

Remote visualization service

Introduction

The goal of this service is to provide the users a GPU accelerated use of OpenGL applications, especially for pre- and post- processing work, where not only the GPU performance is needed but also fast access to the shared file systems of the cluster and a reasonable amount of RAM.

The service is based on integration of open source tools VirtualGL and TurboVNC together with the cluster's job scheduler PBS Professional.

Currently two compute nodes are dedicated for this service with following configuration for each node:

Visualization node configuration CPU 2x Intel Sandy Bridge E5-2670, 2.6GHz Processor cores 16 (2x8 cores) RAM 64 GB, min. 4 GB per core GPU NVIDIA Quadro 4000, 2GB RAM Local disk drive yes - 500 GB Compute network InfiniBand QDR Schematic overview —————

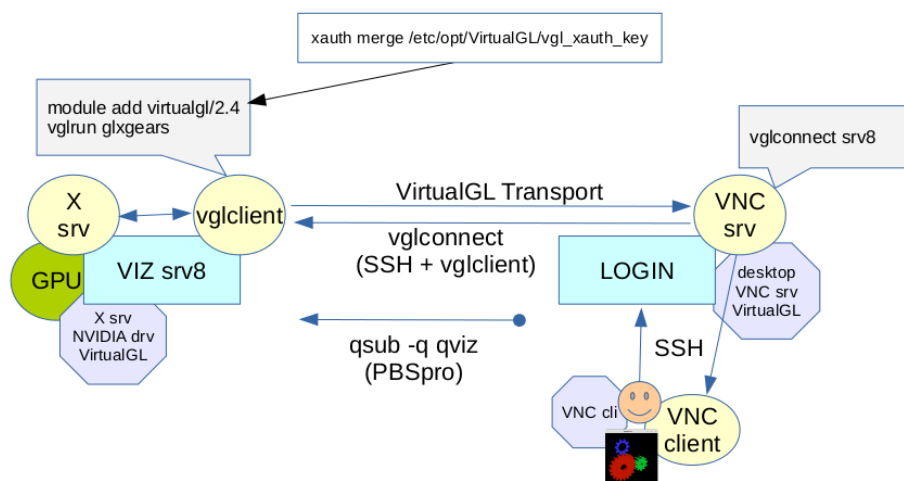


Figure 1: rem_vis_scheme

How to use the service

Setup and start your own TurboVNC server.

TurboVNC is designed and implemented for cooperation with VirtualGL and available for free for all major platforms. For more information and download, please refer to: <http://sourceforge.net/projects/turbovnc/>

Legend

User



Hardware node

LOGIN

Logical Component

VNC
client

Software Component

VNC cli

Hardware component

GPU

Application



Command

vglconnect srv8

Figure 2: rem_vis_legend

Always use TurboVNC on both sides** (server and client) **don't mix TurboVNC and other VNC implementations** (TightVNC, TigerVNC, ...) as the VNC protocol implementation may slightly differ and diminish your user experience by introducing picture artifacts, etc.

The procedure is:

1. Connect to a login node. {#1-connect-to-a-login-node}

Please follow the documentation.

2. Run your own instance of TurboVNC server. {#2-run-your-own-instance-of-turbovnc-server}

To have the OpenGL acceleration, **24 bit color depth must be used**. Otherwise only the geometry (desktop size) definition is needed.

At first VNC server run you need to define a password.

This example defines desktop with dimensions 1200x700 pixels and 24 bit color depth.

```
$ module load turbovnc/1.2.2
$ vncserver -geometry 1200x700 -depth 24
```

Desktop 'TurboVNC: login2:1 (username)' started on display login2:1

Starting applications specified in /home/username/.vnc/xstartup.turbovnc
Log file is /home/username/.vnc/login2:1.log

3. Remember which display number your VNC server runs (you will need it in the future to stop the server). {#3-remember-which-display-number-your-vnc-server-runs-you-will-need-it-in-the-future-to-stop-the-server}

```
$ vncserver -list
```

TurboVNC server sessions:

```
X DISPLAY # PROCESS ID
:1 23269
```

In this example the VNC server runs on display :1.

4. Remember the exact login node, where your VNC server runs. {#4-remember-the-exact-login-node-where-your-vnc-server-runs}

```
$ uname -n  
login2
```

In this example the VNC server runs on **login2**.

5. Remember on which TCP port your own VNC server is running.
{#5-remember-on-which-tcp-port-your-own-vnc-server-is-running}

To get the port you have to look to the log file of your VNC server.

```
$ grep -E "VNC.*port" /home/username/.vnc/login2:1.log  
20/02/2015 14:46:41 Listening for VNC connections on TCP port 5901
```

In this example the VNC server listens on TCP port **5901**.

6. Connect to the login node where your VNC server runs with SSH to tunnel your VNC session.
{#6-connect-to-the-login-node-where-your-vnc-server-runs-with-ssh-to-tunnel-your-vnc-session}

Tunnel the TCP port on which your VNC server is listening.

```
$ ssh login2.anselm.it4i.cz -L 5901:localhost:5901
```

If you use Windows and Putty, please refer to port forwarding setup in the documentation: <https://docs.it4i.cz/anselm-cluster-documentation/accessing-the-cluster/x-window-and-vnc#section-12>

7. If you don't have Turbo VNC installed on your workstation.
{#7-if-you-don-t-have-turbo-vnc-installed-on-your-workstation}

Get it from: <http://sourceforge.net/projects/turbovnc/>

8. Run TurboVNC Viewer from your workstation.
{#8-run-turbovnc-viewer-from-your-workstation}

Mind that you should connect through the SSH tunneled port. In this example it is 5901 on your workstation (localhost).

```
$ vncviewer localhost:5901
```

*If you use Windows version of TurboVNC Viewer, just run the Viewer and use address **localhost:5901**.*

9. Proceed to the chapter "Access the visualization node."
{#9-proceed-to-the-chapter-access-the-visualization-node}

Now you should have working TurboVNC session connected to your workstation.

10. After you end your visualization session. {#10-after-you-end-your-visualization-session}

Don't forget to correctly shutdown your own VNC server on the login node!

```
$ vncserver -kill :1
```

Access the visualization node

To access the node use a dedicated PBS Professional scheduler queue `qviz**`. The queue has following properties:

|queue|active project|project resources

nodes

min ncpus*

priority

authorization

walltimedefault/max || — | — | |qviz Visualization queue |yes |none
required |2 |4 |>150 |no |1 hour / 2 hours |

Currently when accessing the node, each user gets 4 cores of a CPU allocated, thus approximately 16 GB of RAM and 1/4 of the GPU capacity. *If more GPU power or RAM is required, it is recommended to allocate one whole node per user, so that all 16 cores, whole RAM and whole GPU is exclusive. This is currently also the maximum allowed allocation per one user. One hour of work is allocated by default, the user may ask for 2 hours maximum.*

To access the visualization node, follow these steps:

1. In your VNC session, open a terminal and allocate a node using PBSPro `qsub` command. {#1-in-your-vnc-session-open-a-terminal-and-allocate-a-node-using-pbspro-qsub-command}

This step is necessary to allow you to proceed with next steps.

```
$ qsub -I -q qviz -A PROJECT_ID
```

In this example the default values for CPU cores and usage time are used.

```
$ qsub -I -q qviz -A PROJECT_ID -l select=1:ncpus=16 -l walltime=02:00:00
```

*Substitute **PROJECT_ID** with the assigned project identification string.*

In this example a whole node for 2 hours is requested.

If there are free resources for your request, you will have a shell running on an assigned node. Please remember the name of the node.

```
$ uname -n  
srv8
```

In this example the visualization session was assigned to node **srv8**.

2. In your VNC session open another terminal (keep the one with interactive PBSPro job open). {#2-in-your-vnc-session-open-another-terminal-keep-the-one-with-interactive-pbspro-job-open}

Setup the VirtualGL connection to the node, which PBSPro allocated for your job.

```
$ vglconnect srv8
```

You will be connected with created VirtualGL tunnel to the visualization node, where you will have a shell.

3. Load the VirtualGL module. {#3-load-the-virtualgl-module}

```
$ module load virtualgl/2.4
```

4. Run your desired OpenGL accelerated application using VirtualGL script “vglrun”. {#4-run-your-desired-opengl-accelerated-application-using-virtualgl-script-vglrun}

```
$ vglrun glxgears
```

Please note, that if you want to run an OpenGL application which is available through modules, you need at first load the respective module. E. g. to run the **Mentat** OpenGL application from **MARC** software package use:

```
$ module load marc/2013.1  
$ vglrun mentat
```

5. After you end your work with the OpenGL application. {#5-after-you-end-your-work-with-the-opengl-application}

Just logout from the visualization node and exit both opened terminals and end your VNC server session as described above.

Tips and Tricks

If you want to increase the responsibility of the visualization, please adjust your TurboVNC client settings in this way:

To have an idea how the settings are affecting the resulting picture quality three levels of “JPEG image quality” are demonstrated:

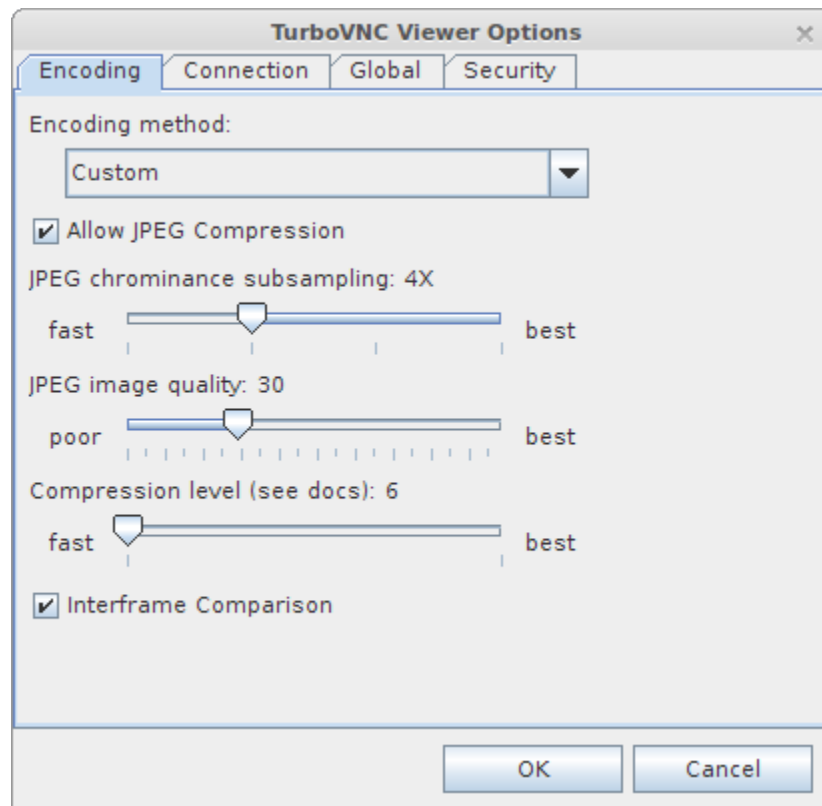


Figure 3: rem_vis_settings

1. JPEG image quality = 30

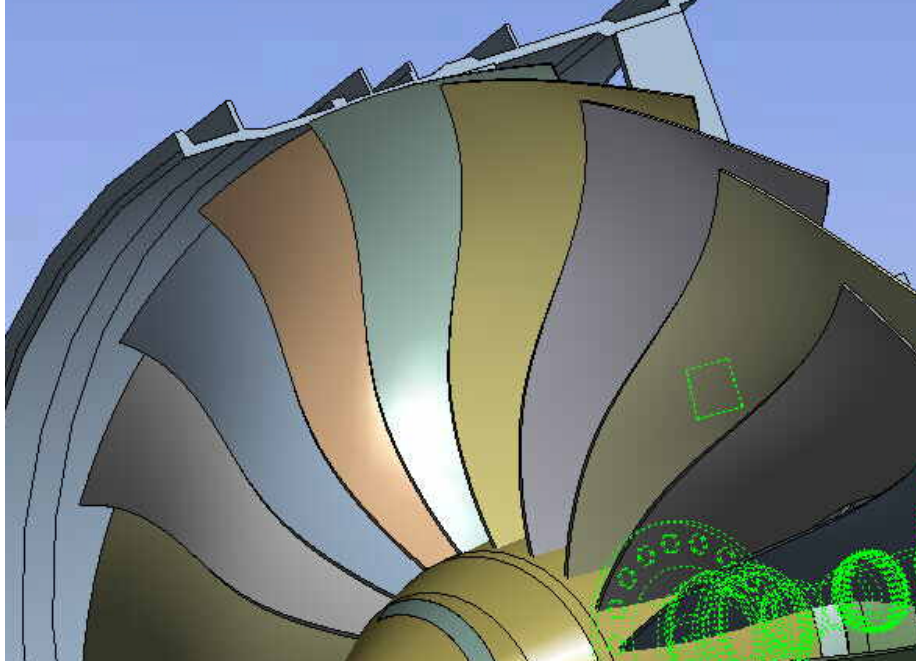


Figure 4: rem_vis_q3

2. JPEG image quality = 15
3. JPEG image quality = 10

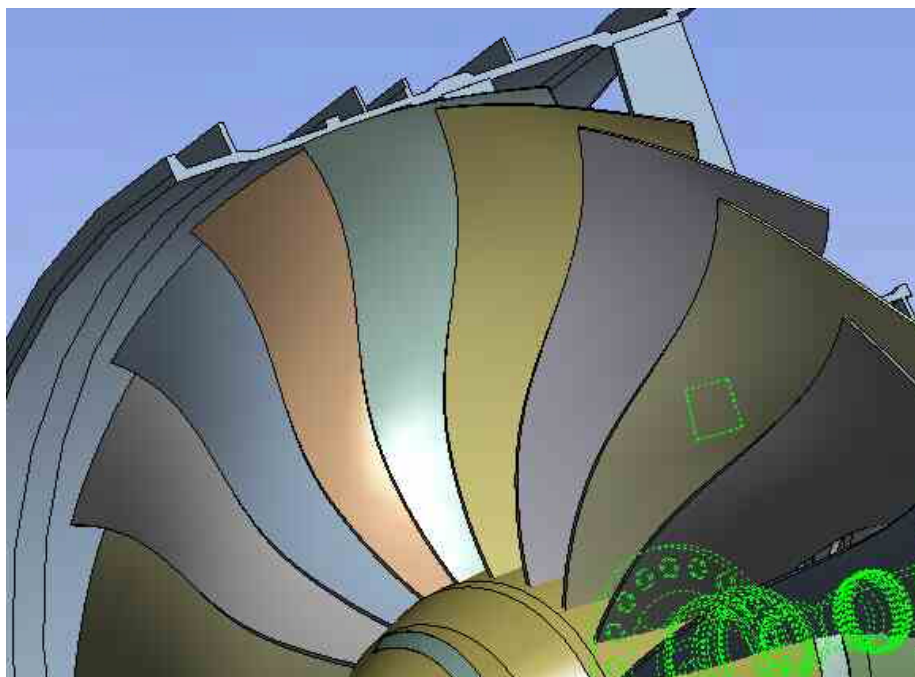


Figure 5: rem_vis_q2

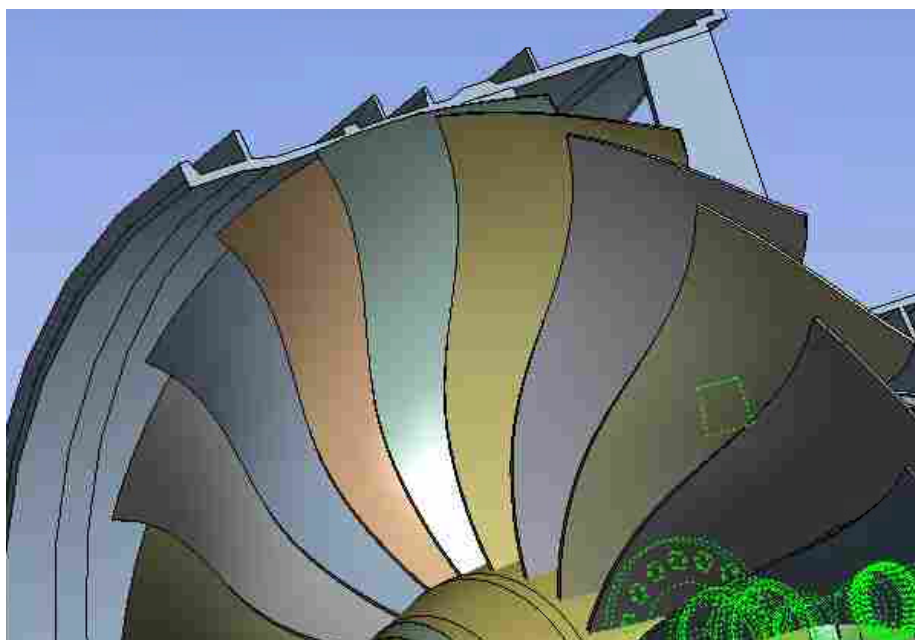


Figure 6: rem_vis_q1