Remote Graphics on the GPC
Client-Server Application and VNC

Ramses van Zon

SciNet User Group Meeting

October 14, 2015
Outline

1. X Forwarding
2. VNC
3. Tunneling & VNC
4. Alternative remote visualization: paraview
5. Planned Visualization Nodes
Remote graphics using X

SSH X forwarding – if an X server has been installed locally (for Linux and MacOS this is often already there by default)

$ ssh -Y login.scinet.utoronto.ca
$ ssh -Y gpc0x

This can be slow, depending on various factors, eg. low-bandwidth/high-latency connections (e.g. home internet connections).
Remote graphics using X

ssh X forwarding – if an X server has been installed locally (for Linux and MacOS this is often already there by default)

$ ssh -Y login.scinet.utoronto.ca
$ ssh -Y gpc0x

This can be slow, depending on various factors, eg. low-bandwidth/high-latency connections (e.g. home internet connections).
Remote graphics using X

ssh X forwarding – if an X server has been installed locally (for Linux and MacOS this is often already there by default)

```
$ ssh -Y login.scinet.utoronto.ca
$ ssh -Y gpc0x
```

This can be slow, depending on various factors, eg. low-bandwidth/high-latency connections (e.g. home internet connections).
Example: X forwarding on GPC compute nodes

▼ Establish the remote connection with X forwarding

$ ssh -Y login.scinet.utoronto.ca
$ gpc
$ debugjob

(gpc and debugjob forward X, but ssh or qsub would need explicit -X options)

▼ Let’s have some fun...

$ module load gnuplot
$ gnuplot
  or ...
$ module load grace
$ xmgr
Example: X forwarding on GPC compute nodes

▼ Establish the remote connection with X forwarding

$ ssh -Y login.scinet.utoronto.ca
$ gpc
$ debugjob

(gpc and debugjob forward X, but ssh or qsub would need explicit -X options)

▼ Let’s have some fun...

$ module load gnuplot
$ gnuplot
  or ...
$ module load grace
$ xmgr
Example: X forwarding on GPC compute nodes

▼ Establish the remote connection with X forwarding

$ ssh -Y login.scinet.utoronto.ca
$ gpc
$ debugjob

(gpc and debugjob forward X, but ssh or qsub would need explicit -X options)

▼ Let’s have some fun...

$ module load gnuplot
$ gnuplot
  or ...

$ module load grace
$ xmgr
Example: X forwarding on GPC compute nodes

▼ Establish the remote connection with X forwarding

$ ssh -Y login.scinet.utoronto.ca
$ gpc
$ debugjob

(gpc and debugjob forward X, but ssh or qsub would need explicit -X options)

▼ Let’s have some fun...

$ module load gnuplot
$ gnuplot
  or ...

$ module load grace
$ xmgr
- VNC: Virtual Network Computing
- VNC uses an X server on the remote machine, and a local viewer.
- VNC behaves as if taking continuous desktop snapshots.
- It uses compression techniques to reduce the required bandwidth, and transfers only the parts of the desktop that have changed.
- Using VNC with an SSH tunnel and a password is quick and secure.
VNC vs X forwarding

- X forwarding will work and be just fine in many cases
- VNC offers a potentially more suitable protocol for such remote connections
- Remote X graphics applications require a local X server, and transmit many little events and data messages. On a network with high latency, the number of roundtrips needed makes X slow and less responsive.
- X’s speed depends more on the type of application than VNC (e.g. java applications tend to be very slow over X, but are OK over VNC).
- VNC typically requires fewer roundtrip, hence is often more responsive.
A bit about client-server setups

![Diagram of client-server setup with clients and server connected through the internet.](image)
A bit about client-server setups

- The server may be running many services
  e.g. `sshd`, `http`, `gfps`, `vnc`, ...

- To distinguish these services, they listen to different "ports".

- The clients talk to these services.
  e.g., `ssh`, `vncviewer`, ...

- Clients are specific to the service.

- To connect, they need to know the hostname of the server and the port number.
A bit about client-server setups

- The server may be running many services, e.g., sshd, http, gfps, vnc, ...

- To distinguish these services, they listen to different “ports”.

- The clients talk to these services, e.g., ssh, vncviewer, ...

- Clients are specific to the service.

- To connect, they need to know the hostname of the server and the port number.
The server may be running many services 
\textit{e.g. sshd, http, gfps, vnc, \ldots}

To distinguish these services, they listen to different "ports".

The clients talk to these services. 
\textit{e.g., ssh, vncviewer, \ldots}

Clients are specific to the service.

To connect, they need to know the hostname of the server and the port number.
A bit about client-server setups

- The server may be running many services, e.g., sshd, http, gfps, vnc, ...
- To distinguish these services, they listen to different “ports”.
- The clients talk to these services, e.g., ssh, vncviewer, ...
- Clients are specific to the service.
- To connect, they need to know the hostname of the server and the port number.
A bit about client-server setups

- The server may be running many services, e.g., `sshd`, `http`, `gfps`, `vnc`, ...
- To distinguish these services, they listen to different "ports".
- The clients talk to these services, e.g., `ssh`, `vncviewer`, ...
- Clients are specific to the service.
- To connect, they need to know the hostname of the server and the port number.
Now via the gateway, login.scinet.utoronto.ca:

Must use port forwarding.

- ssh encrypts: secure over network
- VNC password: no exposure through local or remote port
- only 1 client
- a high level of security
Now via the gateway, login.scinet.utoronto.ca:

Must use **port forwarding**.

- ssh encrypts: secure over network
- VNC password: no exposure through local or remote port
- only 1 client
- a high level of security
Now via the gateway, login.scinet.utoronto.ca:

Must use **port forwarding**.

- **ssh** encrypts: secure over network
- VNC password: no exposure through local or remote port
- Only 1 client
- A high level of security
Now via the gateway, login.scinet.utoronto.ca:

Must use **port forwarding**.

- **ssh** encrypts: secure over network
- VNC password: no exposure through local or remote port
  - only 1 client
  - a high level of security
Now via the gateway, login.scinet.utoronto.ca:

Must use **port forwarding**.

- **ssh encrypts**: secure over network
- VNC password: no exposure through local or remote port
- only 1 client
- a high level of security

- VNC client
- **login.scinet.utoronto.ca**
- **gpc0x**
- VNC/remote-PORT
- VNC server
- X server
Now via the gateway, login.scinet.utoronto.ca:

Must use **port forwarding**.

- ssh encrypts: secure over network
- VNC password: no exposure through local or remote port
- only 1 client
- a high level of security

VNC client

gpc0x

VNC/remote-PORT

VNC server

X server
Prerequisites

- **on your local machines**
  - Install an **ssh client**
    - Linux and MacOS: come with them!
    - Windows: MobaXterm, Cygwin, PuTTY
  - Install a **VNC client**
    - Linux and MacOS: most likely are there already
    - If not: TightVNC or TigerVNC are a good option

- **on the remote machine (on GPC@SciNet)**
  - Require a **VNC server**
    - we will be using *modules*
    - `module load vnc` \(\rightarrow\) VNC server & scripts
    - This module requires the `Xlib`raries module.
Prerequisites

► on your local machines

- Install an **ssh client**
  - Linux and MacOS: come with them!
  - Windows: MobaXterm, Cygwin, PuTTY

- Install a **VNC client**
  - Linux and MacOS: most likely are there already
  - If not: TightVNC or TigerVNC are a good option

► on the remote machine (on GPC@SciNet)

- Require a **VNC server**
  - we will be using modules
  - module load vnc → VNC server & scripts
  - This module requires the Xlibraries module.
Prerequisites

- on your *local machines*
  - Install an **ssh client**
    - Linux and MacOS: come with them!
    - Windows: MobaXterm, Cygwin, PuTTY
  - Install a **VNC client**
    - Linux and MacOS: most likely are there already
    - If not: TightVNC or TigerVNC are a good option

- on the *remote machine* (on GPC@SciNet)
  - Require a **VNC server**
    - we will be using *modules*
    - module load `vnc` ⇝ VNC server & scripts
    - This module requires the *Xlibraries* module.
Set up a VNC session on a devel node

1. Log into scinet.
2. Log into a devel node (gpc01, ...).
3. Stay there.
4. Start a VNC server on that node
5. Using `ssh`, **forward** a local port from your local machine to a port on that devel node, via `login.scinet`.
6. Start the VNC client on local machine
Set up a VNC session on a compute node

1. Log into scinet.
2. Log into a devel node (gpc01, ...).
3. Get an interactive compute node using `qsub -I` or `debugjob`.
4. Start a VNC server on a compute node.
5. Using `ssh`, **forward** a local port from your local machine to a port on that compute node, via `login.scinet`.
6. Start the VNC client on local machine.
Setting up a VNC session - Start the VNC session

1) connect with `ssh` to `login.scinet.utoronto.ca`

```
$ ssh login.scinet.utoronto.ca
```

2) connect with `ssh` to a development node: gpc[01-04]

```
$ ssh gpc0x # or just type gpc
```

3) Hop to a compute node using `qsub`

```
$ qsub -I -q debug -l nodes=1:ppn=8,walltime=02:00:00
```

(or just type `debugjob`)

4) finally, load modules & start vnc

```
$ module load Xlibraries vnc
$ vnc start
```

Prompts for password
Do not leave it blank!

Note down the port number.
1) connect with `ssh` to `login.scinet.utoronto.ca`

```
$ ssh login.scinet.utoronto.ca
```

2) connect with `ssh` to a development node: `gpc[01-04]`

```
$ ssh gpc0x # or just type gpc
```

3) Hop to a compute node using `qsub`

```
$ qsub -I -q debug -l nodes=1:ppn=8,walltime=02:00:00
```

(or just type `debugjob`)

4) finally, load modules & start vnc

```
$ module load Xlibraries vnc
$ vnc start
```

➡️ prompts for password
Do not leave it blank!

➡️ Note down the port number.
Setting up a VNC session - Start the VNC session

1) connect with `ssh` to login.scينет.utoronto.ca

```bash
$ ssh login.scينет.utoronto.ca
```

2) connect with `ssh` to a development node: gpc[01-04]

```bash
$ ssh gpc0x # or just type gpc
```

3) Hop to a compute node using `qsub`

```bash
$ qsub -l nodes=1:ppn=8,walltime=02:00:00
```

(or just type debugjob)

4) finally, load modules & start vnc

```bash
$ module load Xlibraries vnc
$ vnc start
```

- prompts for password
  - Do not leave it blank!
- Note down the port number.
1) connect with `ssh` to login.scinet.utoronto.ca

```bash
$ ssh login.scinet.utoronto.ca
```

2) connect with `ssh` to a development node: gpc[01-04]

```bash
$ ssh gpc0x # or just type gpc
```

3) Hop to a compute node using `qsub`

```bash
$ qsub -I -q debug -l nodes=1:ppn=8,walltime=02:00:00
```

(or just type `debugjob`)

4) finally, load modules & start vnc

```bash
$ module load Xlibraries vnc
$ vnc start
```

.prompts for password
Do not leave it blank!

Note down the port number.
Setting up a VNC session - Start the VNC session

1) connect with `ssh` to `login.scinet.utoronto.ca`

$ ssh login.scinet.utoronto.ca

2) connect with `ssh` to a development node: `gpc[01-04]`

$ ssh gpc0x # or just type `gpc`

3) Hop to a compute node using `qsub`

$ qsub -I -q debug -l nodes=1:ppn=8,walltime=02:00:00

(or just type `debugjob`)

4) finally, load modules & start vnc

$ module load Xlibraries vnc
$ vnc start

---

Prompts for password
Do not leave it blank!
Note down the port number.

SNUG - October 2015
Remote Graphics on GPC
October 14, 2015
Setting up a VNC session - Start the VNC session

1) connect with `ssh` to `login.scinet.utoronto.ca`

```
$ ssh login.scinet.utoronto.ca
```

2) connect with `ssh` to a development node: gpc[01-04]

```
$ ssh gpc0x # or just type gpc
```

3) Hop to a compute node using `qsub`

```
$ qsub -I -q debug -l nodes=1:ppn=8,walltime=02:00:00
```

(or just type `debugjob`)

4) finally, load modules & start vnc

```
$ module load Xlibraries vnc
$ vnc start
```

- prompts for password
  Do not leave it blank!
- Note down the port number.
Setting up a VNC session - Start the VNC session

1) connect with `ssh` to login.scinet.utoronto.ca

   $ ssh login.scinet.utoronto.ca

2) connect with `ssh` to a development node: gpc[01-04]

   $ ssh gpc0x # or just type gpc

3) Hop to a compute node using `qsub`

   $ qsub -I -q debug -l nodes=1:ppn=8,walltime=02:00:00
   (or just type `debugjob`)

4) finally, load modules & start vnc

   $ module load Xlibraries vnc
   $ vnc start

⇒ prompts for password
Do not leave it blank!

⇒ Note down the port number.
Setting up a VNC session - Start the VNC session

1) connect with `ssh` to `login.scinet.utoronto.ca`

```
$ ssh login.scinet.utoronto.ca
```

2) connect with `ssh` to a development node: `gpc[01-04]`

```
$ ssh gpc0x # or just type gpc
```

3) Hop to a compute node using `qsub`

```
$ qsub -l nodes=1:ppn=8,walltime=02:00:00
```

(or just type `debugjob`)

4) finally, load modules & start vnc

```
$ module load Xlibraries vnc
$ vnc start
```

➤ prompts for password
Do not leave it blank!
➤ Note down the port number.
Setting up a VNC session - Setup a secure SSH tunnel

- All external traffic has to go through login.scinet.utoronto.ca

5) ssh allows for port forwarding, which takes a port on a local machine and forwards it to a port on the devel/compute node

```
$ ssh login.scinet.utoronto.ca -Lxxxx:nodeName:PORT -N
```

- nodeName: gpc[01-04] or a compute node
- xxxx: port one local machine (usually 5900 or 15900, your choice!)
- PORT: port of VNC server as returned by vnc start.

- do not exit this shell, or the tunnel will collapse

eg.

```
$ssh login.scinet.utoronto.ca -L15900:gpc03:11950 -N
```

- For faster communication try the following ssh extended command:

```
$ssh -C -c arcfour login.-----ca -L15900:gpc03:11950 -N
```
Setting up a VNC session - Setup a secure SSH tunnel

➤ All external traffic has to go through login.scinet.utoronto.ca

5) ssh allows for port forwarding, which takes a port on a local machine and forwards it to a port on the devel/compute node

```bash
$ ssh login.scinet.utoronto.ca -Lxxxx:nodeName:PORT -N
```

➤ nodeName: gpc[01-04] or a compute node

➤ xxxx: port one local machine (usually 5900 or 15900, your choice!)

➤ PORT: port of VNC server as returned by vnc start.

➤ do not exit this shell, or the tunnel will collapse

eg.

```
$ ssh login.scinet.utoronto.ca -L15900:gpc03:11950 -N
```

➤ For faster communication try the following ssh extended command:

```
$ssh -C -c arcfour login.scinet.utoronto.ca -L15900:gpc03:11950 -N
```
Setting up a VNC session - Setup a secure SSH tunnel

- All *external traffic* has to go through login.scinet.utoronto.ca
- 5) `ssh` allows for port forwarding, which takes a port on a local machine and forwards it to a port on the devel/compute node

```
$ ssh login.scinet.utoronto.ca -Lxxxx:nodeName:PORT -N
```

- `nodeName`: gpc[01-04] or a compute node
- `xxxx`: port one local machine (usually 5900 or 15900, your choice!)
- `PORT`: port of VNC server as returned by `vnc start`.
- Do **not** exit this shell, or the *tunnel* will collapse

eg.

```
$ssh login.scinet.utoronto.ca -L15900:gpc03:11950 -N
```

- For faster communication try the following `ssh` extended command:

```
$ssh -C -c arcfour login...ca -L15900:gpc03:11950 -N
```
Setting up a VNC session - Setup a secure SSH tunnel

- All external traffic has to go through login.scinet.utoronto.ca

5) ssh allows for port forwarding, which takes a port on a local machine and forwards it to a port on the devel/compute node

```
$ ssh login.scinet.utoronto.ca -Lxxxx:nodeName:PORT -N
```

- nodeName: gpc[01-04] or a compute node
- xxxx: port one local machine (usually 5900 or 15900, your choice!)
- PORT: port of VNC server as returned by vnc start.
- do not exit this shell, or the tunnel will collapse

eg.

```
$ssh login.scinet.utoronto.ca -L15900:gpc03:11950 -N
```

- For faster communication try the following ssh extended command:

```
$ssh -C -c arcfour login....ca -L15900:gpc03:11950 -N
```
Setting up a VNC session - Setup a secure SSH tunnel

⇒ All **external traffic** has to go through login.scinet.utoronto.ca
⇒ 5) **ssh** allows for port forwarding, which takes a port on a local machine and forwards it to a port on the devel/compute node

```
$ ssh login.scinet.utoronto.ca -Lxxxx:nodeName:PORT -N
```

⇒ **nodeName**: gpc[01-04] or a compute node
⇒ **xxxx**: port one local machine (usually 5900 or 15900, your choice!)
⇒ **PORT**: port of VNC server as returned by `vnc start`
⇒ do **not** exit this shell, or the **tunnel** will collapse

eg.

```
$ ssh login.scinet.utoronto.ca -L15900:gpc03:11950 -N
```

⇒ For faster communication try the following ssh extended command:

```
$ ssh -C -c arcfour login...ca -L15900:gpc03:11950 -N
```
6) any VNC viewer, can now be *attached* to the remote VNC server

- eg. in Linux,
  $ vncviewer localhost:15901

- eg. in MacOS,
  $ open vnc://localhost:15901

- type the password for the VNC server
- you will get a ‘desktop’ with an Xterm

- there may be options to improve the efficiency of the connection,
  vncviewer -encodings ’copyrect tight hextile’ localhost:15901
  or
  vncviewer -PreferredEncoding ’copyrect tight hextile’ localhost:15901
Setting up a VNC session - Start the VNC client

6) any VNC viewer, can now be attached to the remote VNC server

- eg. in Linux,

  $ vncviewer localhost:15901

- eg. in MacOS,

  $ open vnc://localhost:15901

- type the password for the VNC server

- you will get a ‘desktop’ with an Xterm

- there may be options to improve the efficiency of the connection,

  vncviewer -encodings 'copyrect tight hextile' localhost:15901
  or

  vncviewer -PreferredEncoding 'copyrect tight hextile' localhost:15901
6) any VNC viewer, can now be *attached* to the remote VNC server

⇒ eg. in Linux,

```
$ vncviewer localhost:15901
```

⇒ eg. in MacOS,

```
$ open vnc://localhost:15901
```

type the password for the VNC server

⇒ you will get a ‘desktop’ with an Xterm

⇒ there may be options to improve the efficiency of the connection,

```
vncviewer -encodings 'copyrect tight hextile' localhost:15901
```
or

```
vncviewer -PreferredEncoding 'copyrect tight hextile' localhost:15901
```
Setting up a VNC session - shortcut

- it is possible to combine steps 5+6, when using eg. TightVNC viewer,

  ```
  $ vncviewer -via login.scinet.utoronto.ca gpc03:PORT
  or
  $ vncviewer -via login.scinet.utoronto.ca gpc03:ALTPORT
  ```

  where ALTPORT=PORT-5900

- to control compression for TightVNC’s combining steps 5+6,

  - set environment variable VNC_VIA_CMD, e.g.

  ```
  $ VNC_VIA_CMD='ssh -C -c arcfour -f -L %L:%H:%R %G sleep 20'
  $ export VNC_VIA_CMD
  ```
Setting up a VNC session - shortcut

- it is possible to combine steps 5+6, when using eg. TightVNC viewer,
  
  `$ vncviewer -via login.scinet.utoronto.ca gpc03:PORT`

  or

  `$ vncviewer -via login.scinet.utoronto.ca gpc03:ALTPORT`

  where $ALTPORT=PORT-5900$

- to control compression for TightVNC’s combining steps 5+6,
  
  set environment variable $VNC_VIA_CMD$, e.g.

  `$ VNC_VIA_CMD='ssh -C -c arcfour -f -L %L:%H:%R %G sleep 20'`

  `$ export VNC_VIA_CMD$`
What does it look like?
Client-side usage Details

Light-weight window manager \texttt{twm}

- \texttt{Xterm} starts by default
- Icon, close, maximize and resize buttons are found in title bars
- \texttt{Ctrl-Tab} brings successive windows to the foreground
- Left mouse click on the background pops up the \texttt{twm} menu
- Use ‘Exit’ option from the \texttt{twm} menu to terminate VNC

Implementation

- \texttt{Xvfb} for the Xserver
- \texttt{x11vnc} for the VNC server
### VNC scripts available

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vnc stop</td>
<td>Stop the VNC servers, killing any X applications</td>
</tr>
<tr>
<td>vnc status</td>
<td>Probes whether the VNC server and the X server are running</td>
</tr>
<tr>
<td>vnc detach</td>
<td>Stop the VNC servers, killing any X applications</td>
</tr>
<tr>
<td>vnc help</td>
<td>Display a help message about VNC/X/twm environment</td>
</tr>
<tr>
<td>vnc start</td>
<td>Has some additional options:</td>
</tr>
<tr>
<td></td>
<td>(-r) RESOLUTION (\Rightarrow) set X’s resolution (default: 800x544x16)</td>
</tr>
<tr>
<td></td>
<td>(-s) FRACTION (\Rightarrow) use x11vnc’s scaling feature</td>
</tr>
<tr>
<td></td>
<td>(-v) 0</td>
</tr>
<tr>
<td></td>
<td>(-n) (\Rightarrow) switch on x11vnc’s ncache feature</td>
</tr>
<tr>
<td></td>
<td>(-b) (\Rightarrow) use a blank background</td>
</tr>
<tr>
<td>Directory</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>~/.xinitrc</td>
<td>Initialization of X: start window manager twm and xterm</td>
</tr>
<tr>
<td>~/.twmrc</td>
<td>Settings file for window manager twm</td>
</tr>
<tr>
<td>~/.vnc</td>
<td>Directory with encrypted VNC password and other settings</td>
</tr>
<tr>
<td>~/.fr</td>
<td>Directory with settings for FileRunner</td>
</tr>
</tbody>
</table>
Closing the VNC viewer window instead of using Exit in the twm menu, keeps the X server running on the remote devel/compute node. Try, for instance doing so, and reconnecting the local viewer. Also useful, when the connection is lost...

SciNet usage

- only available on GPC system
Alternative remote visualization: paraview

- Some visualization packages have a built-in server-client setup
- Paraview is a prime example.
- Still need to do port forwarding.
- Server and client versions of paraview must match.

**Example**

1. Setup tunnel:

```
$ ssh login.scinet.utoronto.ca -L11111:nodeName:11111 -N
```

2. Start paraview server

```
$ module load intel/15.0.2 gcc/4.8.1 python/2.7.5
$ module load openmpi/intel/1.6.4 extras paraview/4.1.0
$ mpirun -np 8 pvserver -use-offscreen-rendering
```

3. Start local paraview gui and select “File->Connect”.
Alternative remote visualization: paraview

- Some visualization packages have a built-in server-client setup
- Paraview is a prime example.
- Still need to do port forwarding.
- Server and client versions of paraview must match.

Example

1. **Setup tunnel:**
   ```bash
   $ ssh login.scinet.utoronto.ca -L11111:nodeName:11111 -N
   ```

2. **Start paraview server**
   ```bash
   $ module load intel/15.0.2 gcc/4.8.1 python/2.7.5
   $ module load openmpi/intel/1.6.4 extras paraview/4.1.0
   $ mpirun -np 8 pvserver -use-offscreen-rendering
   ```

3. **Start local paraview gui and select “File->Connect”**.
We are planning to repurpose a couple of nodes dedicated as visualization nodes

- Longer than usual interactive jobs
- Large memory pool (≈ 128 GB)
- Software available: eg. VisIt, ParaView, VMD, ...
- Available both interactively and through job submission
Dedicated Viz Nodes - preloaded modules?

Xlibraries
vnc
git
gcc/4.8.1
intel/15.0.2
python/2.7.5
openmpi/intel/1.6.4
paraview/4.1.0
gnuplot/4.6.1
grace/5.1.22
vmd/1.9
visit/2.6.3
ImageMagick/6.6.7
ffmpeg/2.1.3
hdf5/187-v18-serial-intel
octave/4.0.0
pgplot/5.2.2-intel
ncl/6.1.0

We welcome your input as we set this up!