Visualization User Guide

RIKEN R-CCS

Jun 24, 2024

INDEX:

1 Introduction **1**
- 1.1 Utilized Notations . . . 2
- 1.2 Execution Examples . . . 2
- 1.3 Update History . . . 4

2 Usage Scenarios **6**
- 2.1 Fugaku Open OnDemand . . . 6
- 2.2 CLI (Command Line Interface) . . . 15
- 2.3 GUI (Graphical User Interface) . . . 20
- 2.4 Distributed (Client/Server) . . . 23
- 2.5 VNC (GNOME Desktop) . . . 29

3 AVS/Express **36**
- 3.1 Open OnDemand . . . 36
- 3.2 User PC (Fugaku VPN Connection) . . . 36

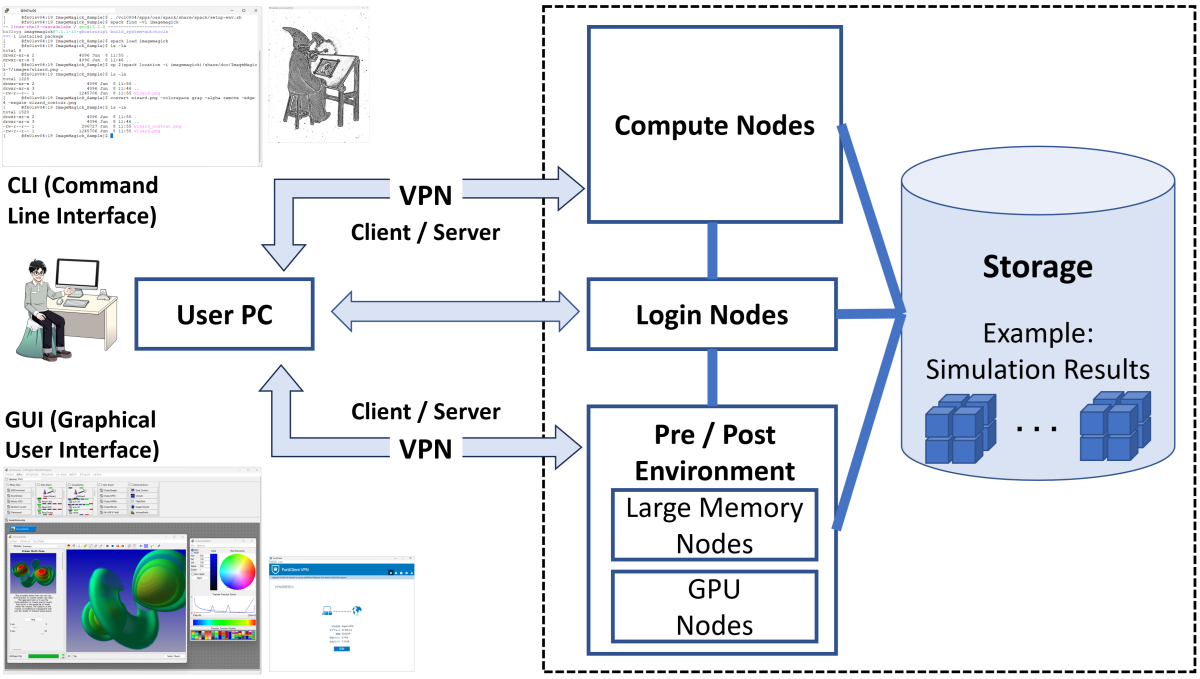
4 Kombyne **40**

CHAPTER

ONE

INTRODUCTION

- This document describes some usage scenarios (**CLI (Command Line Interface)**, **GUI (Graphical User Interface)**, **Distributed (Client/Server)**), and execution environments (**Fugaku Open OnDemand**, **GNOME Remote Desktop**) for using visualization applications on the Fugaku environment.
- *Fugaku environment* here means **Pre/Post Environment (Large Memory Nodes, GPU Nodes)** and **Compute Nodes**.
- Although it is possible to use the **Login Nodes** as the alternative to **Pre/Post Environment**, **it is requested to avoid computationally intensive pre- and post-processing on the Login Nodes as much as possible**. Also, please check the operation notice (https://www.fugaku.r-ccs.riken.jp/operation/20240611_01) regarding the precautions when using Login Nodes.
- Visualization applications range from general-purpose applications (Ex.: ParaView, AVS/Express) to domain specific applications (Ex.: Ncview, GrADS), and this user guide mainly describes how to launch them.
- **Please pay close attention to the license rules and regulations when using software installed by yourself.**



Note: How to use the visualization applications as well as technical problems during their usage are not covered in this document. Please contact the corresponding support desk or the developer if needed.

1.1 Utilized Notations

As shown in the above figure, different systems can be used for the visualization. Therefore, the following notations (for the command prompt) will be used to clarify where the command is being executed: User PC **(UserPC)**; Login Node **(Login)**; Pre/Post Environment **(PrePost)**; or Compute Node **(Compute)**.

Command Promt	Target System
[UserPC]$	Execution on the User's PC
[Login]$	Execution on the Login Node
[PrePost]$	Execution on the Pre/Post Environment
[Compute]$	Execution on the Compute Node

- Sequential numbering (UserPC_1, UserPC_2, ...) will be applied when using multiple CLI terminals.
- **[PrePost]$** corresponds to both **Large-Memory Nodes** and **GPU Nodes**.

1.2 Execution Examples

Some visualization/image processing applications installed on the system side will be used as execution examples in this user guide. Following are some of the installed visualization/image processing applications:

- CLI (Command Line Interface)
  - **imagemagick**: ImageMagick (Image Processing Software)
  - **povray**: POV-Ray (Persistence of Vision Raytracer)
  - **grads**: GrADS (Grid Analysis and Display System)
  - **gmt**: GMT (The Generic Mapping Tools)
  - **gnuplot**: Gnuplot (Command-driven plotting program)
- GUI (Graphical User Interface)
  - **paraview**: ParaView (Data Analysis and Visualization Application)
  - **avs**: AVS/Express (Visual Programming Visualization Tool)
- Client/Server
  - **paraview**: ParaView (Data Analysis and Visualization Application)

Applications provided via Spack (Public Instace) should be loaded in advance (ImageMagick in the following example), and please refer to the Spack Users' Guide for detailed information about the Spack usage.

```
[PrePost]$ . /vol0004/apps/oss/spack/share/spack/setup-env.sh

[PrePost]$ spack find -x
...
-- linux-rhel8-a64fx / fj@4.10.0 ---------------------------------
povray@3.7.0.8       grads@2.2.3     paraview@5.11.2

-- linux-rhel8-cascadelake / gcc@13.2.0 --------------------------
gmt@6.2.0            ncview@2.1.9     gnuplot@5.4.3
imagemagick@7.1.1-11
...

[PrePost]$ spack load imagemagick@7.1.1-11%gcc@13.2.0
```

For applications that are not provided, or application options that are not enabled, in the Spack (Public Instance), it is possible to try building them in a local Spack Private Instance. In addition, if the desired application is not

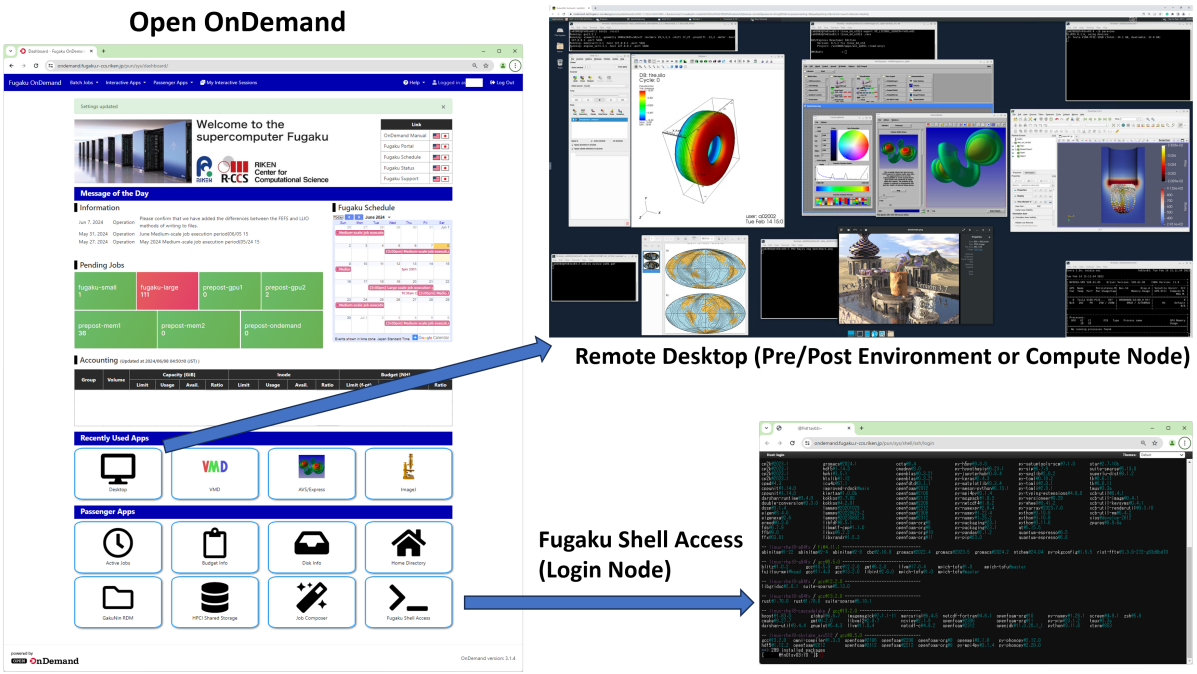
available in the latest Spack envinronment, but provided in a previous Spack environment, it may be possible to use it as shown in the following ParaView example.

```
===========
Spack 0.21.0

[PrePost]$ . /vol0004/apps/oss/spack/share/spack/setup-env.sh

[PrePost]$ spack --version
 0.21.0

[PrePost]$ spack find paraview arch=linux-rhel8-cascadelake

==> No package matches the query: paraview arch=linux-rhel8-cascadelake


===========
Spack 0.19.0

[PrePost]$ . /vol0004/apps/oss/spack-v0.19/share/spack/setup-env.sh

[PrePost]$ spack --version
 0.19.0

[PrePost]$ spack find paraview arch=linux-rhel8-cascadelake
 -- linux-rhel8-cascadelake / gcc@12.2.0 --------------------------

 paraview@5.10.1
 ==> 1 installed package

[PrePost]$ spack load paraview@5.10.1%gcc@12.2.0
```

Note: The above Spack outputs are just examples, and the results can differ due to continuous installations and updates. Also, if a problem, like database consistency error, occurs when using different Spack versions, a Spack cache clearing can be tried to solve it. Please refer to the Spack Users' Guide for detailed information about the Spack usage.

In addition to these applications provided via Spack, the Fugaku Open OnDemand service provides some additional applications that are directly installed on the Open OnDemand side. The list below shows some of these installed applications that can be used without loading via Spack.

- General-purpose visualization applications
- **paraview**: ParaView (Data analysis and visualization application)
- **visit**: VisIt (Open source, interactive, scalable, visualization, animation and analysis tool)
- **avs**: AVS/Express (Visual programming visualization tool)
- Domain-specific visualization applications
- **vmd**: VMD (Molecular visualization program)
- **ovito**: OVITO (Open visualization tools)
- **VESTA**: VESTA (3D visualization program for structural models)
- **pymol**: PyMol (Molecular visualization system)
- **xcrysden**: XCrysDen (Crystalline and molecular structure visualization program)



1.3 Update History

- Version 0.9.1 (Jun. 24, 2024)
- Added a note regarding the license when using user installed software into Section 1 [Introduction]
- Added a note regarding the Anaconda license into Section 2.5 [VNC (GNOME Desktop)]
- Version 0.9 (Jun. 20, 2024)
- Added Section 4 [Kombyne]
- Updated Spack related information due to its version upgrade
- Updated Open OnDemand related information due to its service upgrade
- Update on some figures and explanation
- Version 0.8 (Jul. 12, 2023)
- Chaged [Open OnDemand (Xfce)] to [Open OnDemand]
- Updated Open OnDemand related information due to its service upgrade
- Updated Spack related information due to its version upgrade
- Version 0.7 (Mar. 8, 2023)
- **Chaged the order of Subsections**
  - Moved the [Open OnDemand (Xfce)] to Section 2.1
- Added the pre-installed VisIt and AVS/Express on the Open OnDemand into Section 2.1.1.1 [Running GUI Applications]
- Updated the Spack related information due to its upgrade
- Version 0.6 (Nov. 10, 2022)
- Added Subsection 4 [X Server (WIndows OS)] in Section 2.2 [GUI (Graphical User Interface)]
- Added Section 2.5 [Open OnDemand (Xfce)]
- Added Section 3.3 [Open OnDemand (Xfce)]

- Additional information such as Spack applications and usage
- Version 0.5.1.1 (May 31, 2022)
- Minor update on some usage examples following the Pre/Post Environment Users Guide (Ver. 1.7)
- Version 0.5.1 (May 18, 2022)
- Added memory size and GPU usage options on the interactive job submission command at Sections 2 and 3
- Corrections on the Pre/Post environment related notations following the Pre/Post Environment Users Guide documentation
- Version 0.5 (Apr. 12, 2022)
- Added information for installing AVS/Express on local environment at Section 3.2
- Updated the usage examples by using new queues made available on the Pre/Post Environment
- Version 0.4 (Oct. 25, 2021)
- Added PvPython and PvBatch as usage examples in Section 2.1
- Updated the Pre/Post Environment usage mode in Section 2.3 (requirement of [–x11] option)
- Information renewal due to Spack update
- Version 0.3 (Jul. 26, 2021)
- **Changed the use of [SSH Port Forwarding] to [Fugaku VPN Connection]**
  - Section 2.3 [Distributed (Client/Server)]
  - Section 2.4 [VNC (GNOME Desktop)]
- Added Section 3 [AVS/Express]
- Version 0.2 (May 17, 2021)
- Corrections on the node notations
- Added Section 1.1 [Utilized Notations]
- Added Section 1.2 [Execution Examples]
- Added Section 1.3 [Update History]
- Added Section 2.4 [VNC (GNOME Desktop)]
- First Version (April 23, 2021)

CHAPTER

TWO

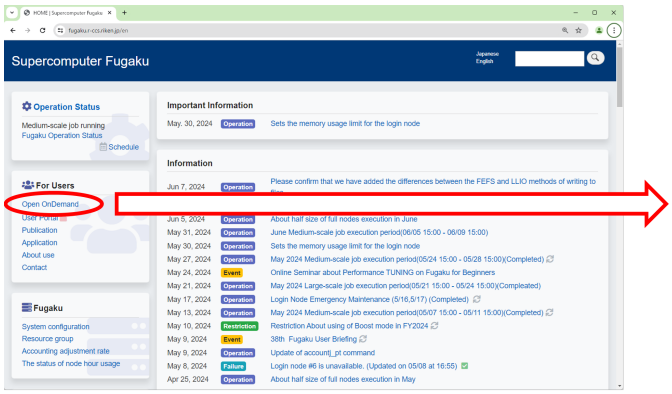
USAGE SCENARIOS

2.1 Fugaku Open OnDemand

Visualization applications can be executed on the Web-based **Fugaku Open OnDemand** service via **CLI (Command Line Interface)** and **GUI (Graphical User Interface)** modes. Please refer to the "Fugaku Open OnDemand Guide" for detailed information about its use. Additional technical information as well as the use of visualization on the Open OnDemand environment can be found in the following **12th Meeting for Application Code Tuning on A64FX Computer Systems (Ease To Use Fugaku Open OnDemand Service)** materials:

- Material: Fugaku Open OnDemand [PDF]
  - https://www.hpci-office.jp/documents/meeting_A64FX/231023/MTGA64FX_12-Fugaku_Open_OnDemand.pdf
- Material: Visualization HandsOn [PDF]
  - https://www.hpci-office.jp/documents/meeting_A64FX/231023/MTGA64FX_12-Visualization_HandsOn.pdf

Fugaku Open OnDemand service can be accessed via dedicated URL (https://ondemand.fugaku.r-ccs.riken.jp/) or via "Open OnDemand" link in the Fugaku Website as follows.



"Default" (Recently Used Apps & Passenger Apps)

Help ➔ Profiles: Switch between "Default" and "All Apps"
"All apps" ("Default" + Batch Jobs + Interactive Apps)

Note: Visualization and analysis oriented applications listed in the "Interactive Apps" group will not appear in the "Default" view mode. Therefore, in such case, it is recommended to switch to the "All apps" view mode.

In the **All apps** view mode, the **Batch Jobs** group mainly lists the applications for batch job execution on the Compute Nodes, and the **Interactive Apps** group mainly lists the applications for interactive use on the Pre/Post Environment.

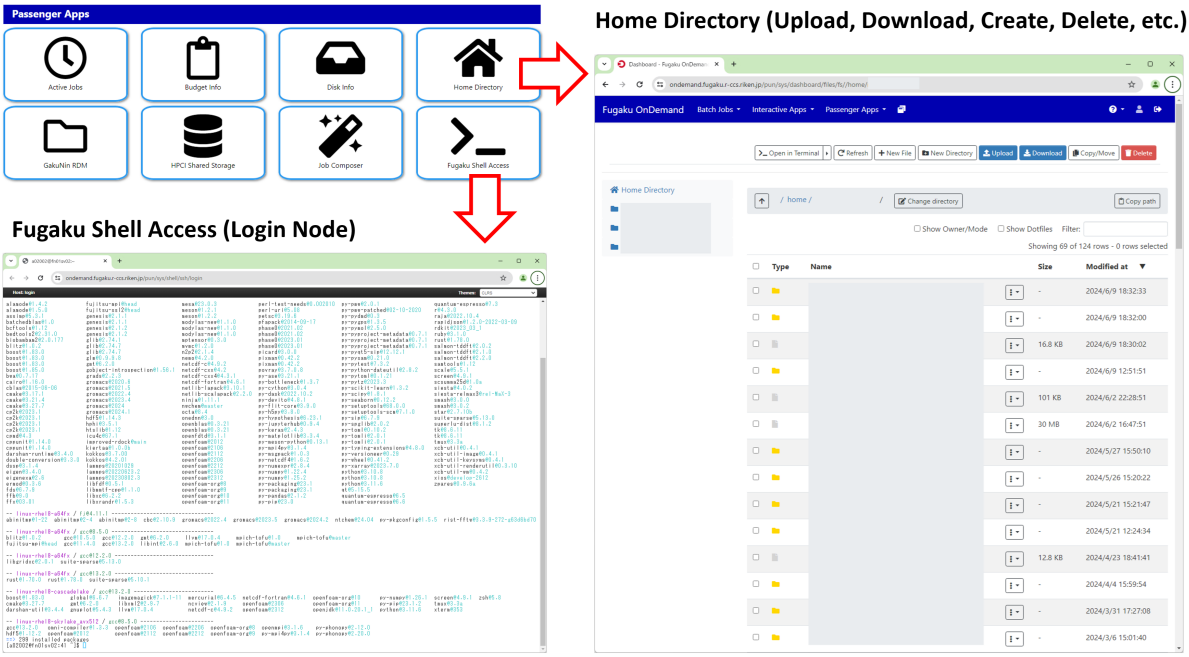
Batch Jobs	Interactive Apps
	

Note: Displayed icons and installed applications may differ from the above examples due to the continuous updates and possible configuration changes in the Open OnDemand service. For detailed information about the Fugaku Open OnDemand service and its use, please refer to the "OnDemand Manual" listed in the "Link" displayed at the top of the Open OnDemand Webpage.

2.1.1 Home Directory & Fugaku Shell Access

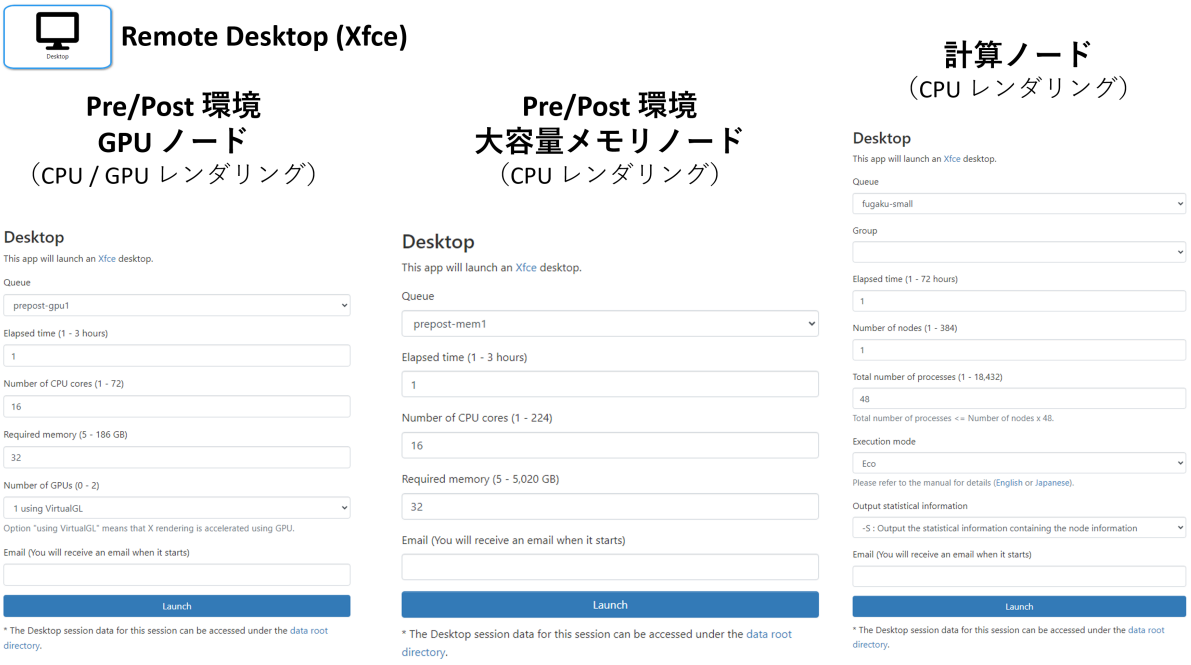
The **Home Directory** tool, listed in the **Passenger Apps** section, can be used to upload or download visualization-oriented data to or from the Fugaku environment. In addition, **CLI terminal (Login Node)** can be launched from the **Fugaku Shell Access** also listed in the **Passenger Apps** section. As mentiond in the Introduction section, **it is requested to avoid computationally intensive processing on the Login Nodes**, and in such case, we ask to use the Pre/Post Environment or Compute Nodes via batch or interactive jobs.

Note: For detailed information about the Passenger Apps, please refer to the "OnDemand Manual" listed in the "Link" displayed at the top of the Open OnDemand Webpage.



2.1.2 Desktop (Xfce Remote Desktop)

It is possible to launch an Xfce-based remote desktop session on the **Pre/Post Environment** as well as **Compute Nodes** by clicking the **Desktop** icon listed in the **Interactive Apps**, and by setting the job queue, amount of computing resources, and desired time to be used as follows.



For the job queues specific to GPU Nodes (Pre/Post Environment), that is **prepost-gpu1** and **prepost-gpu2**,it is possible to select the use of GPU Nodes. When GPU is selected (Number of GPUs > 0), hardware-accelerated rendering (GPU rendering) will be performed via GPU vendor's graphics library (NVIDIA). On the other hand, with no GPU selection (Number of GPUs = 0), or on Large Memory Nodes and Compute Nodes, software rendering (CPU rendering) will be performed via Mesa 3D graphics library.

As shown below, it is possible to check if GPU rendering is enabled in the Remote Desktop environment by verifying if the active OpenGL graphics library is from the GPU vendor (NVIDIA).

```
GPU Node (Number of GPUs > 0)
[PrePost]$ glxinfo | grep OpenGL
Output example:
OpenGL vendor string: NVIDIA Corporation
OpenGL renderer string: Tesla V100-PCIE-32GB/PCIe/SSE2
OpenGL core profile version string: 4.6.0 NVIDIA 535.129.03
...
```

If the active OpenGL graphics library is from Mesa, as shown below, then the rendering process will be performed on the CPU.

```
Large Memory Node, GPU Node (Number of GPUs = 0)
[PrePost]$ glxinfo | grep OpenGL
Output example:
OpenGL vendor string: Mesa/X.org
OpenGL renderer string: llvmpipe (LLVM 15.0.7, 256 bits)
OpenGL core profile version string: 4.5 (Core Profile) Mesa 22.3.0
...

Compute Node
[Compute]$ glxinfo | grep OpenGL
Output example:
OpenGL vendor string: Mesa
OpenGL renderer string: llvmpipe (LLVM 16.0.6, 128 bits)
OpenGL core profile version string: 4.5 (Core Profile) Mesa 23.1.4
```

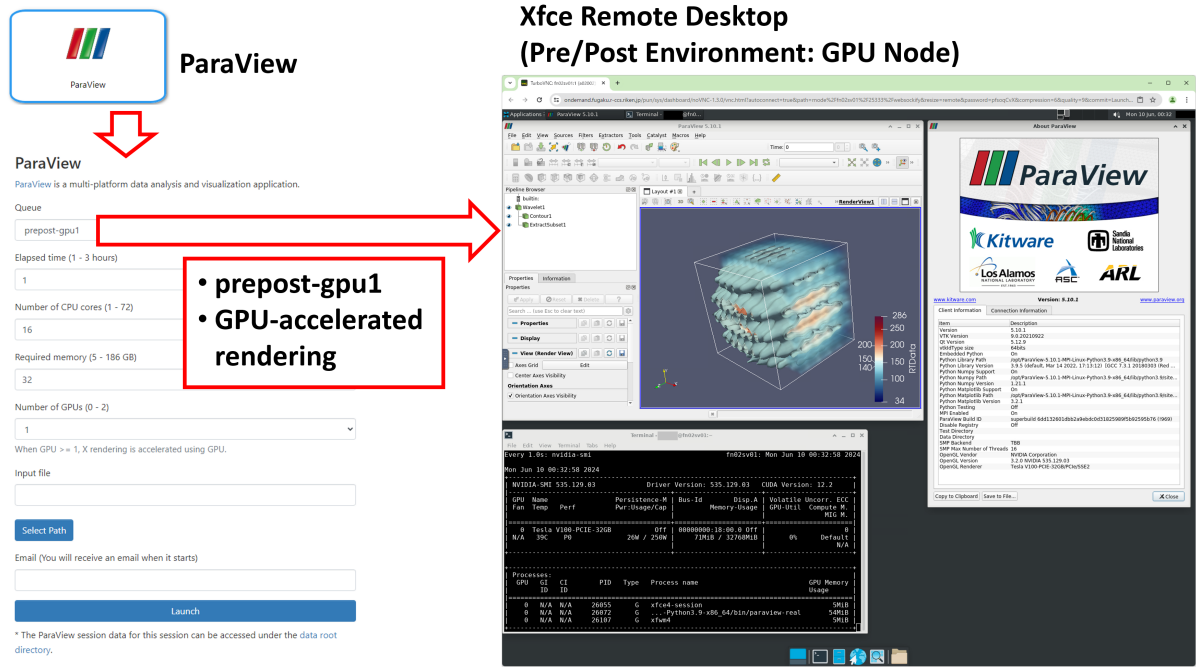
Note: Displayed OpenGL library and its version may differ from the above examples due to the updates or configuration changes.

If GPU rendering is enabled, the "nvidia-smi" tool can be used to monitor the GPU usage (including the applications using GPU resources). In the following usage example, the GPU information is updated every second.

```
[PrePost]$ watch -n 1 nvidia-smi
```

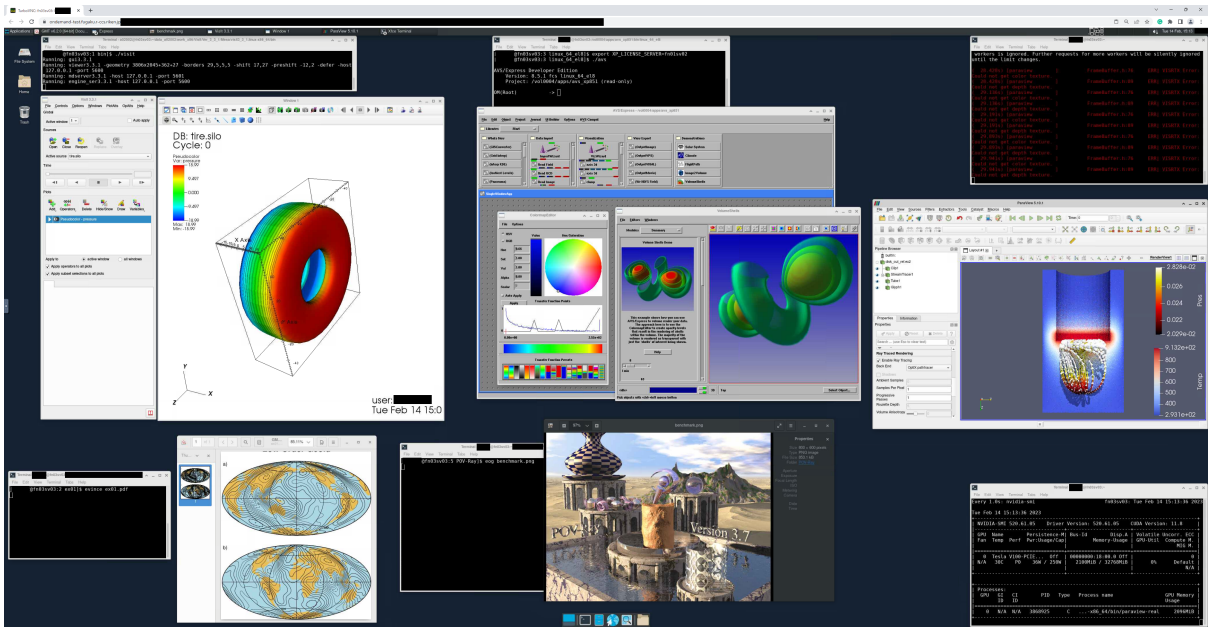
Applications listed in the **Interactive Apps** section can be launched by clicking on their respective icons, and selecting and filling up necessary information on their respective configuration screen. In the example below, ParaView is configured to run on the Pre/Post Environment (GPU Node) via its specific job queue (prepost-gpu1), and it will be launched on the Xfce Remote Desktop environment. Since GPU usage is selected (Number of GPUs > 0), then GPU-based hardware rendering will be enabled by default.

Note: In addition to ParaView, many other applications listed in the "Interactive Apps" section are launched as GUI applications on the Xfce Remote Desktop environment. Therefore, it is possible to start the Xfce Remote Desktop via "Desktop" icon in advance, and launch the desired visualization application from the CLI terminal.



Pre/Post Environment

Below is an example of Xfce Remote Desktop session launched on a GPU Node (Pre/Post Environment). In the upper part, it shows the VisIt, AVS/Express, and ParaView, which are installed on the Open OnDemand side, and in the lower part, GMT and POV-Ray visualization results are displayed by using evince and eog tools.



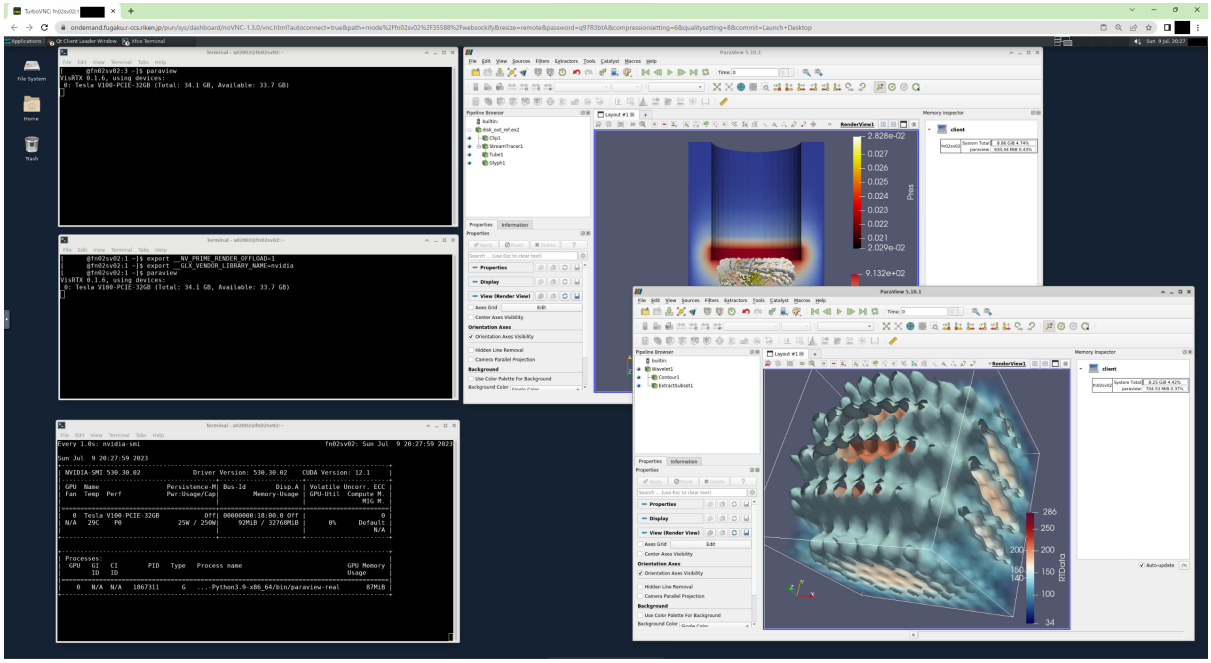
Below are some applications installed on the Open OnDemand side, to be used in the Pre/Post Environment, which can be directly launched from the CLI terminal (by using the bold text portion) without loading them via Spack.

- General-purpose visualization applications
- **paraview**: ParaView (Data analysis and visualization application)
- **visit**: VisIt (Open source, interactive, scalable, visualization, animation and analysis tool)
- **avs**: AVS/Express (Visual programming visualization tool)

- Domain-specific visualization applications
- **vmd**: VMD (Molecular visualization program)
- **ovito**: OVITO (Open visualization tools)
- **VESTA**: VESTA (3D visualization program for structural models)
- **pymol**: PyMol (Molecular visualization system)
- **xcrysden**: XCrysDen (Crystalline and molecular structure visualization program)

a) ParaView

```
[PrePost]$ paraview
```



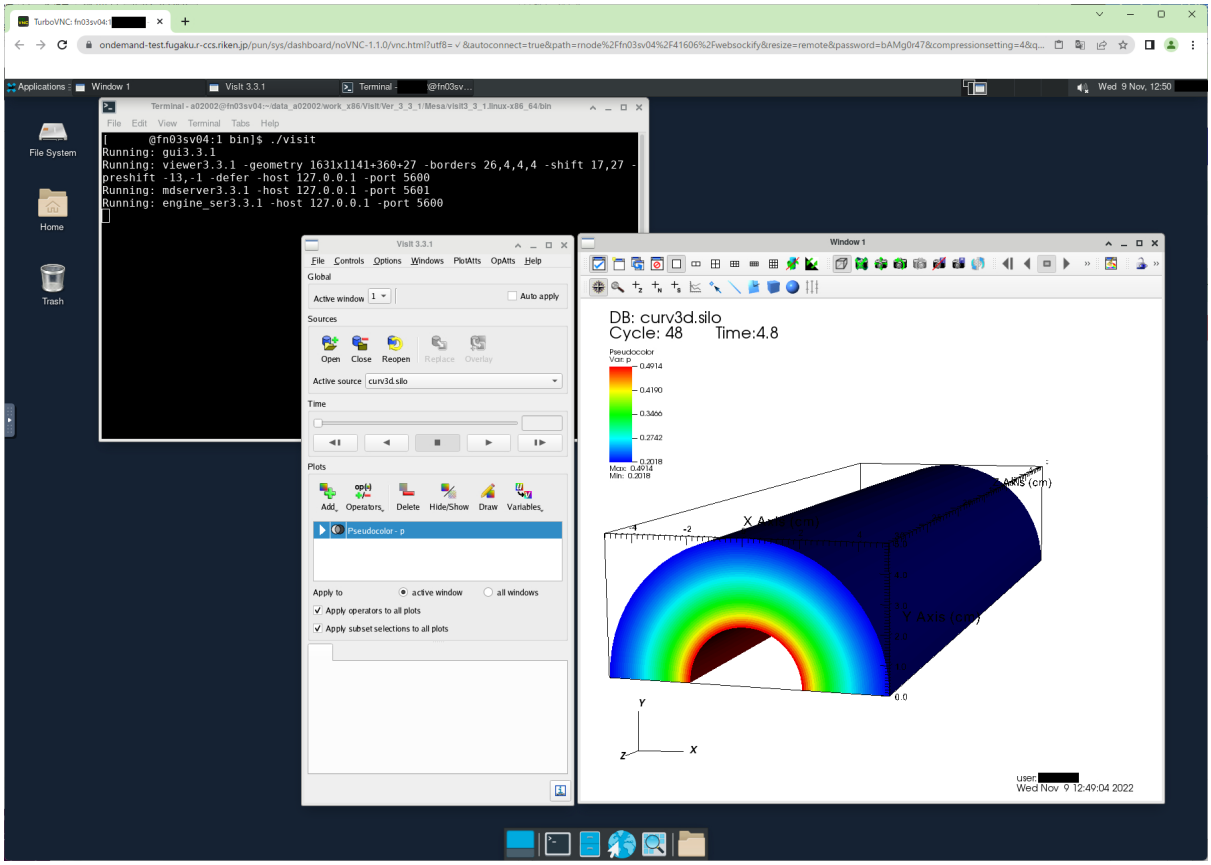
Note: To enable ray tracing capability, there is a need to tick the "Enable Ray Tracing" on the "Ray Traced Rendering" sub-window of ParaView GUI. At this time, it is possible to select the desired renderig engine (CPU-based Intel OSPRay or GPU-based NVIDIA OptiX).

b) VisIt

VisIt is a VTK(Visualization ToolKit)-based application, like ParaView, designed for large data visualization.

```
[PrePost]$ visit
```

c) AVS/Express



AVS/Express is a general-purpose visualization application and its license server is running on the login node 2 (login2: fn01sv02). It can also be used by installing on the local PC as detailed in the "AVS/Express" section.

```
[PrePost]$ avs
```

d) Applications available on the Xfce Desktop side

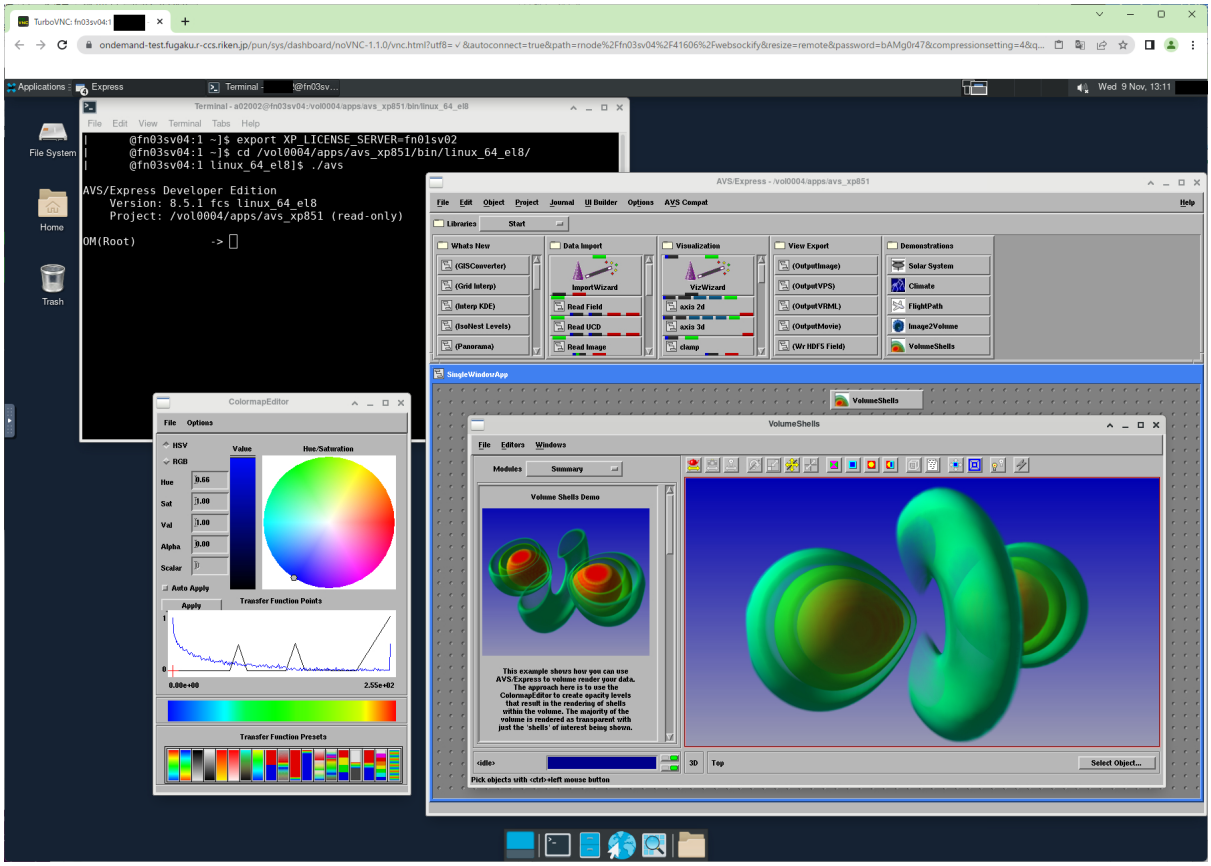
In addition to the applications shown as graphics icons (Termnal, File Browser, Web Browser, etc.) on the Xfce Remote Desktop GUI, it is possible to run other applications installed on the Xfce Desktop side such as:

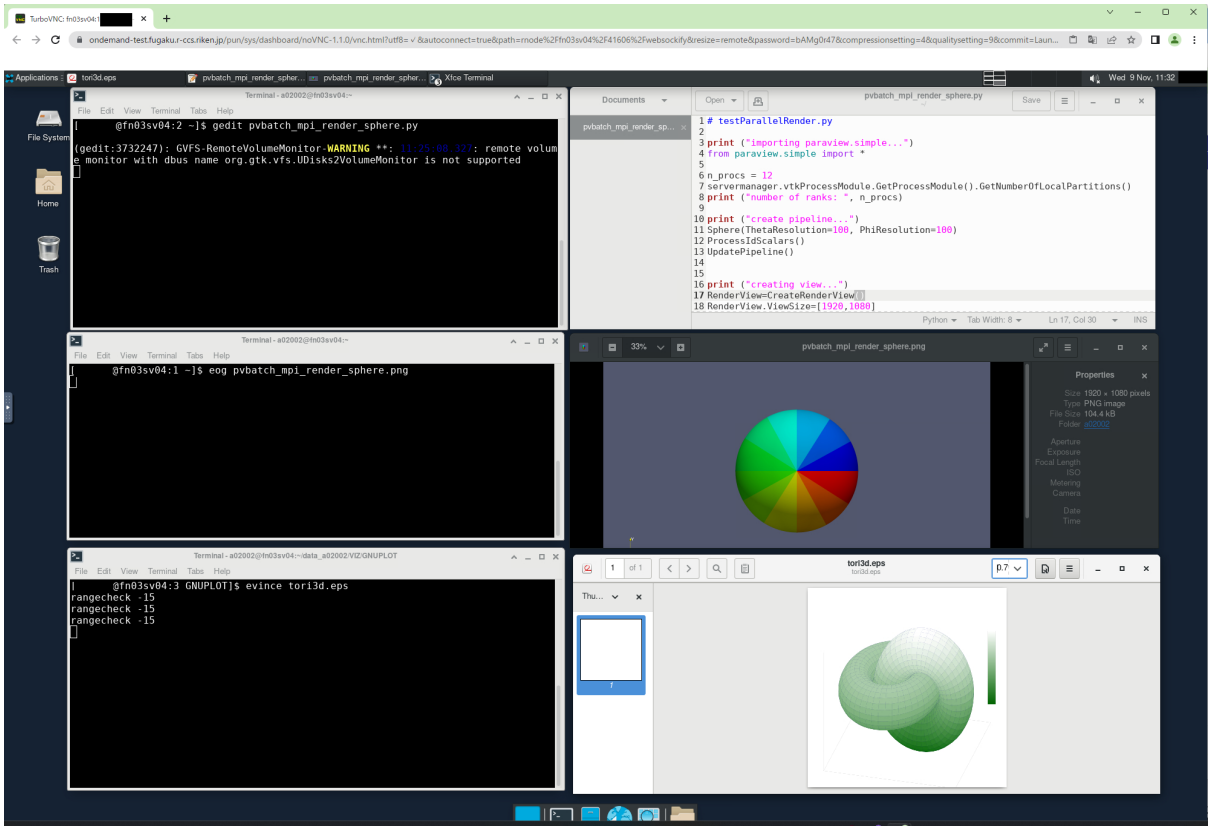
- Text Editor: gedit
- Image Viewer: eog, display (ImageMagick)
- Document VIewer: evince
- Graph Plotting Tool: gnuplot

Note: Above applications may fail to start when some Spack applications are loaded. In this case, open a new CLI terminal and relaunch the desired application.

---

e) Applications available via Spack


File System
Home
Trash
AVS/Express Developer Edition
Version: 8.5.1 fcs linux_64_el8
Project: /vol0004/apps/avs_xp851 (read-only)
OM(Root) ->
ColormapEditor
VolumeShells
Volume Shells Demo


File System
Home
Trash
@fn03sv04:2 ~]$ gedit pvbatch_mpi_render_sphere.py
@fn03sv04:1 ~]$ eog pvbatch_mpi_render_sphere.png
@fn03sv04:3 GNUPLOT]$ evince tori3d.eps
rangecheck -15
rangecheck -15
rangecheck -15

It is possible to use Spack-based applications (from Public Instance) such as those shown in the “CLI (Command Line Interface)” and “GUI (Graphical User Interface)” sub-sections.

f) Applications not available via Spack

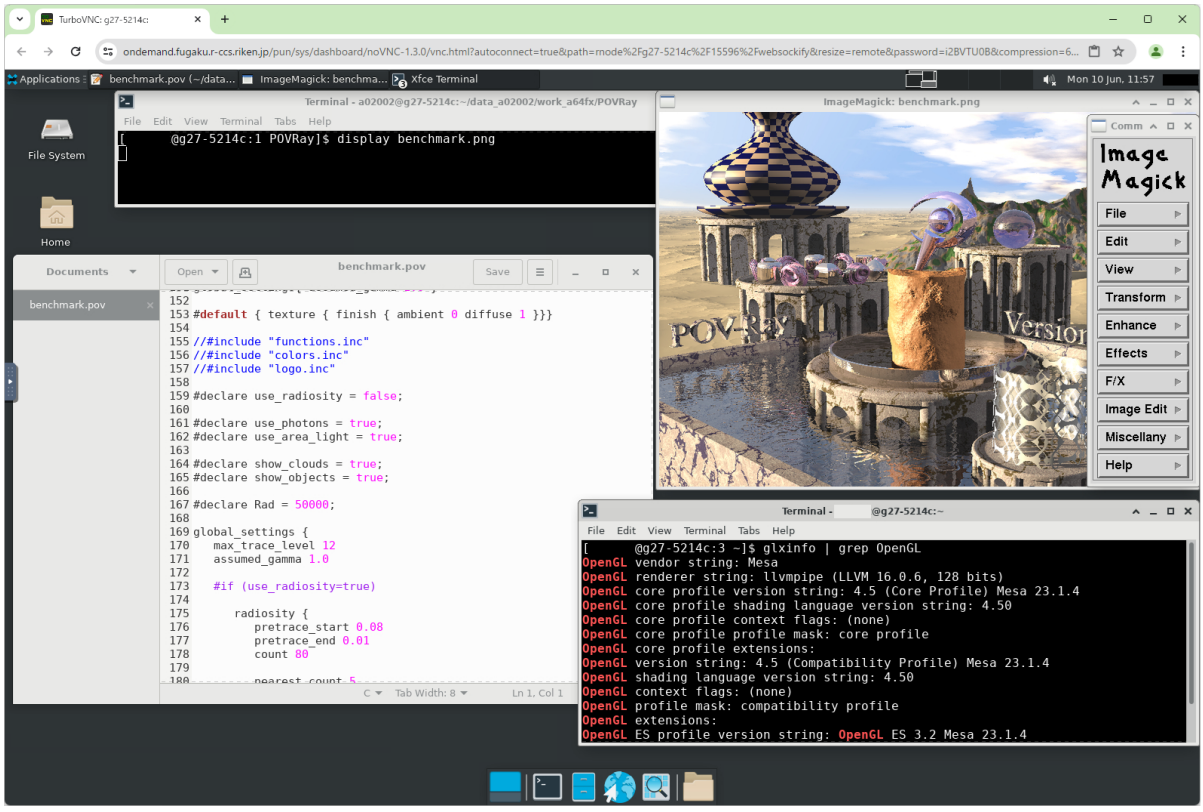
For those applications which were not possible to build via Spack, or those not available on the Spack (no Spack recipe), there is a possibility to run these applications by building from source code or download as binary files from appropriate repositories.

Compute Nodes

a) Applications available on the Xfce Desktop side

In addition to the applications shown as graphics icons (Termnal, File Browser, Web Browser, etc.) on the Xfce Remote Desktop GUI, it is possible to run other applications installed on the Xfce Desktop side such as:

- Text Editor: gedit
- Image Viewer: display (ImageMagick)
- Document VIewer: evince
- Graph Plotting Tool: gnuplot



b) Applications available via Spack

It is possible to use Spack-based applications (from Public Instance) such as those shown in the “CLI (Command Line Interface)” and “GUI (Graphical User Interface)” sub-sections.

Note: Since CPU rendering is performed on Compute Nodes, it is recommended to use the Pre/Post Environment (especially GPU Nodes) for computationally intensive rendering.

2.2 CLI (Command Line Interface)

Use of visualization related applications via CLI terminal. It performs visualization processing and outputs image files, without using GUI interface.

2.2.1 Pre/Post Environment

Following are the step-by-step guide for using an application (**imagemagick** as an example) on the Pre/Post Environment during an interactive job. Some of the commands are abbreviated, and please refer to the Pre/Post Environment Users Guide for detailed information regarding its use.

1. Login to the Fugaku (**Login Node**).

```
[UserPC]$ ssh user@login.fugaku.r-ccs.riken.jp
```

Note: It can also be used from a CLI terminal within VNC GNOME Remote Desktop or Fugaku Shell Access (Open OnDemand). However, the latter cannot be used for visualization processes that require an X server.

---

2. Start an **Interactive Job** on the **Pre/Post Environment**.

(**GPU Node** usage example)

```
[Login]$ srun -p gpu1 -n 4 --mem 16000 --time=0:15:00 --pty bash -i
```

Note: Specify the desired job queue (-p), number of CPUs (-n), memory size (–mem), GPU usage (–gpus=), and max. elapsed time (–time) according to the necessity. Please see the Pre/Post Environment Users Guide for detailed information. However, when using the CLI terminal from Open Ondemand Xfce Remote Desktop (Pre/Post Environment), steps 1 and 2 above can be skipped.

---

3. Run the desired visualization application

- ImageMagick

```
[PrePost]$ . /vol0004/apps/oss/spack/share/spack/setup-env.sh

==========
[PrePost]$ spack find imagemagick

 -- linux-rhel8-cascadelake / gcc@13.2.0 -------------------------
 imagemagick@7.1.1-11
 ==> 1 installed package

[PrePost]$ spack load imagemagick@7.1.1-11%gcc@13.2.0

==========
[PrePost]$ cp $(spack location -i imagemagick)/share/doc/ImageMagick-7/images/wizard.
↪png .

[PrePost]$ convert wizard.png -colorspace gray -alpha remove -edge 4 -negate wizard_
↪contour.png
```

Input File: wizard.png	Output File: wizard_contour.png
	

Note: This execution example uses a sample image (wizard.png) from the ImageMagick sample image folder: ImageMagick_Folder/share/doc/ImageMagick-7/images/. The folder path can differ from above in the case of a package update. Please refer to the Spack Users' Guide for detailed information about the Spack usage.

- ParaView PvBatch

```
===========
Spack 0.21.0

[PrePost]$ . /vol0004/apps/oss/spack/share/spack/setup-env.sh

[PrePost]$ spack --version
0.21.0

[PrePost]$ spack find paraview arch=linux-rhel8-cascadelake

==> No package matches the query: paraview arch=linux-rhel8-cascadelake

===========
Spack 0.19.0

[PrePost]$ . /vol0004/apps/oss/spack-v0.19/share/spack/setup-env.sh

[PrePost]$ spack --version
0.19.0

[PrePost]$ spack find paraview arch=linux-rhel8-cascadelake

-- linux-rhel8-cascadelake / gcc@12.2.0 --------------------------
```

(continues on next page)

(continued from previous page)

```
paraview@5.10.1
==> 1 installed package

[PrePost]$ spack load paraview@5.10.1%gcc@12.2.0

[PrePost]$ mpirun -n 4 pvbatch filename_of_parallel_pvbatch_script.py
```

- ParaView PvPython

```
[PrePost]$ mpirun -n 1 pvpython filename_of_pvpython_script.py
```

If there exist multiple versions for the same package, there is a need to specify the desired package by specifying version, compiler, architecture, or hash value. Please refer to the Spack Users' Guide for detailed information about the Spack usage.

```
===========
Spack 0.19.0

[PrePost]$ . /vol0004/apps/oss/spack-v0.19/share/spack/setup-env.sh

[PrePost]$ spack --version
0.19.0

[PrePost]$ spack load paraview

==> Error: paraview matches multiple packages.
Matching packages:
  oh5qwqo paraview@5.10.1%fj@4.8.1 arch=linux-rhel8-a64fx
  uzxnwov paraview@5.10.1%fj@4.8.1 arch=linux-rhel8-a64fx
  i66ryps paraview@5.10.1%fj@4.8.1 arch=linux-rhel8-a64fx
  6temuv4 paraview@5.10.1%gcc@12.2.0 arch=linux-rhel8-cascadelake
Use a more specific spec (e.g., prepend '/' to the hash).

Example using architecture
[PrePost]$ spack load paraview arch=linux-rhel8-cascadelake

Example using hash value
[PrePost]$ spack load /6temuv4
```

2.2.2 Compute Node

Following are the step-by-step guide for using an application (**povray** as an example) on the Compute Node during an interactive job. Some of the commands are abbreviated, and please see the Fugaku Users' Guide for detailed information.

1. Login to the Fugaku (**Login Node**)

```
[UserPC]$ ssh user@login.fugaku.r-ccs.riken.jp
```

Note: It can also be used from a CLI terminal within VNC GNOME Remote Desktop or Fugaku Shell Access (Open OnDemand). However, the latter cannot be used for visualization processes that require an X server.

2. Start an **Interactive Job** on the **Compute Nodes(s)**.

```
[Login]$ pjsub -x PJM_LLIO_GFSCACHE=/vol0004 --interact -L "eco_state=2" -L "node=1" -
↪-mpi "max-proc-per-node=4" -L "rscgrp=int" -L "elapse=0:15:00" --sparam "wait-
↪time=600"
```

Note: Set the necessary options such as volume (Ex.: -x PJM_LLIO_GFSCACHE=/vol0004) and energy saving mode (Ex.: -L "eco_state=2"), and please check the Fugaku Users Guide for detailed information about the Fugaku usage. However, when using the CLI terminal from Open Ondemand Xfce Remote Desktop (Compute Node), steps 1 and 2 above can be skipped.

---

3. Run the desired visualization application

- POV-Ray

```
[Compute]$ . /vol0004/apps/oss/spack/share/spack/setup-env.sh

[Compute] spack find povray

-- linux-rhel8-a64fx / fj@4.10.0 ---------------------------------
povray@3.7.0.8
==> 1 installed packages

[Compute]$ spack load povray@3.7.0.8%fj@4.10.0

===========
[Compute]$ cp $(spack location -i povray@3.7.0.8)/share/povray-3.7/scenes/advanced/
↪benchmark/benchmark.pov .

[Compute]$ povray benchmark.pov
```

Output File: benchmark.png

Note: This execution example uses a sample scene (benchmark.pov) from the POV-Ray sample scene folder: POV-Ray_Folder/share/povray-3.7/scenes/advanced/benchmark

---

- ParaView PvBatch

If there exist multiple versions for the same package, there is a need to specify the desired package by specifying version, compiler, architecture, build options, or hash value as follow.

```
[Compute]$ . /vol0004/apps/oss/spack/share/spack/setup-env.sh

===========
[Compute]$ spack find paraview

-- linux-rhel8-a64fx / fj@4.10.0 ---------------------------------
paraview@5.11.2  paraview@5.11.2  paraview@5.11.2
==> 3 installed packages

Search for the hash value and build options
[Compute]$ spack find -vl paraview arch=linux-rhel8-a64fx

-- linux-rhel8-a64fx / fj@4.10.0 ---------------------------------
sknjmpn paraview@5.11.2~adios2~advanced_debug~catalyst~cuda+development_files~
↪examples~eyedomelighting~fortran~hdf5~ipo+kits~libcatalyst+mpi~nvindex+opengl2~
↪openpmd~osmesa~pagosa+python~qt~raytracing~rocm+shared~tbb~visitbridge build_
↪edition=canonical build_system=cmake build_type=Release generator=ninja␣
↪patches=02253c7,acb3805 use_vtkm=default
```

(continues on next page)


ImageMagick: benchmark.png (on fn02sv01)
POV-Ray
Version 3.7

(continued from previous page)

```
vx26636 paraview@5.11.2~adios2~advanced_debug~catalyst~cuda+development_files~
↪examples~eyedomelighting~fortran~hdf5~ipo+kits~libcatalyst+mpi~nvindex+opengl2~
↪openpmd+osmesa~pagosa+python~qt~raytracing~rocm~shared~tbb~visitbridge build_
↪edition=canonical build_system=cmake build_type=Release generator=ninja␣
↪patches=02253c7,acb3805 use_vtkm=default
tfrzwlo paraview@5.11.2~adios2~advanced_debug~catalyst~cuda+development_files~
↪examples~eyedomelighting~fortran~hdf5~ipo+kits~libcatalyst+mpi~
↪nvindex+opengl2+osmesa~pagosa~python~qt~raytracing~rocm~shared~tbb~visitbridge␣
↪build_edition=canonical build_system=cmake build_type=Release generator=ninja␣
↪patches=02253c7,acb3805 use_vtkm=default
==> 3 installed packages

===========
Load example using build options and architecture
[Compute]$ spack load paraview +python +osmesa arch=linux-rhel8-a64fx

Load example using hash value
[Compute]$ spack load /vx26636

===========
[Compute]$ mpirun -n 4 pvbatch filename_of_parallel_pvbatch_script.py
```

Note: Please refer to the Spack Users' Guide for detailed information about the Spack usage.

2.3 GUI (Graphical User Interface)

This section presents the use of GUI-based applications. Fugaku Remote Desktop environments (VNC GNOME Desktop and Open OnDemand Xfce Desktop) can directly display the GUI, but when using CLI terminal such as PuTTY, the GUI part is sent to the User's PC via "X11 forwarding" mechanism. In such case, on the Windows OS environment, an **X Server**, such as **VcXsrv (Windows X Server)**, is required to display the GUI (from remote system) on the User's PC.

Note: It is recommended to use Remote Desktop environment when using GUI-based visualization-oriented applications. Please refer to the corresponding sections for using GUI applications on the GNOME Remote Desktop or Open OnDemand Xfce Remote Desktop.

2.3.1 Pre/Post Environment

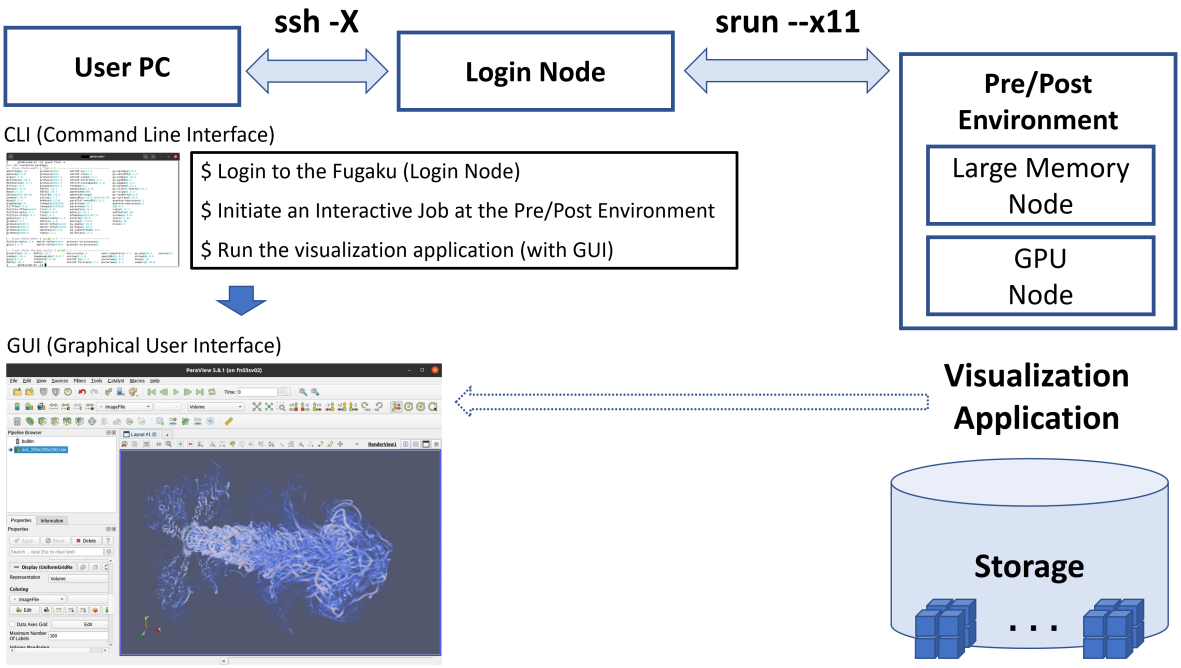
Following is the step-by-step guide for enabling X11 forwarding in order to send the GUI to the User's PC. Some of the commands are abbreviated, and please check the Pre/Post Environment Users Guide for detailed information regarding its use.

1. Login to the Fugaku (**Login Node**) using **"-X"** option.

```
[UserPC]$ ssh -X user@login.fugaku.r-ccs.riken.jp
```

2. Start an **Interactive Job** on the **Pre/Post Environment** using **"–x11"** option.

Note: When using PuTTY, tick the "Enable X11 forwarding" option. Please see the Supercomputer Fugaku Startup Guide for detailed information.



(**GPU Node** usage example)

```
[Login]$ srun --x11 -p gpu1 -n 8 --mem 16000 --time=0:15:00 --pty bash -i
```

Note: Specify the desired job queue (-p), number of CPUs (-n), memory size (–mem), GPU usage (–gpus=), and max. elapsed time (–time) according to the necessity. Please see the Pre/Post Environment Users Guide for detailed information.

3. Run the desired visualization application

- ParaView

Since ParaView is provided on the Open OnDemand, the GUI version (Qt version) of the ParaView will no longer be available via Spack. Therefore, launch the Open OnDemand Xfce Remote Desktop for using paraview (GUI-based ParaView). The pvserver (for Client/Server visualization) and pvbatch (for batch job) will continue being available via Spack.

- ImageMagick

```
[PrePost]$ . /vol0004/apps/oss/spack/share/spack/setup-env.sh

==========
[PrePost]$ spack find imagemagick

 -- linux-rhel8-cascadelake / gcc@13.2.0 -------------------------
 imagemagick@7.1.1-11
 ==> 1 installed package
```

(continues on next page)

(continued from previous page)

```
[PrePost]$ spack load imagemagick@7.1.1-11%gcc@13.2.0

===========
[PrePost]$ cp $(spack location -i imagemagick)/share/doc/ImageMagick-7/images/wizard.
↪png .

[PrePost]$ display wizard.png
```

Panel	Display Window
	

- X Terminal

```
[PrePost]$ . /vol0004/apps/oss/spack/share/spack/setup-env.sh

===========
[PrePost]$ spack find xterm

-- linux-rhel8-cascadelake / gcc@13.2.0 --------------------------
xterm@353
==> 1 installed package

[PrePost]$ spack load xterm@353%gcc@13.2.0
```

(continues on next page)

(continued from previous page)

```
==========
[PrePost]$ xterm
```

2.4 Distributed (Client/Server)

Use of visualization application with distributed processing capability (Client/Server), which uses socket communication for connecting the server and client-side modules. It uses Fugaku VPN Service to enable the communication between the Client (Local PC) and Server (Fugaku Compute Node or Pre/Post Environment).

- In the example case of **ParaView**:
  - Server: **ParaView Server (pvserver)**
  - Client: **ParaView GUI Client** installed on the user side PC (Client and Server versions should match)
    - Download site: https://www.paraview.org/download/
  - Port: **11111 (Default)**

2.4.1 Pre/Post Environment

It is assumed that the Fugaku VPN Connection is already enabled, and in the example case of ParaView, a single- or multi-process ParaView Server (pvserver) runs on the server-side, and the ParaView GUI Client runs on the User PC side. Some of the commands are abbreviated, and please see the "Fugaku", "Prepost environment", and "Fugaku VPN Service" Users' Guide for detailed information.

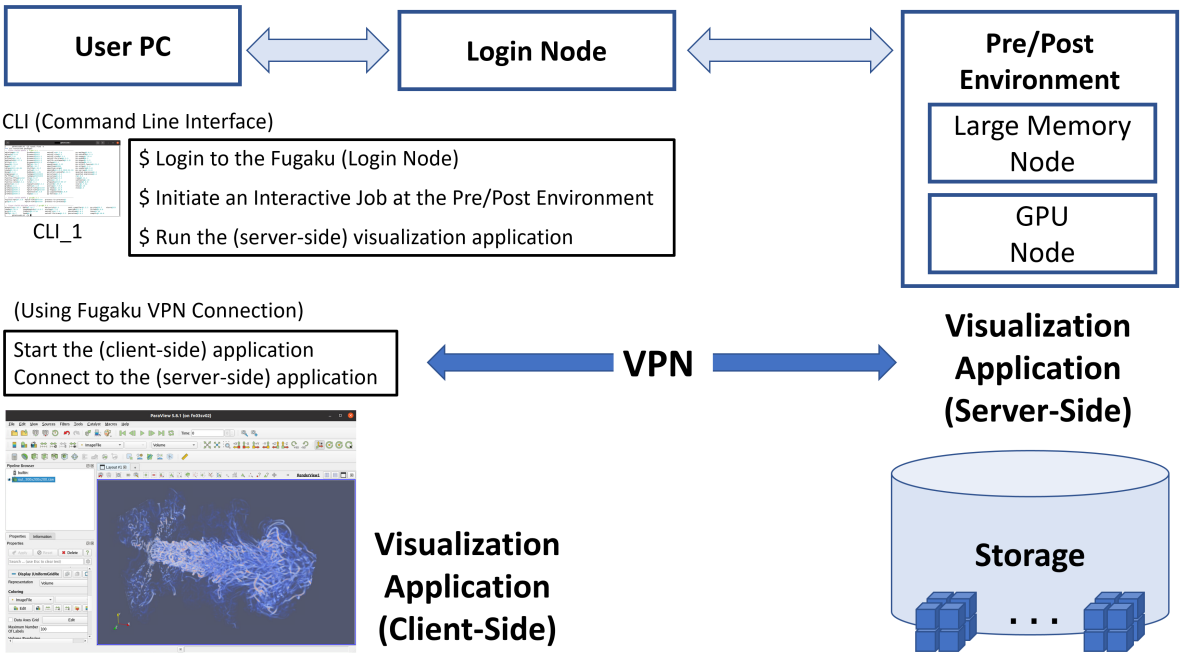
1. Login to the Fugaku (**Login Node**).

```
[UserPC_1]$ ssh user@login.fugaku.r-ccs.riken.jp
```

2. Start an **Interactive Job** on the **Pre/Post Environment**.

(**GPU Node** usage example)

```
[Login]$ srun -p gpu1 -n 4 --mem 32000 --time=0:15:00 --pty bash -i
```



Note: Specify the desired job queue (-p), number of CPUs (-n), memory size (–mem), GPU usage (–gpus=), and max. elapsed time (–time) according to the necessity. Please see the "Pre/Post Environment" Users Guide for detailed information.

3. Run the visualization application (in a **single-** or **multi-process** mode)

(Example of running the **pvserver** (ParaView Server) with 4 processes)

```
===========
Spack 0.21.0

[PrePost]$ . /vol0004/apps/oss/spack/share/spack/setup-env.sh

[PrePost]$ spack --version
0.21.0

[PrePost]$ spack find paraview arch=linux-rhel8-cascadelake

==> No package matches the query: paraview arch=linux-rhel8-cascadelake

===========
Spack 0.19.0

[PrePost]$ . /vol0004/apps/oss/spack-v0.19/share/spack/setup-env.sh

[PrePost]$ spack --version
0.19.0

[PrePost]$ spack find paraview arch=linux-rhel8-cascadelake

-- linux-rhel8-cascadelake / gcc@12.2.0 --------------------------
paraview@5.10.1
==> 1 installed package
```

(continues on next page)

(continued from previous page)

```
[PrePost]$ spack load paraview@5.10.1%gcc@12.2.0

===========
[PrePost]$ mpirun -n 4 pvserver

Example output:
Waiting for client...
Connection URL: cs://fn02sv02:11111
Accepting connection(s): fn02sv02:11111
```

Note: If there exist multiple versions for the same package, there is a need to specify the desired package by using the version (Ex.: 5.10.1), compiler (Ex.: gcc@12.2.0), architecture (Ex.: linux-rhel8-cascadelake), or hash value (Ex.: /6temuv4). Please check the Spack Users' Guide for detailed information about the Spack usage.

4. Start the client-side application

It is assumed that the Fugaku VPN Connection is already enabled, and the following **VPN Hostname** will be used for the corresponding **Hostname** shown after the **Connection URL: cs://** in the previous Step 3.

Hostname	VPN Hostname
fn02sv01	pps01.pp.fugaku.r-ccs.riken.jp
fn02sv02	pps02.pp.fugaku.r-ccs.riken.jp
fn02sv03	pps03.pp.fugaku.r-ccs.riken.jp
fn02sv04	pps04.pp.fugaku.r-ccs.riken.jp
fn03sv01	pps05.pp.fugaku.r-ccs.riken.jp
fn03sv02	pps06.pp.fugaku.r-ccs.riken.jp
fn03sv03	pps07.pp.fugaku.r-ccs.riken.jp
fn03sv04	pps08.pp.fugaku.r-ccs.riken.jp
fn02sv05	ppm01.pp.fugaku.r-ccs.riken.jp
fn03sv05	ppm02.pp.fugaku.r-ccs.riken.jp
fn06sv01	ppm03.pp.fugaku.r-ccs.riken.jp

(For the ParaView case, the server side information can be passed as an option during the startup)

For the example shown in the Step 3, the *VPN Hostname* will be **pps02.pp.fugaku.r-ccs.riken.jp** since the *Hostname* is **fn02sv02**.

```
[UserPC_2]$ ./paraview --server-url=cs://pps02.pp.fugaku.r-ccs.riken.jp:11111
```

Note: Using an additional CLI terminal (UserPC_2).

In the case of success, **Client connected.** message will appear on the server side.

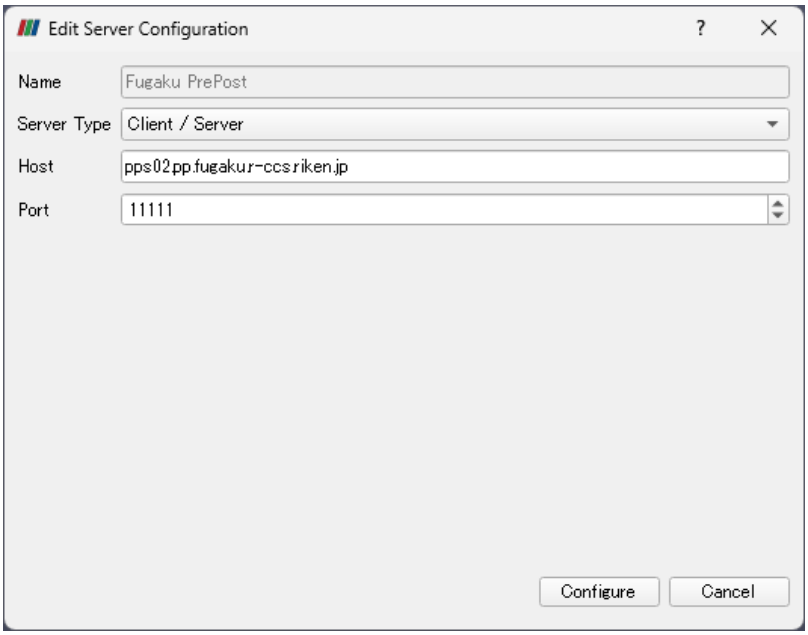
```
Waiting for client...
Connection URL: cs://fn02sv02:11111
Accepting connection(s): fn02sv02:11111
Client connected.
```

Server configuration via GUI:

It can be done via **Choose Server Configuration** window by selecting the **[File→Connect]** on the Tab Window or pressing the **[Connect]** button. New connection can be created via **[Add Server]** button. Except the Name, other initial information should be kept unmodified.

- Name: Fugaku

- Server Type: Client / Server
- Host: VPN Hostname (**pps02.pp.fugaku.r-ccs.riken.jp** for the example in the Step 3)
- Port 11111



ParaView GUI Client on Windows OS being connected to a ParaView Server running with 4 processes.

2.4.2 Compute Nodes

It is assumed that the Fugaku VPN Connection is already enabled, and in the example case of ParaView, a single- or multi-process ParaView Server (pvserver) runs on the server-side, and the ParaView GUI Client runs on the User's PC. Some of the commands are abbreviated, and please see the "Fugaku", and "Fugaku VPN Service" Users' Guide for detailed information.

1. Login to the Fugaku (**Login Node**)

```
[UserPC_1]$ ssh user@login.fugaku.r-ccs.riken.jp
```

2. Start an **Interactive Job** on the **Compute Nodes(s)**.

(Allocation example of 4 nodes)

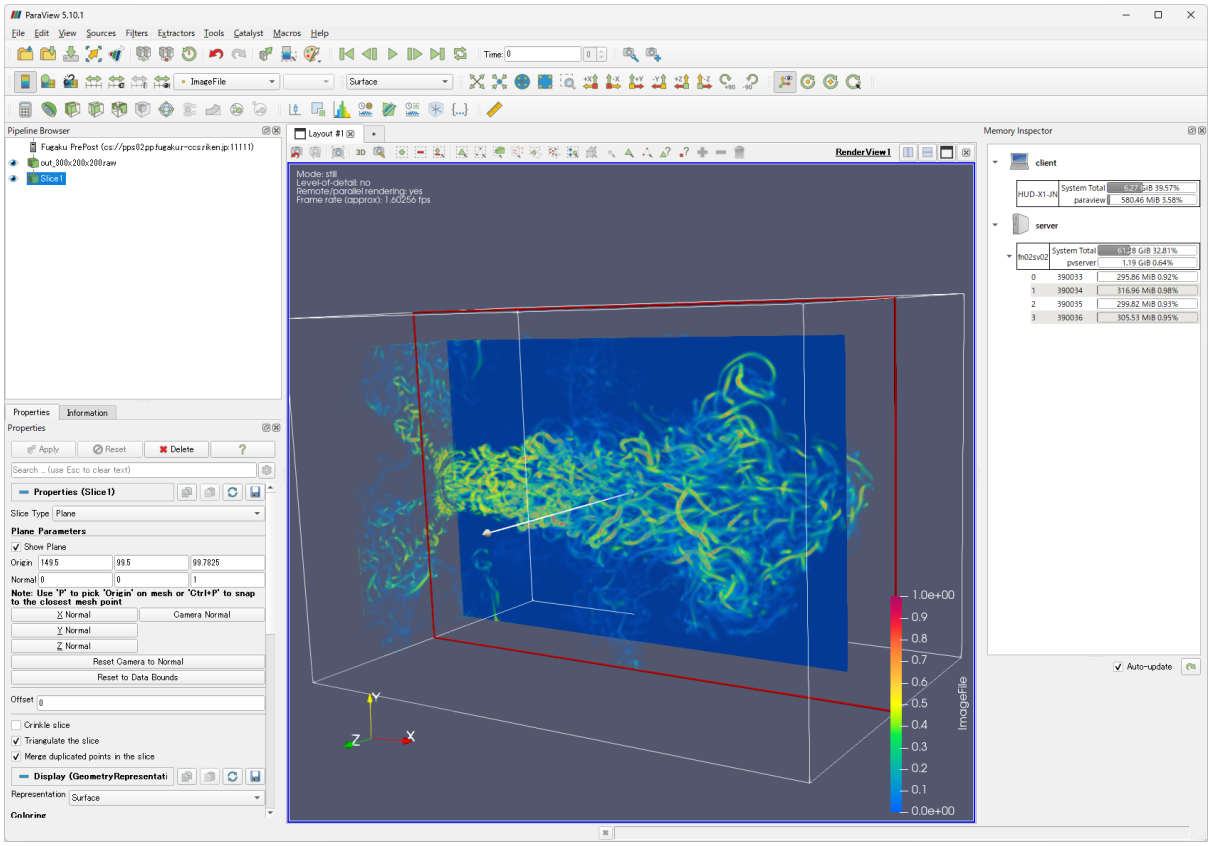
```
[Login]$  pjsub -x PJM_LLIO_GFSCACHE=/vol0004 --interact -L "eco_state=2" -L "node=4"␣
↪-L "rscgrp=int" -L "elapse=0:15:00" --sparam "wait-time=600"
```

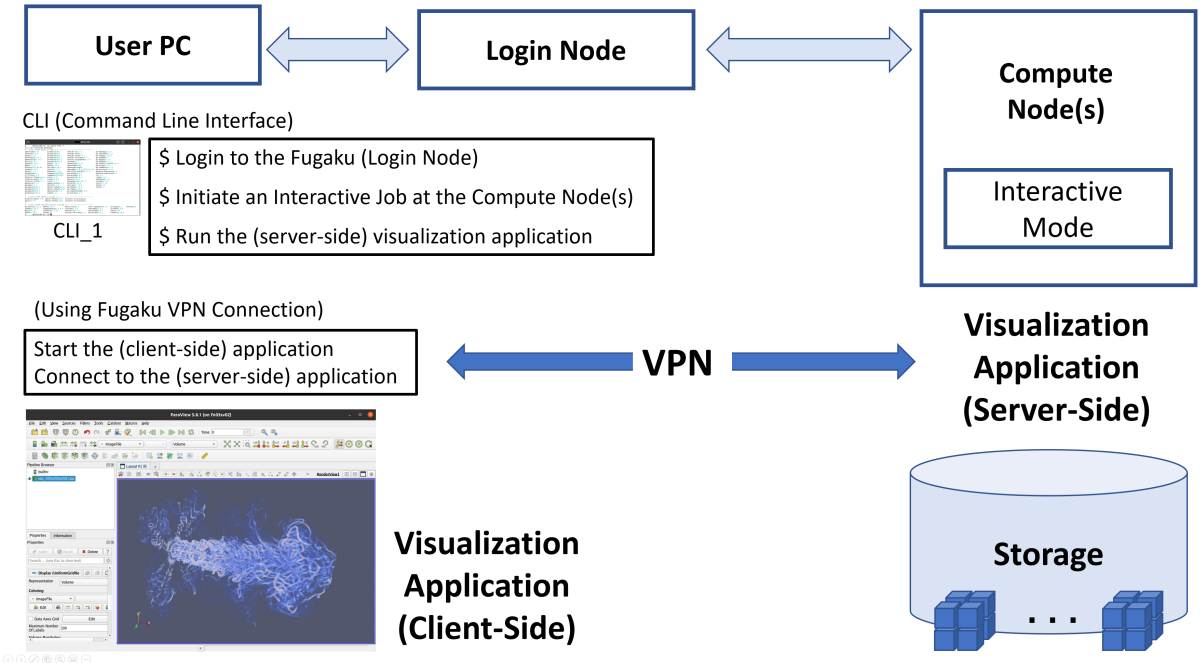
Note: Set the necessary options such as volume (Ex.: -x PJM_LLIO_GFSCACHE=/vol0004) and energy saving mode (Ex.: -L "eco_state=2"). In addiiton, allocate the same number of nodes as the processes since a long initialization time is required when running multiple processes on the same node. Please check the Fugaku Users Guide for detailed information about the Fugaku usage.

---

3. Start the server-side visualization application

(Example of running the **pvserver** (ParaView Server) with 4 processes)


ParaView 5.10.1
File Edit View Sources Filters Extractors Tools Catalyst Macros Help
Pipeline Browser
Properties
Information
Memory Inspector
RenderView1
Mode: still
Level-of-detail: no
Remote/parallel rendering: yes
Frame rate (approx): 1.60256 fps
ImageFile


User PC
Login Node
Compute Node(s)
Interactive Mode
CLI (Command Line Interface)
CLI_1
$ Login to the Fugaku (Login Node)
$ Initiate an Interactive Job at the Compute Node(s)
$ Run the (server-side) visualization application
(Using Fugaku VPN Connection)
Start the (client-side) application
Connect to the (server-side) application
VPN
Visualization Application (Server-Side)
Storage
Visualization Application (Client-Side)

```
[Compute]$ . /vol0004/apps/oss/spack/share/spack/setup-env.sh

===========
[Compute]$ spack find paraview

-- linux-rhel8-a64fx / fj@4.10.0 ---------------------------------
paraview@5.11.2  paraview@5.11.2  paraview@5.11.2
==> 3 installed packages

[Compute]$ spack load paraview +python +osmesa arch=linux-rhel8-a64fx

===========
[Compute]$ mpirun -n 4 pvserver

===========
Verify the stdout file if the server is ready (From another CLI terminal):
[Compute]$ cat output.[Job_ID]/0/[Execution_ID]/stdout.1.0

Waiting for client...
Connection URL: cs://[Compute_Node_ID]:11111
Accepting connection(s): [Compute_Node_ID]:11111
```

Note: The package version provided via Spack can differ from above in the case of a package update. Please check the Spack Users' Guide for detailed information about the Spack usage.

4. Start the client-side visualization application

It is assumed that the Fugaku VPN Connection is already enabled, and the "VPN Hostname" will be the hostname (shown after the "Connection URL: cs://" in the previous Step 3) added by the ".cn.fugaku.r-ccs.riken.jp". Please see the "Fugaku VPN Service" Users' Guide for detailed information.

Hostname	VPN Hostname
example	example.cn.fugaku.r-ccs.riken.jp

For instance, if the hostname is **a25-5012c**, then the VPN hostname will be **a25-5012c.cn.fugaku.r-ccs.riken.jp**.

```
[UserPC_2]$ ./paraview --server-url=cs://a25-5012c.cn.fugaku.r-ccs.riken.jp:11111
```

Note: Using an additional CLI terminal (UserPC_2).

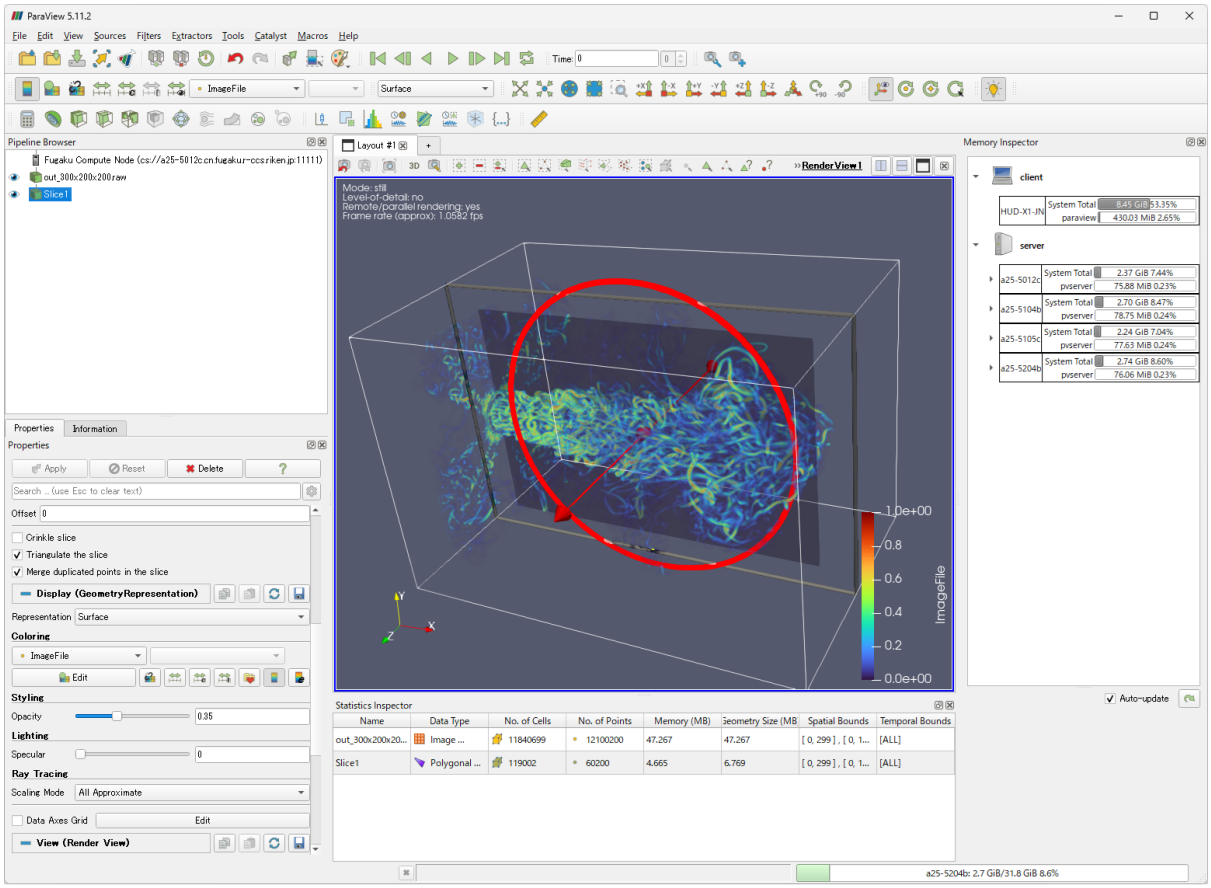
In the case of success, **Client connected.** message will appear on the server side.

```
Waiting for client...
Connection URL: cs://a25-5012c:11111
Accepting connection(s): a25-5012c:11111
Client connected.
```

(ParaView GUI Client connected to the Server running with 4 MPI processes at **a25-5012c.cn.fugaku.r-ccs.riken.jp**)

Server configuration via GUI

It can be done via **Choose Server Configuration** window by selecting the **[File->Connect]** on the Tab Window or pressing the **[Connect]** button. New connection can be created via **[Add Server]** button. Except the Name, other initial information should be kept unmodified.



- Name: Fugaku Compute Node
- Server Type: Client / Server
- Host: VPN Hostname (**a25-5012c.cn.fugaku.r-ccs.riken.jp** for the above example)
- Port 11111

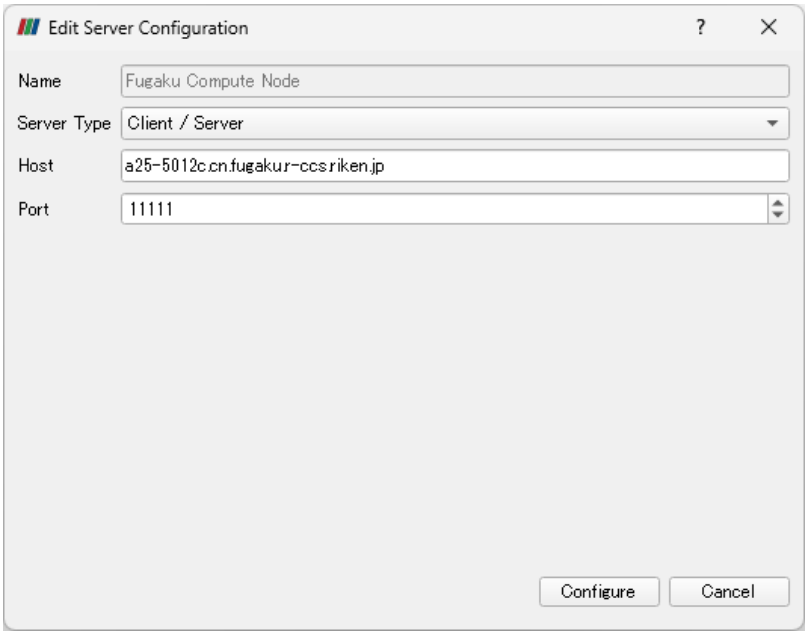
2.5 VNC (GNOME Desktop)

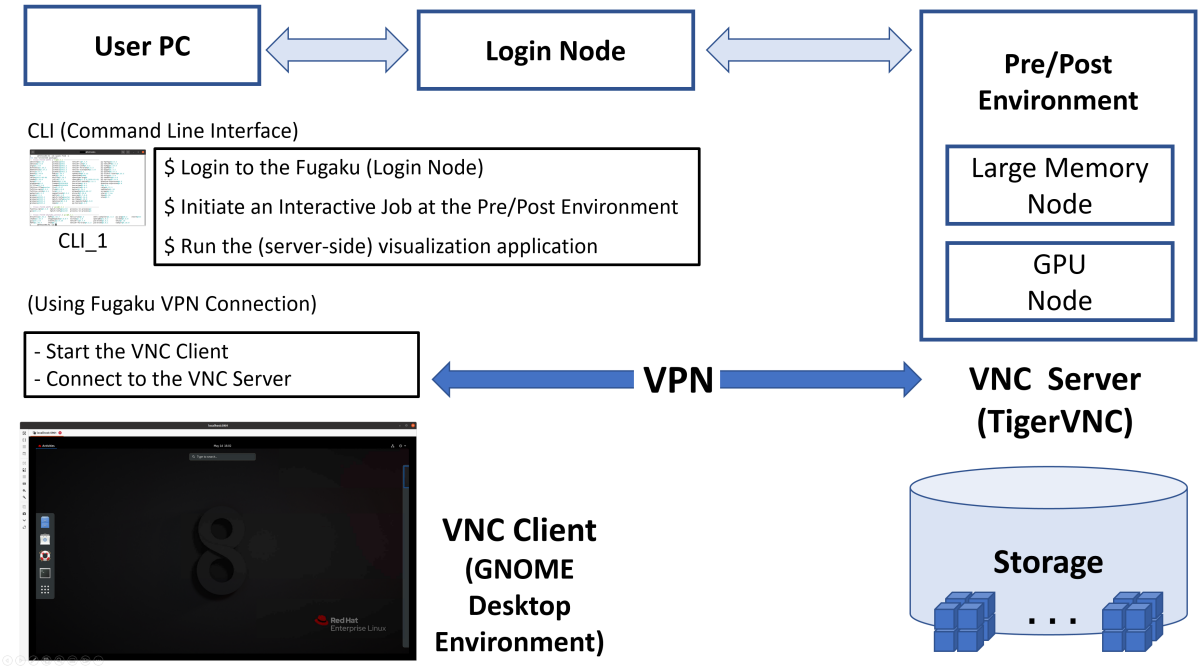
Use of visualization oriented application on the **GNOME Remote Desktop** via **VNC (Virtual Network Computing)** connection. Fugaku VPN service will be used to connect the VNC server (on the Fugaku side) and VNC client (on the User PC side). Remote Desktop can be used in the Open OnDemand environment with simpler settings, thus can be considered as a recommended alternative.

2.5.1 Pre/Post Environment

- VNC Server
  - TigerVNC
- VNC Client
  - Example: Remmina (Ubuntu 20)
- Port number: **5901 (Default)**

Note: VNC Client (Remmina) on a Linux PC (Ubuntu 20) is used just as an example usage, and other VNC Client applications from other OS (such as Windows and MacOS) can also be used as VNC Clients. VNC connection


Edit Server Configuration
Name
Fugaku Compute Node
Server Type
Client / Server
Host
a25-5012c.cn.fugaku.r-ccs.riken.jp
Port
11111
Configure
Cancel


User PC
Login Node
Pre/Post Environment
Large Memory Node
GPU Node
CLI (Command Line Interface)
CLI_1
$ Login to the Fugaku (Login Node)
$ Initiate an Interactive Job at the Pre/Post Environment
$ Run the (server-side) visualization application
(Using Fugaku VPN Connection)
- Start the VNC Client
- Connect to the VNC Server
VPN
VNC Server (TigerVNC)
VNC Client (GNOME Desktop Environment)
Storage

problem was reported on Anaconda installed environment. Please be aware that a paid license is required to use Anaconda at RIKEN.

Following are the step-by-step guide for using the GNOME Desktop (running at the Pre/Post Environment) as a remote desktop on the User's PC via VNC mechanism. It is assumed that the Fugaku VPN Connection is already enabled, and some of the commands are abbreviated, and please see the "Pre/Post Environment" and "Fugaku VPN Service" Users' Guide for detailed information.

1. Login to the Fugaku (**Login Node**).

```
[UserPC]$ ssh user@login.fugaku.r-ccs.riken.jp
```

2. Start an **Interactive Job** on the **Pre/Post Environment**.

(**GPU Node** usage example)

```
[Login]$ srun -p gpu1 -n 16 --mem 64000 --gpus=1 --time=0:15:00 --pty bash -i
```

Note: Specify the desired job queue (-p), number of CPUs (-n), memory size (–mem), GPU usage (–gpus=), and max. elapsed time (–time) according to the necessity. Please see the "Pre/Post Environment" Users Guide for detailed information.

---

3. Initial Setup (Only required on the first time use)

(Setup a password (for the VNC connection) and configure the startup script (for GNOME session))

a) Start a vncserver session

```
[PrePost]$ vncserver

You will require a password to access your desktops.

Password:
Verify:
Would you like to enter a view-only password (y/n)? n
A view-only password is not used

New 'fn02sv02:1 (user)' desktop is fn02sv02:1

Creating default startup script /home/group/user/.vnc/xstartup
Creating default config /home/group/user/.vnc/config
Starting applications specified in /home/group/user/.vnc/xstartup
Log file is /home/group/user/.vnc/fn02sv02:1.log
```

b) Close the vncserver session

```
[PrePost]$ vncserver -kill :1
Killing Xvnc process ID ProcessID
```

c) Configure the startup script (xstartup)

Include the command (gnome-session &) on the startup script file ($HOME/.vnc/xstartup) generated at the step a).

```
#!/bin/sh

unset SESSION_MANAGER
unset DBUS_SESSION_BUS_ADDRESS
/etc/X11/xinit/xinitrc
# Assume either Gnome will be started by default when installed
```

(continues on next page)

(continued from previous page)

```
# We want to kill the session automatically in this case when user logs out. In case␣
↪you modify
# /etc/X11/xinit/Xclients or ~/.Xclients yourself to achieve a different result, then␣
↪you should
# be responsible to modify below code to avoid that your session will be␣
↪automatically killed
if [ -e /usr/bin/gnome-session ]; then
    vncserver -kill $DISPLAY
fi
gnome-session &
```

4. Start a VNC Server session

```
[PrePost]$ vncserver

New 'fn02sv03:1 (user)' desktop is fn02sv03:1

Starting applications specified in /home/group/user/.vnc/xstartup
Log file is /home/group/user/.vnc/fn02sv03:1.log
```

It is possible to set the resolution (screen size) during the startup.

(Example: [PrePost]$ vncserver -geometry 1920x1080)

```
[PrePost]$ vncserver -help

usage: vncserver [:<number>] [-name <desktop-name>] [-depth <depth>]
                 [-geometry <width>x<height>]
                 [-pixelformat rgbNNN|bgrNNN]
                 [-fp <font-path>]
                 [-cc <visual>]
                 [-fg]
                 [-autokill]
                 [-noxstartup]
                 [-xstartup <file>]
                 <Xvnc-options>...
```

Warning message informing the presence of lock files (for the VNC server session) can appear when the VNC servers are not correctly closed. In this case, a VNC server session with subsequent numbering (of X Display) will start.

(In the following example ":2" will start because ":1" is locked)

```
[PrePost]$ vncserver

Warning: fn02sv02:1 is taken because of /tmp/.X1-lock
Remove this file if there is no X server fn02sv02:1

New 'fn02sv02:2 (user)' desktop is fn02sv02:2

Starting applications specified in /home/group/user/.vnc/xstartup
Log file is /home/group/user/.vnc/fn02sv02:2.log
```

By removing the lock files, the VNC server session will start from ":1". It is recommended to check the ownership of the lock files since there is a possiblity of mixing lock files from other users.

```
[PrePost]$ rm /tmp/.X?-lock
[PrePost]$ rm /tmp/.X11-unix/X?
```

5. Connect to the VNC Server (from a VNC Client)

It is assumed that the Fugaku VPN Connection is already enabled, and the **Hostname:X Display #** (shown after the **desktop is** at the Step 3) will be converted to **VPN Hostname:Port #** as shown in the following Table.

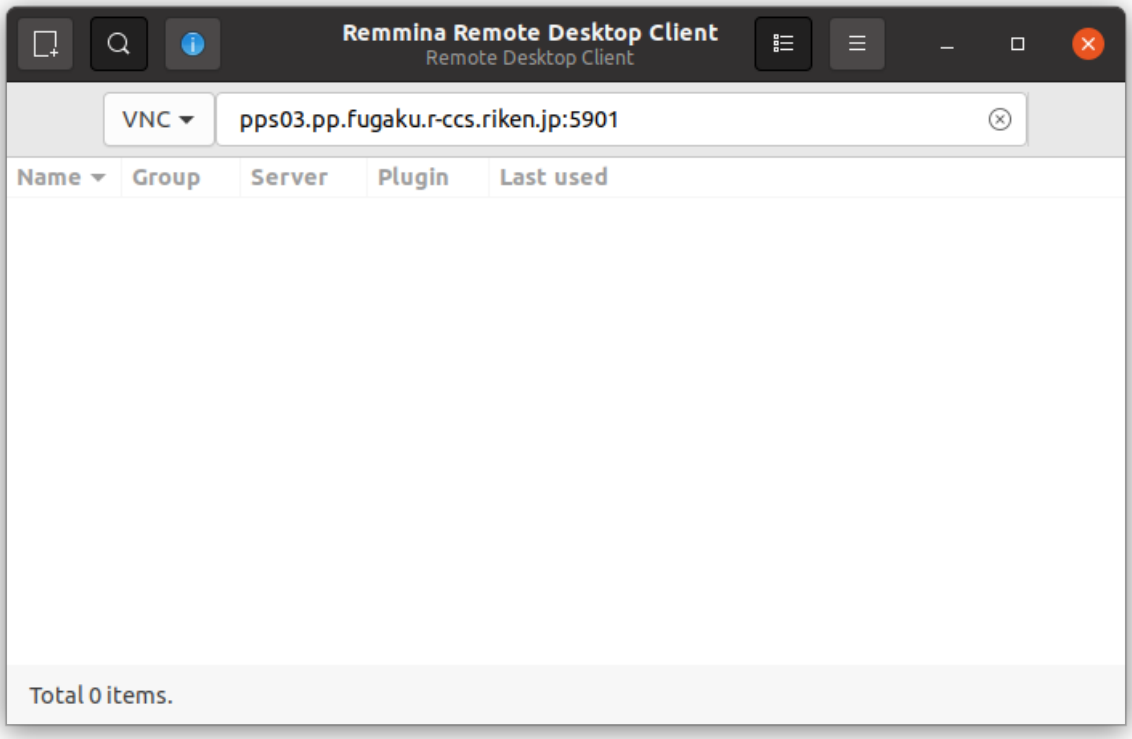
Hostname	VPN Hostname
fn02sv01	pps01.pp.fugaku.r-ccs.riken.jp
fn02sv02	pps02.pp.fugaku.r-ccs.riken.jp
fn02sv03	pps03.pp.fugaku.r-ccs.riken.jp
fn02sv04	pps04.pp.fugaku.r-ccs.riken.jp
fn03sv01	pps05.pp.fugaku.r-ccs.riken.jp
fn03sv02	pps06.pp.fugaku.r-ccs.riken.jp
fn03sv03	pps07.pp.fugaku.r-ccs.riken.jp
fn03sv04	pps08.pp.fugaku.r-ccs.riken.jp
fn02sv05	ppm01.pp.fugaku.r-ccs.riken.jp
fn03sv05	ppm02.pp.fugaku.r-ccs.riken.jp
fn06sv01	ppm03.pp.fugaku.r-ccs.riken.jp

X Display #	Port #
:1	5901
:2	5902
:3	5903
...	...

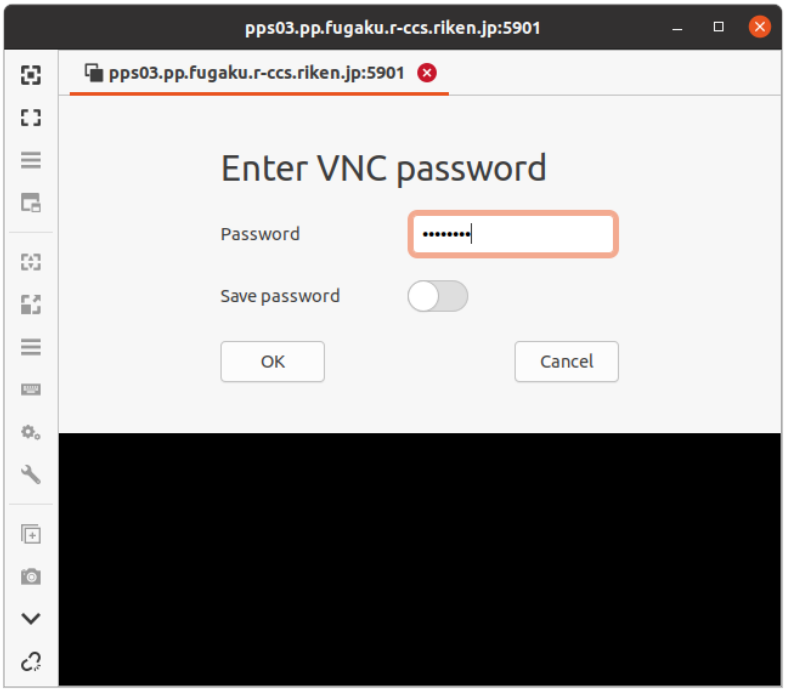
Connect to the **VPN Hostname:Port #**.

Since the **Hostname:X Display #** for the example (in the Step 4) is **fn02sv03:1**, then the **VPN Hostname:Port #** will be **pps03.pp.fugaku.r-ccs.riken.jp:5901**.

Example: Using the Remmina from Ubuntu



In the case of success, the VNC password window will appear.



After the password authentication process, the GNOME Desktop will appear on the VNC CLient. (For the first time use, it is expected to appear the following window [gnome-initial-setup])

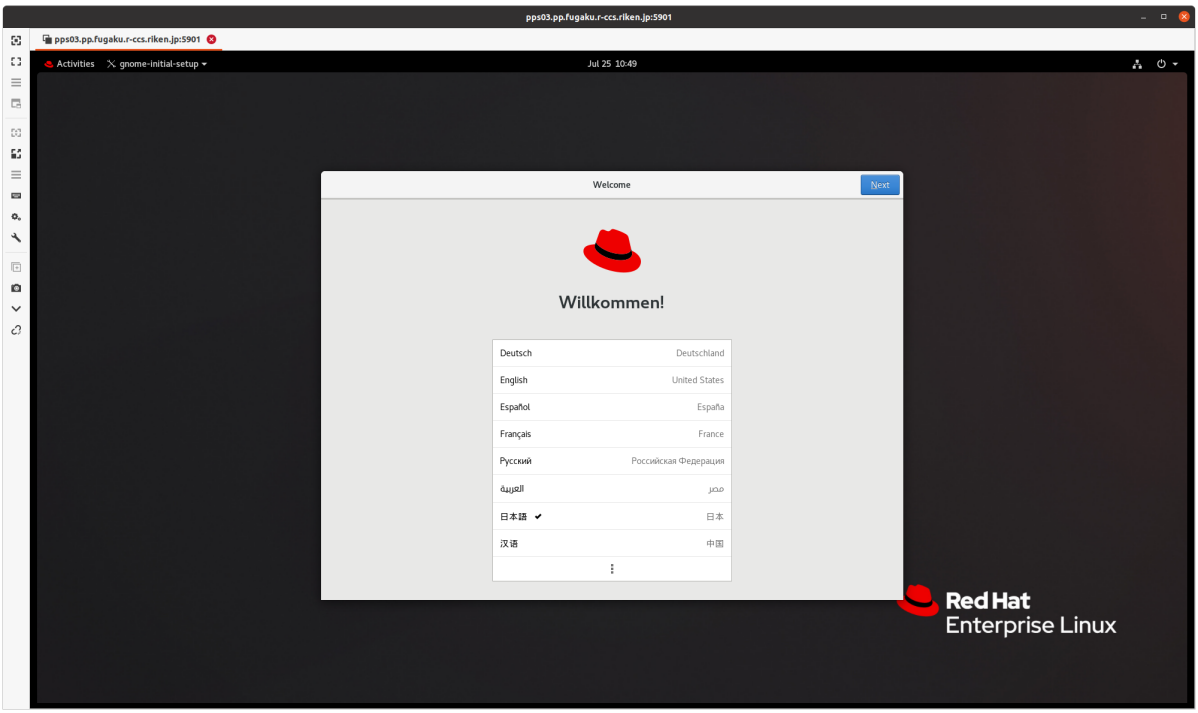
6. Closing the VNC server session

Use [-verify] to check the active VNC server sessions.

```
[PrePost]$ vncserver -list

TigerVNC server sessions:

X DISPLAY #  PROCESS ID
:1           ProcessID_1
:2           ProcessID_2
```

Use the X display number (X DISPLAY #) as a parameter for closing [-kill] the VNC server session.

```
[PrePost]$ vncserver -kill 1
Killing Xvnc process ID ProcessID_1
[PrePost]$ vncserver -kill :2
Killing Xvnc process ID ProcessID_2
```


pps03.pp.fugaku.r-ccs.riken.jp:5901
pps03.pp.fugaku.r-ccs.riken.jp:5901
Activities
gnome-initial-setup
Jul 25 10:49
Welcome
Next
Willkommen!
Deutsch
Deutschland
English
United States
Español
España
Français
France
Русский
Российская Федерация
العربية
مصر
日本語
日本
汉语
中国
Red Hat
Enterprise Linux

CHAPTER

THREE

AVS/EXPRESS

This subsection presents the step-by-step guide for using the AVS/Express, a genral purpose ISV visualization application, on the PrePost Environment or on a Local PC. Following are the information regarding the installation directory (Install Dir.), application directory (Application Dir.), installation files directory (Download Dir.), help information directory (Help Dir.), and AVS/Express license server.

AVS/Express	Developer 8.5.1
Install Dir.	/vol0004/apps/avs_xp851
Application Dir.	/vol0004/apps/avs_xp851/bin/linux_64/el8
Download Dir.	/vol0004/apps/avs_xp851/download
Help Dir.	/vol0004/apps/avs_xp851/runtime/help
License Server	fn01sv02 (login2.fugaku.r-ccs.riken.jp)

3.1 Open OnDemand

AVS/Express can be used on the Fugaku Open OnDemand (Xfce Desktop), as follows:

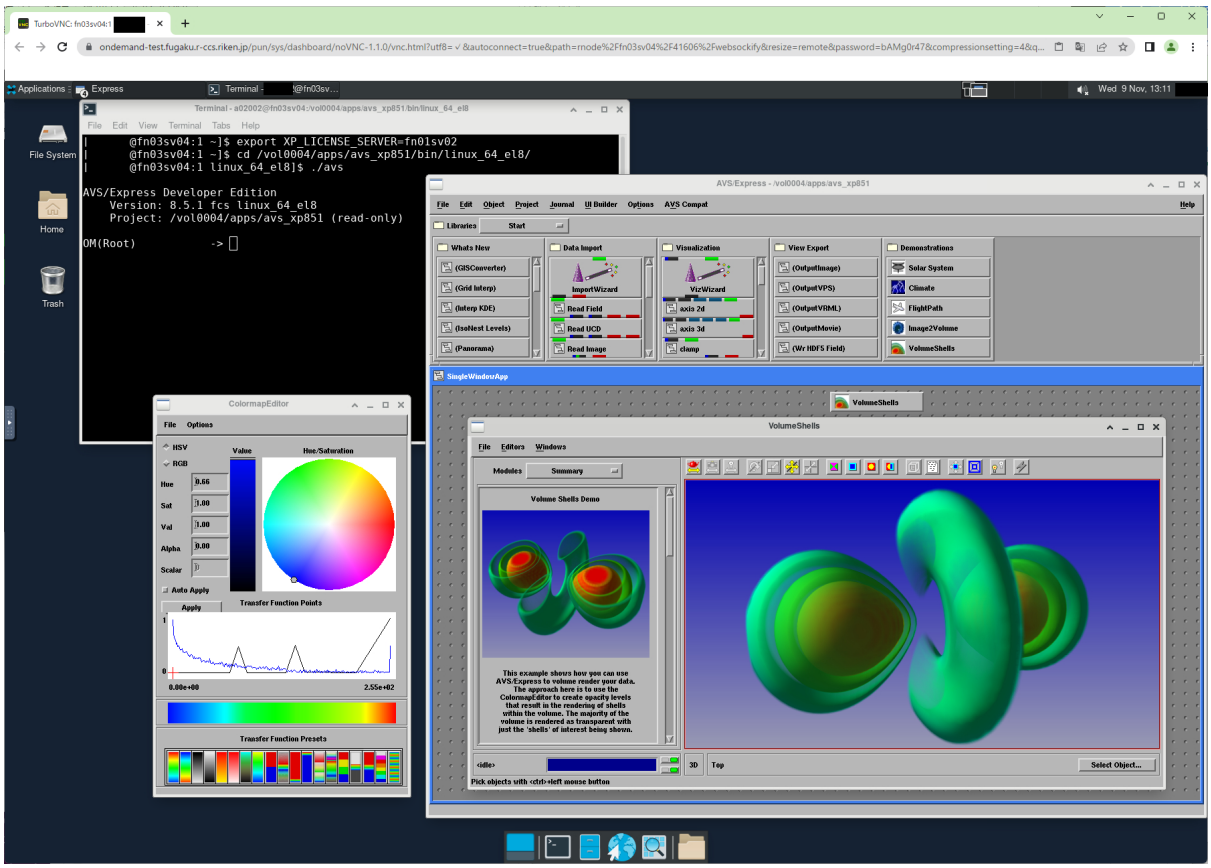
```
[PrePost]$ avs
```

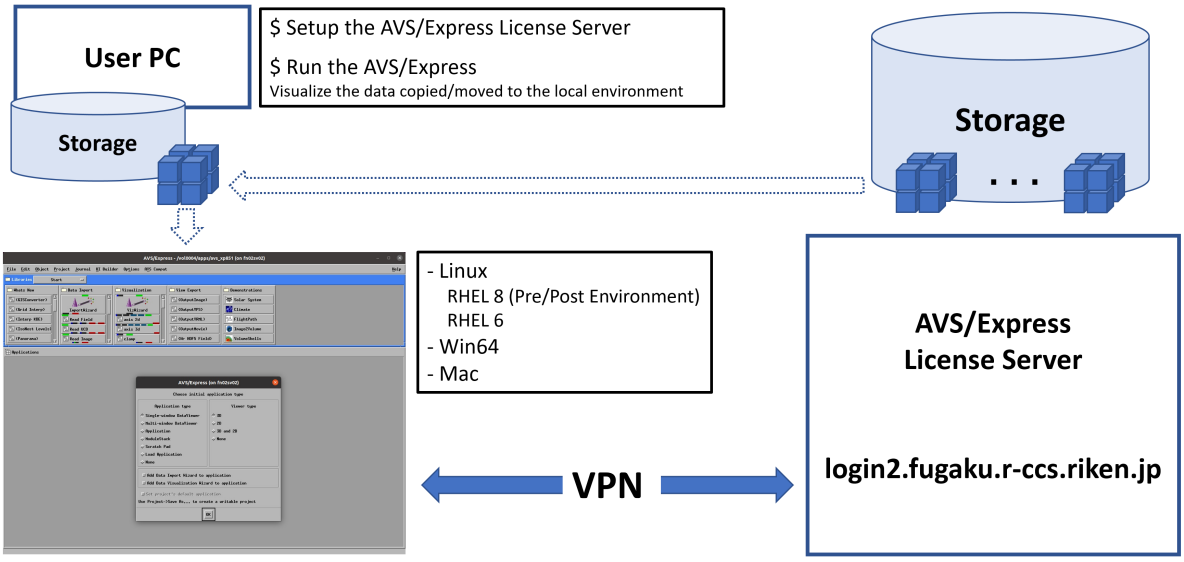
Note: Please refer to the "Fugaku Open OnDemand Guide", available on the Fugaku Website, for detailed information about the Open OnDemand usage.

3.2 User PC (Fugaku VPN Connection)

1. Install on the User PC

Download the corresponding installation file from the "Download Dir." shown in the beginning of this section, then decompress and proceed to the installation.

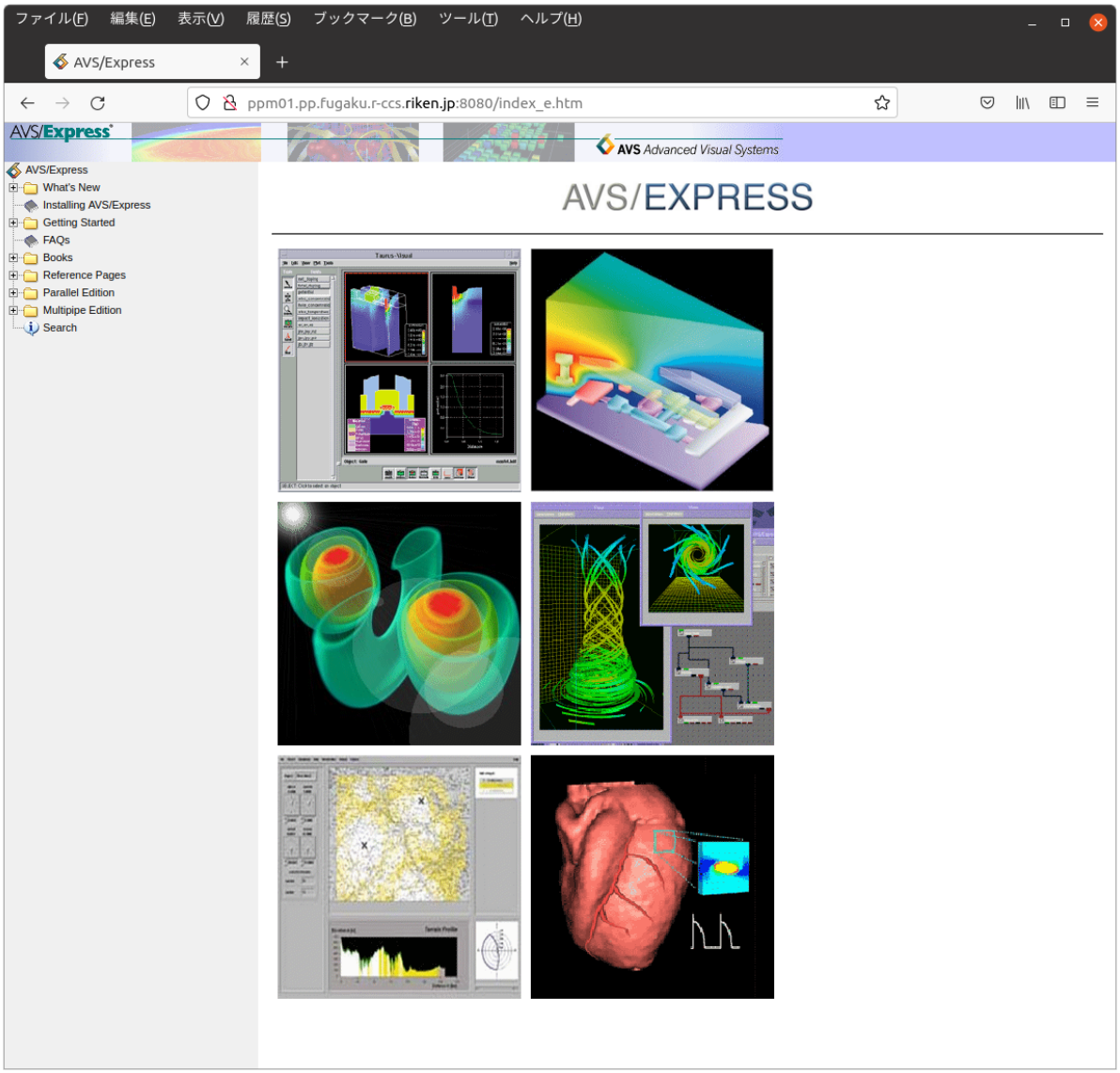

TurboVNC: fn03sv041
ondemand-test.fugaku.r-ccs.riken.jp/pun/sys/dashboard/noVNC-1.1.0/vnc.html?utf8=✓&autoconnect=true&path=rnode%2Ffn03sv04%2F41606%2Fwebsockify&resize=remote&password=bAMg0r47&compressionsetting=4&q...
Applications
Express
Terminal
Wed 9 Nov, 13:11
File System
Home
Trash
Terminal - a02002@fn03sv04:/vol0004/apps/avs_xp851/bin/linux_64_el8
File Edit View Terminal Tabs Help
@fn03sv04:1 ~]$ export XP_LICENSE_SERVER=fn01sv02
@fn03sv04:1 ~]$ cd /vol0004/apps/avs_xp851/bin/linux_64_el8/
@fn03sv04:1 linux_64_el8]$ ./avs
AVS/Express Developer Edition
Version: 8.5.1 fcs linux_64_el8
Project: /vol0004/apps/avs_xp851 (read-only)
OM(Root) ->
AVS/Express - /vol0004/apps/avs_xp851
File Edit Object Project Journal UI Builder Options AVS Compat Help
Libraries Start
What's New
Data Import
Visualization
View Export
Demonstrations
ImportWizard
Read Field
Read UCD
Read Image
VizWizard
axis 2d
axis 3d
clamp
Solar System
Climate
FlightPath
Image2Volume
VolumeShells
SimpleWindowApp
ColormapEditor
File Options
HSV
RGB
Value
Hue/Saturation
Hue
Sat
Val
Alpha
Scalar
Auto Apply
Apply
Transfer Function Points
Transfer Function Presets
VolumeShells
File Editors Windows
Modules
Summary
Volume Shells Demo
This example shows how you can use AVS/Express to volume render your data. The approach here is to use the ColormapEditor to create opacity levels that result in the rendering of shells within the volume. The majority of the volume is rendered as transparent with just the 'shells' of interest being shown.
<idle>
3D
Top
Select Object...
Pick objects with <ctrl>+left mouse button


User PC
Storage
$ Setup the AVS/Express License Server
$ Run the AVS/Express
Visualize the data copied/moved to the local environment
Storage
- Linux
RHEL 8 (Pre/Post Environment)
RHEL 6
- Win64
- Mac
VPN
AVS/Express
License Server
login2.fugaku.r-ccs.riken.jp

OS	Folder	File Package
RHEL8	Linux/linux_64_el8	express85.tgz
RHEL7	Linux/linux_64_el7	express85.tgz
RHEL6	Linux/linux_64_el6	express85.tgz
Windows	Win64	Win64.zip
OSX	Mac	darwin_64.zip

The “Installation Guide” is availble on the “Help Dir.”, shown in the beginning of this section, or in the “runtime/help” folder inside the decompressed installation package (express85.tgz) avaialble for Linux. Open the “index_e.htm” file from a Web browser to acess the “Installation Guide” as well as additional information such as “User’s Guide”.

(using the Firefox as the Web Browser)

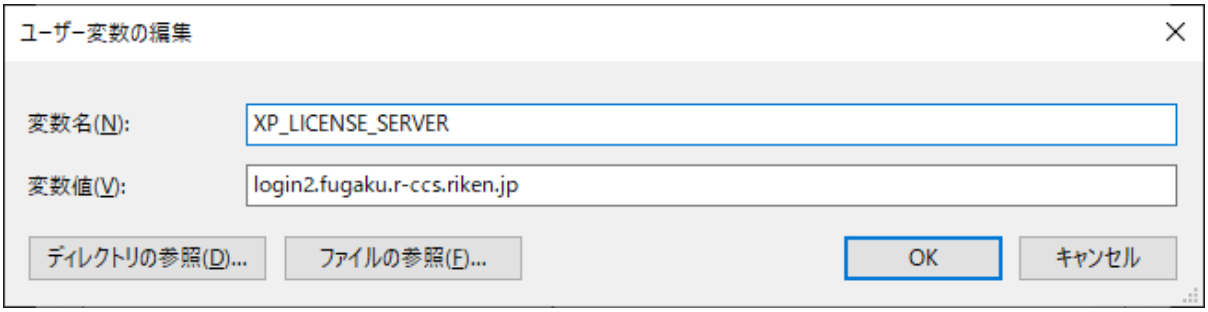


2. Setup the environment variable (AVS/Express License Server)

A) Linux

```
[UserPC]$ export XP_LICENSE_SERVER=login2.fugaku.r-ccs.riken.jp
```

B) Windows



3. Fugaku VPN Connection

It is necessary to have the Fugaku VPN Connection enabled to connect to the AVS/Express License Server running on the following Pre/Post Environment's node.

Hostname	VPN Hostname
fn01sv02	login2.fugaku.r-ccs.riken.jp

4. Run the AVS/Express

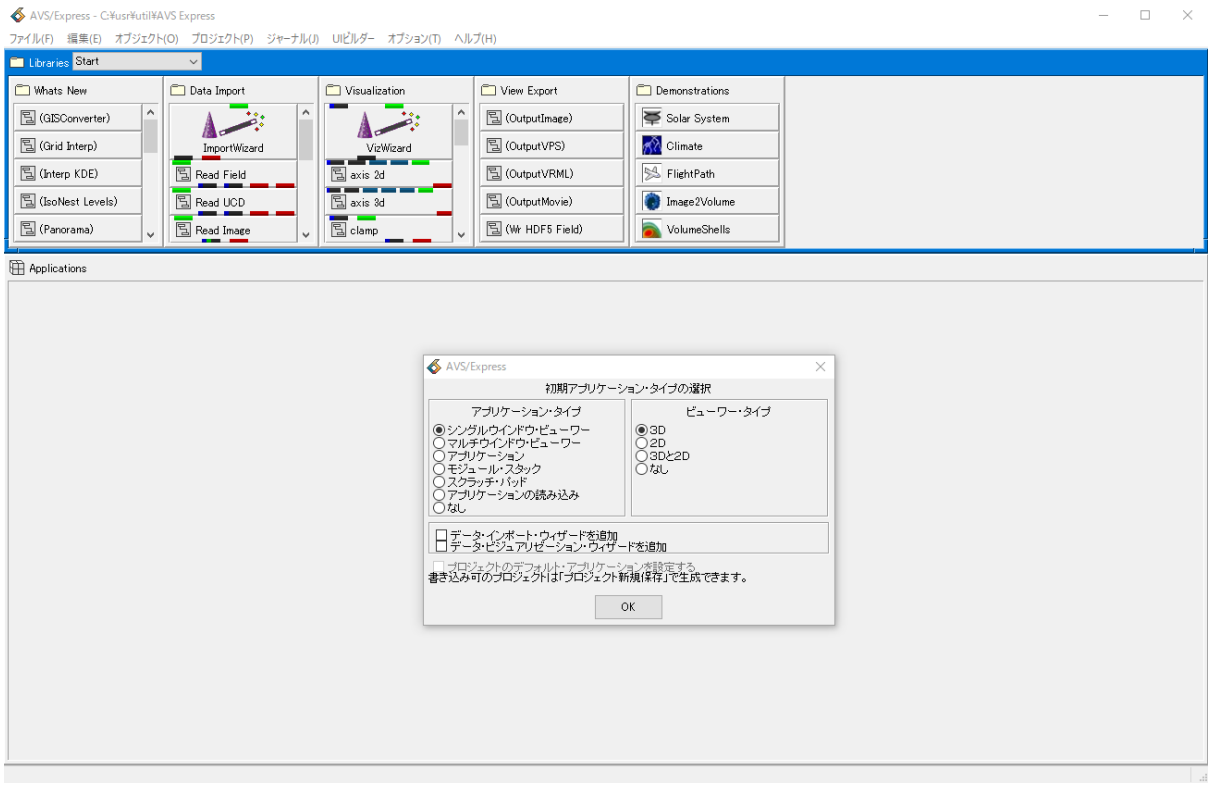
A) Linux

(Ex.: RHEL8)

```
[UserPC]$ cd bin/linux_64_el8/
[UserPC]$ ./express.static -mavs
```

B) Windows

On Windows 10, it is usually possible to run the AVS/Express by clicking the corresponding icon (Installed_Dir/bin/pc16_64/express.exe -mavs).



CHAPTER

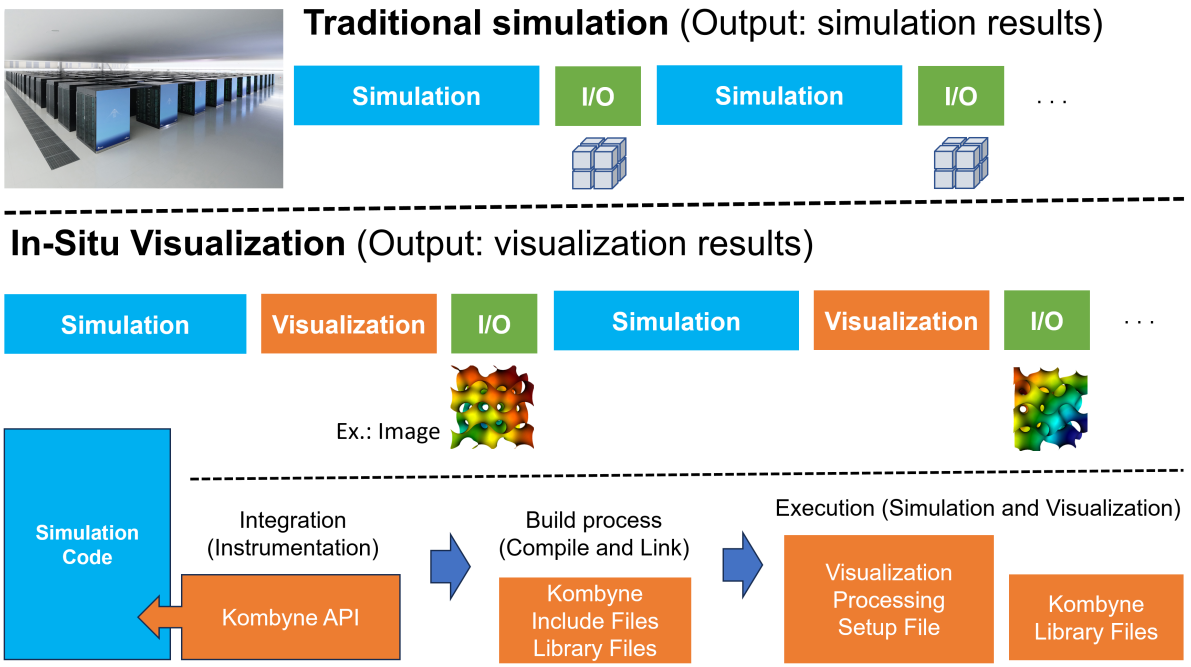
FOUR

KOMBYNE

Kombyne is an *In-Situ Visualization*-oriented commercial ISV tool that enables visualization processing during the simulation runs, and is currently provided in experimental mode.

As shown below, unlike traditional simulation where the results are output to storage for subsequent visualization and analysis tasks, visualization processing is performed while the simulation result is still in the memory, and then outputs the visualization result to the storage. As a result, it is expected to greatly reduce the I/O cost and has recently been receiving increasing attention. This approach was also mentioned in the HPCI Roadmap 2023 (https://cs-forum.github.io/roadmap-2023/) [in Japanese].

It is worth mentioning that there are other OSS-based in-situ visualization tools (e.g.: Catalyst, Libsim, and KVS), but in the Kombyne case, technical user support can be expected from the developer (via Fugaku Support Site).



Note: The above "Visualization" processing includes not only the rendering process that outputs image (PNG format), but also the intermediate data generation process such as during polygon data extraction of an isosurface.

Since in-situ visualization performs visualization processing in conjunction with simulation, thus it is necessary to integrate the provided Kombyne API into the target simulation code. Currently supported programming languages are as follows.

- C/C++
- FORTRAN
- Julia

- Python

Following is the installation path (Install Dir.) on the Fugaku. The subdirectories contain the libraries (lib64) and include files necessary for implementation and execution, as well as examples and the manual (docs).

Kombyne	Version 1.5
Install Dir.	/vol0004/apps/opt/kombyne
Library Dir.	/lib64
Include Dir.	/include
Examples	/examples
Manual	/docs/Kombyne.pdf

It is recommended to set in advance the following environment variables (PATH and LD_LIBRARY_PATH) to facilitate its use.

```
[Compute]$ export PATH=/vol0004/apps/opt/kombyne/bin:$PATH
[Compute]$ export LD_LIBRARY_PATH=/vol0004/apps/opt/kombyne/lib64:$LD_LIBRARY_PATH
```

Following **/bin/kombyne-config** tool can be used to obtain the compiler and linker flags for building C/C++ and FORTRAN codes integrated with Kombyne API.

```
[Compute]$ kombyne-config --c-flags
[Compute]$ kombyne-config --c-libs

[Compute]$ kombyne-config --cxx-flags
[Compute]$ kombyne-config --cxx-libs

[Compute]$ kombyne-config --fortran-flags
[Compute]$ kombyne-config --fortran-libs
```

Below is the output example when using FORTAN code.

```
[Compute]$ kombyne-config --fortran-flags
-DMPICH_SKIP_MPICXX -DOMPI_SKIP_MPICXX -D_MPICC_H -I/vol0004/apps/opt/kombyne/
↪thirdparty/ZeroMQ/include -I/vol0004/apps/opt/kombyne/include/kombyne -I/vol0004/
↪apps/opt/kombyne/thirdparty/Conduit/include -I/vol0004/apps/opt/kombyne/thirdparty/
↪Conduit/include/conduit

[Compute]$ kombyne-config --fortran-libs
-Wl,-rpath,/vol0004/apps/opt/kombyne/lib64:/vol0004/apps/opt/kombyne/thirdparty/
↪ZeroMQ/lib64:/vol0004/apps/opt/kombyne/thirdparty/Conduit/lib:/vol0004/apps/opt/
↪kombyne/thirdparty/ADIOS2/lib64 /vol0004/apps/opt/kombyne/lib64/libkombyne.so /
↪vol0004/apps/opt/kombyne/thirdparty/ZeroMQ/lib64/libzmq.so.5.2.2 /vol0004/apps/opt/
↪kombyne/thirdparty/Conduit/lib/libconduit_relay.so -ldl -lrt /vol0004/apps/opt/
↪kombyne/thirdparty/Conduit/lib/libconduit_blueprint.so /vol0004/apps/opt/kombyne/
↪thirdparty/Conduit/lib/libconduit.so -Wl,-rpath-link,/vol0004/apps/opt/kombyne/
↪thirdparty/ADIOS2/lib64
```

(continues on next page)

(continued from previous page)

Following is an example of executing in-situ visualization using **bin/producer**, that generates data for visualization, and **examples/pipeline/render.yaml** that defines the desired rendering processing.

```
[Compute]$ cp /vol0004/apps/opt/kombyne/bin/producer .
[Compute]$ cp /vol0004/apps/opt/kombyne/examples/pipelines/render.yaml .

4 MPI processs
Domain decomposition: 2x2x1
Time steps: 4
[Compute]$ mpirun -n 4 ./producer -domains 2,2,1 -nsteps 4 -pipeline ./render.yaml

Output (PNG image files) of visualization processing, defined in "render.yaml"
[Compute]$ ls *.png
render000000.png  render000001.png  render000002.png  render000003.png
```

Output: render000000.png