JÜLICH
FORSCHUNGSZENTRUM

# Remote 3D Visualization at Jülich Supercomputing Centre

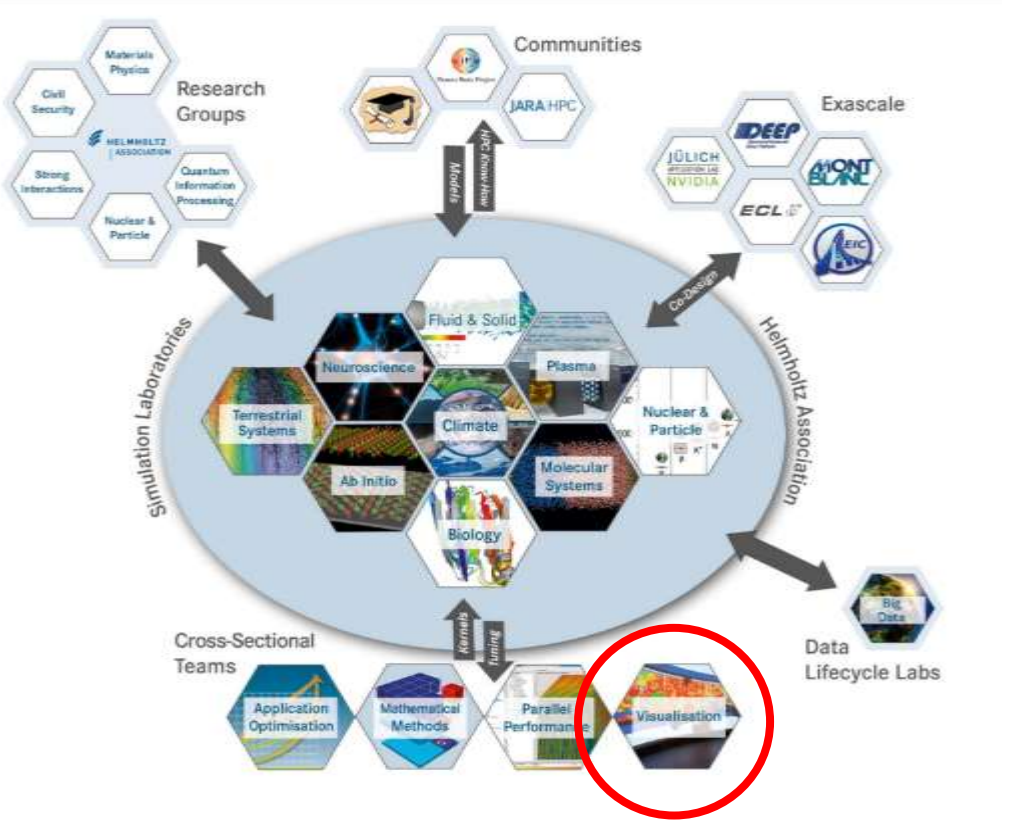
[1] Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, Germany
Cross-Sectional-Team „Visualization“
h.zilken@fz-juelich.de, j.goebbert@fz-juelich.de

# 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




http://www.fz-juelich.de/ias/jsc/EN/Expertise/Support/Visualization/_node.html

# Visualization at JSC

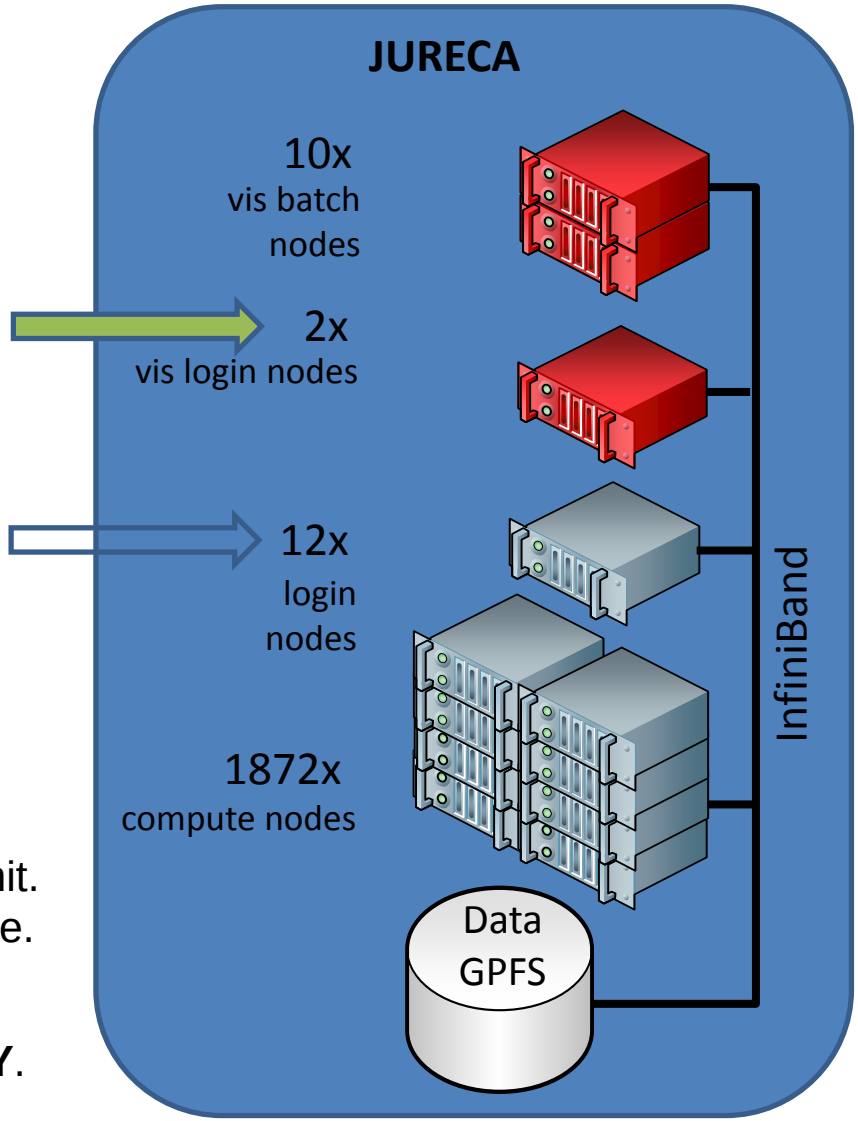
## General Hardware Setup

**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)*
- **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. SSH to allocated resource possible.

**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

# Visualization at JSC

## Usage Model for Vis Nodes

**1. As of today:**

- JURECA projects: (no problem at all)
  - access to vis resources with the normal project contingent
  - BUT: use vis nodes for visualization purpose only!
- JUQUEEN projects:
  - send a request to sc@fz-juelich.de
  - get a small contingent on JURECA for vis nodes
- Non HPC-Project Users:
  - send a request to sc@fz-juelich.de
  - get a small contingent on JURECA for vis nodes

**small contingent means 1000 core h**

**2. Future plans: more formal access (starting maybe next application period):**

- JURECA projects, JUQUEEN projects:
  - request vis nodes in the application form
  - (probably requests also possible later while project is already running)
  - get a small contingent for vis nodes in addition to project contingent
- Non HPC-Project Users:
  - apply for test project
  - get a small contingent for vis nodes only

# Remote 3D Visualization

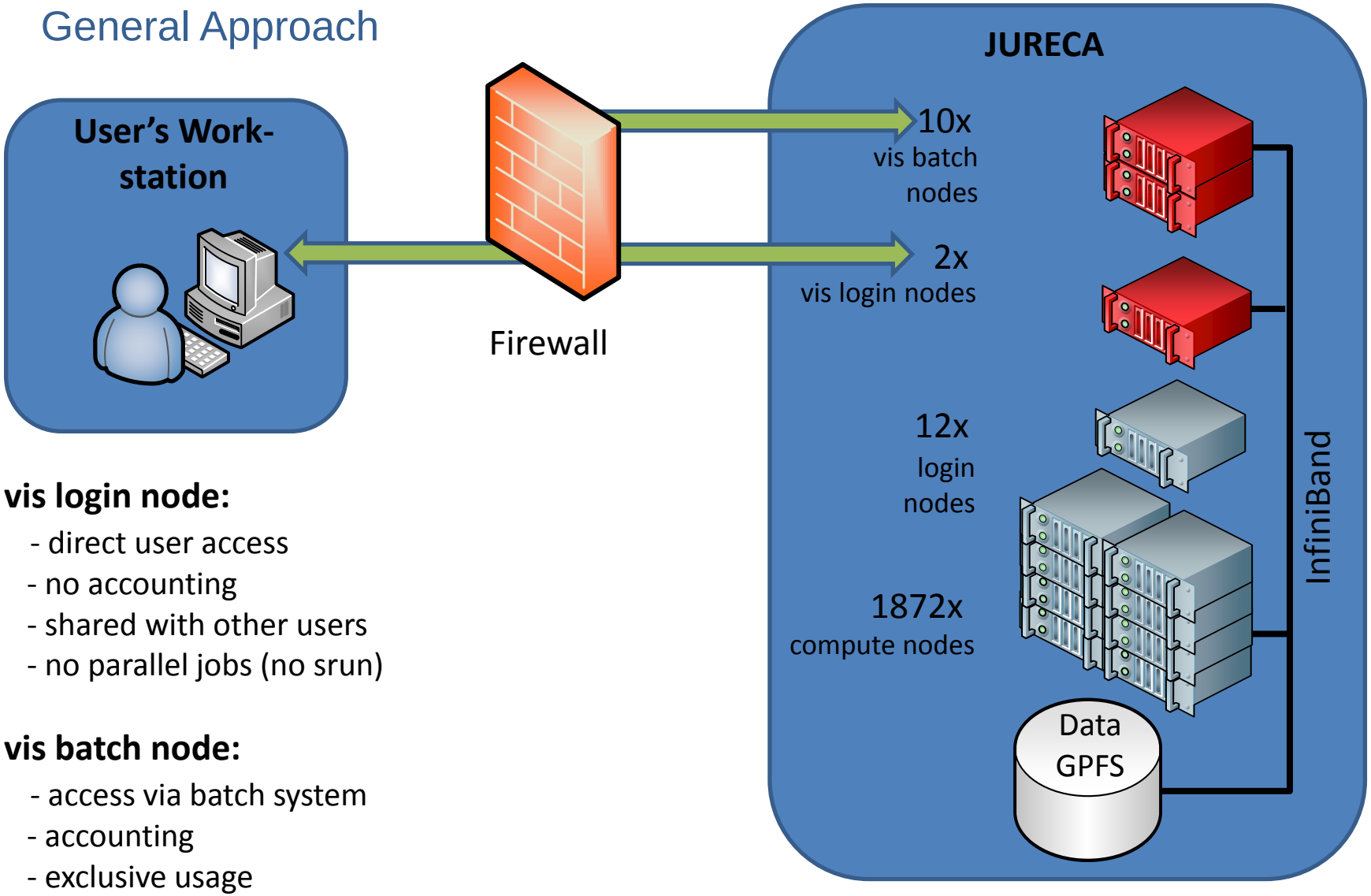
# Remote 3D Visualization

at Jülich Supercomputing Centre

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

# Remote 3D Visualization

## General Approach




**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

## 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 command are encaplusated 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 requied OpenGL.
  - User´s workstation defines the **quality and speed** of the visualization.
  - User´s workstation requires **all data needed** to visualize the 3d scene.

**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 for 3D visualization.**

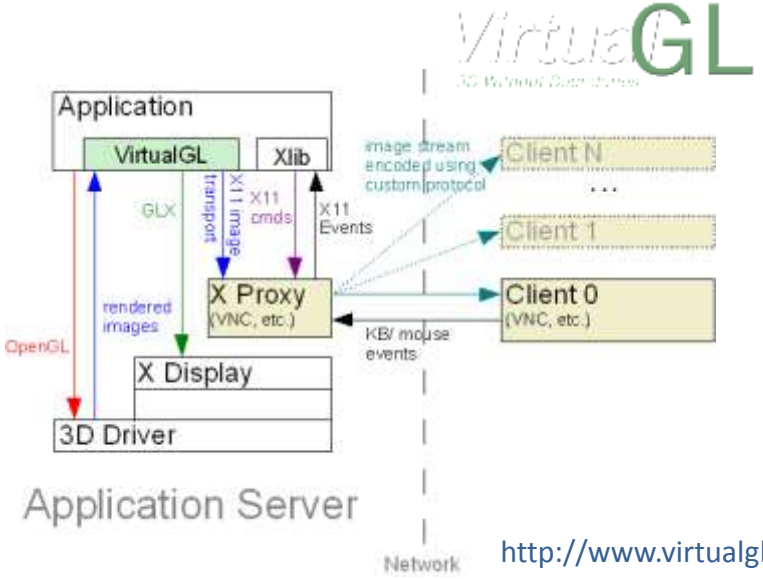
# Remote 3D Visualization

## 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.




http://www.virtualgl.org

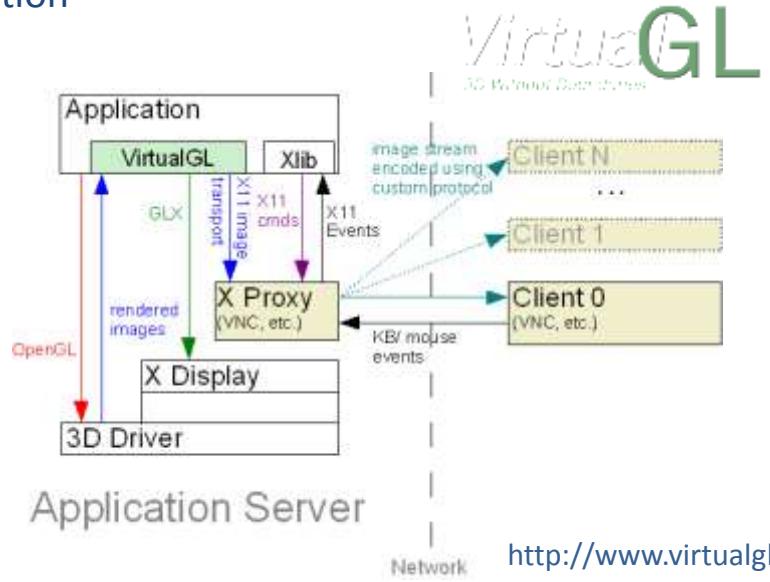
# Remote 3D Visualization

## with VNC (Virtual Network Computing) + VirtualGL

### VNC + VirtualGL

**vglrun <application>**

- Recommended solution for any OpenGL application
  e.g. ParaView, VisIt, IDL, Vapor, …
- 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, …)

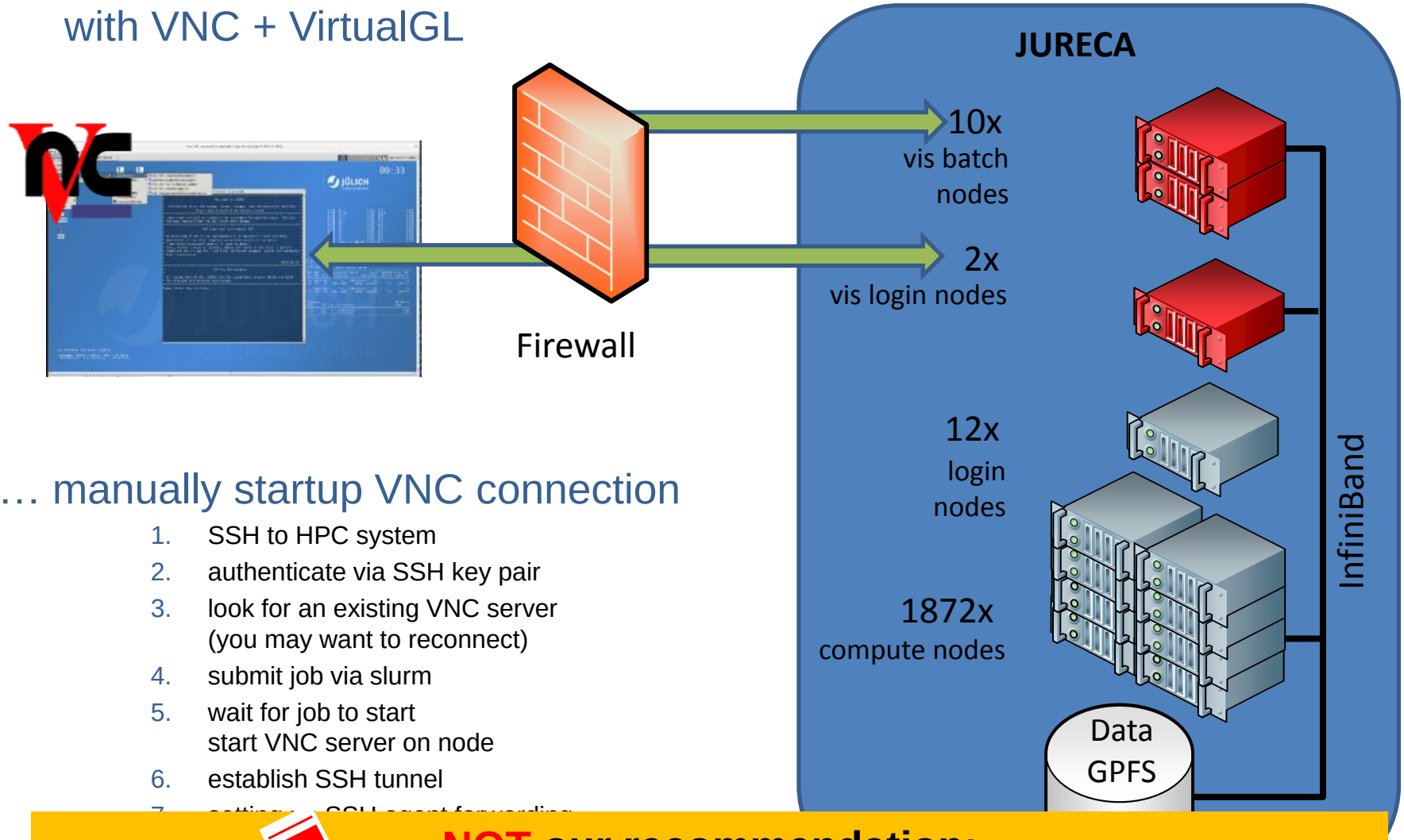



http://www.virtualgl.org

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

# Remote 3D Visualization

## with VNC + VirtualGL




## … manually startup VNC connection

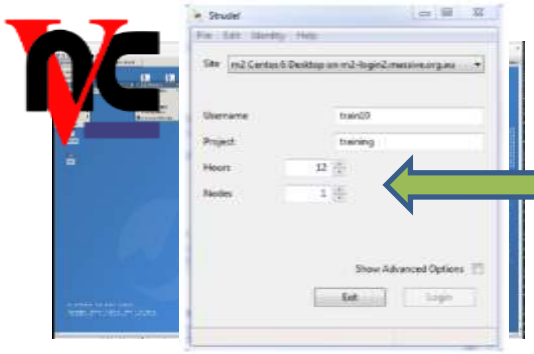
1. SSH to HPC system
2. authenticate via SSH key pair
3. look for an existing VNC server
   (you may want to reconnect)
4. submit job via slurm
5. wait for job to start
   start VNC server on node
6. establish SSH tunnel

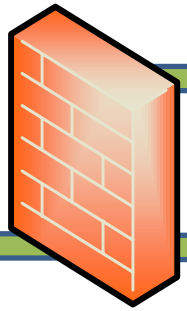
**NOT our recommendation:**
**This is far too time consuming.**

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

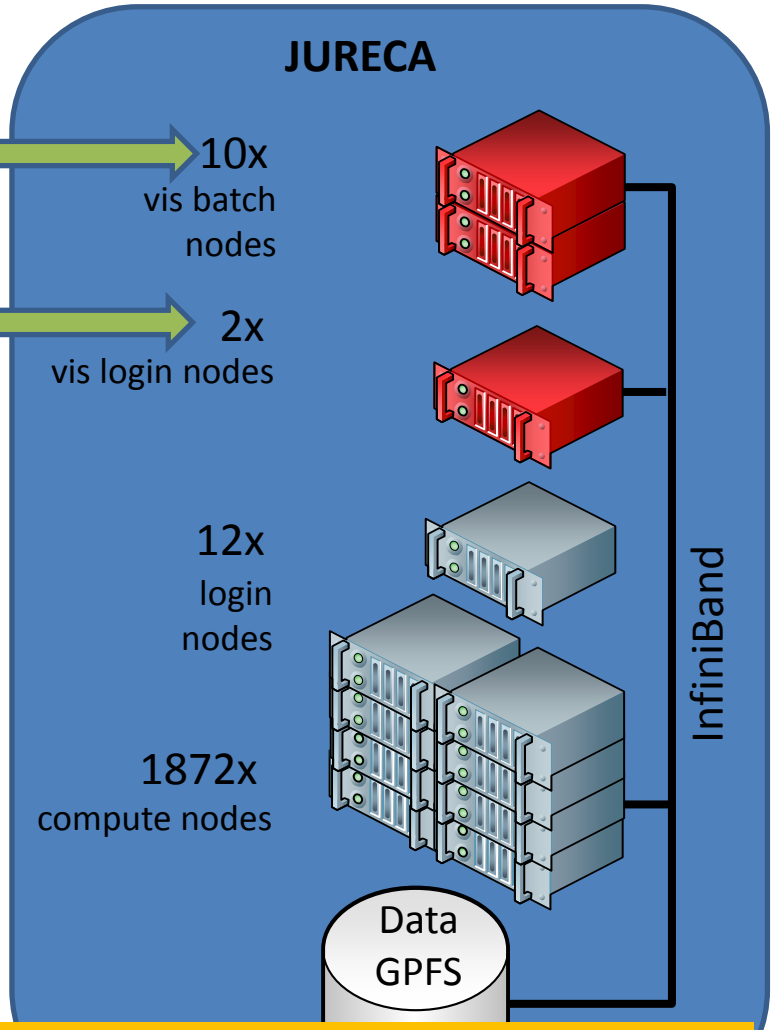
# Remote 3D Visualization

## with VNC + VirtualGL





Firewall




## … **Strudel** = automatically startup VNC

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

MASSIVE

**Our recommendation:**
**Use ‘Strudel’ to start a VNC session.**

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

# Remote 3D Visualization

## with Xpra (X Persistent Remote Applications) + VirtualGL

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

**xpra start ssh:<USERID>@<SERVER> --start-child=<JURECA_XAPP>**

- 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

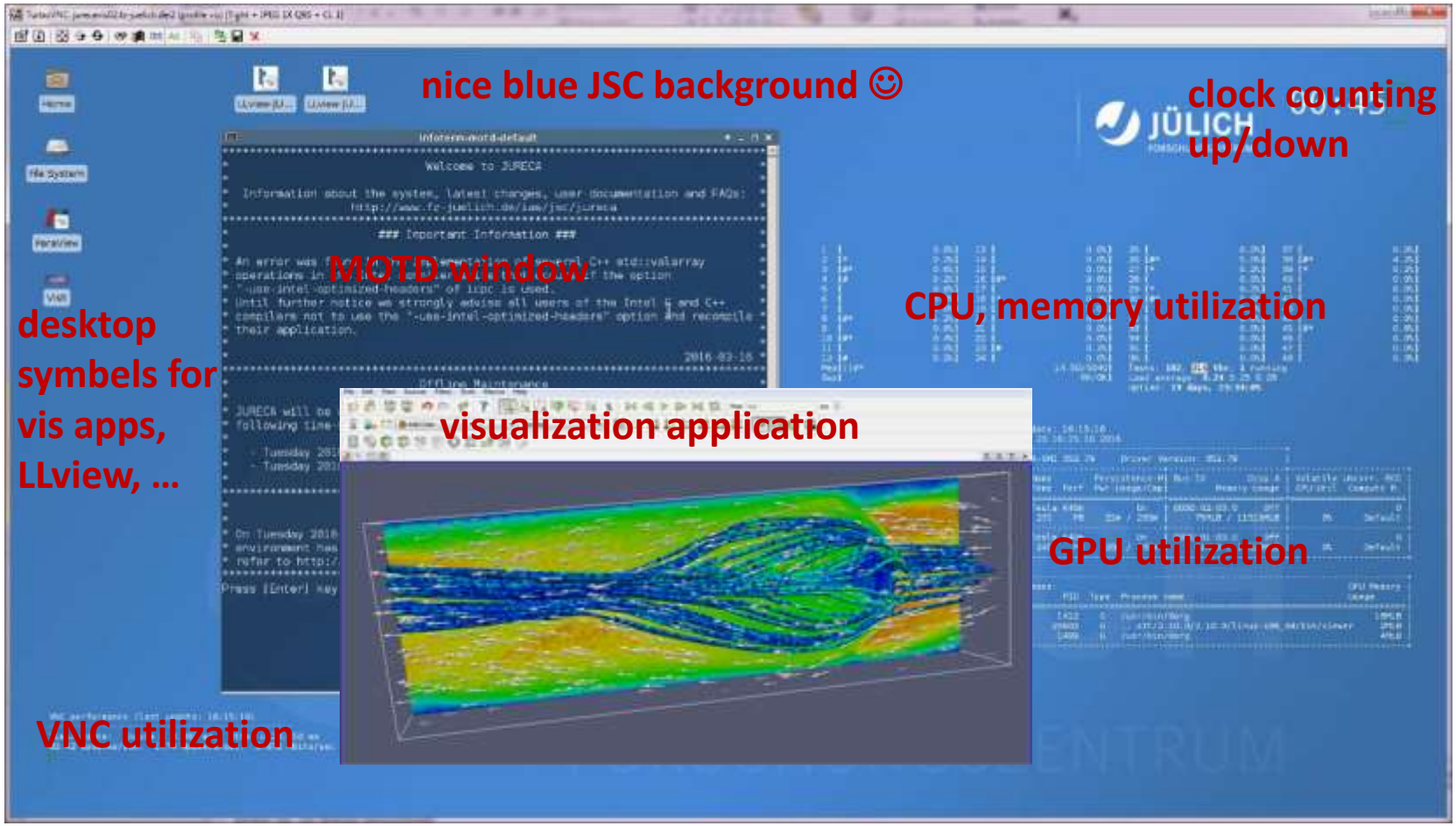
- **advantages**
  - **No X is required** on user´s workstation (X display on server).
  - **No OpenGL is required** on user´s workstation (only images are send).
  - Quality of visualization does **not depend** on user´s workstation.
  - Data size send is **independent** from data of 3d scene.
  - Disconnection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

**Our recommendation:**
**Use 'Xpra' as 'ssh –X' replacement.**

**https://trac.version.fz-juelich.de/vis/wiki/vnc3d/xpra**

# Remote 3D Visualization

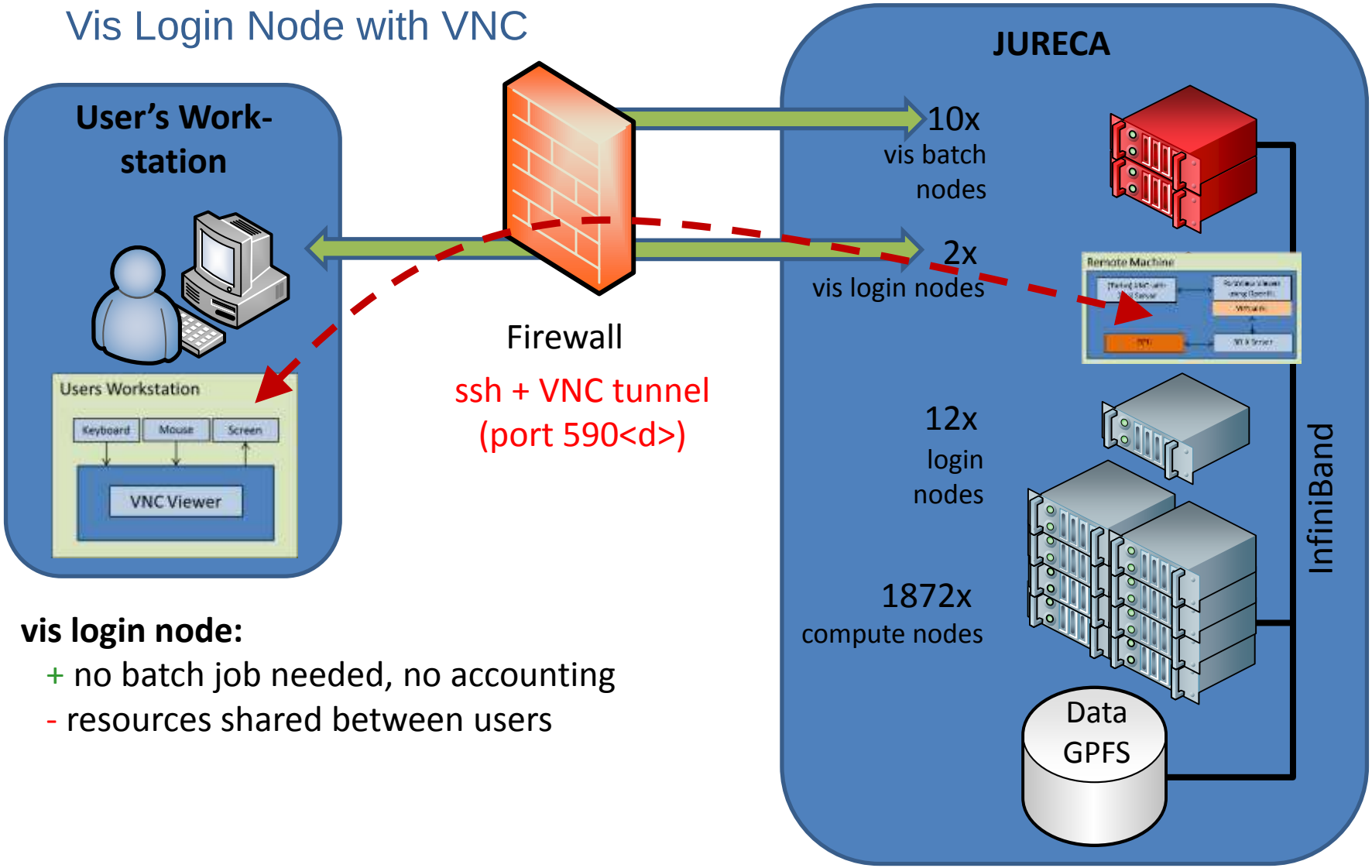
## with VNC + VirtualGL



# Remote 3D Visualization (possible scenarios)

# Visualization Scenario 1:

## Vis Login Node with VNC

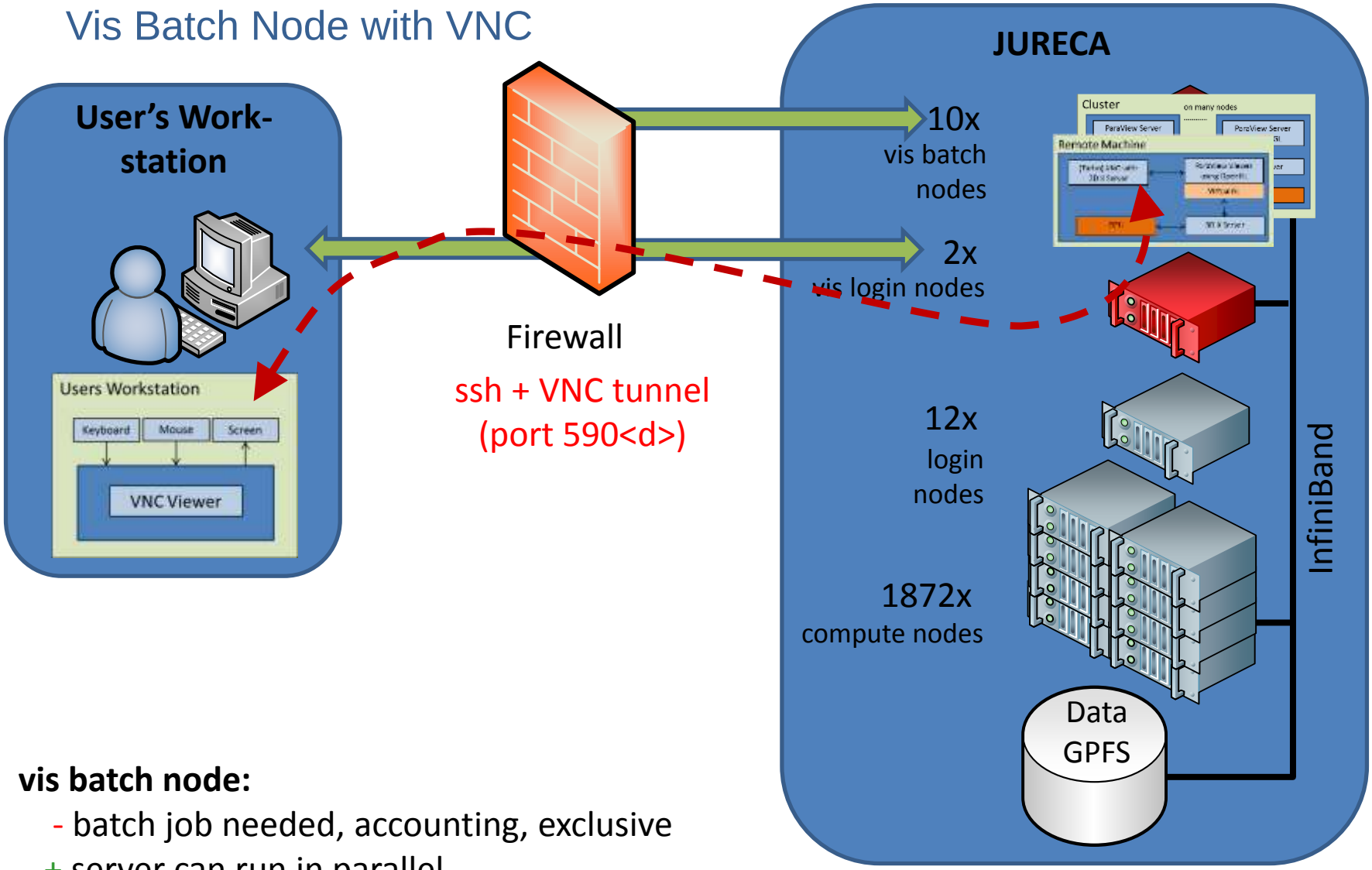



**vis login node:**

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

# Visualization Scenario 2:

## Vis Batch Node with VNC

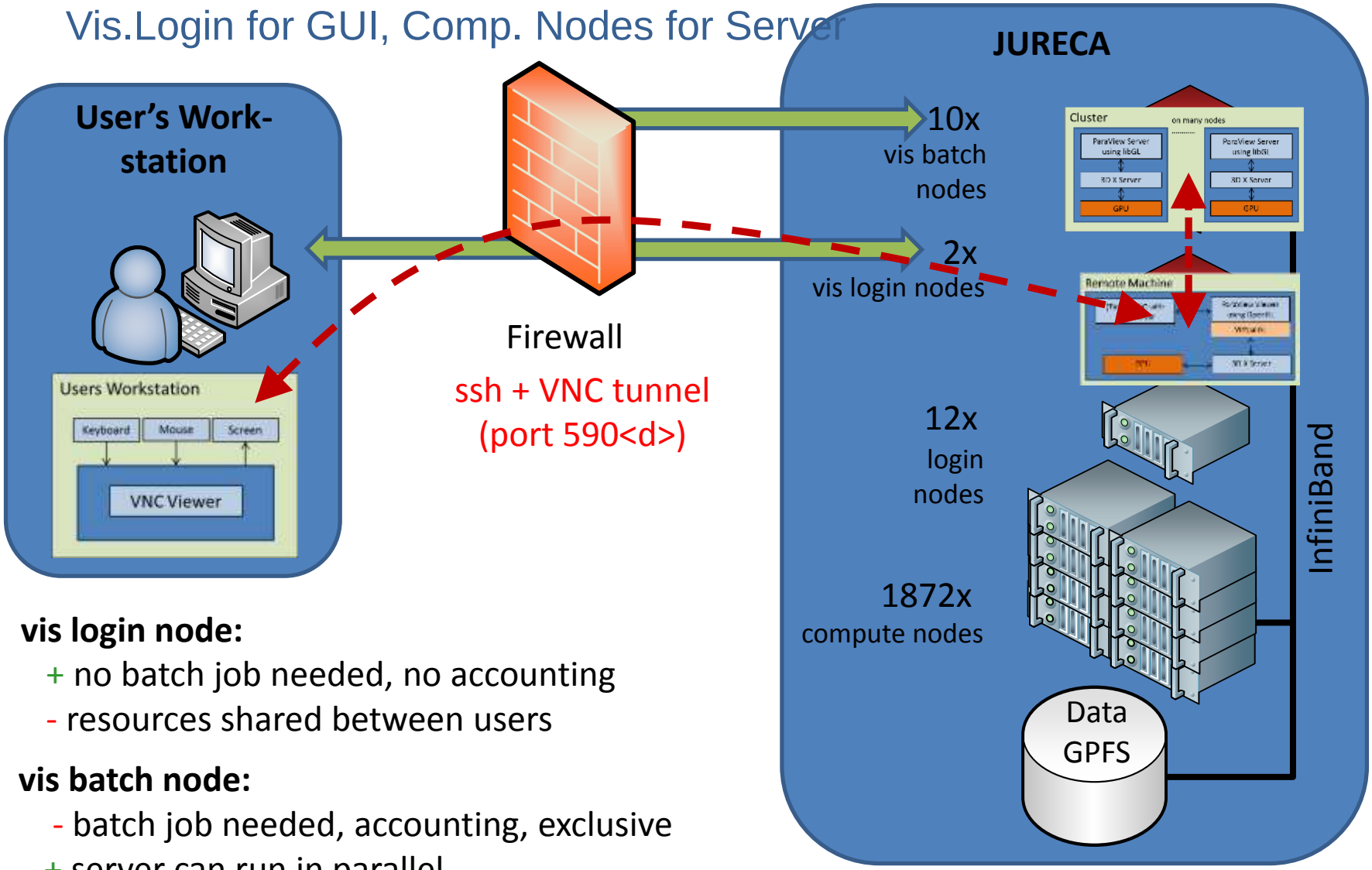



**vis batch node:**

- \- 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:**

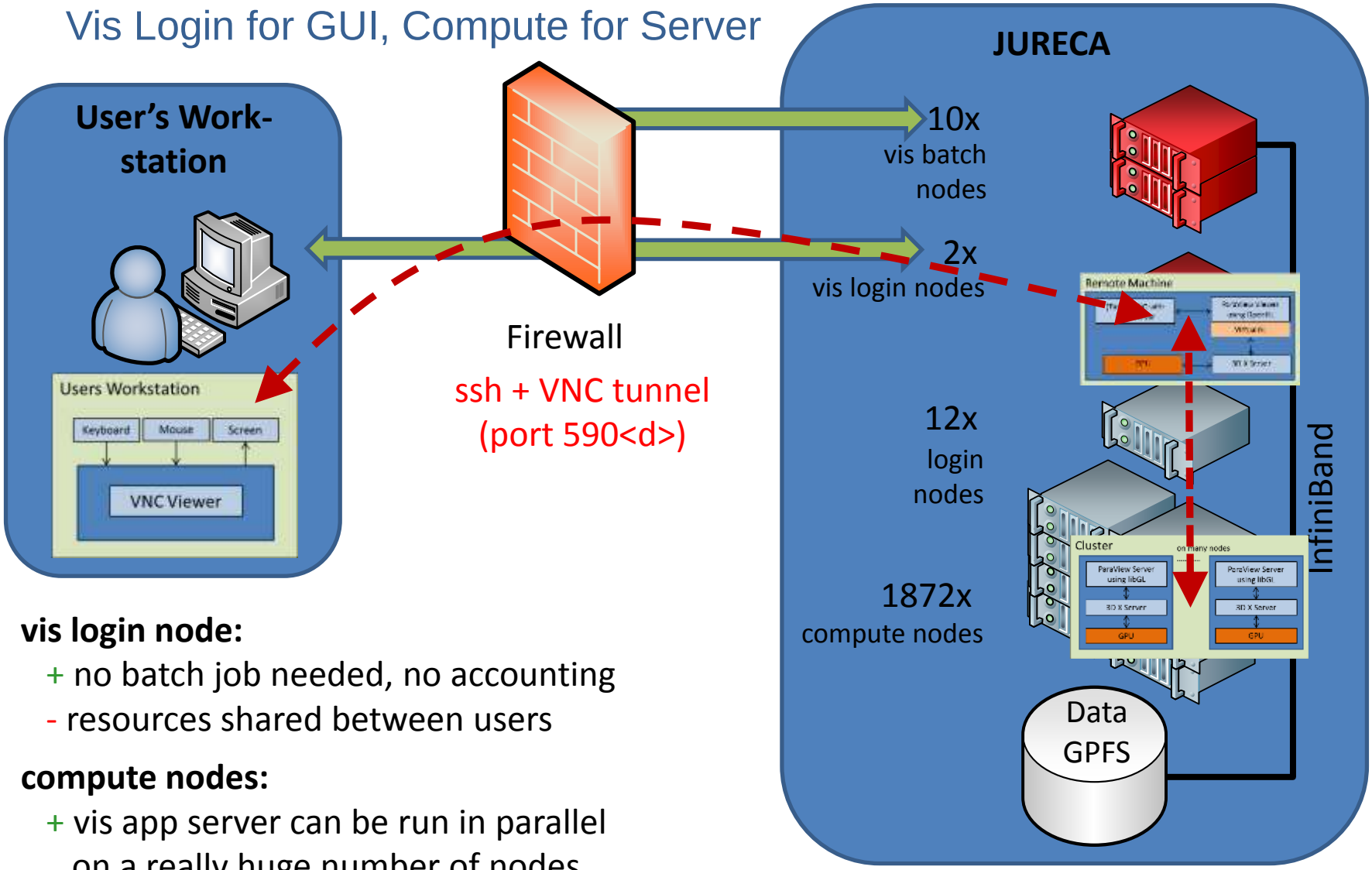
+ no batch job needed, no accounting
- resources shared between users

**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:**

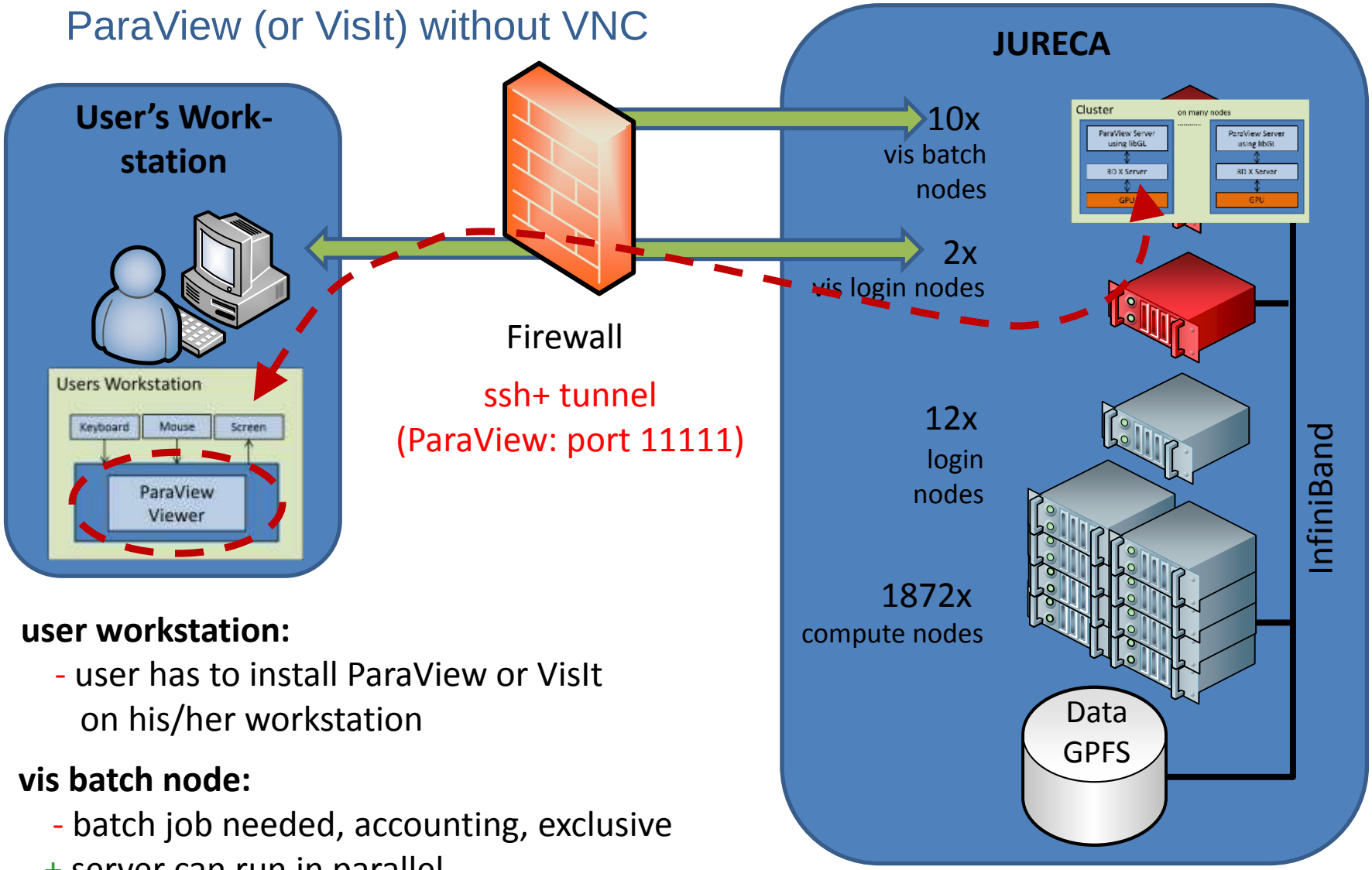
+ no batch job needed, no accounting
- resources shared between users

**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:**

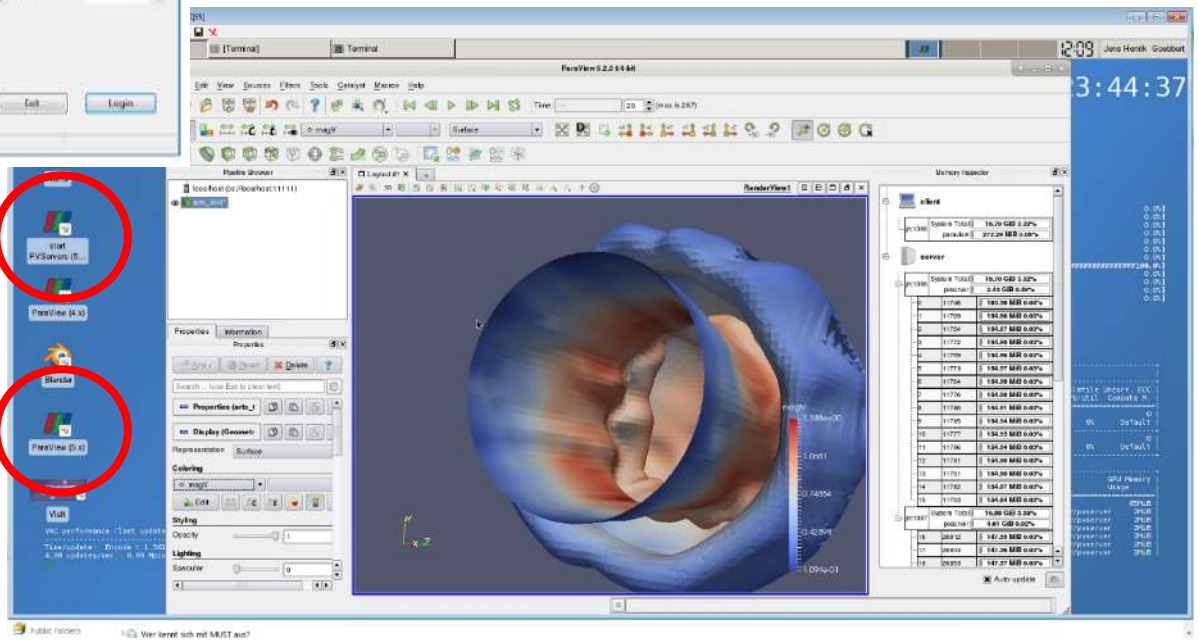
- batch job needed, accounting, exclusive
+ server can run in parallel (but number of vis nodes limited to 4)

# Visualization Software (possible scenarios)

# Scenario 1:

## Parallel ParaView on Vis Batch nodes

Once you have established a VNC session
on one (or more) **vis batch nodes** (not vis login nodes),
you can start ParaView in parallel.



1. Start ParaView Servers
2. Open ParaView GUI (load modules, start “vglrun paraview”)
3. Connect GUI to the pvserver (localhost, port 11111)
   - Click on ‘Connect’ icon in toolbar and add a new server at localhost:11111

# Scenario 2:

## Parallel VisIt on Vis Batch nodes


Once you have established a VNC session
on one (or more) **vis batch nodes** (not vis login nodes),
you can start ParaView in parallel.

Notice:

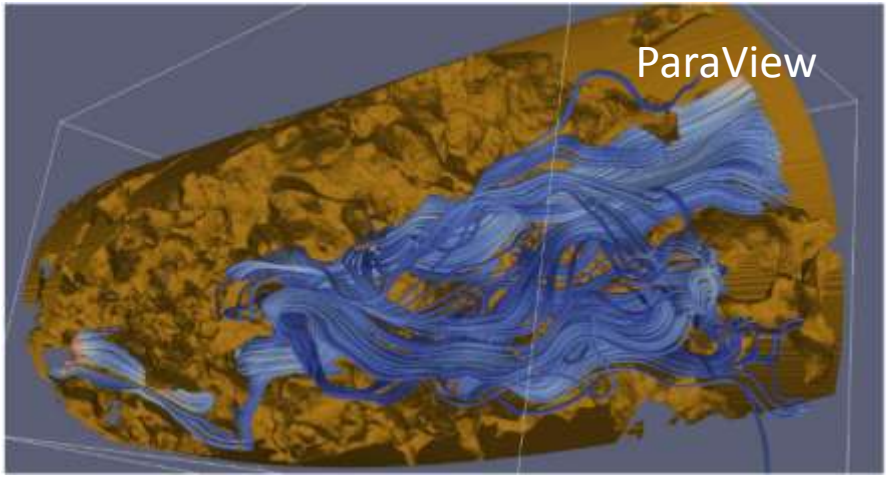
all resources (nodes) are already allocated
after starting the VNC server with sbatch or strudel

1. Open VisIt GUI (load modules, start “vglrun visit –hw-accel”)
2. Inside the VisIt GUI select the proper host profile for JURECA Vis Batch Node (documentation and download link for predefined host profiles here: https://trac.version.fz-juelich.de/vis/wiki/VisIt/Jureca)
3. Select “File open”, in the file-browser choose “JURECA Vis Batch Node” as host.
4. Select a File, choose “localhost” as launch profile, choose number of processors

# Nice To Know

# Nice to know:

## OSPRay

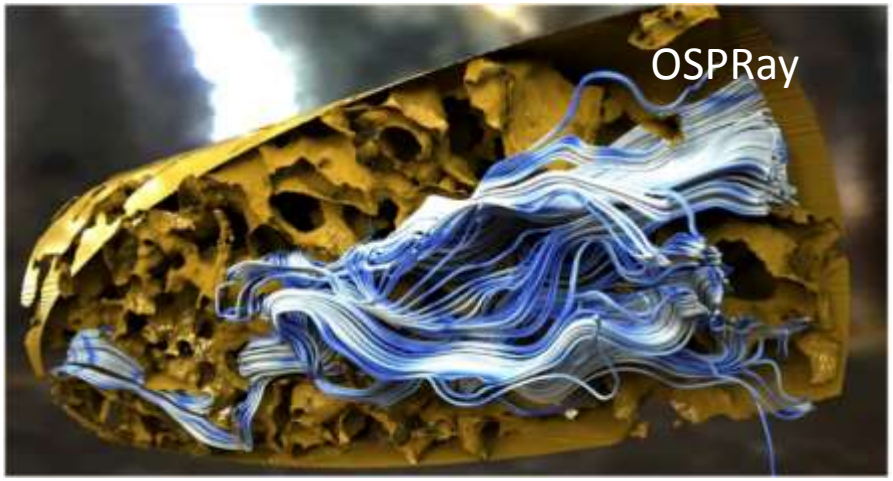



### CPU ray tracing framework for scientific vis. rendering

- 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:

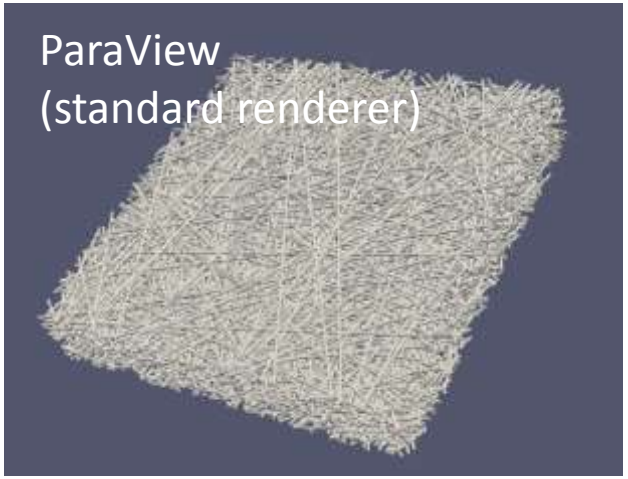
## OSPRay with ParaView

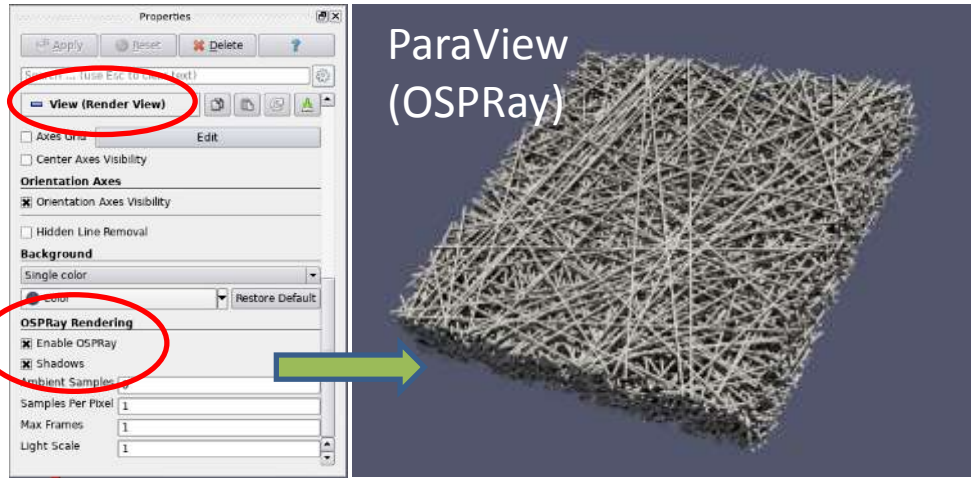
### Ray tracing within ParaView

- 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:
CPUs: 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*

# Summary & Conclusion

## JURECA Visualization Related Documentation

**Please visit**

**https://trac.version.fz-juelich.de/vis/**

**http://www.fz-juelich.de/ias/jsc/EN/Expertise/Support/Visualization/_node.html**

**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