JAXAにおけるOpen OnDemand導入の狙いと期待

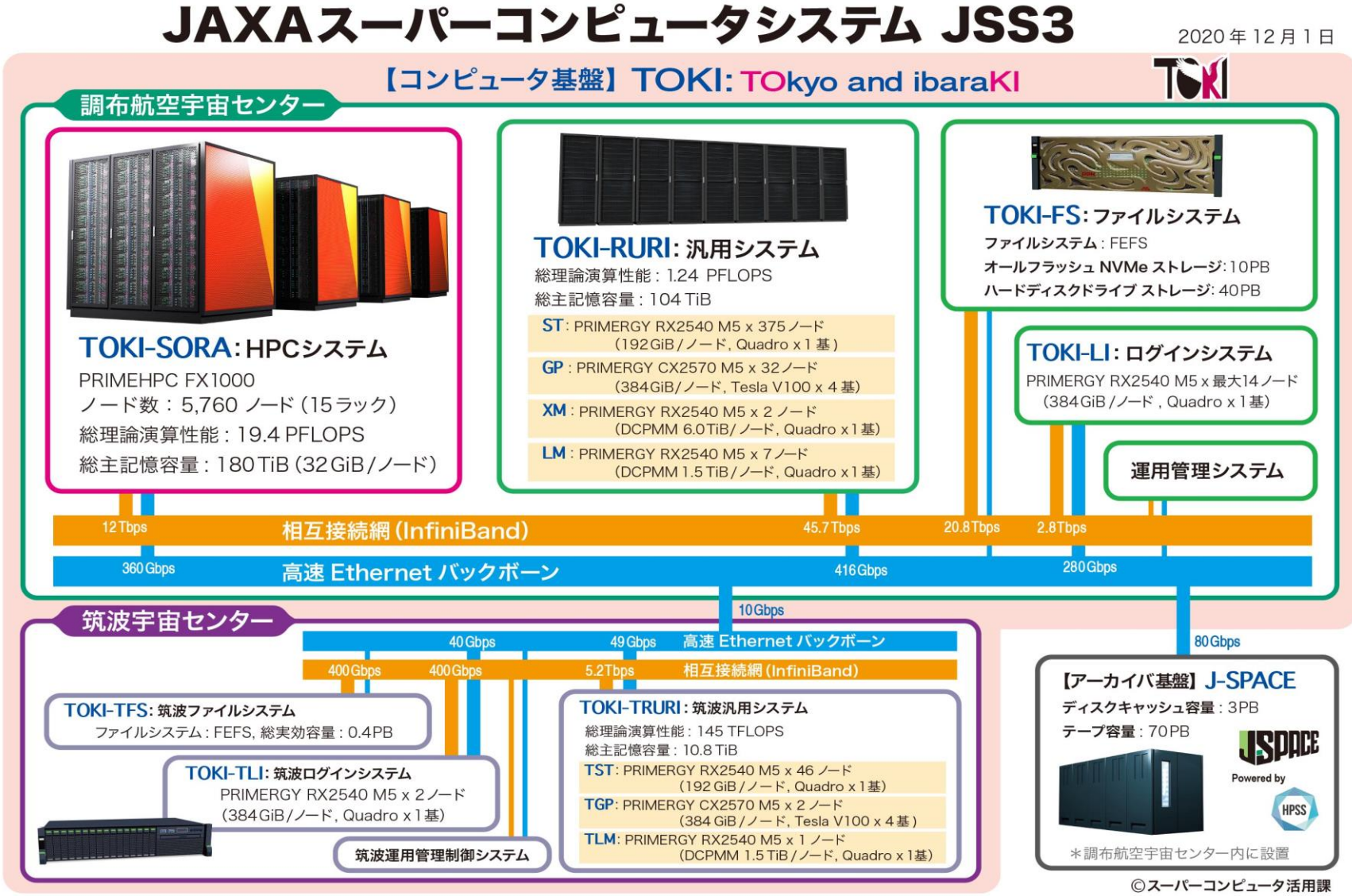
2023/10/11

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

JSS3の紹介


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：180TiB（32GiB/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
（192GiB/ノード, Quadro x1基）
GP: PRIMERGY CX2570 M5 x 32ノード
（384GiB/ノード, Tesla V100 x 4基）
XM: PRIMERGY RX2540 M5 x 2ノード
（DCPMM 6.0TiB/ノード, Quadro x1基）
LM: PRIMERGY RX2540 M5 x 7ノード
（DCPMM 1.5TiB/ノード, Quadro x1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
（384GiB/ノード, Quadro x1基）
運用管理システム
12Tbps
相互接続網（InfiniBand）
45.7Tbps
20.8Tbps
2.8Tbps
360Gbps
高速 Ethernet バックボーン
416Gbps
280Gbps
10Gbps
80Gbps
筑波宇宙センター
40Gbps
49Gbps
高速 Ethernet バックボーン
400Gbps
400Gbps
5.2Tbps
相互接続網（InfiniBand）
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
（384GiB/ノード, Quadro x1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
（192GiB/ノード, Quadro x1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 1.5TiB/ノード, Quadro x1基）
【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
Powered by
HPSS
＊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

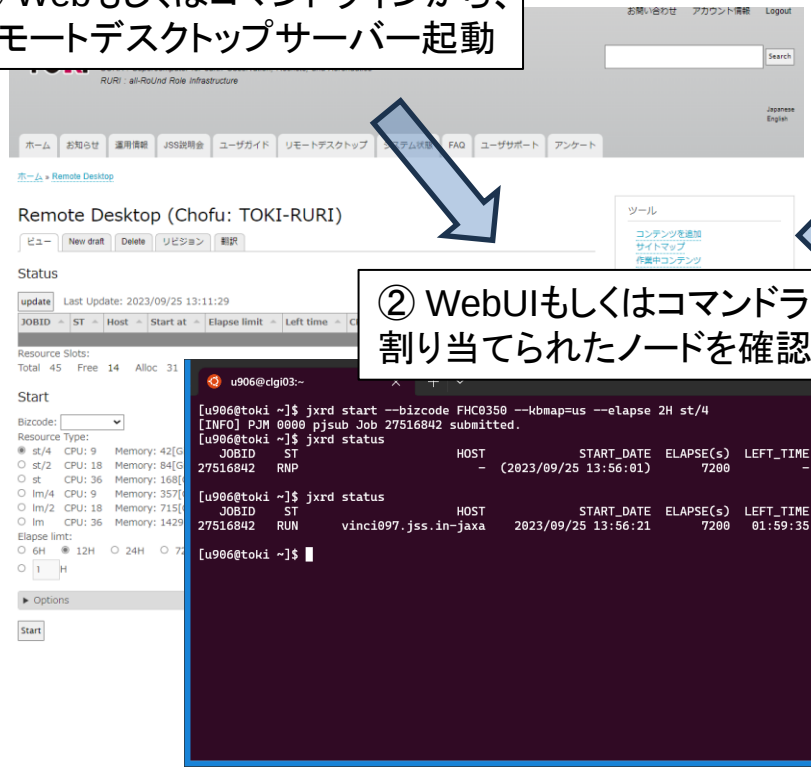
JSS3における新たな利用形態

- 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

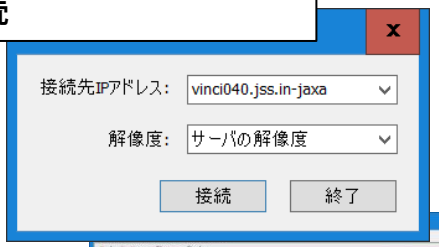
- 日中は汎用システムの**1割程度**がリモートデスクトップ利用
- 富士通製「COLMINA Design Review 高速リモートデスクトップ」を利用
- WebUI or コマンドライン（jxrdコマンド）からサーバ起動
- 現状は、Windowsクライアントからのみ接続可能

① Webもしくはコマンドラインから、リモートデスクトップサーバー起動

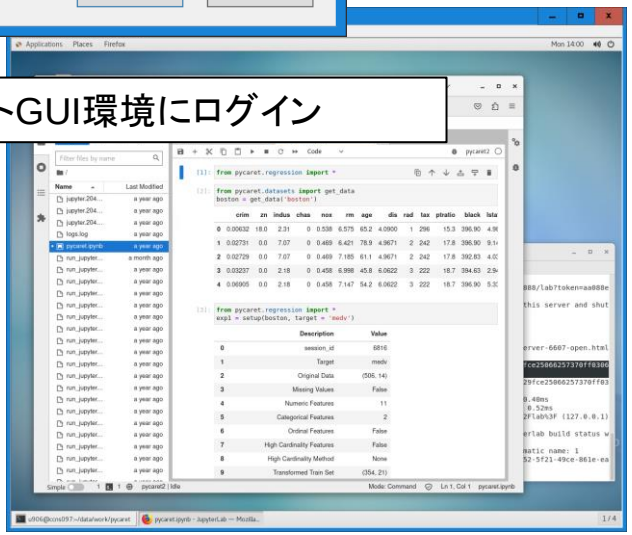


② WebUIもしくはコマンドラインから、割り当てられたノードを確認

③ Windowsのリモートデスクトップクライアントから接続



④ リモートGUI環境にログイン

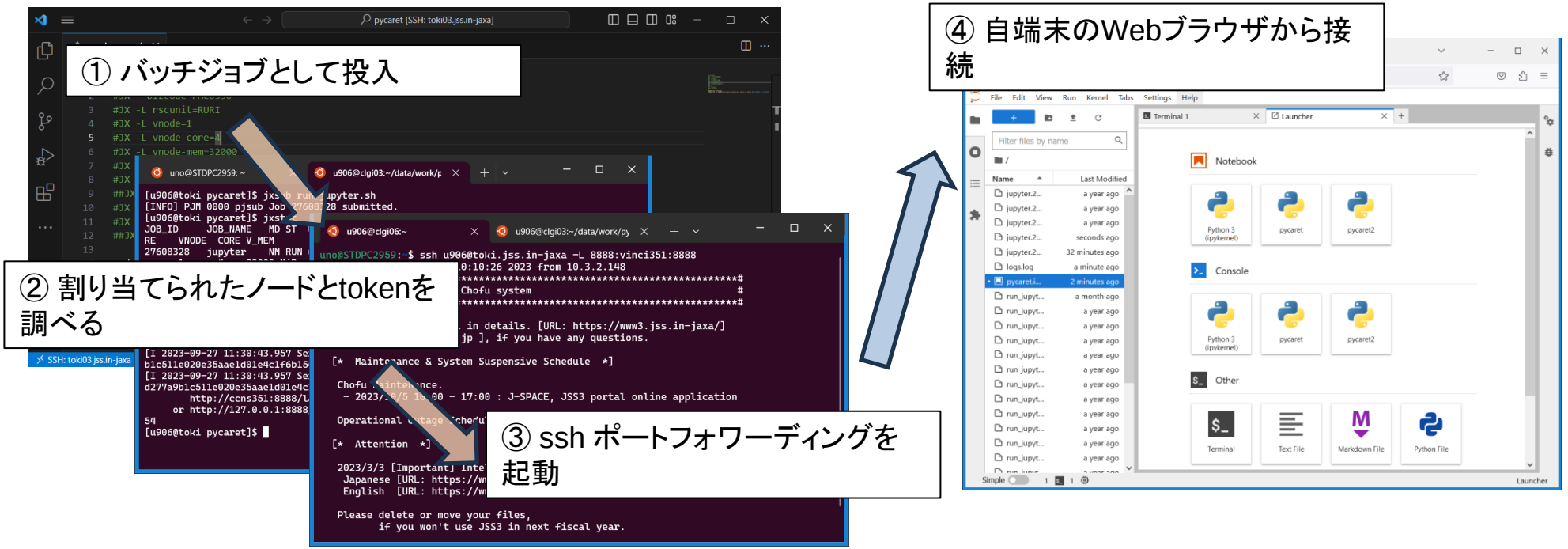


JupyterLab

- 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

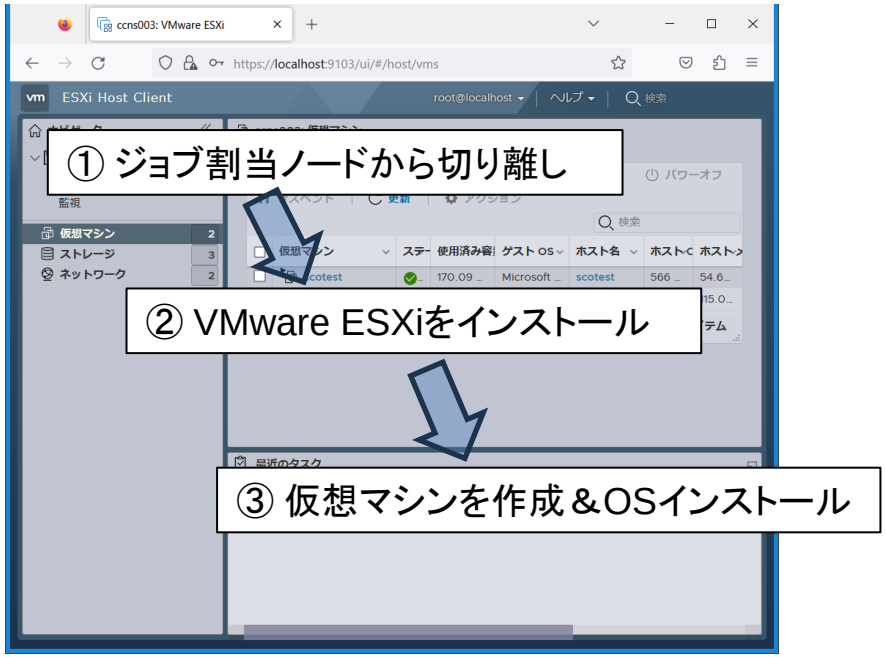
- JupyterHubの導入も検討 ⇒ そこまでは需要が多くない



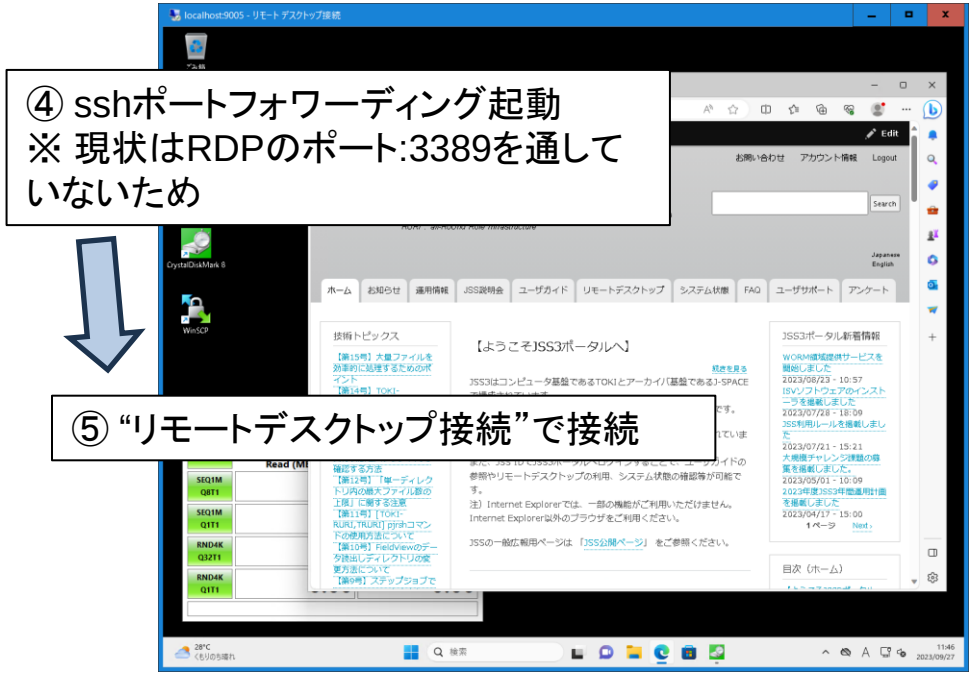
Windows仮想マシン

- 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】

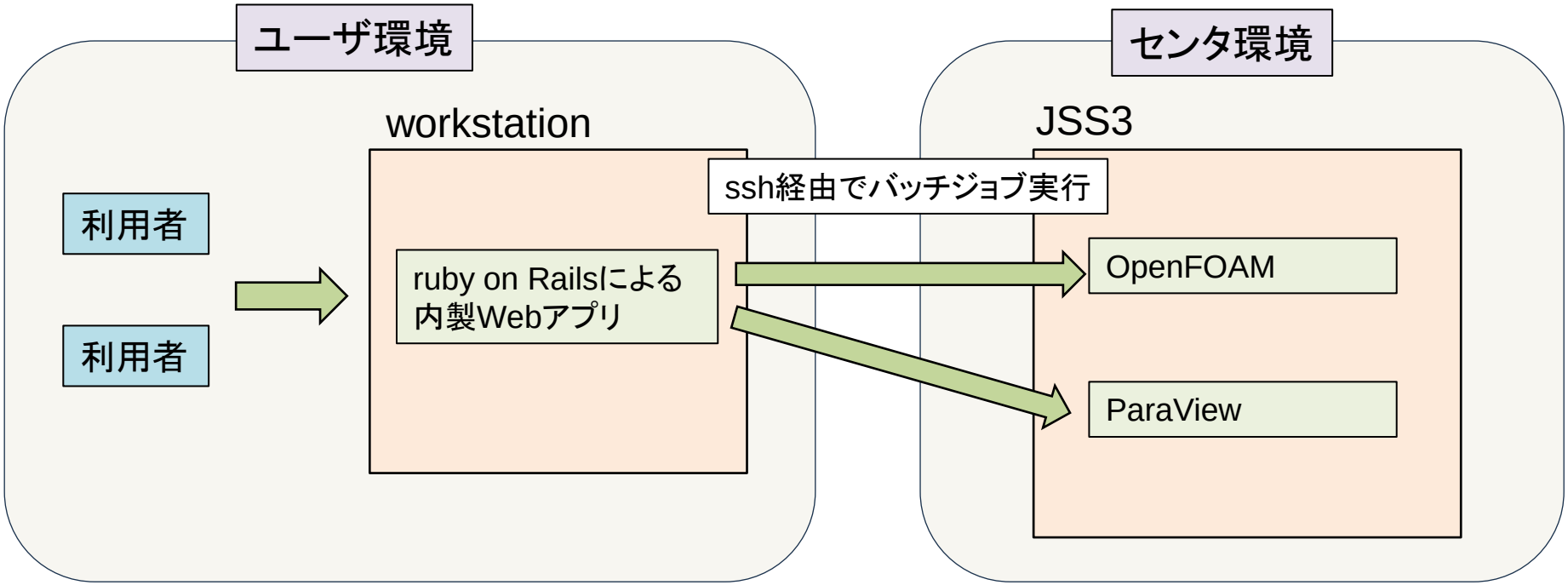


【Windows仮想マシン接続時】



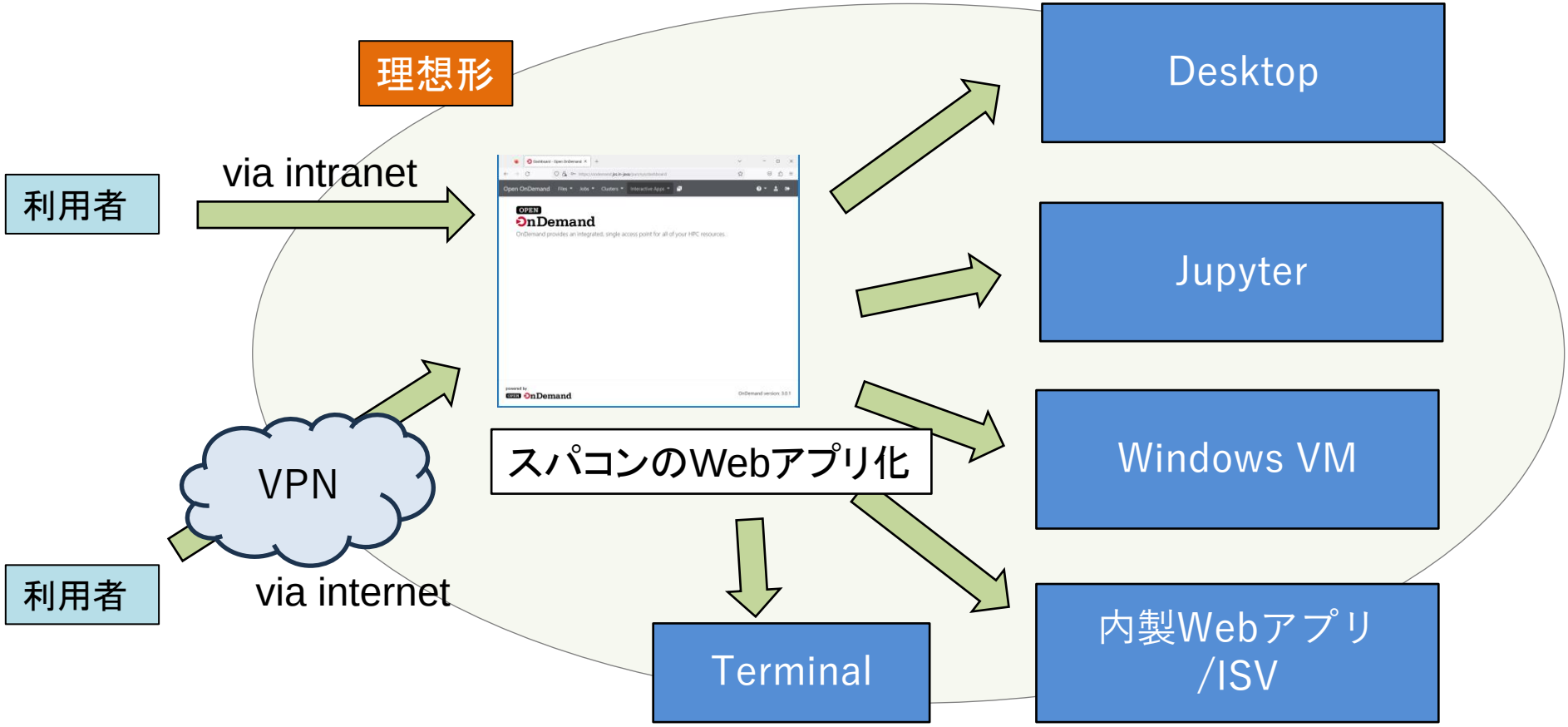
定型処理用Webアプリ

- 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