

JAXAにおけるOpen OnDemand 導入の狙いと期待

2023/10/11

宇宙航空研究開発機構

宇野 俊司

アジェンダ

- JSS3の紹介
- JSS3における新たな利用形態
- Open OnDemand導入の狙い
- Open OnDemandへの期待
- まとめ

JAXAスーパーコンピュータシステム JSS3

2020年12月1日

【コンピュータ基盤】 TOKI: TOKyo and ibaraKI



調布航空宇宙センター



TOKI-SORA: HPCシステム

PRIMEHPC FX1000
 ノード数: 5,760 ノード (15 ラック)
 総理論演算性能: 19.4 PFLOPS
 総主記憶容量: 180 TiB (32 GiB/ノード)



TOKI-RURI: 汎用システム

総理論演算性能: 1.24 PFLOPS
 総主記憶容量: 104 TiB

- ST:** PRIMERGY RX2540 M5 x 375 ノード (192 GiB/ノード, Quadro x 1 基)
- GP:** PRIMERGY CX2570 M5 x 32 ノード (384 GiB/ノード, Tesla V100 x 4 基)
- XM:** PRIMERGY RX2540 M5 x 2 ノード (DCPMM 6.0 TiB/ノード, Quadro x 1 基)
- LM:** PRIMERGY RX2540 M5 x 7 ノード (DCPMM 1.5 TiB/ノード, Quadro x 1 基)



TOKI-FS: ファイルシステム

ファイルシステム: FEFS
 オールフラッシュ NVMe ストレージ: 10PB
 ハードディスクドライブ ストレージ: 40PB

TOKI-LI: ログインシステム

PRIMERGY RX2540 M5 x 最大14 ノード (384 GiB/ノード, Quadro x 1 基)

運用管理システム

12 Tbps

相互接続網 (InfiniBand)

45.7 Tbps

20.8 Tbps

2.8 Tbps

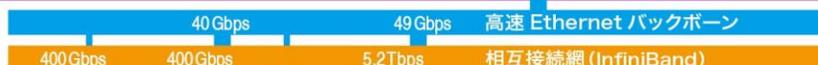
360 Gbps

高速 Ethernet バックボーン

416 Gbps

280 Gbps

筑波宇宙センター



TOKI-TFS: 筑波ファイルシステム

ファイルシステム: FEFS, 総実効容量: 0.4PB

TOKI-TLI: 筑波ログインシステム

PRIMERGY RX2540 M5 x 2 ノード (384 GiB/ノード, Quadro x 1 基)

筑波運用管理制御システム

TOKI-TRURI: 筑波汎用システム

総理論演算性能: 145 TFLOPS
 総主記憶容量: 10.8 TiB

- TST:** PRIMERGY RX2540 M5 x 46 ノード (192 GiB/ノード, Quadro x 1 基)
- TGP:** PRIMERGY CX2570 M5 x 2 ノード (384 GiB/ノード, Tesla V100 x 4 基)
- TLM:** PRIMERGY RX2540 M5 x 1 ノード (DCPMM 1.5 TiB/ノード, Quadro x 1 基)

80 Gbps

【アーカイバ基盤】 J-SPACE

ディスクキャッシュ容量: 3PB
 テープ容量: 70PB



JSPACE

Powered by



*調布航空宇宙センター内に設置

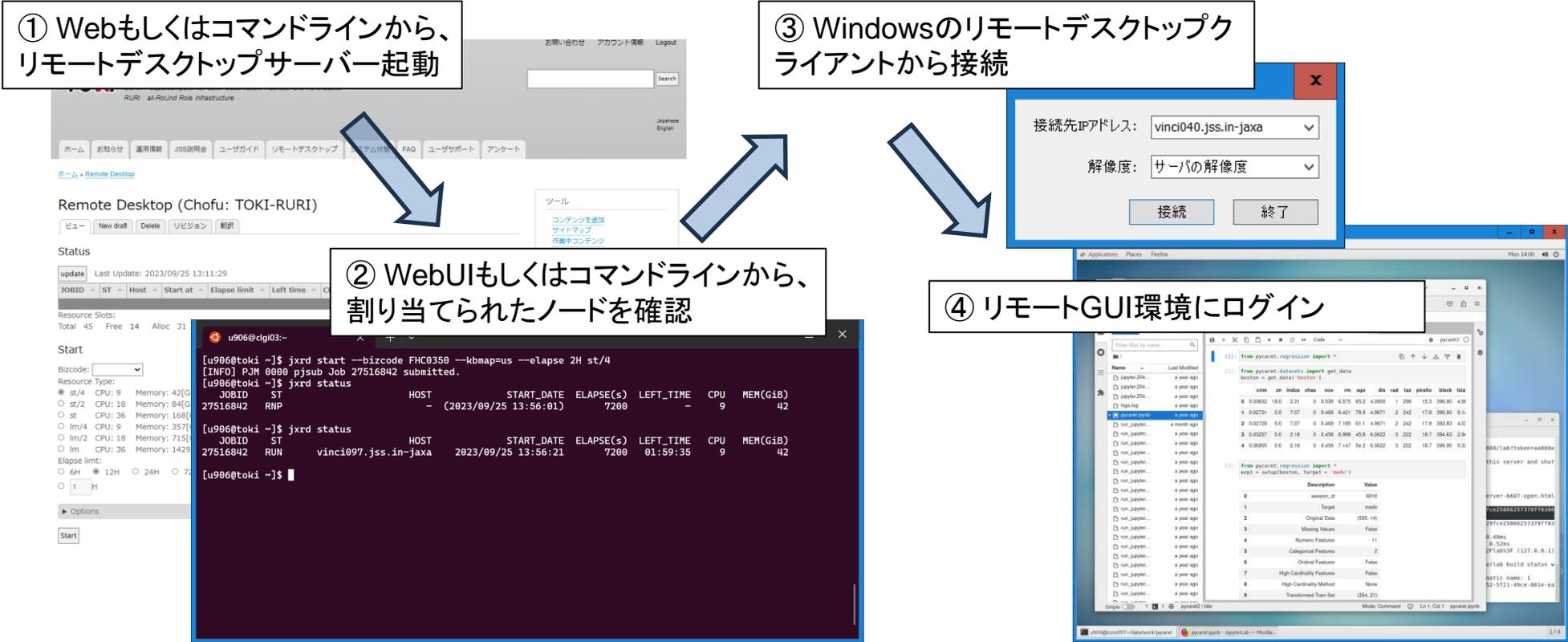
©スーパーコンピュータ活用課

JSS3における新たな利用形態

- リモートデスクトップ
- JupyterLab
- Windows on 仮想マシン(VMware ESXi)
- 定型処理のアプリケーション化

リモートデスクトップ

- 日中は汎用システムの 1割程度 がリモートデスクトップ利用
- 富士通製「COLMINA Design Review 高速リモートデスクトップ」を利用
- WebUI or コマンドライン (jxrdコマンド) からサーバ起動
- 現状は、Windowsクライアントからのみ接続可能



JupyterLab

- リモートデスクトップでの利用
 - バッチジョブでの利用
 - Jupyterlabをバッチジョブ投入 (Singularityイメージ用意)
 - 割当られた計算ノードを調べた後、sshポートフォワーディングで接続
- ⇒ "jxrd"のような専用コマンドは未作成。ドキュメントのみで対応
- JupyterHubの導入も検討 ⇒ そこまでは需要が多くない

① バッチジョブとして投入

```

3 #JX -L rscunit=RURI
4 #JX -L vnode=1
5 #JX -L vnode-core=
6 #JX -L vnode-mem=32000
7 #JX
8 #JX uno@STDPC2959: ~
9 #JX [u906@toki pycaret]$ jxst jupyter.sh
10 #JX [INFO] PJM 0000 pjsub Job 27600: 28 submitted.
11 #JX [u906@toki pycaret]$ jxst
12 #JX JOB_ID JOB_NAME MD ST
13 #JX RE VNODE CORE V_MEM jupyter NM RUN
14 #JX 27608328 jupyter NM RUN
    
```

② 割り当てられたノードとtokenを調べる

```

uno@STDPC2959: ~$ ssh u906@toki.jss.in-jaxa -L 8888:vinci351:8888
0:10:26 2023 from 10.3.2.148
*****
Chofu system
*****
.in details. [URL: https://www3.jss.in-jaxa/]
jp ], if you have any questions.

[* Maintenance & System Suspensive Schedule *]
Chofu Maintenance.
- 2023/09/05 11:00 - 17:00 : J-SPACE, JSS3 portal online application
Operational Outage: cheru

[* Attention *]
2023/3/3 [Important] inte
Japanese [URL: https://w
English [URL: https://w

Please delete or move your files,
if you won't use JSS3 in next fiscal year.
    
```

③ sshポートフォワーディングを起動

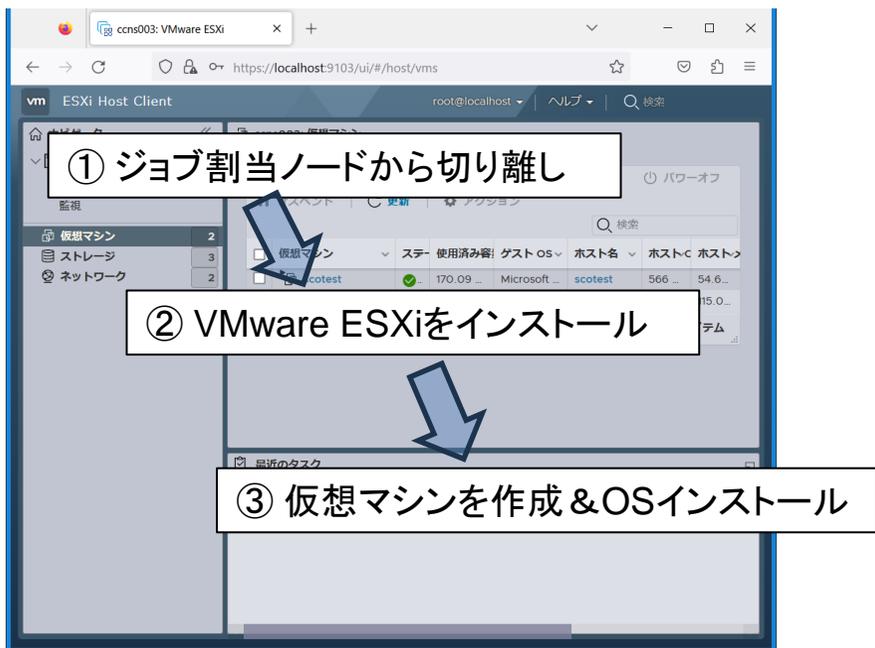
④ 自端末のWebブラウザから接続

The JupyterLab interface shows a file browser with a list of files and folders, and a notebook launcher with options for Python 3 (ipykernel), pycaret, and pycaret2.

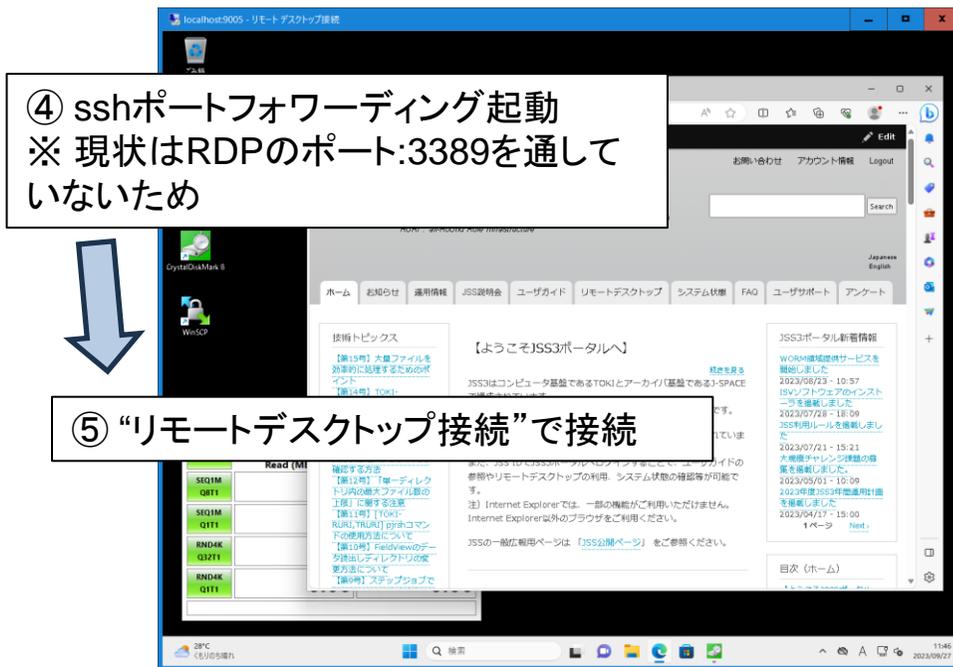
Windows仮想マシン

- 一部計算ノードにVMware ESXiインストール
- 仮想マシンとしてWindowsをインストール
- sshポートフォワーディング+「リモートデスクトップ接続」で利用
- 仮想マシンは基本常に稼働状態（ジョブスケジューラ対象外）

【仮想マシン設定時】

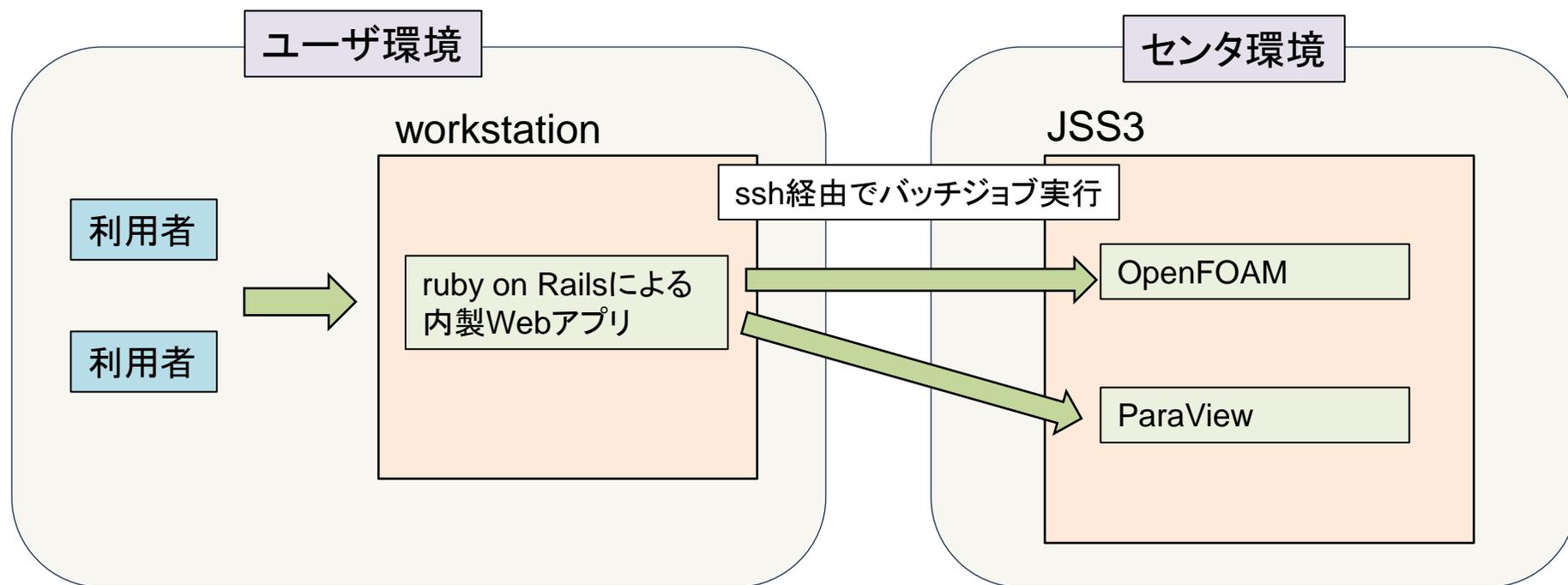


【Windows仮想マシン接続時】



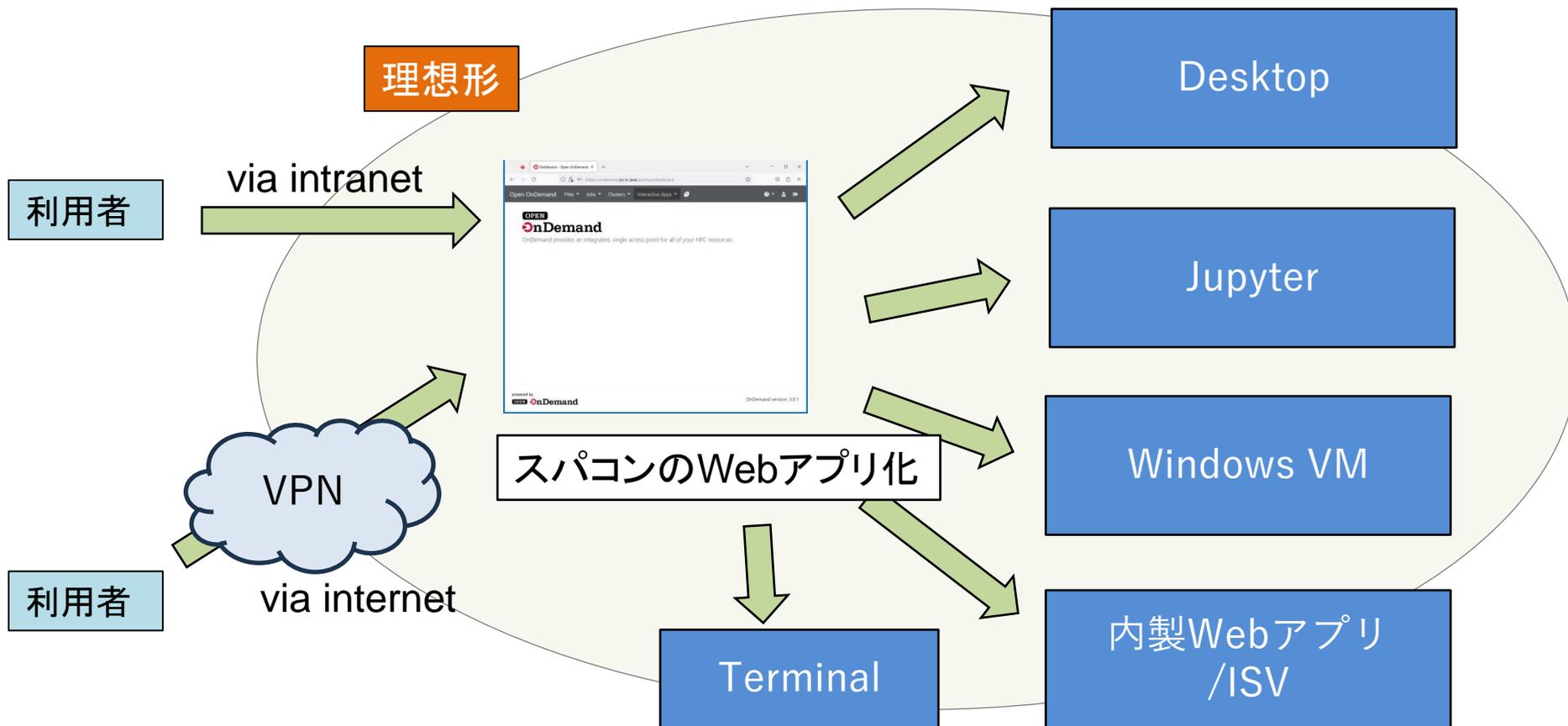
定型処理用Webアプリ

- 一部ユーザは独自のWebアプリを作成。解析／可視化部分をスパコン上でバッチジョブ実行。
- Webアプリ本体は自前のサーバを用意
- パラメータ入力UI→OpenFOAMによる解析→Paraviewで可視化→Webブラウザに表示の流れ



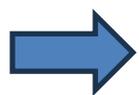
Open OnDemand導入の狙い

- 各種利用事例をOpen OnDemandで一元管理
- Web Reverse Proxyによりsshポートフォワーディングを不要に
- ユーザごとにカスタマイズ可能に



Open OnDemandへの期待（現実編）

- 富岳互換の操作感の実現
 - ひいては、他スパコンとのUI統一化も
- まずはJSS3上で以下の実現に向けて準備中
 - JupyterLab
 - リモートデスクトップ接続(VNC)
 - WHEEL



公開されている情報／リソースで実現可能の見込み

<https://osc.github.io/ood-documentation/latest/>

https://github.com/RIKEN-RCCS/ondemand_fugaku



お試しで環境構築できるdocker-composeがあると便利
例) <https://github.com/ubccr/hpc-toolset-tutorial>

Open OnDemandへの期待（妄想編）

- 新規Interactive appの実装
 - VMの起動/停止 & Windows RDP接続
 - 内製Webアプリの開発推進

さらに...

- Interactive apps store(or hub)
 - 一般ユーザでも簡単にサービスを追加できる仕組みを想定
 - アプリベンダも積極的にプラグインを提供してくれるとありがたい
 - 当面は現行のGitHubでの共有レベルでも問題なし

懸念点は...

- インタラクティブ利用の増加による、CPU利用率の低下
 - ジョブ充填率にかわる新たな指標が必要
 - 理想的にはダイナミックな資源割当も必要か？

まとめ

Open OnDemandで以下の改善に大きく期待

【利用者視点】

- スパコンのGUI/インタラクティブ利用の改善（資源割当、接続）
- リモート（特にVPN経由）での利用改善
- センタごとの操作感の統一
- ユーザごとのメニューのカスタマイズ

【センタ管理者視点】

- システム共通化による導入コスト/管理コストの削減
- QA対応の削減にも期待

まずは実験運用を始めることで、Open OnDemandの普及に貢献

ご清聴ありがとうございました