open OnDemand を用いて運用しているバッチジョブの投入手順は，図 2 の (1) と (2) のみを利用している．バッチジョブの Web フォームの例を**図 3** に示す．図 3 で入力された情報と**図 4** に示すジョブスクリプトのテンプレートからジョブスクリプトを生成する．ジョブスクリプトのテンプレートの形式は ERB（Embedded RuBy）が埋め込まれた YAML (YAML Ain't Markup Language) であり，<%= %>の箇所は Web フォームで入力された値に置換される．1〜6 行目はジョブスケジューラに渡すパラメータを定義し，8〜18 行目はジョブスクリプトを定義している．

このようなバッチジョブの投入手順は，アプリケーションの実行コマンドやジョブスケジューラのディレクティブを完全に隠蔽できるため，1 章で述べた Job Composer の課題を解決していると言える．しかし，そのトレードオフ

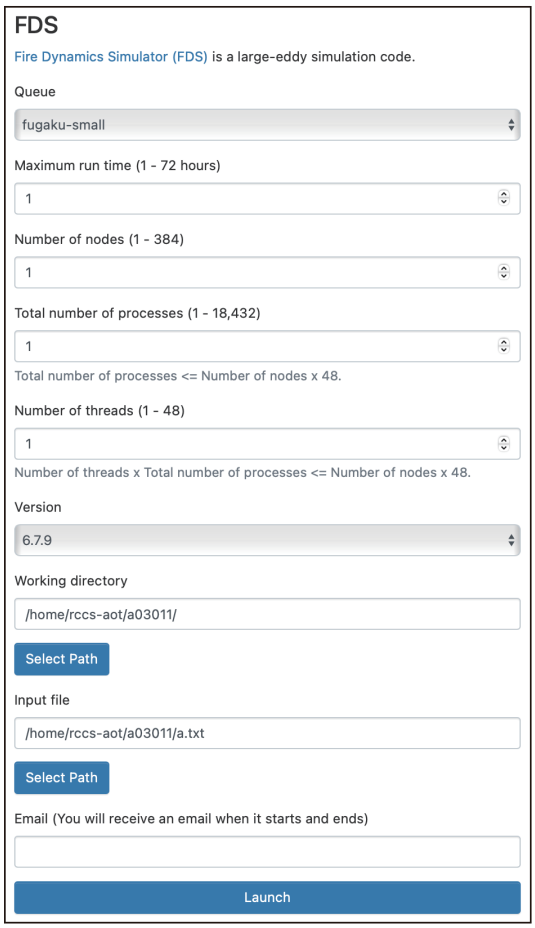

図 3 Example of a web form for a real application

```
1  script:
2    queue_name: <%= queue %>
3    email: <%= email %>
4    native:
5      - -L elapse=<%= hours %>:00:00,node=<%= nodes %>
6      - --mpi proc=<%= procs %>
7
8  batch_connect:
9    template: "basic"
10   script_wrapper: |
11     #!/usr/bin/env bash
12
13     . /vol0004/apps/oss/spack/share/spack/setup-env.sh
14     spack load fds@<%= version %>
15
16     export OMP_NUM_THREADS=<%= threads %>
17     cd <%= working_dir %>
18     mpiexec fds_mpi_fugaku <%= input_file %>
```

図 4 Template of a job script

として，生成されるジョブスクリプトの記述自由度の低さが新たな課題となっている．事前に用意した Web フォームとテンプレートからジョブスクリプトが生成されるため，例えばユーザはアプリケーションに対して任意のコマンドを追加することはできない．

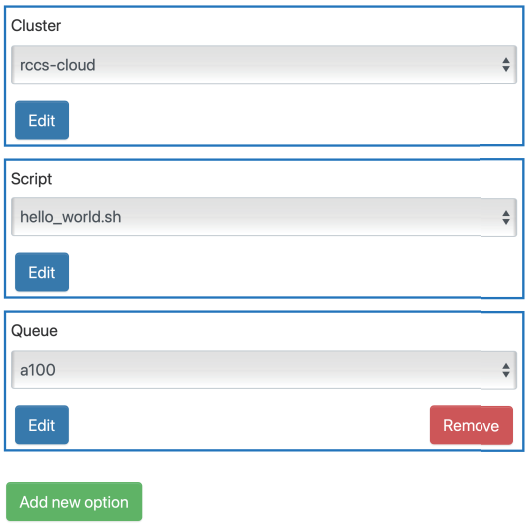

図 5 Project Manager on Open OnDemand

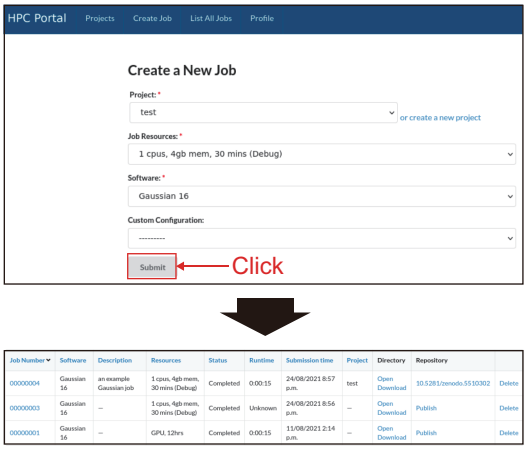

図 6 CHAMP on Open OnDemand[6]

2.2 Open OnDemand 上で動作するアプリケーション

Open OnDemand は Project Manager という Web アプリケーションがベータ版としてプリインストールされている[*1]（**図 5**）．Web フォームからジョブスケジューラの投入コマンドにオプションを渡す機能（ディレクティブの記述と同等の機能）を持つ．しかし，アプリケーションの実行コマンドなどはユーザが手動で記入する必要がある．

CHAMP[6] は Open OnDemand 上で動作する Web アプリケーションであり，FAIR（Findable, Accessible, Interoperable, and Reusabe）原則に沿った研究データの生成と公開を自動的に行う機能を持つ（**図 6**）．CHAMP の特徴として，CHAMP から実行されたジョブの状態を確認できる Web ページを提供しており，その Web ページからジョブの結果をデータリポジトリに公開できる．CHAMP は 2.1 節で述べた方法と同様に Web フォームとテンプレートを用いたジョブの投入を行うため，ジョブスクリプトの記述自由度が低いという課題がある．

[*1] 原稿執筆時点（2024 年 11 月）で最新版の Open OnDemand 3.1.10 ではデフォルトで無効になっている．

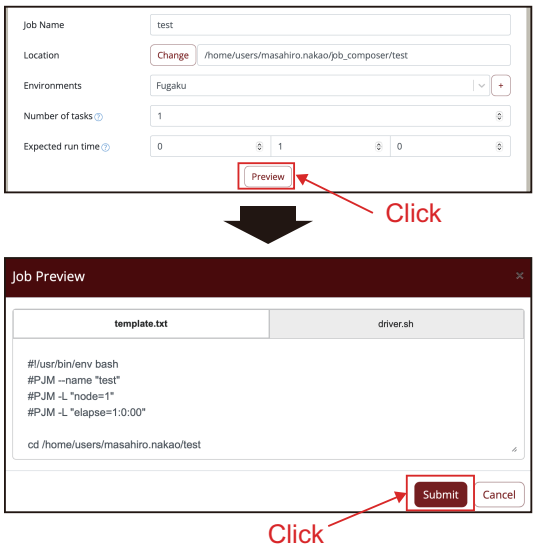

図 7 Drona Composer on Open OnDemand

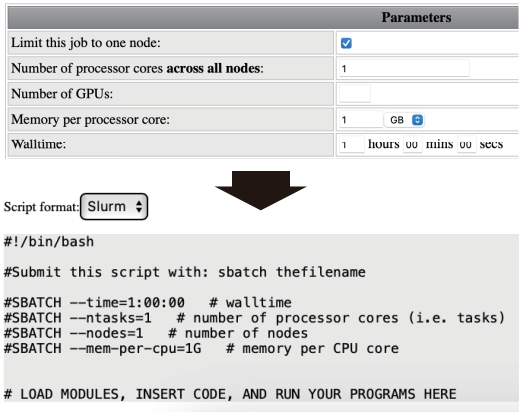

図 8 BYU Job Script Generator

Drona Composer[7] は Open OnDemand 上で動作する Web アプリケーションであり，アプリケーションの実行コマンドとジョブスケジューラのディレクティブを含んだジョブスクリプトを生成できる（**図 7**）．図 7 上でユーザは Web フォームの項目を入力し，「Preview」ボタンをクリックすると，図 7 下が表示され，ジョブスクリプトの確認および編集を行った後，「Submit」ボタンでジョブの投入を行う．図 7 下のタブの「driver.sh」では，ジョブを投入する際の前処理を設定できる．Drona Composer はジョブの状況確認や削除などを行う機能を持たないが，それらは既存の Open OnDemand の機能で代替可能である[*2]．

2.3 Open OnDemand 以外の Web アプリケーション

BYU Job Script Generator[8] は，ジョブスケジューラである Slurm と PBS のディレクティブを生成する Web アプリケーションである（**図 8**）．BYU Job Script Generator の特徴として，Web フォームにパラメータを入力すると，リアルタイムでジョブスクリプトが更新される画面も表示されるため，ジョブスクリプトの確認が行いやすい点が挙げられる．他にも，Slurm に限定しているが同様の Web アプリケーションが多数存在する [9], [10], [11], [12]．しかし，これらの Web サービスはディレクティブ部分を生成するのみであり，アプリケーションの実行コマンドは生成できない．また，ジョブスケジューラと連携する機能も持たないため，ジョブの投入などを行うことはできない．

3. Open Composer の開発

3.1 要件

Open Composer は，ジョブスクリプトの作成に伴う学習コストを削減するとともに，バッチジョブを HPC クラスタに容易に投入可能な環境を提供することにより，初心者から熟練者までの幅広いユーザが HPC クラスタを効率的に活用できることを目指している．そのための要件として，下記が挙げられる．

- ジョブスクリプトの自動生成とエラーチェック機能
- 様々な実アプリケーション用のジョブスクリプトを生成できる柔軟なカスタマイズ性
- ジョブ投入やジョブ管理などを行う機能
- HPC クラスタ間の差異を隠蔽する統一的なインタフェース

3.2 設計

Open OnDemand は Web ブラウザを通じて HPC リソースを利用できる Web アプリケーションのデファクトスタンダードであり，世界中の HPC クラスタで利用されている．そのため，Open Composer は Open OnDemand 上で動作する Web アプリケーションとして作成する．この設計により，Open Composer のデプロイを簡略化でき，さらに Open OnDemand の認証機能やデータ転送機能などをそのまま利用できるため，Open Composer に対する開発コストの低減を図ることができる．

Open Composer は，2.2 節で述べた Drona Composer と同様の下記の機能を持つ．(1) アプリケーション毎に用意された Web フォームからのジョブスクリプトの生成，(2) Web フォームで入力したパラメータに対するエラーチェック，(3) ユーザがジョブスクリプトを直接編集できる機能，(4) ジョブを投入する際の前処理の定義．また，ジョブスクリプトの確認を行いやすくするため，2.3 節で述べた BYU Job Script Generator と同様に，リアルタイムでジョブスクリプトが更新される画面も表示する．さらに，ジョブの投入と管理を行う機能も追加する．ここで，ユーザがジョブスクリプトを作成する手順としては，そのユーザが過去に作成したジョブスクリプトの設定を少し変更して使用す

[*2] Drona Composer の Youtube 動画（`https://www.youtube.com/watch?v=tgpP9LPGlYQ`）では，Open OnDemand 上で動作する他の Web アプリケーションを使ってジョブ管理を行っているが，このアプリケーションが公開されているかどうかは不明．

る場合が多いと考えられる．そのため，投入したジョブのパラメータなどの情報をジョブ管理機能に渡すことにより，ジョブスクリプトを簡易に再利用できる機能も開発する．

3.3 実装

Open OnDemand は Ruby on Rails で作成されているため，Open OnDemand がインストールされた環境では必ず Ruby が利用できる．そのため，Open Composer のバックエンドには Ruby を用いたオープンソースの軽量フレームワーク Sinatra[13] を用いた．フロントエンドにはレスポンシブなサイトを構築するためのツールキットである Bootstrap とジョブスクリプトをリアルタイムで更新するために HTML5 と JavaScript を用いた．

Open Composer では，統一的な操作でジョブ管理を行うため，ジョブスケジューラ間の差異を吸収するアダプタを作成する必要がある．そこでアダプタのためのスーパークラスを定義し，ジョブスケジューラ毎にそのサブクラスを作成する設計にした．スーパークラスで定義した関数は，ジョブを投入するための `submit()`，ジョブを削除するための `cancel()`，ジョブの状態を確認するための `query()` である．現在は，「富岳」で利用されているジョブスケジューラの Fujitsu_TCS と Slurm のアダプタを作成しており，それぞれのアダプタを開発するのに要した Ruby の行数は 79 と 76 である．

Open Composer の実装に要した各言語の行数（2024 年 11 月時点）は，Ruby は 1,269 行，JavaScript は 737 行，ERB を含む HTML5 は 291 行である．Open Composer はオープンソースソフトウェアとして，`https://github.com/RIKEN-RCCS/OpenComposer` で公開する予定である．

4. Open Composer の利用方法

4.1 概要

Open Composer は「トップページ」・「アプリケーションページ」・「履歴ページ」から構成される．本節ではそれぞれの概要について説明する．

トップページを**図 9** に示す．トップページではカテゴリ毎にアプリケーションのアイコンが表示される．そのアイコンをクリックするとアプリケーションページに遷移する．

アプリケーションページを**図 10** に示す．図 10 左の Web フォームでパラメータの入力を行うと，図 10 右のテキストエリアにジョブスクリプトがリアルタイムで生成される．ジョブスクリプトの編集も可能である．図 10 右下の「Submit」ボタンをクリックすると，そのジョブスクリプトがジョブスケジューラに投入される．

履歴ページを**図 11** に示す．履歴ページでは，Open Composer を通じて投入された過去のジョブの履歴の閲覧と実行中のジョブの状態の確認などを行うことができる．また，

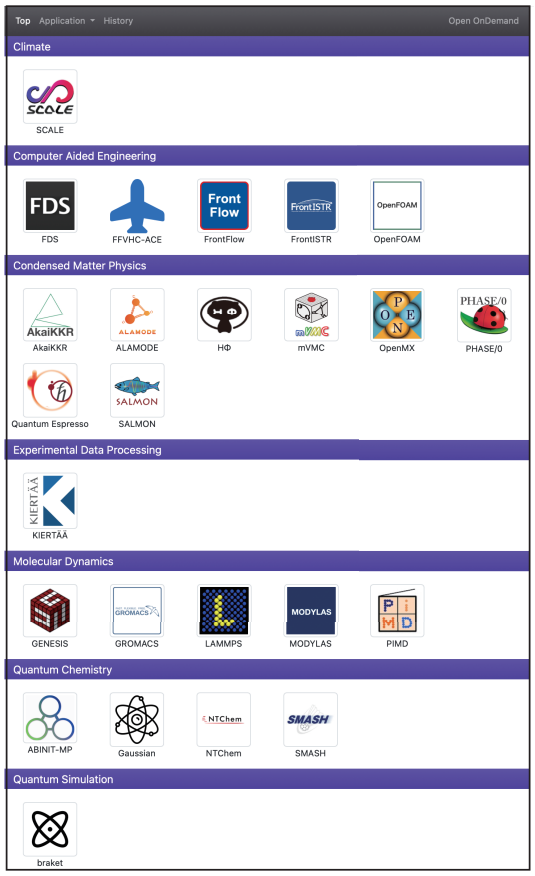

図 9 Top page on Open Composer

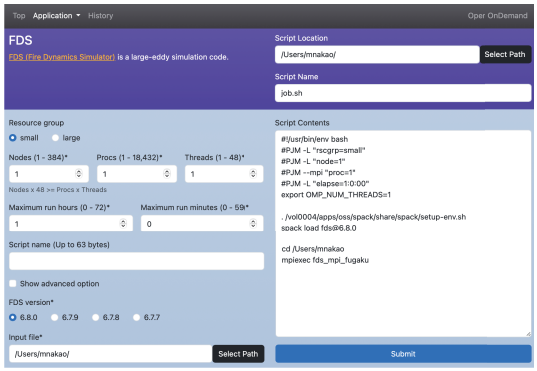

図 10 Application page on Open Composer

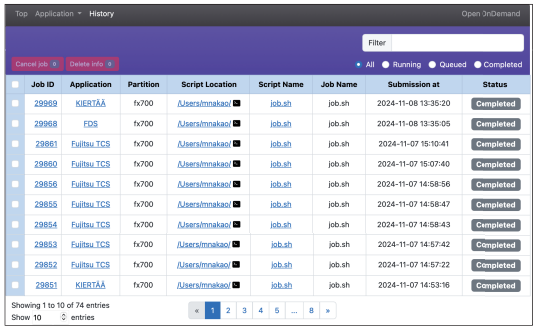

図 11 History page on Open Composer

過去のジョブスクリプトを簡易に再利用できる仕組みも持つ．「Script Name」列の文字列をクリックすると，**図 12**

図 12 Recycling a job script

```
1 name: Test
2 category: Computer Aided Engineering
3 icon: icon.png
4 description: This application is test.
```

図 13 Sample of `manifest.yml`

のようなウィンドウが表示され，「Load Parameters」をクリックすると，そのジョブスクリプトがロードされた状態のアプリケーションページに遷移する．

4.2 アプリケーションの設定例

Open Composer のインストール先（デフォルトでは `/var/www/ood/apps/sys/OpenComposer`）にアプリケーション毎のディレクトリを作成し，その中に YAML 形式の設定ファイル `manifest.yml` と `form.yml` を作成する．それぞれに ERB を含めたい場合は，ファイル名を `manifest.yml.erb` と `form.yml.erb` にする．

`manifest.yml` にはアプリケーションの説明を記述する．`manifest.yml` の例を**図 13** に示す．`name` はアプリケーション名，`category` はアプリケーションのカテゴリ，`icon` はアイコン画像へのパスであり，それらはトップページの表示に用いる．4 行目の `description` はアプリケーションページの左上でアプリケーションの説明を表示するために用いる．

`form.yml` は `form`，`script`，`check` というブロックで構成されており，ジョブスクリプトの生成に用いる．`form` ブロックはジョブ情報を入力するためのウィジットの定義，`script` ブロックはジョブスクリプトのテンプレート，`check` ブロックは生成されたジョブスクリプトのエラーチェックに用いる．Open Composer は下記のウィジットを用意している．これらのウィジットや 4.3 節で述べる Dynamic Form Widget は，Open OnDemand の仕様を参考にしている[*3]．

- `number`：数値を入力する．
- `text`：テキストを入力する．
- `email`：email アドレスを入力する．
- `select`：セレクトボックスから項目を 1 つ選択する．
- `multi_select`：セレクトボックスから項目を複数選択する．

[*3] https://osc.github.io/ood-documentation/latest/how-tos/app-development/interactive/dynamic-form-widgets.html

```
1  form:
2    partition:
3      widget: radio
4      label: Partition
5      value: Small partition
6      direction: horizontal
7      options:
8        - [Small partition, small]
9        - [Large partition, large]
10   nodes:
11     widget: number
12     label: Number of nodes (1 - 128)
13     value: 4
14     min: 1
15     max: 128
16     step: 1
17   time:
18     widget: number
19     size: 2
20     label: [Max time (0-24 h), Max time (0-59 m)]
21     value: [ 1, 0]
22     min: [ 0, 0]
23     max: [72, 59]
24     step: [ 1, 1]
25
26 script: |
27   #!/usr/bin/env bash
28   #SBATCH -p #{partition}
29   #SBATCH -N #{nodes}
30   #SBATCH -t #{time_1}:#{time_2}:00
31    mpiexec ./a.out
32
33 check: |
34   if @time_1.to_i == 24 && @time_2.to_i > 0
35     halt 500, "Exceeded Time"
36   end
```

図 14 Sample of `form.yml`

- `radio`：ラジオボタンから項目を 1 つ選択する．
- `checkbox`：チェックボックスから項目を複数選択する．
- `path`：Open Composer が動作しているサーバ上のファイルやディレクトリのパスを選択する．

`form.yml` と生成されるアプリケーションのページの例を**図 14** と**図 15** に示す．アプリケーションページで Web ブラウザの画面サイズが広い場合は図 10 のように 2 段組みになるが，スマートフォンなどからの利用で画面サイズが狭い場合は図 15 のように 1 段組になる．図 15 上の「Script Location」と「Script Name」は固定された `path` ウィジットと `text` ウィジットであり，それぞれジョブスクリプトの保存場所とジョブスクリプト名を入力する．`path` ウィジットで表示される「Select Path」ボタンをクリックすると，**図 16** のような画面が表示され，ディレクトリやファイルのパスをクリック操作で入力できる．図 14 の 2，10，17 行目はウィジット名を示す．その次の行の `widget` と `label` と `value` は，それぞれウィジットの種類，Web フォームに表示されるラベル，初期値を示す．5 行目の

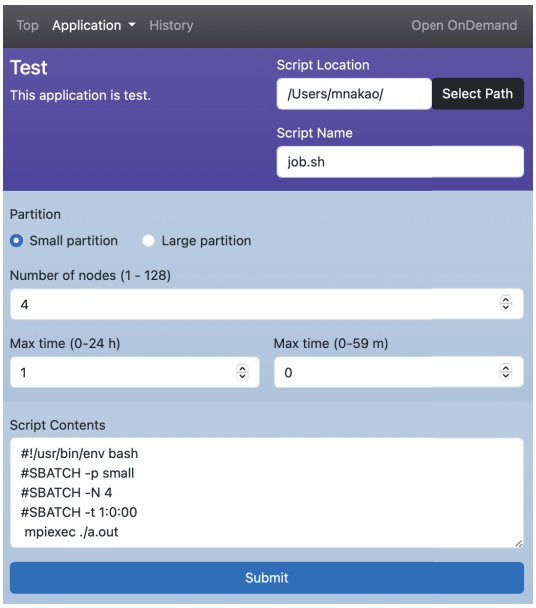

図 15 Sample of application page

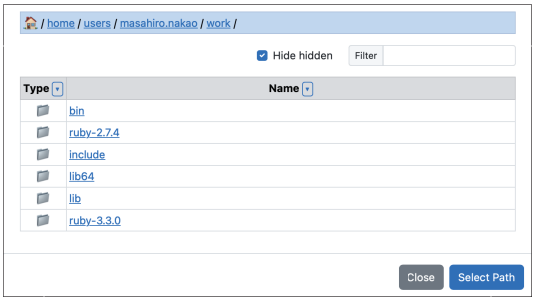

図 16 Path widget on Open Composer

`direction` は，ラジオボタンの選択肢の並ぶ方向を示す．7〜9 行目はラジオボタンの選択肢を示す．1 つ目の要素（例：Small Partition）はラジオボタンに表記されるラベルであり，2 つ目の要素（例：small）は `script` ブロック中で用いられる値である．`number` ウィジットで用いられている `min`，`max`，`step` は，それぞれ最小値，最大値，ステップ幅を示す．19 行目の `size` は入力数を表し，そのウィジットの項目は多要素を持つ配列として記述する．26〜31 行目の `script` ブロックでは，ジョブスクリプトを定義している．このブロック内で，`#{WidgetName}` で記述された箇所は，そのウィジットの値に置換される．`size` に 2 以上の値が用いられている場合，`#{WidgetName_n}` のようにアンダースコアと数字をウィジット名の後に追加して指定する．33〜36 行目の `check` ブロックでは，入力した値をチェックするための Ruby スクリプトを記述する．この例では，`time` のウィジットで 24 時間を超過する値を入力した場合，ジョブスクリプトの投入前にエラーが表示される．

4.3 前処理機能

表 1 に示した実アプリケーションの 1 つである KIERTÄÄ を実行する手順は，まず**図 17** のようなパラメータファイルを記述し，そのパラメータファイルから KIERTÄÄ が提供するスクリプトを用いてジョブスクリプトを生成する．このように Web フォームから直接ジョブスクリプトを生成しないアプリケーションに対応するために，ジョブを投入する前の処理を定義できる機能を開発した．**図 18** のような `submit.yml.erb` をアプリケーションのディレクトリに保存し，このファイルがある場合はその `script` ブロックがジョブの投入前に実行される仕組みである．`submit.yml.erb` 中の`@SCRIPT_LOCATION` と `@SCRIPT_NAME` は，それぞれ図 15 上の「Script Location」と「Script Name」で入力した値に置換される．

4.2 節および 4.3 節の機能を用いることで，表 1 に示したすべての実アプリケーションが Open Composer 上で動作することを確かめた．

```
1 user_id = "mnakao"
2 mode = "jasri_iap_s"
3 filter = ramp
4 zero_padding = 4
```

図 17 Parameter file of KIERTÄÄ

```
1 script: |
2   #!/usr/bin/env bash
3   cd <%= @SCRIPT_LOCATION %>
4   mv <%= @SCRIPT_NAME %> parameters.conf
5   genjs_ct parameters.conf > <%= @SCRIPT_NAME %>
```

図 18 submit.yml.erb

```
1  form:
2    partition:
3      widget: radio
4      label: Partition
5      value: Normal partition
6      options:
7        - [Normal partition, small, disable-cuda]
8        - [GPU partition, gpu]
9    cuda:
10     widget: number
11     label: CUDA version (12, 13, 14)
12     value: 12
13     min: 12
14     max: 14
15     step: 1
```

図 19 Sample of `form.yml` of Dynamic Form Widget

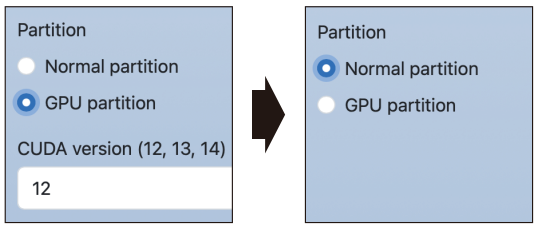

図 20 Sample of application page of Dynamic Form Widget

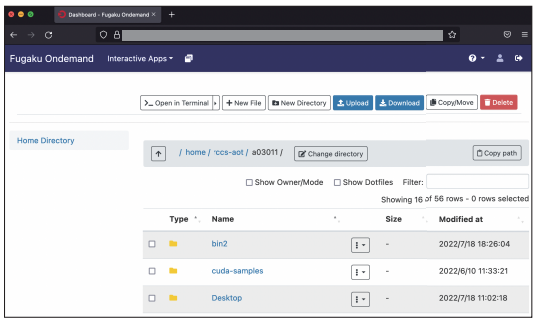

図 21 Home directory on Open OnDemand

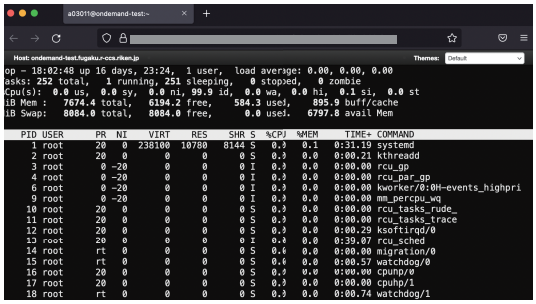

図 22 Shell on Open OnDemand

4.4 Dynamic Form Widget

Open Composer は動的なウィジットを作成するための機能である Dynamic Form Widget を提供している．Dynamic Form Widget は選択肢を持つウィジットにおいて，ある選択肢を選択すると，他のウィジットの値の変更・無効化・非表示化を行う機能である．本節では Dynamic Form Widget の無効化の機能について，**図 19** と**図 20** を用いて説明する．図 19 の 2〜8 行目ではラジオボタンを定義している．7 行目の 3 つ目の要素の `disable-cuda` は `cuda` という名のウィジットを無効化することを意味している．図 20 に示す通り，ラジオボタンで `Normal partiton` を選択すると `cuda` ウィジットが無効化される．

4.5 Open OnDemand との連携

図 11 に示す履歴ページでは，ジョブの結果の確認などを簡易に行うために，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ワンクリックで起動する機能を提供している．あるジョブ ID の行の「Script Location」の列の文字列をクリックすると，Web ブラウザからデータの送受信やサーバ上のファイルの編集を行える `Home directory` アプリケーションが起動する（**図 21**）．また，その文字列の横にあるターミナルのアイコンをクリックすると，ターミナル接続を行える（**図 22**）．これらの機能により，ジョブの出力ファイルに素早くアクセスすることができ，必要な場合は Linux コマンドを実行することができる．

5. まとめ

本稿では，ジョブスクリプトの生成と投入を行うことができる Web アプリケーション Open Composer の開発を行った．Open Composer は，Web フォームと編集可能なテキストエリアの双方を活用することで，高い記述自由度を維持しつつ，ジョブスケジューラの学習コストを低減することを目指している．また，ジョブの管理機能を備えており，ジョブ履歴ページから過去のジョブスクリプトを容易に再利用できる仕組みを提供している．これらの機能により，HPC クラスタ環境におけるジョブ投入作業の煩雑さを解消し，ユーザの生産性向上に寄与すると考えている．

参考文献

[1] 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.

[2] 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.

[3] 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.

[4] 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 の運用. 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-191, 2023.

[5] Masahiro et al Nakao. Introducing open ondemand to supercomputer fugaku. In *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, SC-W '23, p. 720–727, New York, NY, USA, 2023. Association for Computing Machinery.

[6] Christopher Cave-Ayland, Michael Bearpark, Charles Romain, and Henry S. Rzepa. Champ is a hpc access and metadata portal. *Journal of Open Source Software*, Vol. 7, No. 70, p. 3824, 2022.

[7] Texas A&M High Performance Research Computing. Drona Composer. `https://hprc.tamu.edu/kb/User-Guides/Portal/Drona_composer/`.

[8] Brigham Young University. BYU Job Script Generator. `https://byuhpc.github.io/BYUJobScriptGenerator/`.

[9] National Energy Research Scientific Computing Center. Jobscript Generator. `https://my.nersc.gov/script_generator.php`.

[10] University of Virginia. Slurm Script Generator. `https://www.rc.virginia.edu/userinfo/hpc/slurm-script-generator/`.

[11] University of Michigan. Job script generator for Slurm. `https://websites.umich.edu/~greatlakes/jobscript_generator/`.

[12] George Mason University. Slurm Script Generator. `https://wiki.orc.gmu.edu/mkdocs/slurm_generator/slurm_script_generator.html`.

[13] Sinatra. `https://sinatrarb.com`.