

Scientific Visualization at Jülich Supercomputing Centre

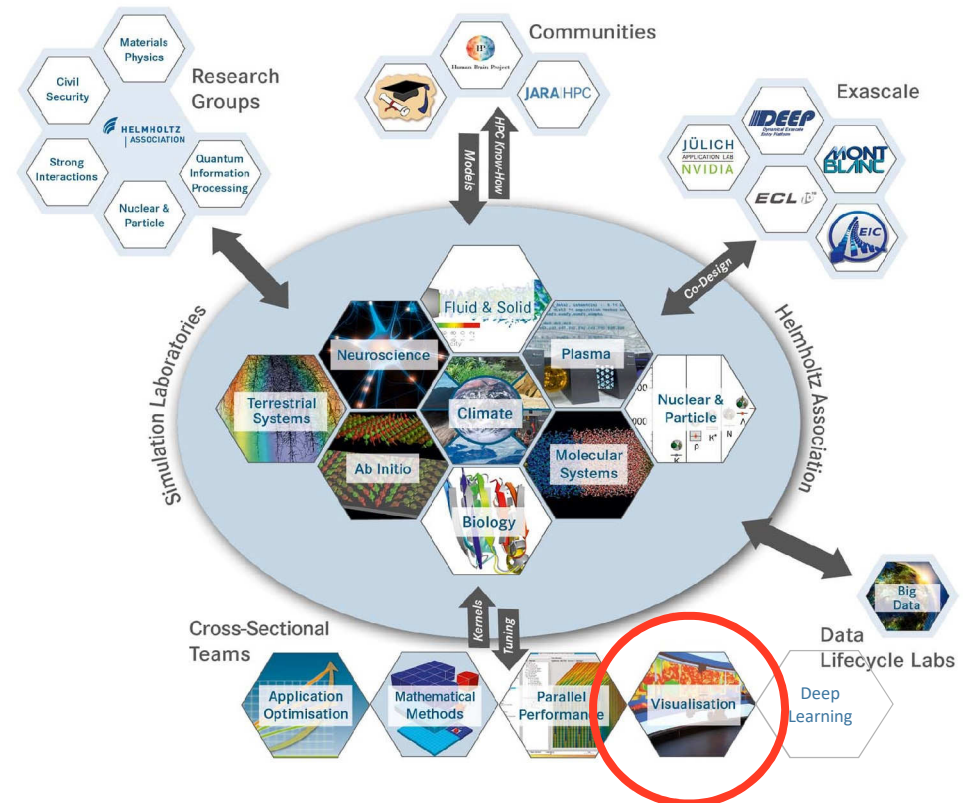
¹ Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, Germany
Cross-Sectional-Team „Visualization“
h.zilken@fz-juelich.de, j.goebbert@fz-juelich.de

Visualization at JSC

Cross-Sectional Team “Visualization”

- **Scientific Visualization**
 - R&D + support for visualization of scientific data
- **Virtual Reality**
 - VR systems for the analysis and presentation
- **Multimedia**
 - multimedia productions for websites, presentations or on TV

Domain-specific User Support and Research at JSC



Visualization at JSC

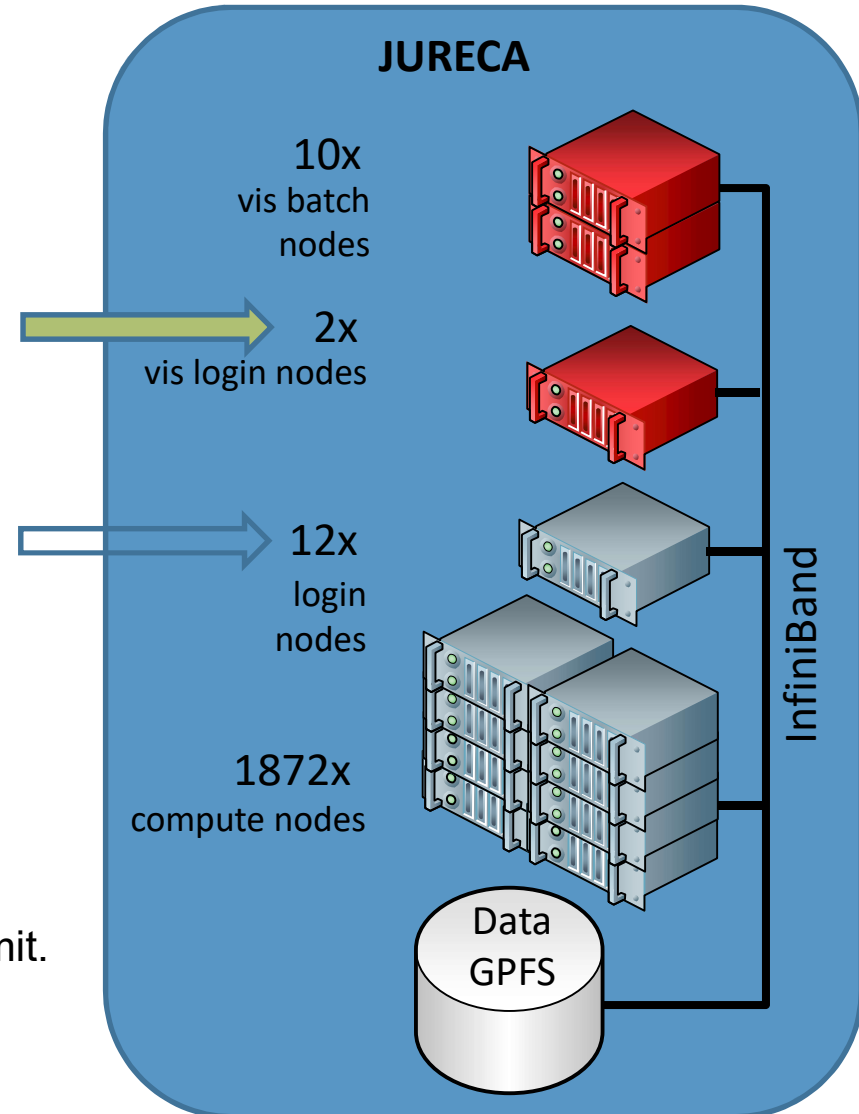
General Hardware Setup

12x Visualization Nodes

- 2 GPUs Nvidia Tesla K40 per node
- 12 GB RAM on each GPU
- **2x Vis. Login Nodes**
 - jurecavis.fz-juelich.de
 - (*jurecavis01 or jurecavis02 in round-robin fashion*)
 - 512 GB RAM each
- **10x Vis. Batch Nodes**
 - 8 nodes with 512 GB RAM
 - 2 nodes with 1024 GB RAM
 - special partition: **vis**
 - Use vis. batch nodes via job submit.

Keep in mind:

Visualization is **NOT** limited to vis. nodes **ONLY**.
(software rendering is possible on any node)



Visualization at JSC

General Software Setup

Special Software Stack on Vis Nodes:

Base Software:



X-Server, X-Client (Window-Manager)



OpenGL (libGL.so, libGLU.so, libglx.so), Nvidia

Middleware:



Virtual Network Computing: VNC-Server, VNC-Client



VirtualGL



Strudel

Parallel and Remote Rendering Apps, In-Situ Visualization:



ParaView



Visit

Other Visualization Packages:

VMD, PyMol, Blender, GPicView, GIMP

Visualization at JSC

Usage Model for Vis Nodes

JURECA projects:

- Every project gets a small contingent (1000 core h) for vis nodes in addition to project contingent
- Contingent is valid for partition “vis”
- If you need more compute time for visualization: send a request to sc@fz-juelich.de

JUWELS projects:

- Visualization will be possible on 4 login nodes
- No visualization batch nodes
- As of today, visualization software stack not installed, available end Q1 2019
- If you need visualization right away: send a request to sc@fz-juelich.de to get contingent on JURECA

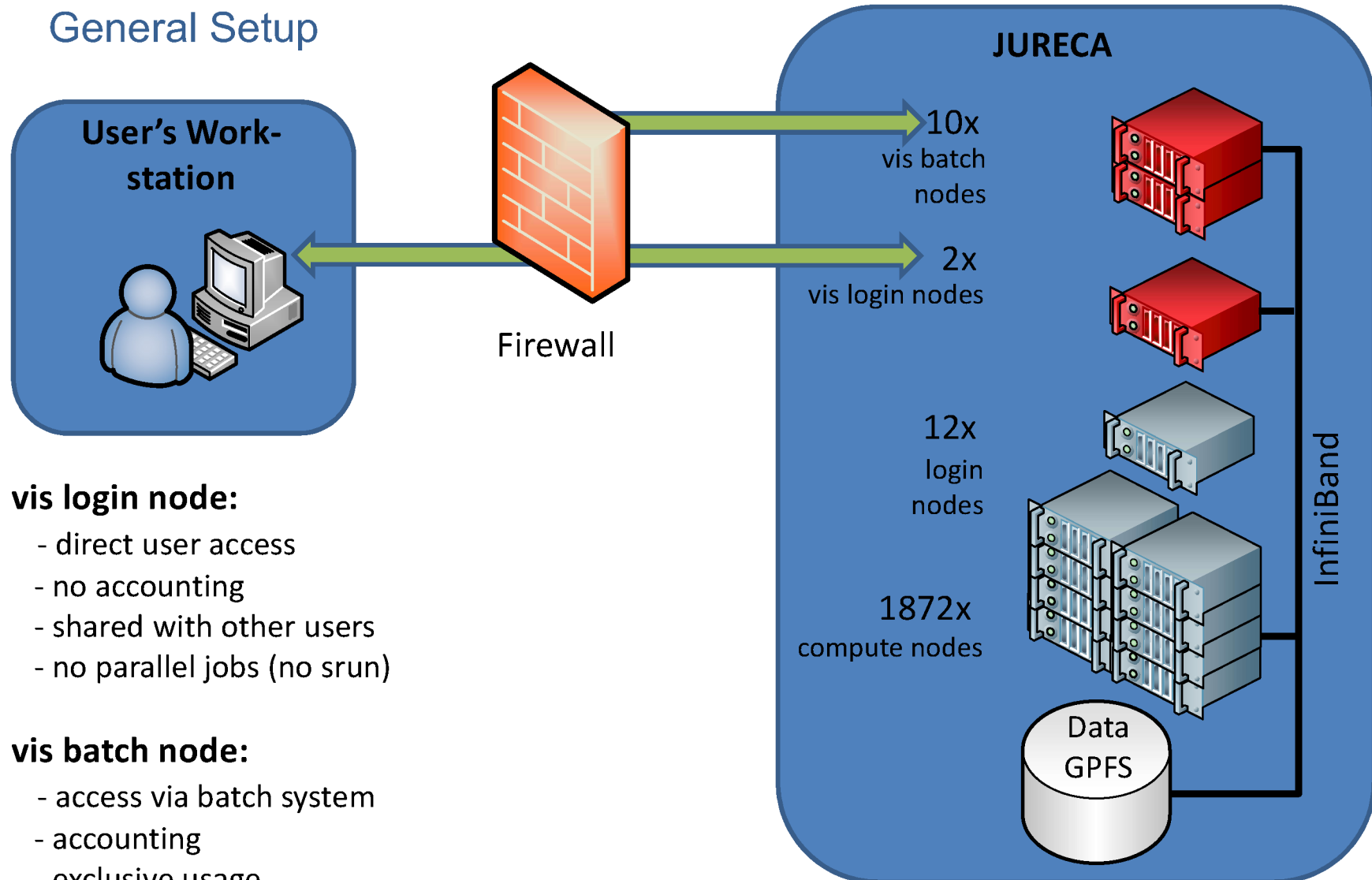
Non HPC-Project Users:

- apply for test project
- get a small contingent for vis nodes (1000 core h)

Remote 3D Visualization

Remote 3D Visualization

General Setup



vis login node:

- direct user access
- no accounting
- shared with other users
- no parallel jobs (no srun)

vis batch node:

- access via batch system
- accounting
- exclusive usage
- parallel jobs possible

Remote 3D Visualization

at Jülich Supercomputing Centre

- X forwarding + Indirect Rendering
slow, maybe incompatible → bad idea
- “remote aware” visualization apps (ParaView, VisIt)
application dependent error-prone setup
- VNC (Virtual Network Computing) + VirtualGL
our recommendation → good idea
- Xpra - stream application content with H.264 + VirtualGL
alternative recommendation → good idea

Remote 3D Visualization

with X Forwarding + Indirect Rendering

Traditional Approach (X forwarding + Indirect Rendering)
ssh -X <USERID>@<SERVER>

- uses GLX extension to X Window System
- X display runs on user workstation
- OpenGL commands are encapsulated inside X11 protocol stream
- OpenGL commands are executed on user workstation
- **disadvantages**
 - User's workstation requires a running **X server**.
 - User's workstation requires a **graphic card** capable of the required OpenGL.
 - User's workstation defines the **quality and speed** of the visualization.
 - User's workstation requires **all data needed** to visualize the 3d scene.
 - This approach is known to be error prone (OpenGL version mismatch, ...)

Try to **AVOID** for 3D visualization.

Remote 3D Visualization

with VNC (Virtual Network Computing) + VirtualGL

State-of-the-Art Approach (VNC with VirtualGL)
vncserver, vncviewer



- platform independent
- application independent
- session sharing possible
- **advantages**
 - **No X is required** on user's workstation (X display on server, one per session).
 - **No OpenGL is required** on user's workstation (only images are send).
 - Quality of visualization does **not depend** on user's workstation.
 - Data size send is **independent** from data of 3d scene.
 - Disconnection and reconnection possible.

<http://www.virtualgl.org>

Try to **USE** this approach for 3D visualization.

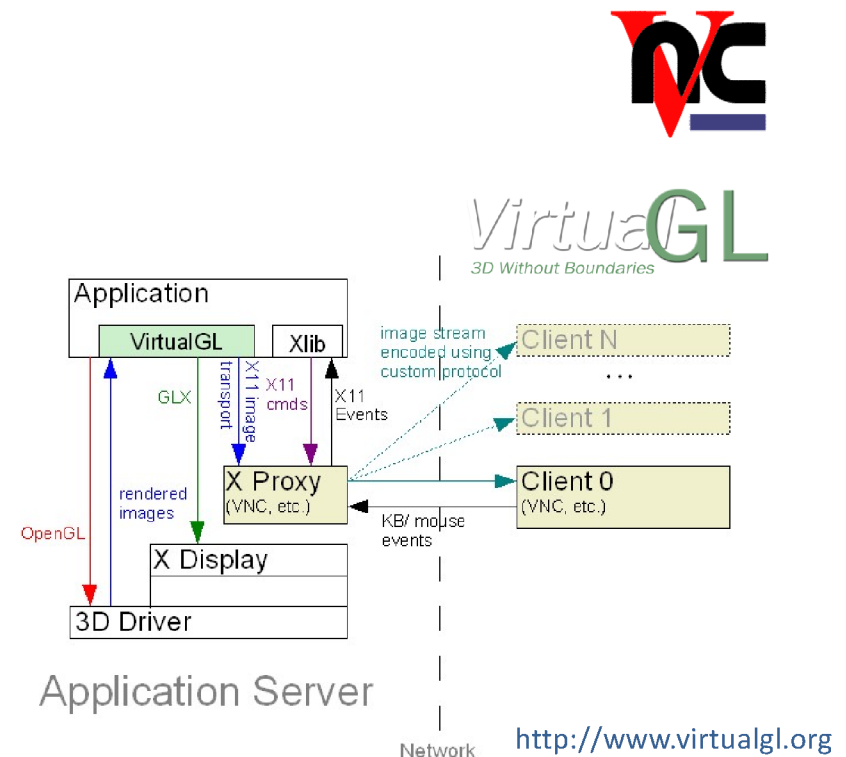
Remote 3D Visualization

with VNC (Virtual Network Computing) + VirtualGL

VNC + VirtualGL

vglrun <application>

- OpenGL applications send both GLX and X11 commands to the same X display.
- Once **VirtualGL** is preloaded into an OpenGL application, it **intercepts the GLX** function calls from the application and **rewrites them**.
- The corresponding GLX commands are then sent to the X display of **the 3d X server**, which has a 3D hardware accelerator attached.



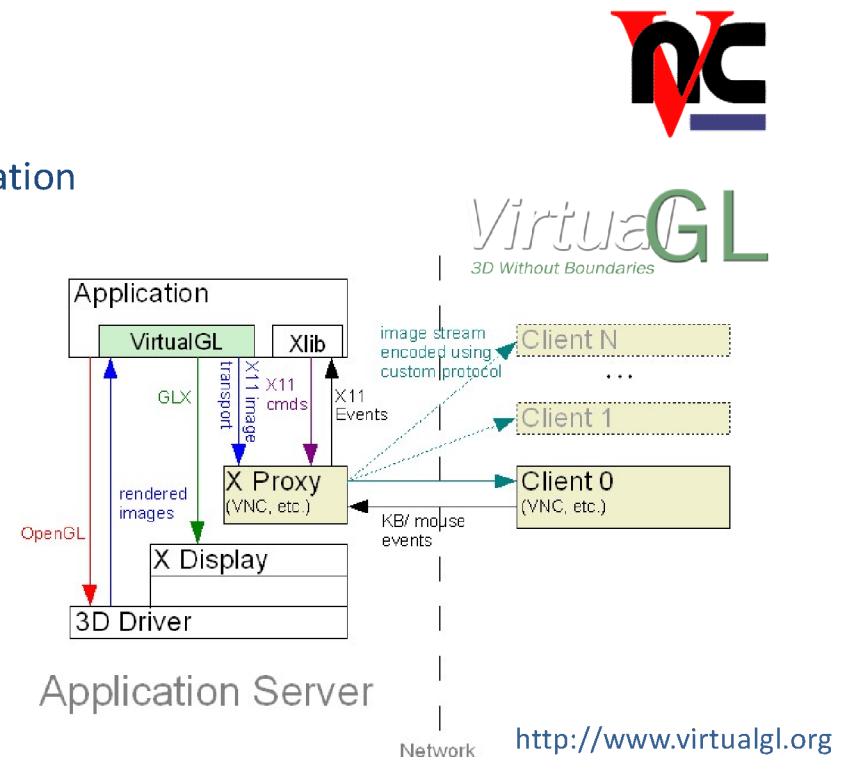
Remote 3D Visualization

with VNC (Virtual Network Computing) + VirtualGL

VNC + VirtualGL

vglrun <application>

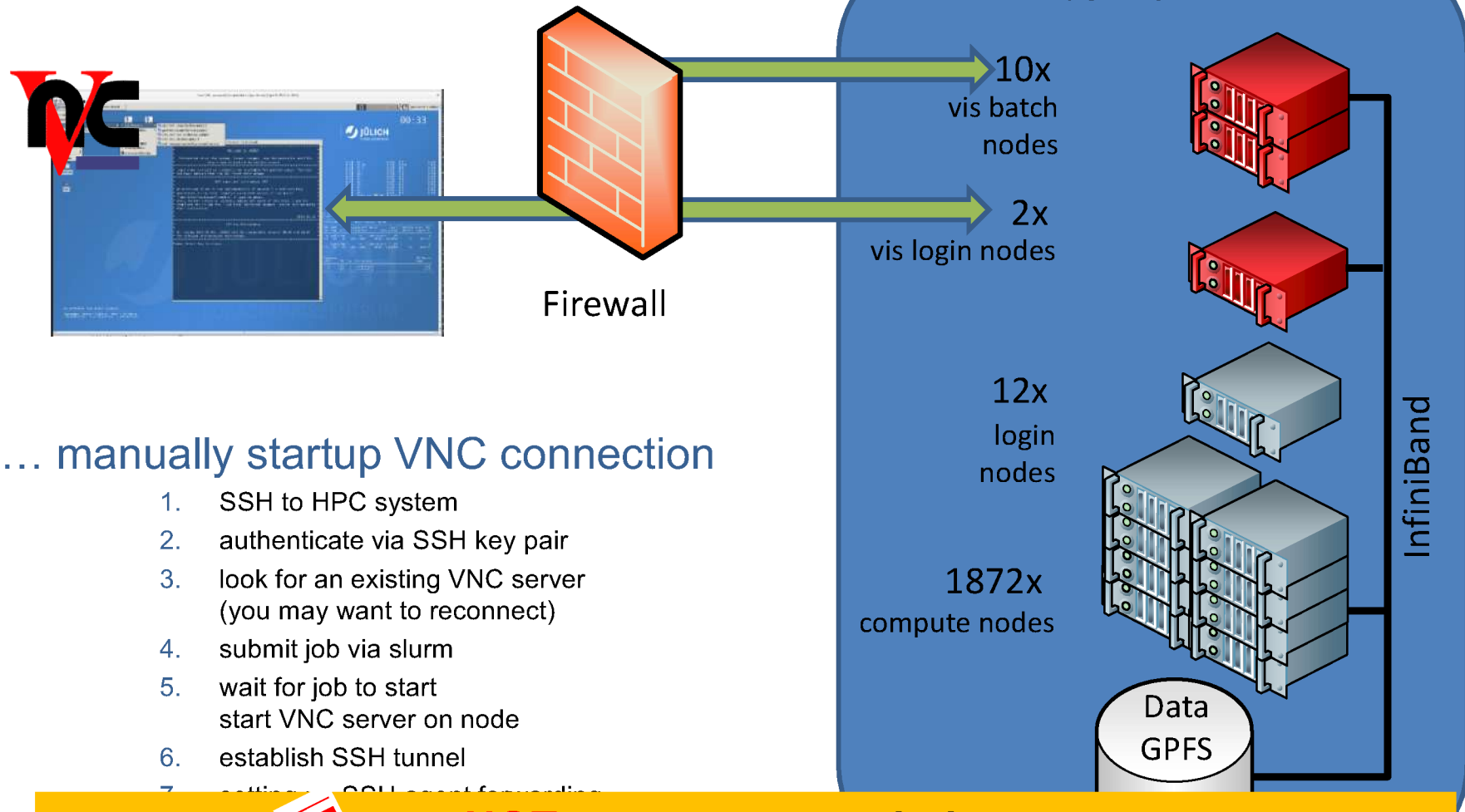
- Recommended solution for **any OpenGL application**
e.g. ParaView, VisIt, IDL, VMD, PyMol, ...
- Allows fast and reliable **server-side hardware rendering**
(GPU acceleration) with VirtualGL
- User only installs local VNC viewer.
- Desktop sharing possible
- Should also be used for the frontend
of “remote aware” applications
(e.g. for ParaView and VisIt, ...)



**Our recommendation:
Use VNC for remote rendering on JURECA.**

Remote 3D Visualization

with VNC + VirtualGL



... manually startup VNC connection

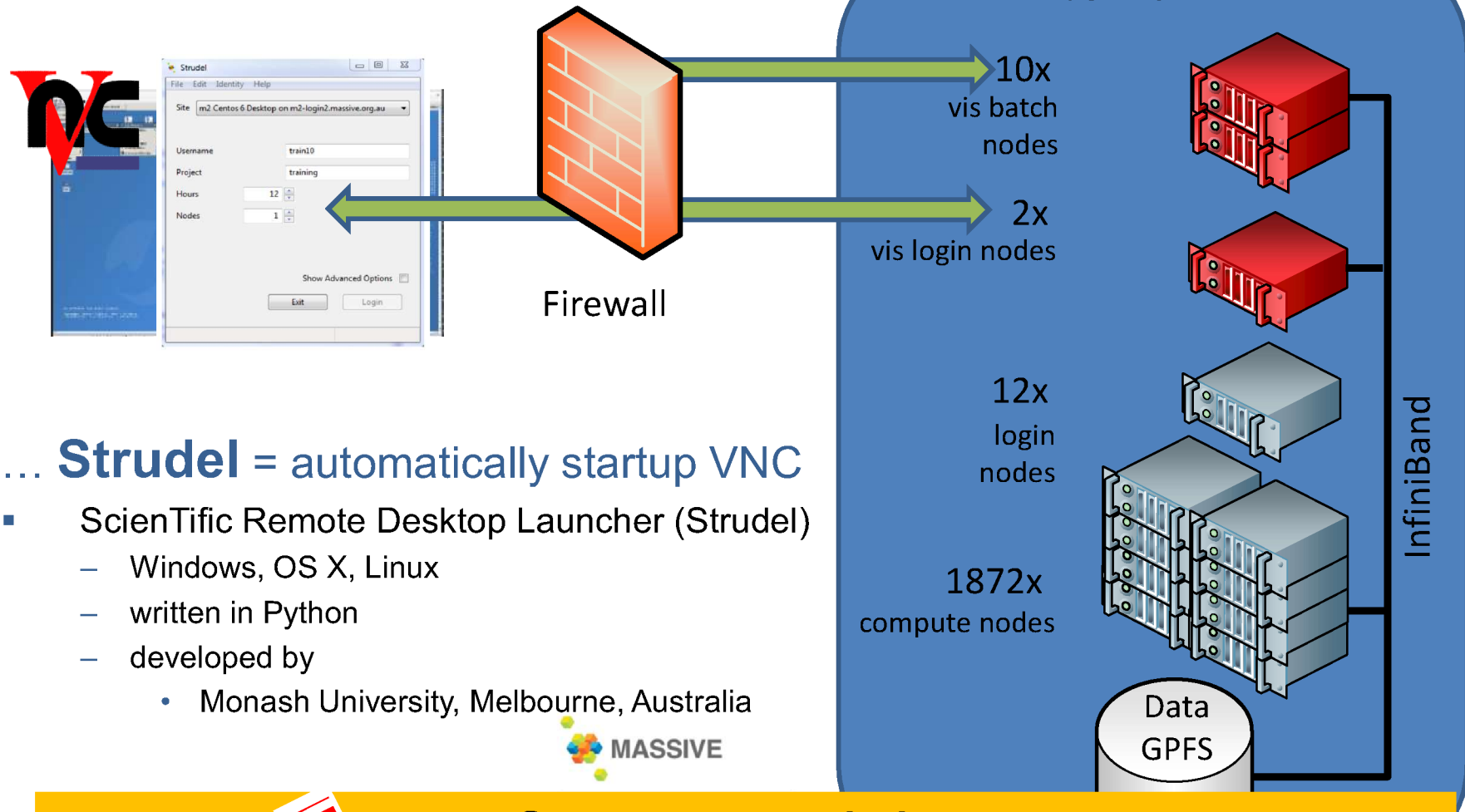
1. SSH to HPC system
2. authenticate via SSH key pair
3. look for an existing VNC server (you may want to reconnect)
4. submit job via slurm
5. wait for job to start
start VNC server on node
6. establish SSH tunnel
7. ...

NOT our recommendation:
This is far too time consuming.

<https://trac.version.fz-juelich.de/vis/wiki/vnc3d/manual>

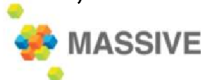
Remote 3D Visualization

with VNC + VirtualGL



... **Strudel** = automatically startup VNC

- ScienTific Remote Desktop Launcher (Strudel)
 - Windows, OS X, Linux
 - written in Python
 - developed by
 - Monash University, Melbourne, Australia

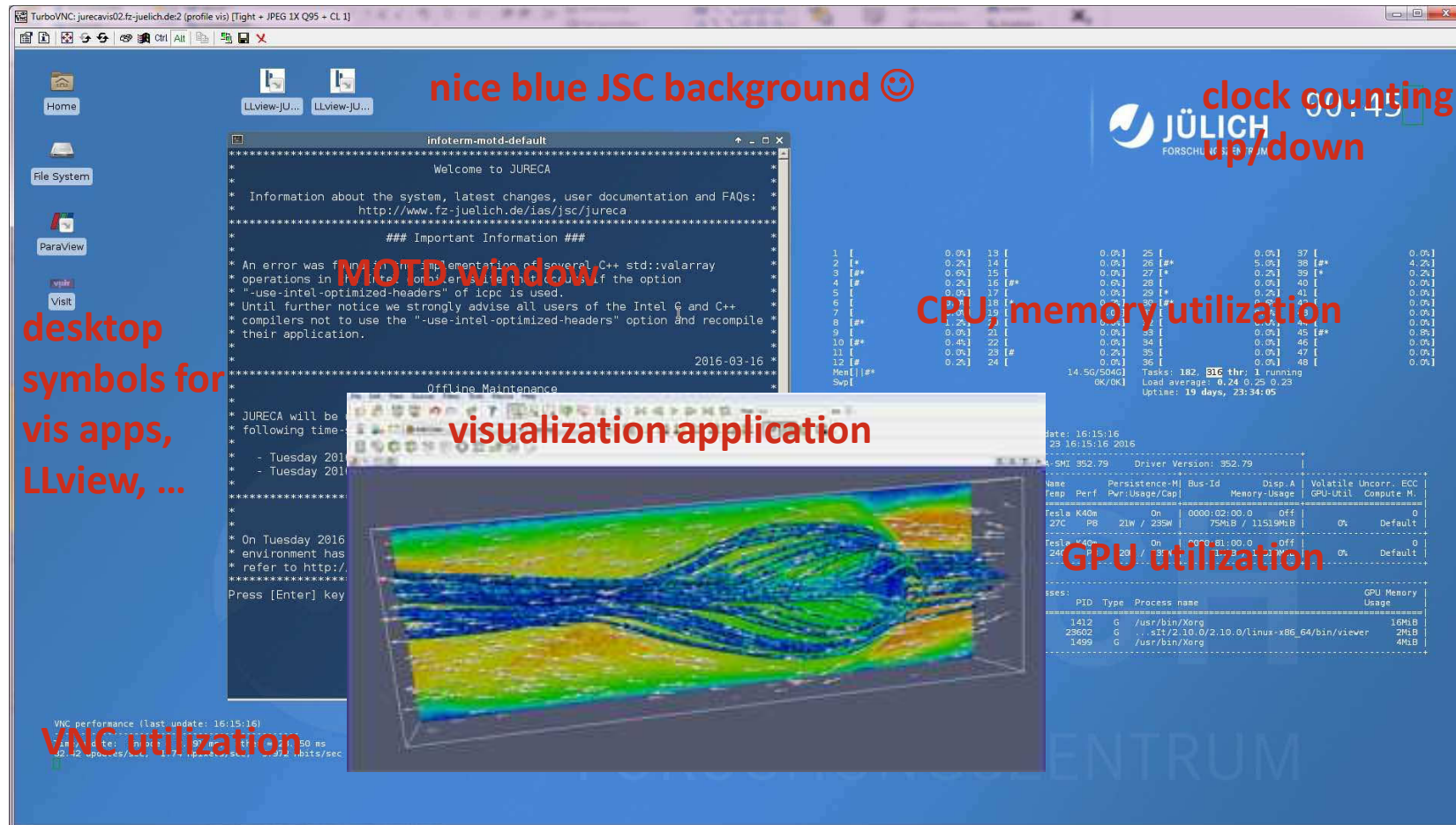


Our recommendation:
Use 'Strudel' to start a VNC session.

<https://trac.version.fz-juelich.de/vis/wiki/vnc3d/strudel>

Remote 3D Visualization

Desktop Profiles with VNC + VirtualGL



nice blue JSC background 😊

clock counting up/down

desktop symbols for vis apps, LLview, ...

MDT window

visualization application

GPU memory utilization

GPU utilization

VNC utilization

JÜLICH FORSCHUNGSZENTRUM

info term-motd-default

```

*****
Welcome to JURECA

Information about the system, latest changes, user documentation and FAQs:
http://www.fz-juelich.de/ias/jsc/jureca

*** Important Information ***

An error was found in the implementation of several C++ std::valarray
operations in the Intel compiler. This can be fixed if the option
"-use-intel-optimized-headers" of icpc is used.
Until further notice we strongly advise all users of the Intel C and C++
compilers not to use the "-use-intel-optimized-headers" option and recompile
their application.

2016-03-16

*****
Offline Maintenance

JURECA will be
following time-
- Tuesday 201
- Tuesday 201

*****

On Tuesday 2016
environment has
refer to http://
*****
Press [Enter] key
  
```

VNC performance (last update: 16:15:16)

Client	Server	Bandwidth	Latency	Input	Output
1412	1499	10.00 MB/sec	10 ms	10.00 MB/sec	10.00 MB/sec

GPU Memory Usage

PID	Type	Process name	GPU Memory Usage
1412	G	/usr/bin/Xorg	16MB
23602	G	./sli/2.10.0/2.10.0/linux-x86_64/bin/viewer	2MB
1499	G	/usr/bin/Xorg	4MB

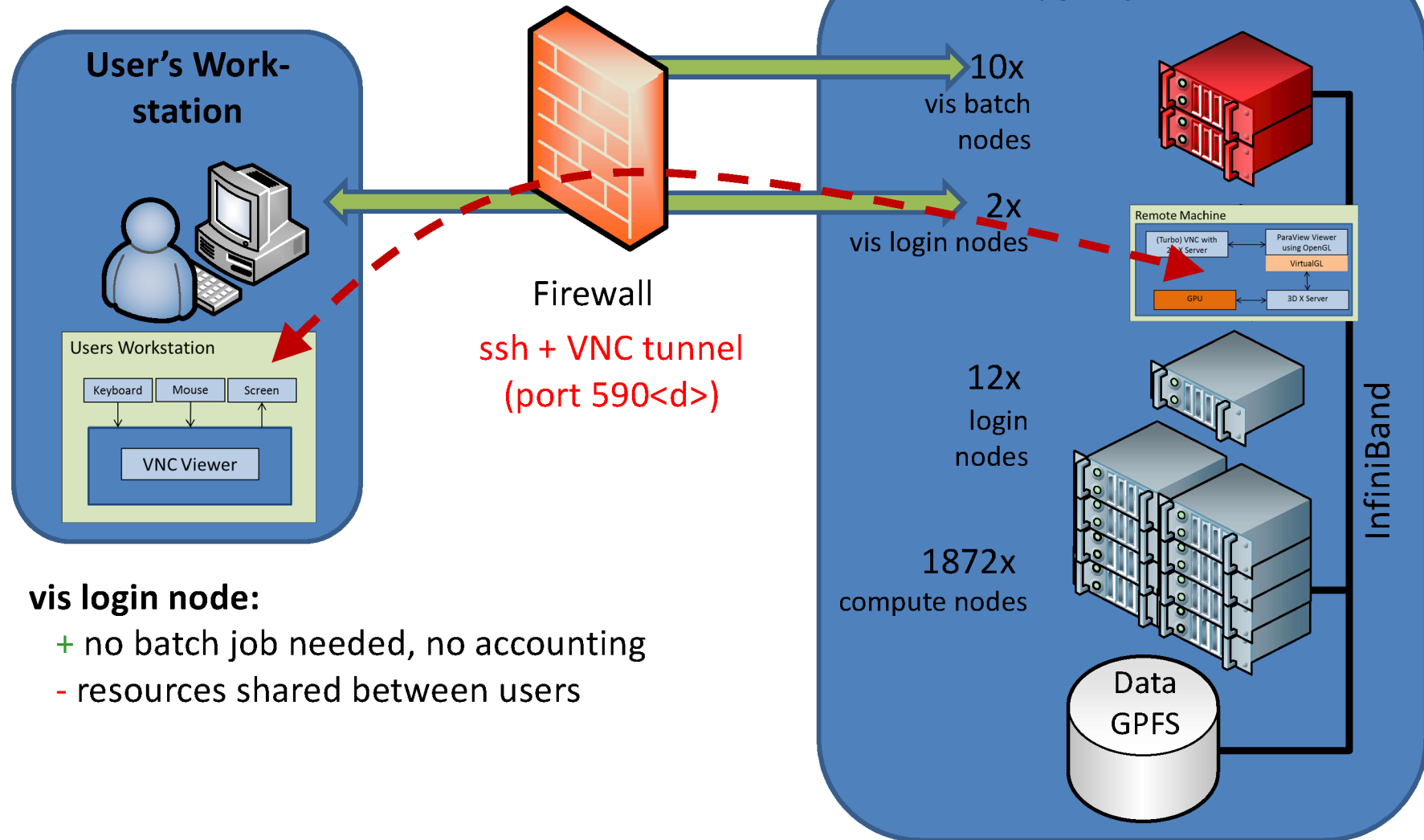
GPU Utilization

Name	Persistence-M	Bus-Id	Disp.A	Volatile	Uncorr.	ECC
Fesla K40m	On	0000:02:00.0	Off	0%	Default	0
Fesla K40m	On	0000:01:00.0	Off	0%	Default	0

Remote 3D Visualization (possible scenarios)

Visualization Scenario 1:

Vis Login Node with VNC

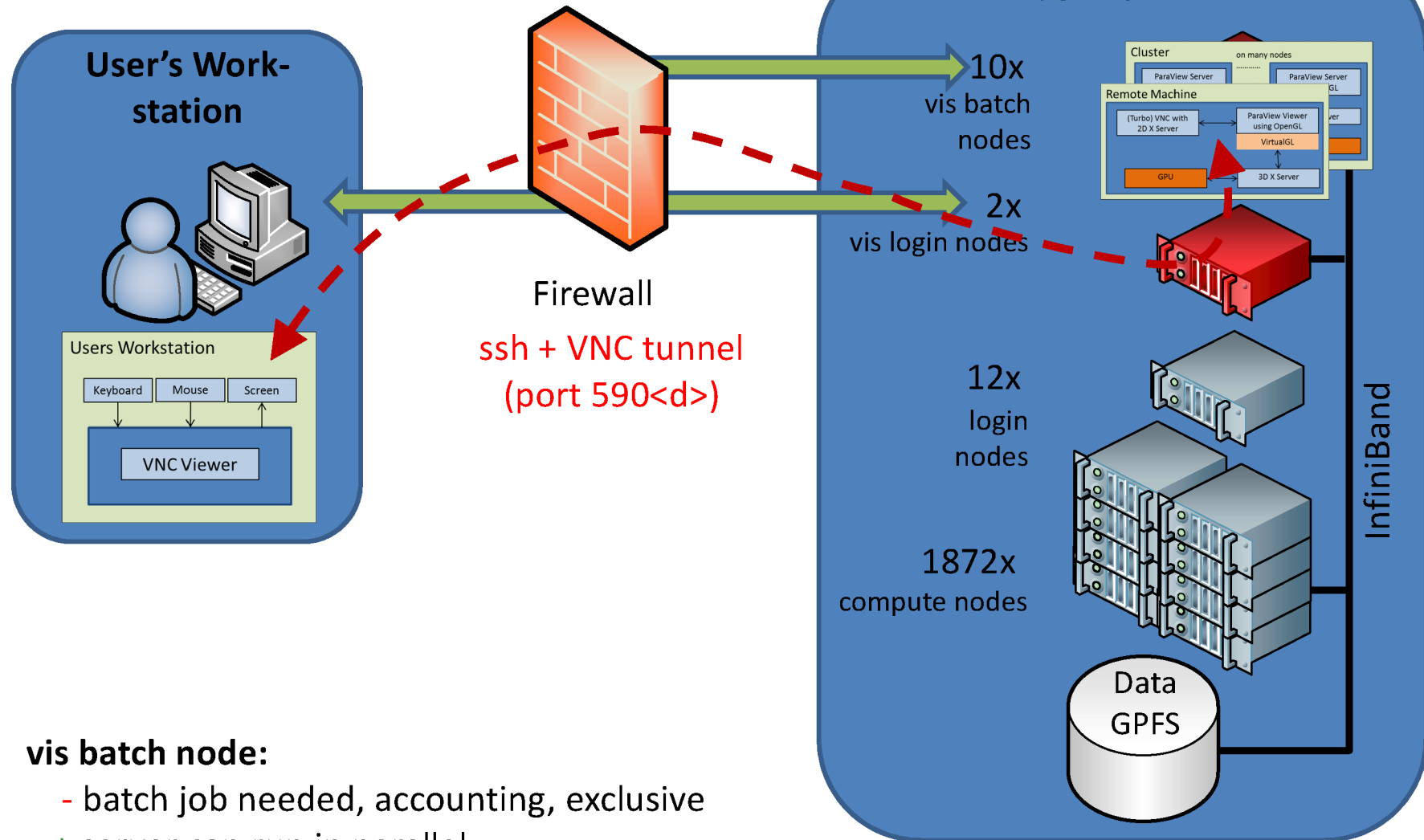


vis login node:

- + no batch job needed, no accounting
- resources shared between users

Visualization Scenario 2:

Vis Batch Node with VNC

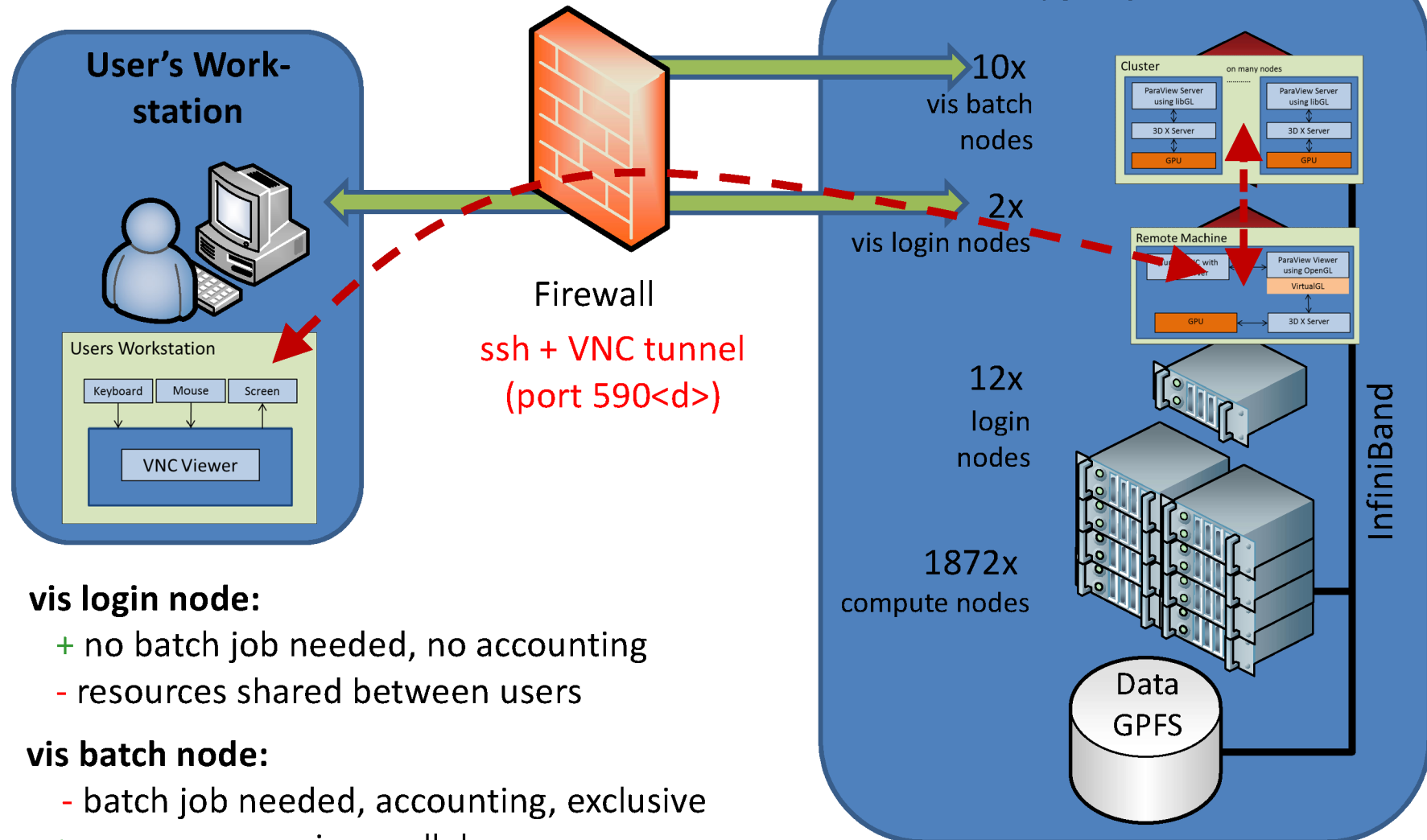


vis batch node:

- batch job needed, accounting, exclusive
- + server can run in parallel
- (but number of vis nodes limited to 4)

Visualization Scenario 3:

Vis.Login for GUI, Comp. Nodes for Server



vis login node:

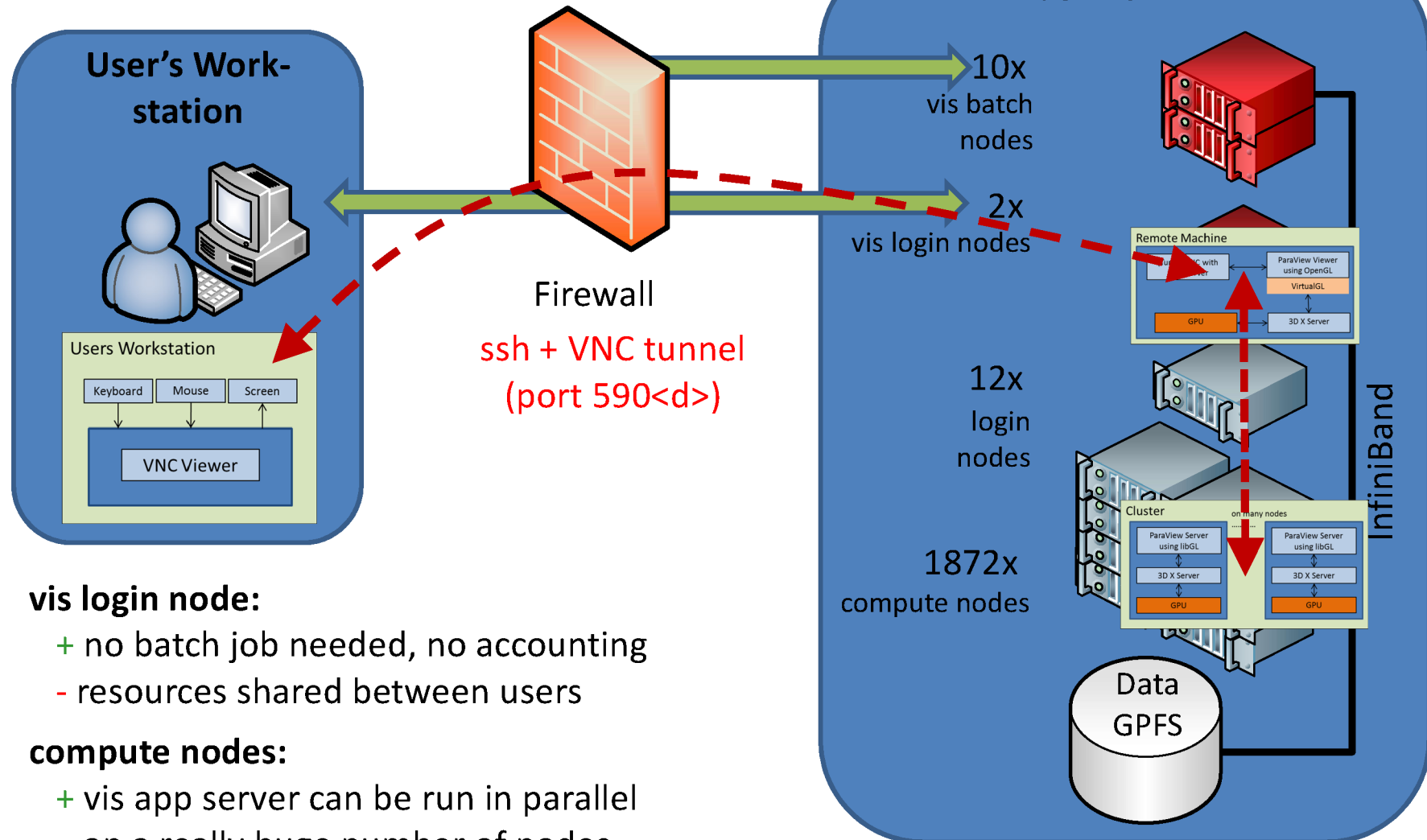
- + no batch job needed, no accounting
- resources shared between users

vis batch node:

- batch job needed, accounting, exclusive
- + server can run in parallel
(but number of vis nodes limited to 4)

Visualization Scenario 4:

Vis Login for GUI, Compute for Server



vis login node:

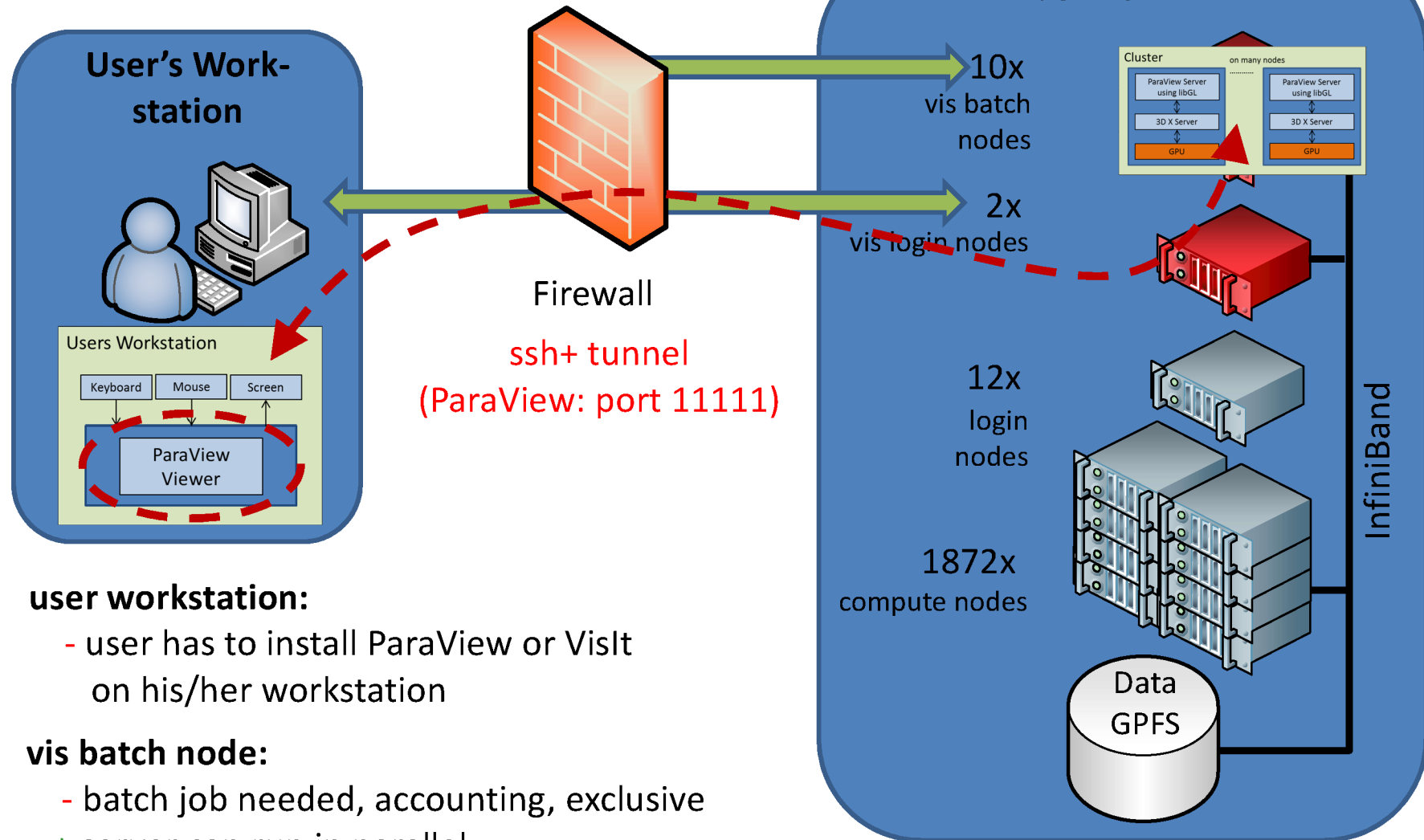
- + no batch job needed, no accounting
- resources shared between users

compute nodes:

- + vis app server can be run in parallel on a really huge number of nodes
- only software rendering

Visualization Scenario 5:

ParaView (or VisIt) without VNC



user workstation:

- user has to install ParaView or VisIt on his/her workstation

vis batch node:

- batch job needed, accounting, exclusive
- + server can run in parallel (but number of vis nodes limited to 4)

Remote 3D Visualization (alternatives)

Remote 3D Visualization

with Xpra (X Persistent Remote Applications) + VirtualGL

"screen for X11,, (stream application content with H.264 + VirtualGL)

```
xpra start ssh:<SERVER>:<DISPLAY> \  
--ssh-key=<PATH_TO_SSH-PRIVATE-KEY> \  
--start-child=<XAPPLICATION>
```

- X-applications forwarded by Xpra appear on the local desktop as normal windows
- allows disconnection and reconnection without disrupting the forwarded application
- Xpra protocol is self-tuning and relatively latency-insensitive
- **advantages**
 - **No X is required** on user's workstation (X display on server).
 - **No OpenGL is required** on user's workstation (only images are send).
 - Quality of visualization does **not depend** on user's workstation.
 - Data size send is **independent** from data of 3d scene.
 - Disconnection and reconnection possible.



Our recommendation:
Use 'Xpra' as 'ssh -X' replacement.

<https://xpra.org/>

Nice To Know

Nice to know:

OSPRay

CPU ray tracing framework
for scientific vis. rendering

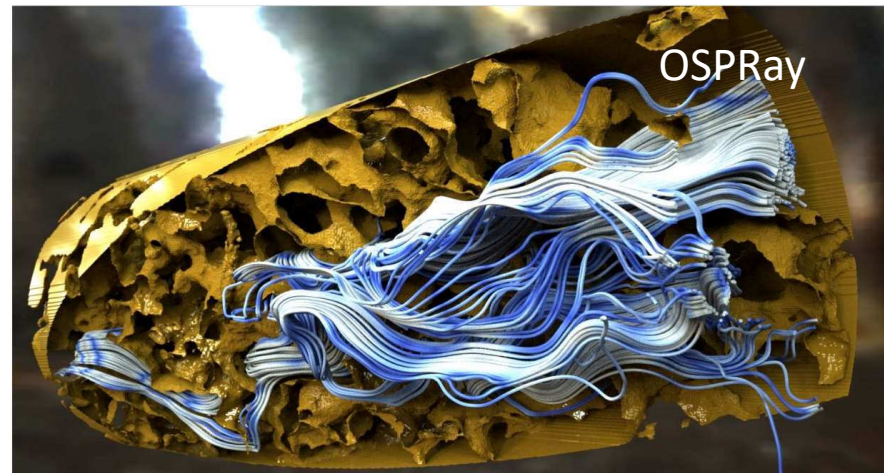
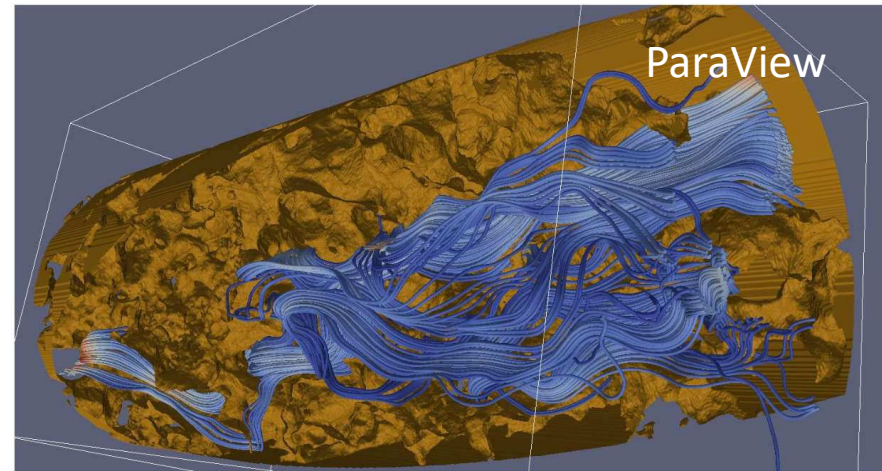
- efficient rendering on CPUs
- ray tracing / high fidelity rendering
- made for scientific visualization

Built on top of

- Embree (Intel ray tracing kernels)
- Intel SIMD Program Compiler

Integrated into

- ParaView, VMD, VisIt, VL3,
EasternGraphics,...



"FIU" Ground Water Simulation

Texas Advanced Computing Center (TACC) and Florida International University

<http://www.ospray.org/>

http://www.sdvis.org/ospray/download/talks/IEEEVis2016_OSPray_talk.pdf

Nice to know:

OSPRay with ParaView

Ray tracing within ParaView

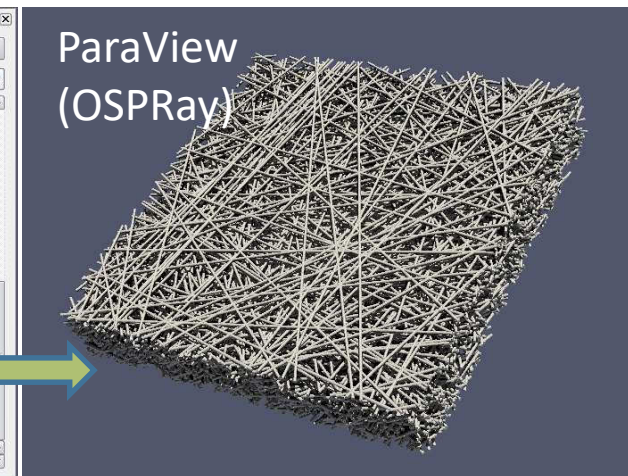
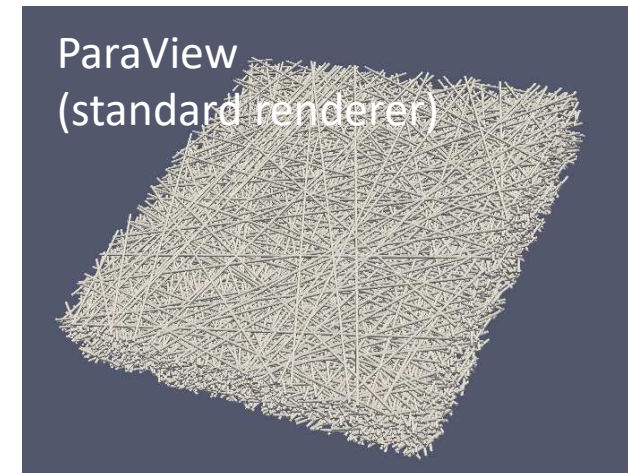
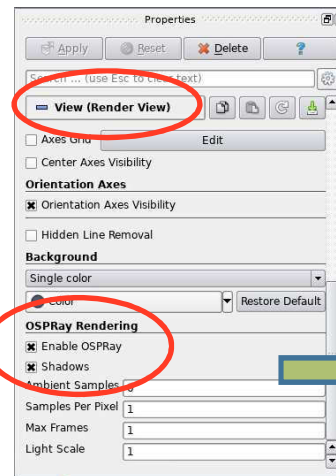
- Build option in ParaView 5.2 by default

Why ray tracing?

- gives more realistic results
- adds “depth” to your image
- can be faster on large data

Requirement:

CPU: Anything SSE4 and newer
(in part, including Intel®
Xeon Phi™ Knights Landing)



<http://www.ospray.org/>
http://www.sdvis.org/ospray/download/talks/IEEEVis2016_OSPray_talk.pdf

Cooperation with Electrochemical Process Engineering (IEK-3)
Jülich Forschungszentrum GmbH, Germany

Documentation

JURECA Visualization Related Documentation

Please visit

<https://trac.version.fz-juelich.de/vis/>

Please send us your feedback.

h.zilken@fz-juelich.de **j.goebbert@fz-juelich.de**

Questions ... ?



rendered with Blender from a DNS of a diesel injection spray of ITV, RWTH Aachen University

Interactive Supercomputing at Jülich Supercomputing Centre

¹ Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, Germany
Cross-Sectional-Team „Visualization“
h.zilken@fz-juelich.de, j.goebbert@fz-juelich.de

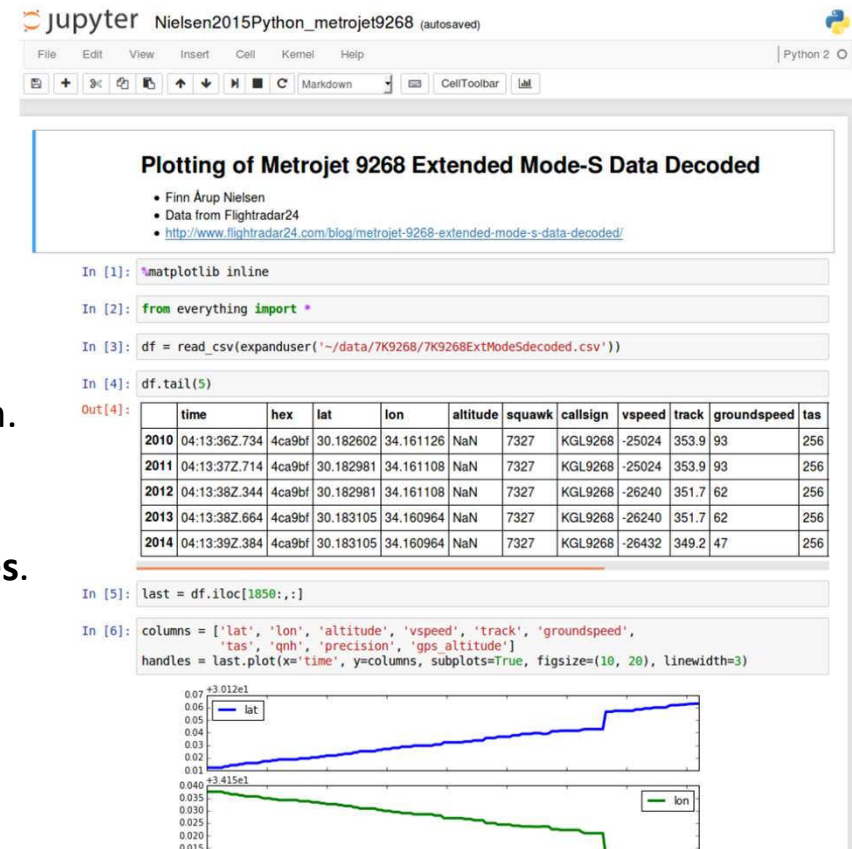
Interactive Supercomputing

What is Jupyter

Jupyter ...

- ... is an interactive computational environment **in the web browser**.
- ... combines code execution, rich text, mathematics, plots and interactive visualization.
- ... is the interface to Jupyter Notebooks for **creating reproducible computational narratives**.
- ... supports multiple programming languages beside Python (C++, Julia, JavaScript, ...).
- ... was formerly called IPython notebook

<https://trac.version.fz-juelich.de/vis/wiki/Jupyter>



Interactive Supercomputing

Why you should use Jupyter

Researchers today across all academic disciplines often need to **write computer code** in order to collect and process data, carry out statistical tests, run simulations or draw figures.

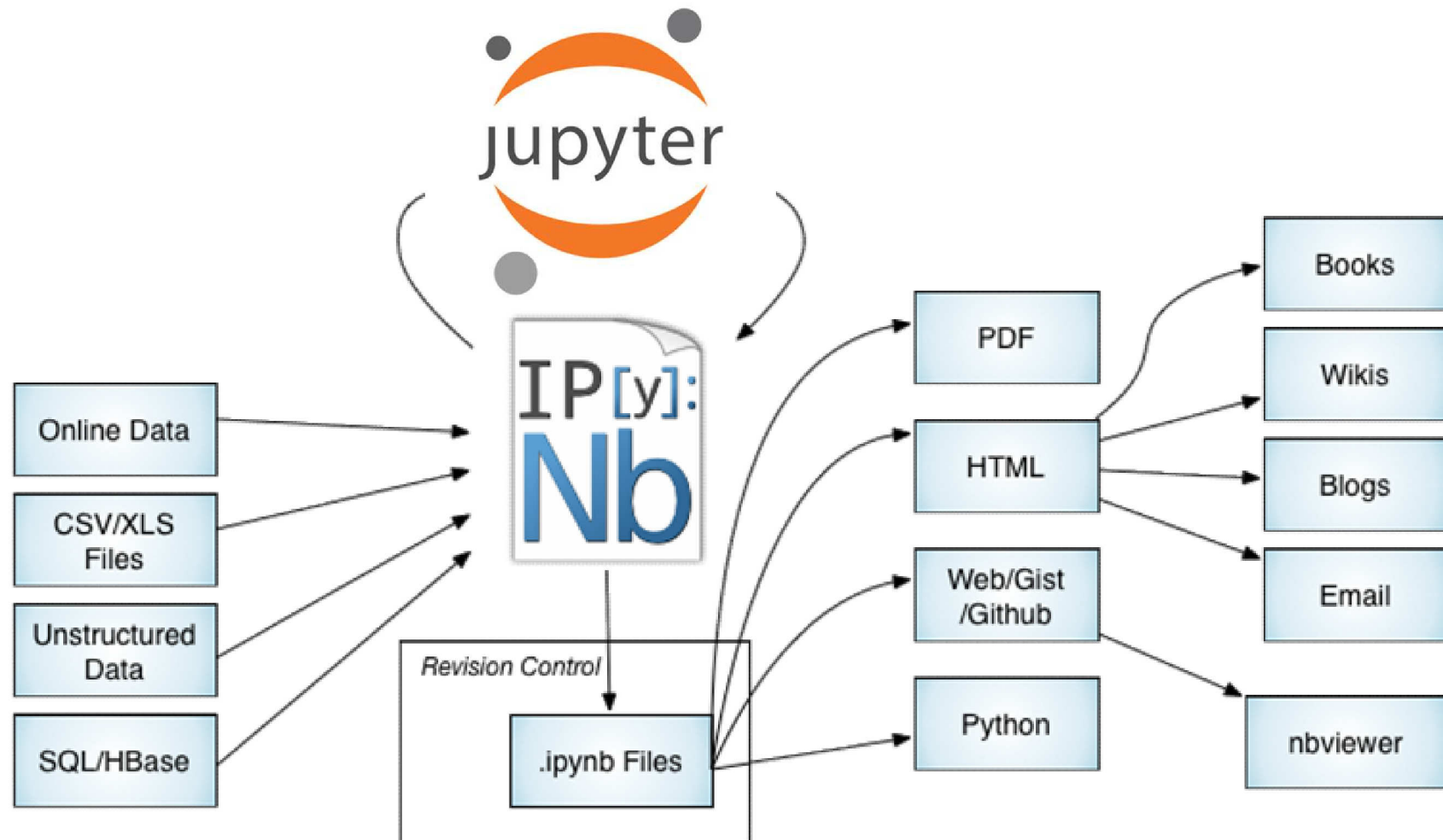
Even if the widely applicable libraries and tools for this are often open source projects (such as NumPy, TensorFlow, ...), the specific code researchers write for a particular piece of work is often **left unpublished, hindering reproducibility**.

Jupyter Notebooks integrating prose, code and results offer **a way to publish a computational method** which can be readily read and replicated.



Interactive Supercomputing

Why you should use Jupyter



Interactive Supercomputing

How to use Jupyter

<https://jupyter-jsc.fz-juelich.de>

- Apply for a JSC Web Service Account
- Sign in with your JSC Web Service Account
- Accept the usage and data protection agreement.

... and you are ready to start with Jupyter.



Sign in with JSC Webservice Account

Register for a JSC Webservice Account

Username:

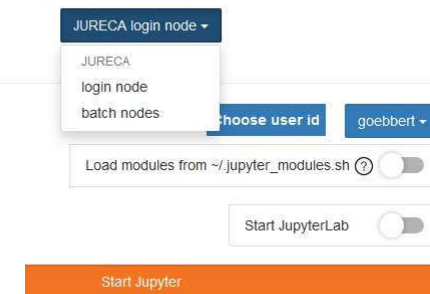
JSC Webservice Account Password:

Authenticate

Configure My Jupyter

Jupyter configuration

Your server is starting up.
You will be redirected automatically when it's ready for you.



JURECA login node ▾

JURECA
login node
batch nodes

choose user id goebbert ▾

Load modules from ~/jupyter_modules.sh ⓘ ☐

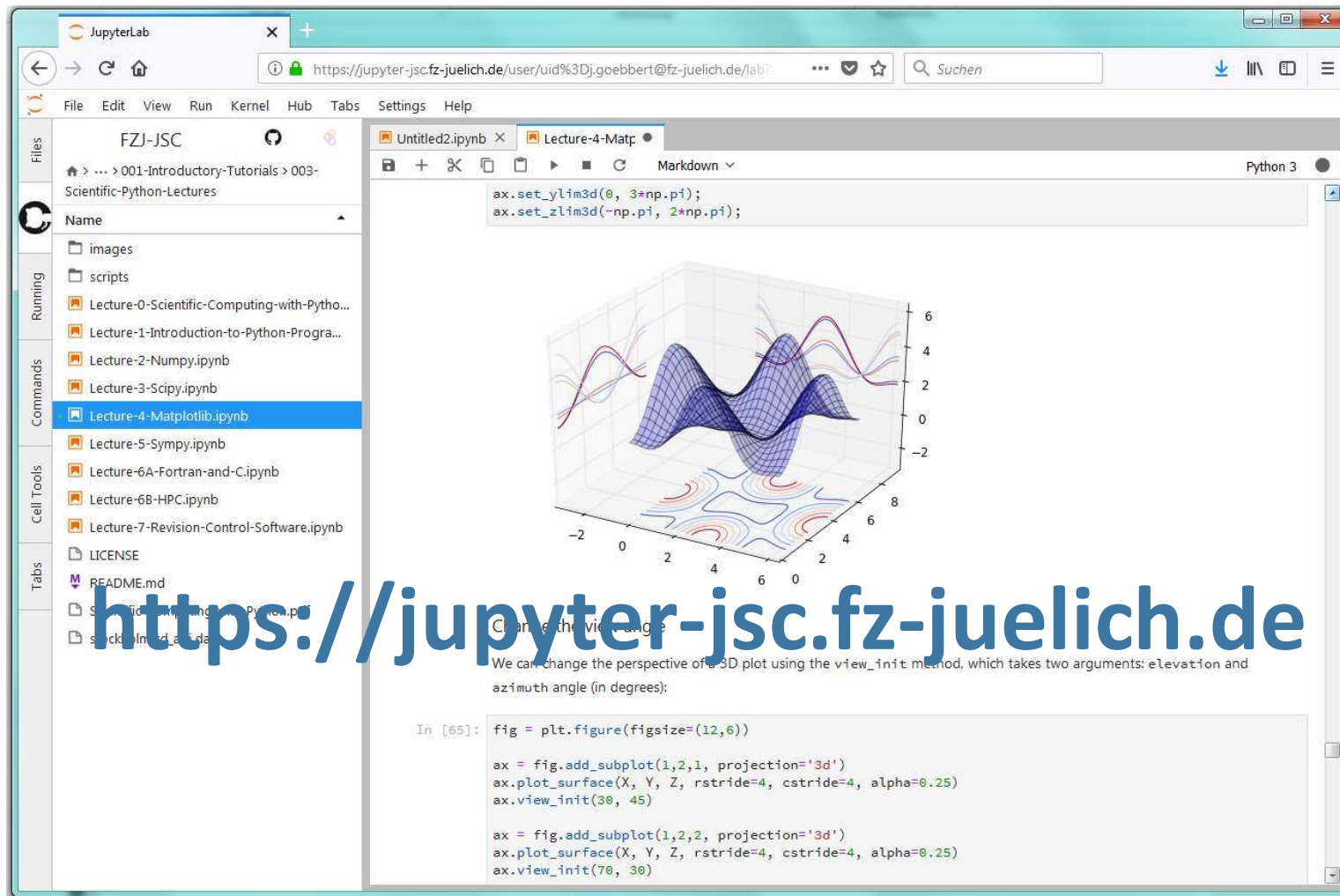
Start JupyterLab ☐

Start Jupyter

<https://trac.version.fz-juelich.de/vis/wiki/Jupyter>

Interactive Supercomputing

JupyterLab



<https://jupyter-jsc.fz-juelich.de>