Remote visualization

While the cluster is mainly designed to run batch jobs without user interaction, we realize that in some cases it is necessary to also interactively work on a node - for example when setting up a 3D-model that is too large to fit into your workstation's memory in a simulation software, or if you need to postprocess/filter large amounts of data that has been generated by a job in your BIGWORK directory. To facilitate that, we have installed a special node that has specifically been set up to support interactive/graphical use. We will watch its usage and possibly add further nodes in the future, depending on user demand. CURRENTLY, THIS IS JUST A TEST.

Please note: in this initial phase and due to the very limited resources, only one visualization session (job) lasting no longer than 3 hours is allowed per user at the same time to give everyone a chance to test the service. When the session expires, it will be terminated automatically and without warning (!). A job can request a maximum of 8 CPU cores and 32 GB of main memory, and a maximum of three simultaneous sessions can run on the visualization node at any time to limit the influence of other users on the experience of others.

Please note: If you do not require an intensive 3D hardware acceleration, please use a regular remote desktop session instead to interactively work with your GUI applications (accessed via the menu **Interactive Apps > Cluster Remote Desktop** on the web portal).

Concept of a visualization server

The traditional approach to remote data visualization relies on tunneling the X11 protocol through an SSH connection to run graphical applications. That means adding up latencies between client and server, resulting in slow reactions and a “sluggy” handling of large 3D-geometries in particular. The approach of a visualization server is to perform all rendering on the server's graphics hardware, transferring only the resulting 2D-images to the remote workstation for display. This permits to operate 3D applications efficiently over standard network connections. In addition, the local workstation neither requires a special dedicated graphics card nor large memory or an installation of 3D processing tools.

Since the cluster visualization servers are managed by SLURM via the partition `vis`, you need to submit a job to this partition to get access to the server's resources. Currently, the `vis` partition consist of only one OpenGL-capable server with a single NVIDIA Quadro P4000 GPU, 20 CPU cores and 92 GB of main memory. The node runs a 3D X server and has all the required software tools installed, incuding TurboVNC (Virtual Network Computing) and VirtualGL (an open source software that intercepts 3D remote-rendering commands). The server has a fast connection to both the BIGWORK and PROJECT file-systems (where your datasets are supposed to be placed).

Starting a visualization session

To use the remote visualization service of the LUIS cluster, you need to authenticate to the cluster's web portal and on the dashboard activate the **Cluster Visualization Desktop** item in the **Interactive Apps** menu, cf. figure 1. On the service page that opens, leave the checkbox `Enable`

`OpenGL desktop session` deactivated for the time being (see below the section Optimization). Set the remaining session(job) parameters according to your requirements and click the **Launch** button at the bottom of the service page.

Fig. 1: Remote visualization settings page

You will be redirected to the session information page, where you can see the status of your job, see the top window in figure 2. Wait for the job to get into running state, then click **Launch Cluster Visualization Desktop**, see the bottom window in figure 2, to connect to the remote VNC session that is allocated on the visualization server.

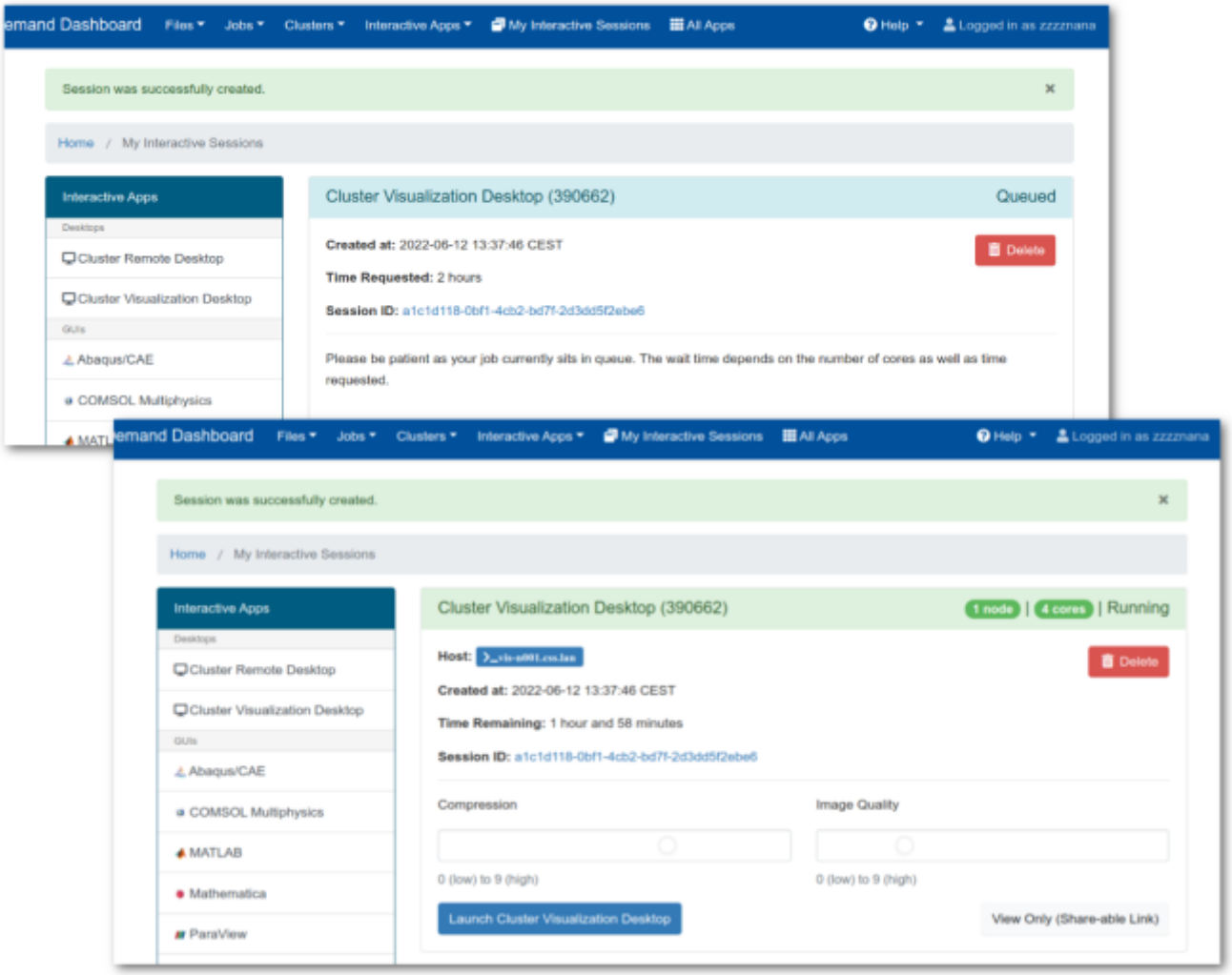

Fig. 2: Remote visualization session job

Once in the desktop session, invoke a terminal (`Terminal Emulator` or similar, e.g. `Terminator` from the sub-menu **Applications > System**) from the menu **Applications** at the top-left of the screen as shown in figure 3.

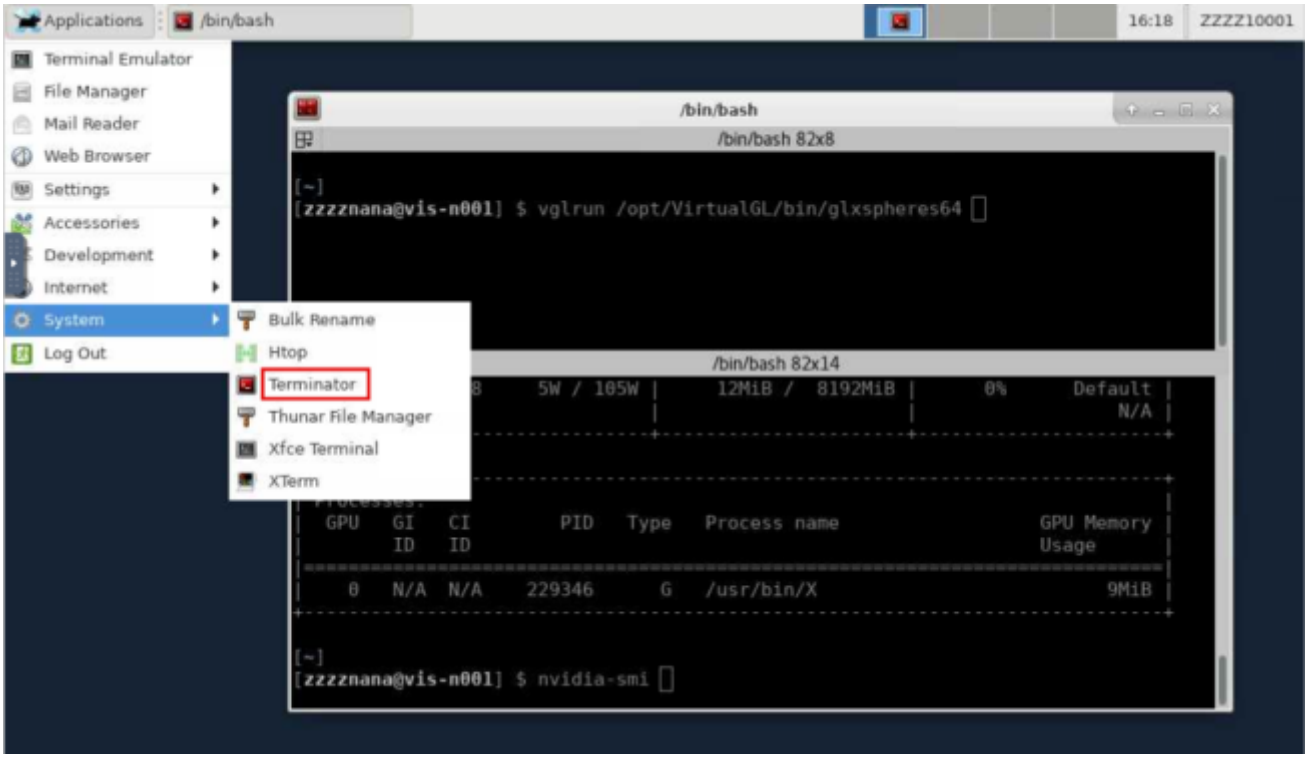

Fig. 3: Remote visualization desktop

To terminate your visualization session either click the item **Applications > Log Out** (figure 3) or cancel the session job using the **Delete** button on the **My Interactive Sessions** page (figure 2).

First test

As a first test, we'll run the OpenGL program `glxspheres64` supplied with the VirtualGL package (cf. figure 4):

```
[user@vis-n001 ~]$ glxspheres64
```

You should see a new window containing some colourful animated spheres. Notice the frame rate. Exit the program (ESC) and start it anew, this time using:

```
[user@vis-n001 ~]$ vglrun glxspheres64
```

You should see a noticeable difference in framerate and measured performance. Exit the tool. You do not need to do this every time, this was just a demonstration to give you a feel for the difference to expect.

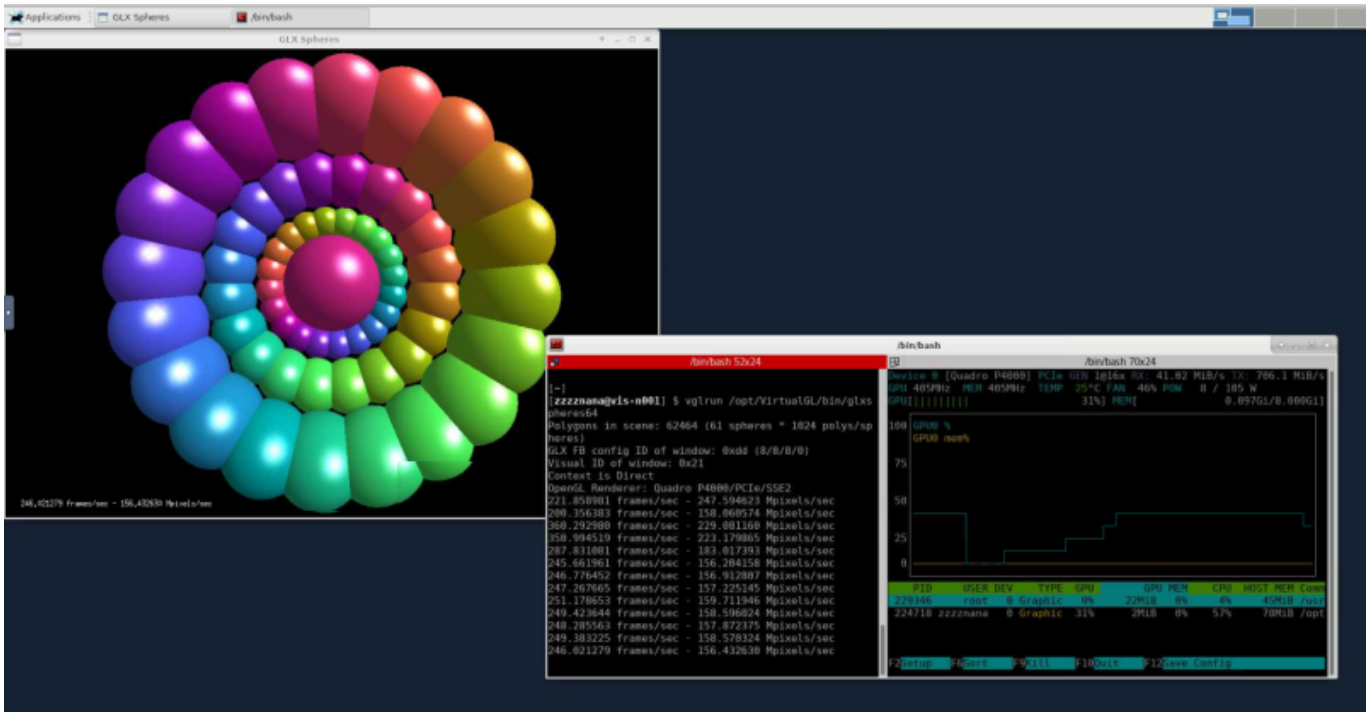

Fig. 4: Running glxspheres64 on the cluster

You can now activate the application you want to use (by loading the appropriate modules, activating your conda environment, etc.) and run it using the `vglrun` prefix:

```
[user@vis-n001 ~]$ vglrun <your-application> <your-application-options>
```

To check whether the graphics cards of the server are actually being used by your application, open a second terminal window and run the command `nvtop`. nvtop (Neat Videocard TOP) is a `(h)top` -like tool that provides real-time information about the usage of NVIDIA and AMD GPUs as well as the processes executing on the GPUs (see the right tab of the terminal window in figure 4). The `nvidia-smi` utility may also be useful.

Optimization

Here are some further optimizations you could try:

- If you activate the checkbox near `Enable OpenGL desktop session` in the session setup, the X server itself will be started using `vglrun`. This should have the effect that software started from within a terminal opened on that desktop should automatically start with OpenGL acceleration enabled (without explicitly using `vglrun`). In some cases, this may not work as expected — then just leave the option deactivated and start your software using `vglrun`.
- In case your internet connection bandwidth is small, you can tell vglrun to use a smaller frame rate. Compare: `vglrun -fps 60 glxspheres64`, `vglrun -fps 20 glxspheres64` and `vglrun -fps 10 glxspheres64`.
- You may also find other vglrun-parameters to tune interactivity by just typing the command by itself. Namely the options `-c`, `-np`, `-q` could be of use over a limited connection.

Application examples

Here are instructions on how to run some applications to enable hardware accelerated OpenGL rendering.

ABAQUS

In a terminal window, load the ABAQUS module and type:

```
vglrun abaqus cae
```

ANSYS Fluent

Load the ANSYS module and run:

```
vglrun fluent -driver opengl
```

If ANSYS Fluent is launched from ANSYS Workbanch, the following variables must be set, otherwise ANSYS Fluent will not use hardware rendering.

```
export FLUENT_WB_OPTIONAL_ARGS="-driver opengl"
export CORTEX_PRE=/opt/VirtualGL/bin/vglrun.
```

COMSOL

Load the COMSOL module and execute:

```
vglrun comsol -3drend ogl
```

GaussView

Load the GaussView module and execute:

```
vglrun gview
```

MATLAB

Load the MATLAB module and execute:

```
vglrun matlab -nosoftwareopengl
```

ParaView

```
module load GCC/11.2.0 OpenMPI/4.1.1 ParaView/5.9.1-mpi
vglrun paraview
```

VMD

```
module load GCC/10.2.0  OpenMPI/4.0.5 VMD/1.9.4a51
vglrun vmd
```