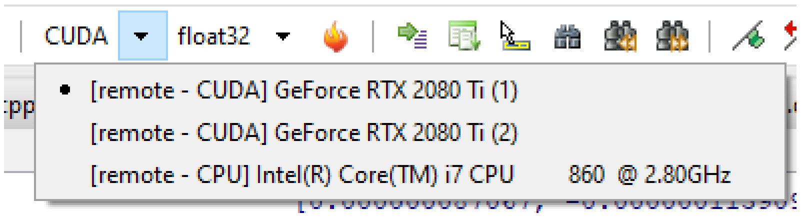

CUDA
float32
[remote - CUDA] GeForce RTX 2080 Ti (1)
[remote - CUDA] GeForce RTX 2080 Ti (2)
[remote - CPU] Intel(R) Core(TM) i7 CPU 860 @ 2.80GHz

Figure 1: Remote device selection

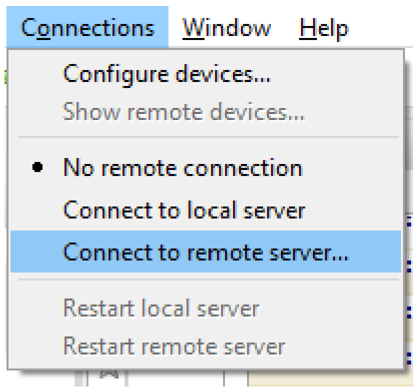


Figure 2: Connections menu

Remote Debugging

Redshift now offers a brand new feature: *remote debugging*. Using remote debugging, you can use Redshift on low-end computers with no discrete GPU with the possibility to connect to a powerful remote machine with one (or multiple) discrete GPUs. This also opens up a way to share GPUs between multiple users (e.g. in a shared remote server). In Redshift, remote debugging has a total integration that allows the user to develop, run, debug and profile code without restrictions. At program startup, the user connects to a remote debugging server. When the connection succeeds, the server IP and host name are shown in the title bar. The remote devices are displayed in the device selection combo box, with prefix remote:

The source code of the program is stored on the local file system. When the Quasar program is started, the program is transmitted to the remote debugging server and executed on the server, in its entirety. The program executes at the native speed of the server. There is still possibility to place breakpoints in the programming code, add watch expressions, obtain tooltips. In this case, communication between the local client and debugging server occurs.

Remote debugging is available through the "Default debugger" (see menu Debug/Program start modes in Redshift). Also a new connections menu has been added:

The following options can be selected:

- **Do not use a remote connection** (legacy and default mode): this executes the Quasar program locally, within the Redshift process

- **Connect to local server**: this option runs the debugger in a separate process. This has the advantage that the Redshift IDE does not crash or needs to be restarted when the Quasar program gets stuck or crashes (e.g., due to a sticky CUDA error or an unchecked memory access).
- **Connect to remote server**: this option is similar to the previous option, with the difference that the debugger process runs on a remote machine. Communication with the remote machine is done via TCP.

When selecting the last option, a connection dialog box appears in which you can configure "friendly" and trusted servers. For security reasons, this needs to be done manually.

It suffices to enter the host name (or IP address) and port on which the Quasar debugger (Quasar.Spectroscope.exe) is running. See below for options for setting up the Remote debugger.

The following features are currently supported during remote debugging sessions:

- watch variables
- source code tooltips during debugging
- breakpoints
- print, disp, imshow, plot, scatter, barplot functions are redirected to the client (so they work as if the program would run locally)

Currently not supported:

- user interface functions, such as form(), OpenGL renderers.

Local and remote debugging servers can be restarted at any moment of operation. First, a soft reset is attempted (allowing the server process to terminate gracefully). If this does not work, a hard reset is performed (killing the process).

File storage

By default, all files accessed by the remotely debugged process most be available on the remote server. This can be achieved by setting up a shared file system between the local client and the remote server.

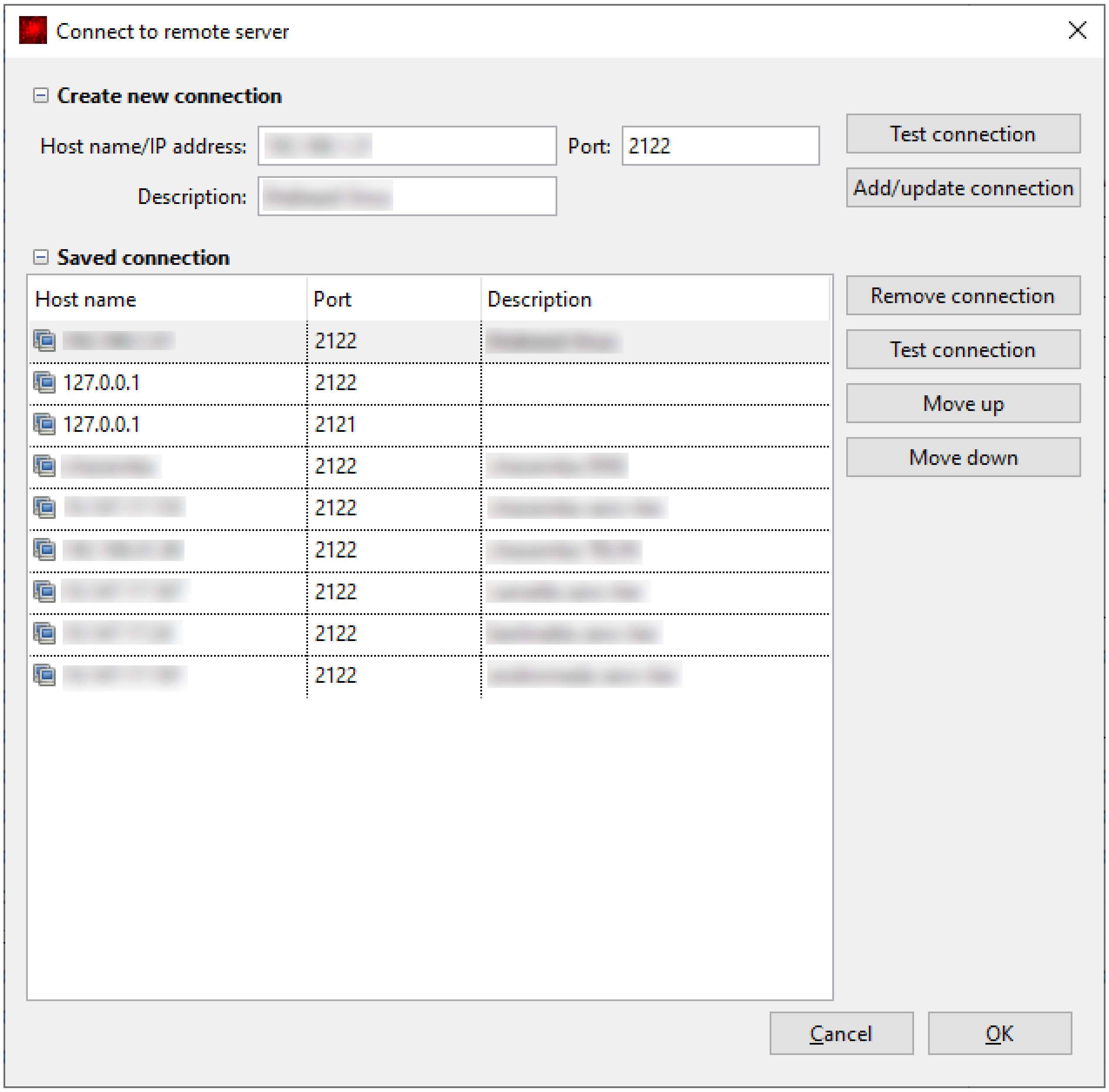
An alternative is to use the option "Redirect library function to local file system" during debugging (see Program Settings/Debugging). In this option is checked, file reading and writing functions (fopen, load, save, imread, imwrite, csvread, csvwrite) will use the local file system. For this purpose, the data is transferred over the network connection.

Remote connectivity options

Remote debugging in a LAN

The simplest way to set up a Quasar remote debugging session is by running the debugging server on a remote host within the local area network (LAN). Redshift then directly connects with a host within the same network.

To set up the remote debugging server, first install Quasar on the server. Then start the debugging server as follows:



Create new connection

Host name/IP address: [redacted] Port: 2122

Description: [redacted]

Saved connection

Host name	Port	Description
[redacted]	2122	[redacted]
127.0.0.1	2122	
127.0.0.1	2121	
[redacted]	2122	[redacted]
[redacted]	2122	[redacted]
[redacted]	2122	[redacted]
[redacted]	2122	[redacted]
[redacted]	2122	[redacted]
[redacted]	2122	[redacted]

Figure 3: Picture of connection dialog box

- Windows: Quasar.Spectroscope.exe −−server −−port=2122
- Linux: mono Quasar.Spectroscope.exe −−server −−port=2122

Important: in case the server has multiple network interface cards (NICs), it is also required to specify which network card to use, via the bindto property:

```
Quasar.Spectroscope.exe --server --port=2122 --bindto=10.149.18.150
```

Remote debugging via SSH tunnels

Remote debugging via SSH tunnels currently not supported. The reason is that the server may dynamically set up connections with the client using auto-assigned ports. This is currently not practical to achieve via SSH tunnels.

Remote debugging in a virtualized network

VPN is currently not supported.

An alternative is Zero-tier (https://www.zerotier.com)

1. Create zero-tier account (choose free account, with up to 100 devices)
2. Click on the networks tab, save the network ID (a hexadecimal string of 16 characters)
3. Select "EASY IPv4 auto-assign" and select a network range (for example 10.147.18.*)
4. Click on the downloads tab, install zero-tier software on the client and on the server (or on any host system you want to use within the virtualized network)
5. In windows, enter the network ID. In linux, run the following command from a terminal: zerotier −cli join $(NETWORK_ID) (fill in your network ID)
6. Go back to the networks tab in the Members pane enable each machine you have connected. You may give each machine a descriptive name.
7. Note down the IP addresses of the machines. You will need it do setup a remote connection

Security considerations

Currently it is not recommended to run the debugging server on a machine with static IP or that is directly exposed to the internet. Always use the debugging server on a trusted network behind NAT/firewall or use the virtualized network.