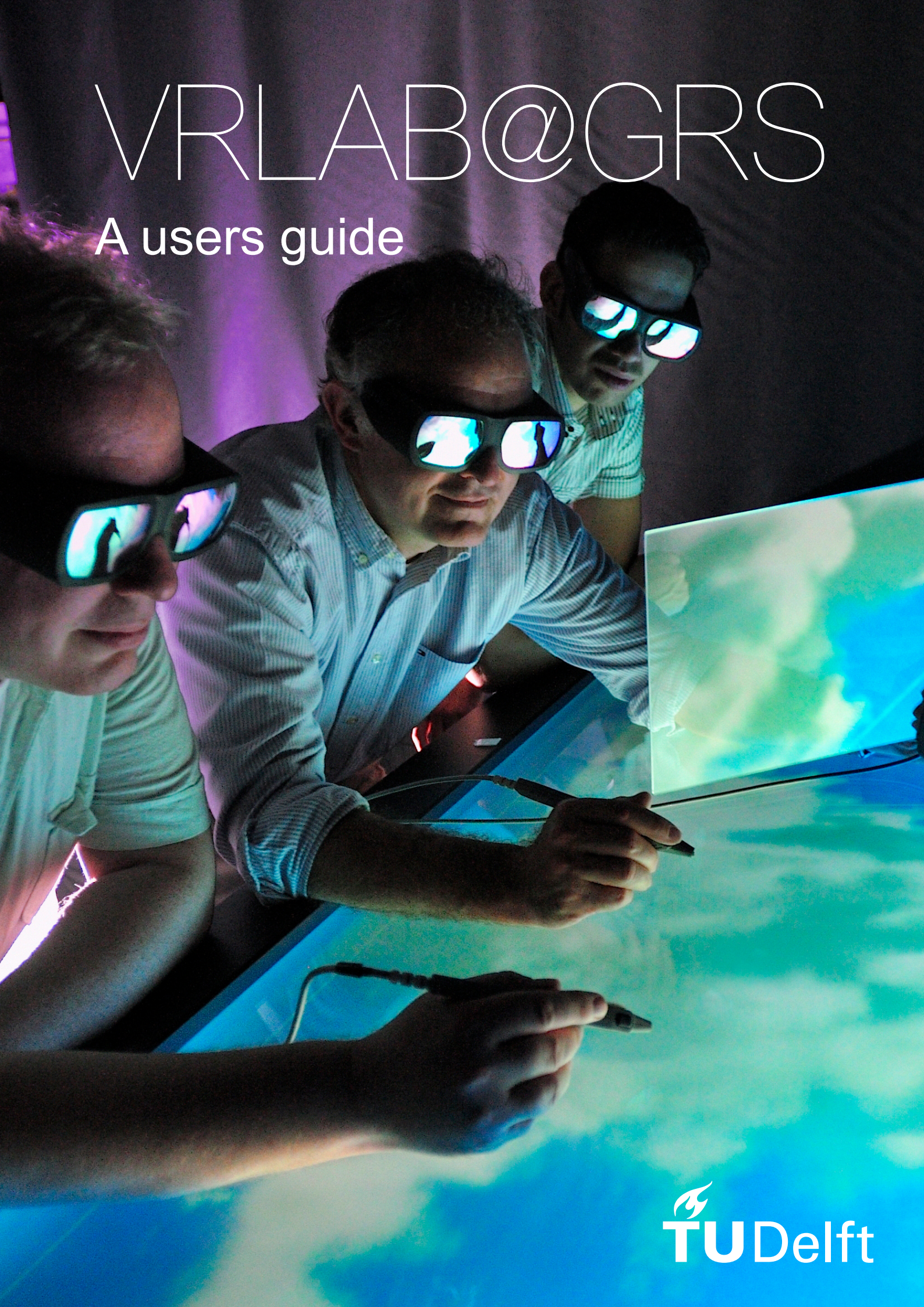
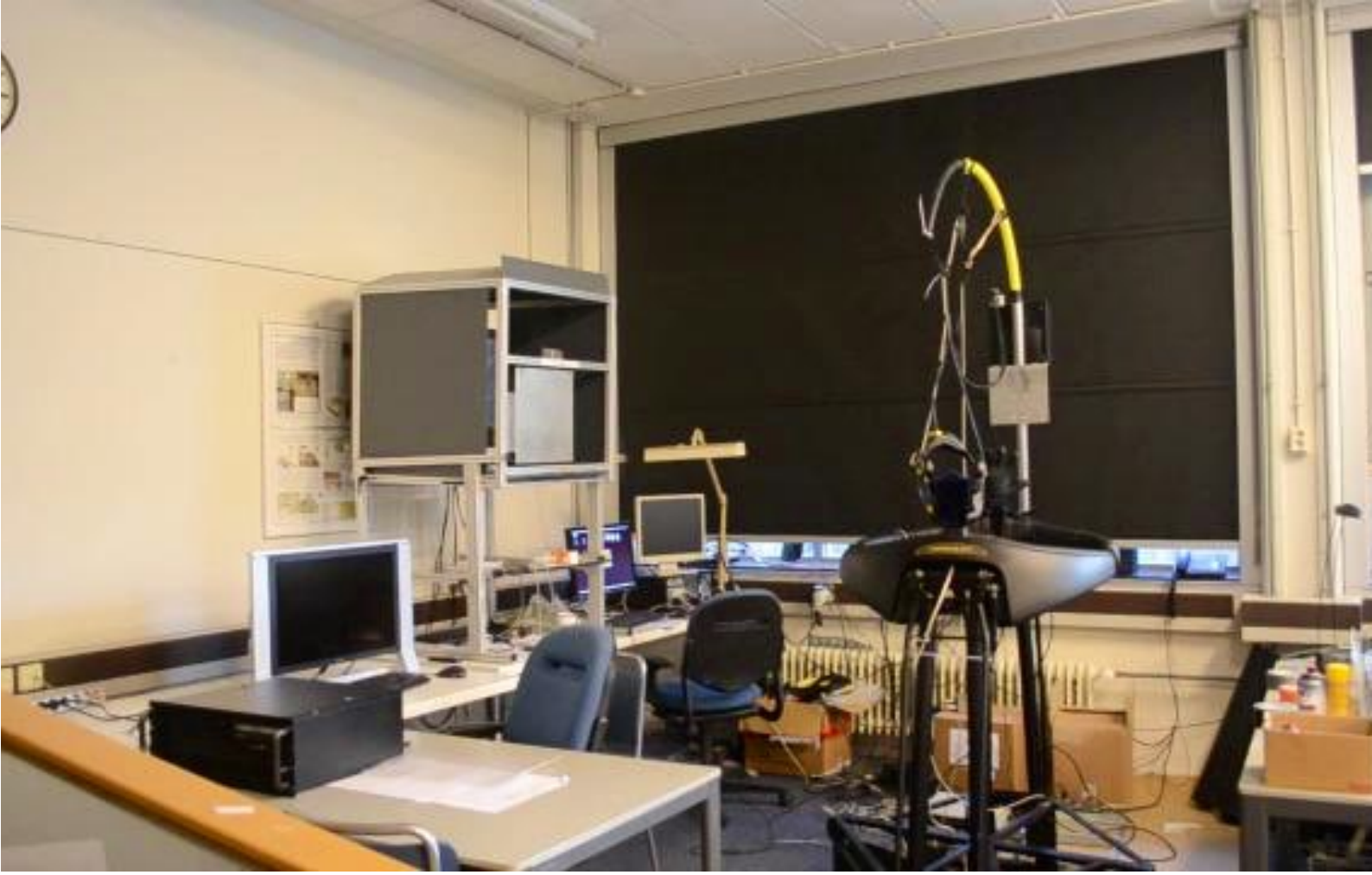
VRLAB@GRS

A users guide



TU Delft



Former VRLAB at TNW

Introduction

The VRLAB, originally an EWI visualisation facility, was refurbished in 2005 with an NCF/NWO grant and the financial support of EWI and TNW faculties of the TU Delft. Since then there has been tight collaboration between the two faculties in the exploitation of the lab.

When the TU Delft decided to centralise it's ICT server a few years later, it soon became apparent that the centralised servers had the disadvantage of having high latency on the network connections, mainly caused by the physical distance between workstations and the servers a few kilometers away in the data center. The visualisation machines in the lab have therefore always been connected to a local high speed network. Over time our desktop workstation have become more and more visualisation and software development workstations too. Firstly as a result of technological progress in GPU's. Secondly because of the increased requirements in performance. To take advantage of the local high speed, low latency VRLAB network, our desktop workstations and servers were soon connected to the VRLAB network, instead of being connected to centralised servers in the data center. This situation is still the same today for users of VRLAB connected machines. Although network technology has progressed enough to get 1 GB/s connections to almost everyone, latency, being determined mostly by the speed at which the signal goes through our optical fibers and network equipment, is still our main performance killer on connections to the data center.

The specialised high performance ICT environment the VRLAB gives us, is one the cornerstones needed for outstanding research. Due to the TU Delft policy that centralisation and standardisation are not a goal in itself but a tool, there is plenty of room for special infrastructures like HPC clusters and facilities like the VRLAB, keeping the door open for scientific research that otherwise would be impossible. The credo is ‘Standardisation where possible, specialisation where needed’ is paying of here. The current VRLAB implementation is a textbook example where standard solutions are used to implement a specialised solution for our needs. Based on standard protocols such as NFS, Kerberos, ethernet and Infiniband we have created a high performance environment that outperforms centralised solutions on nearly all aspects.

Who is it for?

The VRLAB network is targeted at users of High Performance Computing facilities. Where scientist traditionally ran their jobs and visualisations on centralised computing facilities, such as the supercomputer at SurfSara in Amsterdam, using their workstation merely as a terminal, computing and visualisation power are shifting towards the desktop.

Although supercomputers will always be needed to do the largest of computations and analyses, it is now possible to do visualisation, data analysis and computations on local machines, given a suitable network and server infrastructure around them.

On the vrlab computations can be done on local many-core- and/or GPU machines. This area of computational power, positioned between single desktop machines and supercomputers, is the part of the computational spectrum the VRLAB is at its best and performs better than centralised computational cluster, the latter being being sub-optimal for interactive tasks as software development and visualisation. Typical applications used on the VRLAB are LES simulations, processing of big data such as supercomputer results or experiment measurements (e.g. processing of radar data) and visualisation. Due to the low network latency and high speed connections between machines (up to 40Gb/s) and the possibility to use substantial local GPU and CPU power in the network, the VRLAB is also an excellent environment to do software development and testing.

For many (student) projects the VRLAB is powerful enough to run simulations and create visualisations that are not possible on a single desktop machine and would otherwise require access to a supercomputer or cluster.

One the the first applications in the VRLAB were 3D data visualisations on 3D data sets. In those days, custom written applications utilising OpenGL were required to create 3D views of dataset. Today, many general purpose applications are available to do this A few examples can be viewed on our Youtube channel (search for ‘AtmosPhysDelft’). These modern tools require none or very little programming by users. In the physical lab at EWI there is a large stereo screen (similar to 3D screens you know from the cinema) that is available to all to look at data in stereo on a large screen. In many cases, depending

Access and Apps

The lab network is suitable for Unix and Unix-based systems like Linux and MacOS clients. For mainly technical reasons (and some policies) we have our own local authentication and authorisation servers in the VRLAB network that is not connected to the central TU Delft system. Users need a local account to access resources on the VRLAB. Authentication is done against a

Share	Mountpoint	name of server	Usage
home	/home/<logonname>	nfs1	Your private home folder.
labdata	/net/labdata/<logonname>	nfs4	Bulk storage for data
scratch	/net/scratch/<logonname>	nfs4	Temporary storage
shared-public	/net/shared-public/…	nfs4	Shared data.
shared-staff	/net/shared-staff/..,	nfs4	Shared data with limited group access.

Kerberos server with authorization done from an LDAP system utilising industry standard, opensource backends. The entire local authentication and authorisation system is redundant for fault tolerance with servers at EWI and CiTG.

The lab is standardised on one Linux distribution (Currently OpenSuSE 42.1 Leap) for reasons of maintainability and uniformity. As a result, users can use any (linux) workstation connected to the VRLAB and have all their files and software available while reducing the administrative effort needed to maintain the VRLAB at the same time. Support for MacOS clients is a bit limited. Although Macs can be used to access resources on the VRLAB, things like automatic software installation and automatic mounting of network storage are not available. Support for Windows systems is non-existent, other than providing ssh access to Linux systems connected to the VRLAB and file up- and download services via ssh-based protocols like scp, sftp, rsync. via our bastion host.

Linux machines

Linux machines are maintained and ready to use. A complete software stack is available on all machines. Basically everything that is offered in the linux distribution is installed all machines, augmented with tools like Matlab, Intel Fortran, Paraview and many other tools. The philosophy is that users do not need to install software themselves, as that would make each machine an unique incarnation of Linux tools. Instead software installs are done centrally, so that all machines have the same tools available in the same version. Obviously this is also much more efficient than installing tools on each workstation individually. The software packages that are not available in the linux distribution, are available by loading them via the ‘module’ command. (See side bar). This way we can have multiple versions of many packages available, without conflicts. The ‘module load’ command basically manipulates the

GNU Module

MODULE AVAIL
LISTS AVAILABLE MODULES

MODULE LIST
LISTS LOADED MODULES

MODULE SEARCH <ARG>
SEARCH FOR A MODULE
MODULE LOAD <NAME>
LOAD THE NAMED MODULE

MODULE UNLOAD <NAME>
UNLOAD THE NAMED MODULE

users environment by adding paths to environment variables such as ‘PATH’, ‘LD_LIBRARY_PATH” etc. It is possible that a module depends on other modules and load these dependencies automatically.

MacOS machines

On the MacOS machine, things are slightly different. At the moment there is no support for the module command, neither is there a network repository where application and tools are available. This means that all applications on a Mac (such as Matlab, Paraview) need to be installed by the user manually. Most applications are available for download (some only after authentication as a TU Delft employee) from the Internet. The Intel Fortran compile is not licensed on MacOS and not available.

Remote logon

Remote logon is possible from every internet connected machine via ssh. The external address of our bastion host is “bastion-grs.vrlab.tudelft.nl" On this machine, all disks are mounted, to allow access to all your files. However, there is no software installed on the bastion machine. In order to run applications, you need to logon from the bastion, using ssh, to your desktop machine or one of the public machines available.

Storage facilities

Our policy is that all data is stored on our servers and not on local disks. There are many reasons to enforce this, but the most important one is that the single disks in workstation and laptops will fail at some point and that would cause loss of data and possibly loss of careers if the data was important.

Available storage

We have several servers in the VRLAB, providing a total of 70TB of storage (sept. 2016). For performance reasons, each server has a dedicated function. Currently we have 2 storage servers providing storage The machines provide the following shares:

Some of the services can temporarily run on another servers due to maintenance work. On Linux the mounts are automatic and end users won't notice this. Mac users however, need to temporarily connect to the replacement machine manually.

home

Home folders are on a separate server with high protection. The reason behind that, keys and other sensitive files are stored in your home folder. All traffic to and from the server is encrypted using strong encryption. Use this folder for anything that is important, such as private files, source code and texts for publications. Home is not suitable for storing data. Home is not guaranteed to be available in jobs run on the grid.

labdata

Labdata is our big storage server, which can be expanded tot Petabyte sizes if the need arises. This is the place to store your data from any source. This folder can be made accessible to your supervisor if you are a student.

scratch

Scratch is for temporary data. Use for anything that you don't need to keep. Files that are older than a couple of weeks are automatically deleted from this share. A typical use case is a simulation run that produces lots of files, of which you only need to keep a few. Once the simulation is finished, you copy the files you need from scratch to 'labdata'. The left-overs are automatically removed after a while. The retention period is roughly 60 days.

shared-public

Shared-public is for datasets that need to be shared with multiple people (labdata is private). Typical use is measurement data or simulation results. A toplevel directory, can be requested. One user will get read/write access to that folder, while the rest of our

ACCESS FROM A MAC

On the Mac, the shares are not automatically mounted. To mount a share manually, press CMD+k in the Finder. For the url, enter the full name of the server. For example, something like

afp://nfs4.grs.vrlab.tudelft.nl

After entering your credentials, a Finder window will open, showing the shares that are available to you. On the Mac, shares are mounted below '/Volumes/…'.

users get read-only access. Shared-public is not suitable for data that needs to be shielded from some people.

shared-staff

Shared-staff is similar to shared-public, except for the access rules. A folder can be created, to which a given group of users will have full read/write access, while any other user won’t have access at all. This share is suitable for shared projects that need to be kept secret, such as exams, confidential papers etc.

Using the Grid

Introduction

The VRLAB is in essence a grid network, where jobs can run in many places. At the moment we have a few dedicated job processors, but the goal is to configure the network in such a way, that idle workstation can also run jobs, for example at night or in the weekend. The grid consists solely of Linux machines. Mac users need to logon to a linux machine, in order to submit jobs and use the grid.

Currently we have around 100 CPU threads and 8 GPU's available. When users submit jobs to the grid, they do not specify on which machine the job should go. The task of selection the proper machine is dedicated to the grid scheduler. The scheduler makes it's selection based on the resources requested by the user. A job specification needs to include:

- Number of CPU threads needed
- Number of GPU's needed
- The amount of RAM memory needed
- The amount of time needed to run the job

It is important to set reasonable values for these resources, otherwise it get take a long time before a job is scheduled to run and it may keep resources occupied that could be made available for other jobs. For example, if you set the time needed to 24 hours and the jobs takes 2 hours, it will never get scheduled in a slot of 12 hours available on a cpu on a workstation.

Submitting a job

Jobs should not be started in your home folder, but should run from /net/labdata or /net/scratch. A good strategy can be to run the job from scratch and copy the data the you want to keep to /net/labdata. As scratch gets cleaned automatically, this will make sure the leftovers of a job are automatically removed. You can, of course, run directly from labdata and do any cleanup yourself. Jobs should never be run from your home folder, as home is too slow for this and may be unavailable on the job processor due to expired kerberos tickets.

To run a job, you will need to small (bash) shell script that runs the job. In the example below, I'll use '/net/labdata/erwin', but of course you will need to replace that with your own labdata folder. The first small example uses the 'bc' command to compute 4096^{4096}. The script could look like this:

```
#!/bin/bash
#$ -l h_vmem=1G
#$ -l h_rt=120

echo 4096^4096|bc
```

Save this in a file called "RUN" and submit it with

```
qsub RUN
```

The default value for the number of threads is 1. The default number of GPU's is zero, so for this single threaded job that can't use multiple cpu's or gpu's, we can leave them unspecified. With the 'qstat' command you can see the status of your job. The 'qdel' command can delete jobs, if needed.

```
erwin@goofy:/net/labdata/erwin/TEST> qstat
job-ID  prior   name       user         state submit/start at     queue                          slots ja-task-ID
-----------------------------------------------------------------------------------------------------------------
  23321 0.55617 job.015    stephan      r     01/17/2017 10:50:19 cirrus-b@cirrus-b.grs.vrlab.tu     8
  23393 0.55617 job.014    stephan      r     01/18/2017 10:54:49 cirrus-a@cirrus-a.grs.vrlab.tu     8
  23625 0.45617 RUN        erwin        qw    01/25/2017 09:58:57                                    1
  23394 0.00000 job.016    stephan      hqw   01/18/2017 10:56:42                                    8
  23427 0.00000 Job520     patrick      hqw   01/18/2017 18:00:58                                    1
  23428 0.00000 Job320     patrick      hqw   01/18/2017 18:02:57                                    1
  23430 0.00000 Job220     patrick      hqw   01/18/2017 18:03:54                                    1
```

The job will, by default, output a job output file (RUN.oxxxxx) and and error file (RUN.exxxxx) where xxxxx is the job id that was assigned by the scheduler. You can override the names of these file if you want.