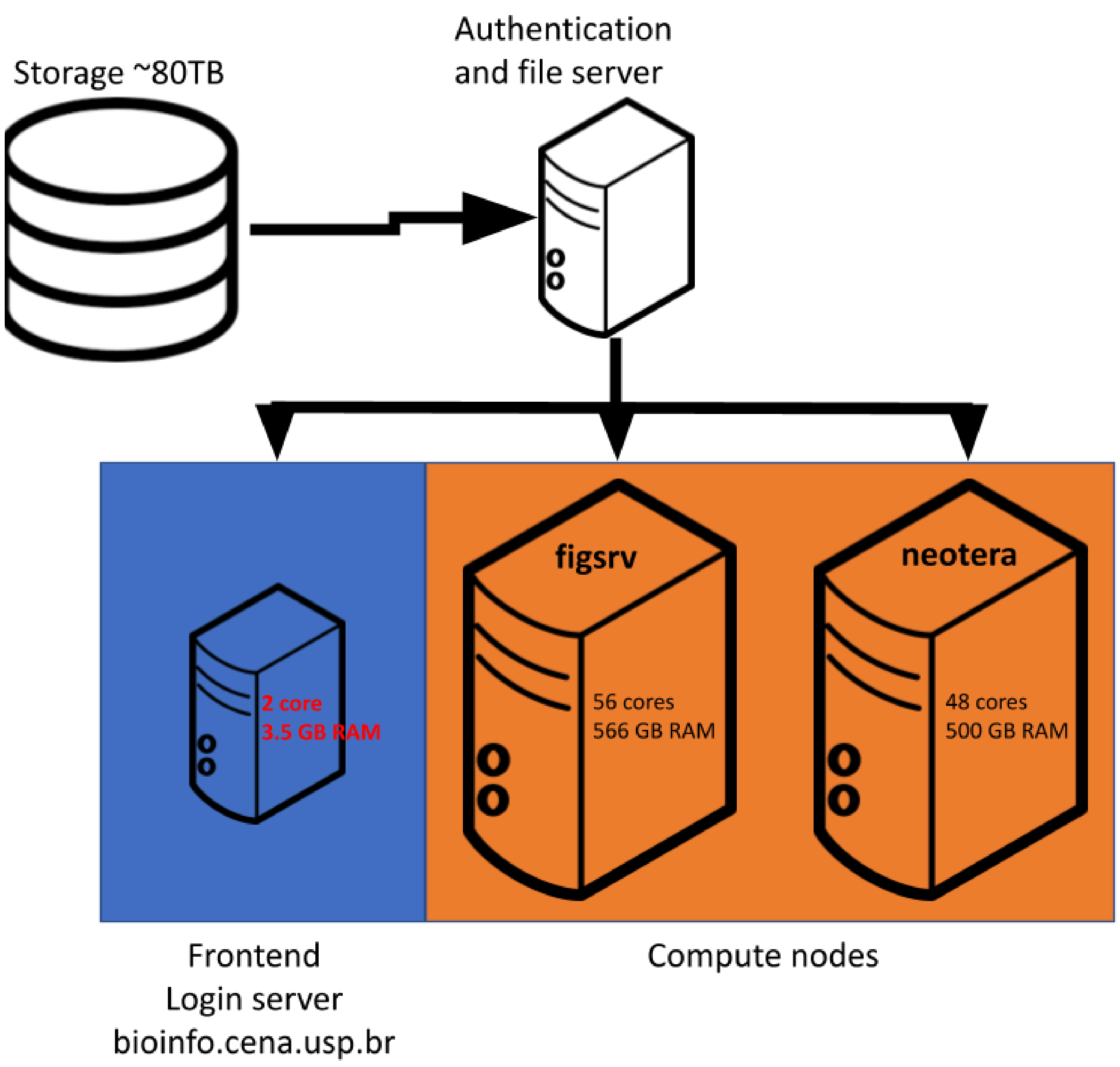
Bioinformatics Cluster at CENA/USP

Version history

Version	Date	Author	Rationale
0.5	22/12/2018	DMRP	First draft, with policy and usage guide
0.9	27/12/2018	DMRP	First complete version. Needs approval from compute node owners and STI
0.91	23/08/2023	DMRP	Updating infrastructure, new server added, old servers decommissioned
0.92	12/12/2023	DMRP	Adding instructions of how to request a specific compute node

Infrastructure



Policy usage

1. The policy usage of the Bioinformatics cluster at CENA/USP is based on trust and fairness, we want to maximise the use of the infrastructure, and we trust on their rational use by the users.
2. The cluster runs on Linux operating systems (CentOS 7), so only software that runs on these could be used.
3. The cluster is targeted for large scale computation, so interactive usage is discouraged, as well as the use of Software with graphical user interfaces. Only software packages that have a command line interface should be used.
4. The cluster is available to faculty at CENA/USP and their students.
5. Requests for use should be evaluated by the cluster administrator.
6. All laboratories that gain access to the cluster must have a single point of contact between the STI and the Laboratory. This is one, and only one person, that will request the creation of users and software installation to the STI.
7. Requests for software installation must be for packages that will be routinely used. If a user needs a package for a one time use, then they are advised to install it temporarily in their HOME directory and remove it afterwards.
8. The jobs submitted to the cluster are managed by SGE (Son of Grid Engine), which implements a fair use policy. Any new user must be trained on how to submit jobs via SGE before having their login to the cluster authorized. Part of that training is reading the guide in the following pages.
9. The cluster is intended for large scale computation, not for medium or long term data storage. So, after computations are finished, input and output files should be removed to release disk space for other users.
10. If you are a PI at CENA and you (or if the PI of your group) are thinking on acquiring computational infrastructure for your research, come to talk to us, and try to join forces to improve/grow our consolidated and shared infrastructure for scientific computation.

Cluster usage

IMPORTANT
Never run any computation command on the frontend node

General Info

The frontend node (bioinfo.cena.usp.br) is a very small machine (2 core, 3.5GB RAM) that is only intended to allow login over the internet and the submission of jobs to the compute nodes via SGE (Son of Grid Engine).
Before using the cluster you MUST become familiar and feel comfortable using Linux operative systems, and using the bash shell. Follow these two guides/courses if needed: http://korflab.ucdavis.edu/unix_and_perl/ and http://swcarpentry.github.io/shell-novice/.
You can log in to the frontend using any SSH client using the port 1000. You will be asked for your password. If this is your first login please change your password with the command **`passwd`**. Linux and MacOS systems have a built in SSH client on the command line (ssh), for MS Windows you could use putty (https://www.putty.org/) or MobaXterm (https://mobaxterm.mobatek.net/).
The administration of user jobs is based on queues, The user send jobs to Son of Grid Engine (SGE in the following), SGE put the jobs in a queue and if there are computational resources available SGE will run the job from the queue.

At this moment we have a single queue: **`all.q`**, and both compute nodes had been assigned to that queue.

Cluster Stats

The SGE command **`qstat`** (all SGE command start with the letter '**`q`**', that comes from queue), will inform you about what is currently running or scheduled to run on the cluster. If you run **`qstat`** on your shell prompt, it will show you what is scheduled to run, or is running, on YOUR behalf, with your username.

Oftentimes it is more interesting to get a global picture of what is running on the cluster, or scheduled to run, by any user. For that you can use different arguments passed to **`qstat`**. For the general case I recommend to use the arguments:

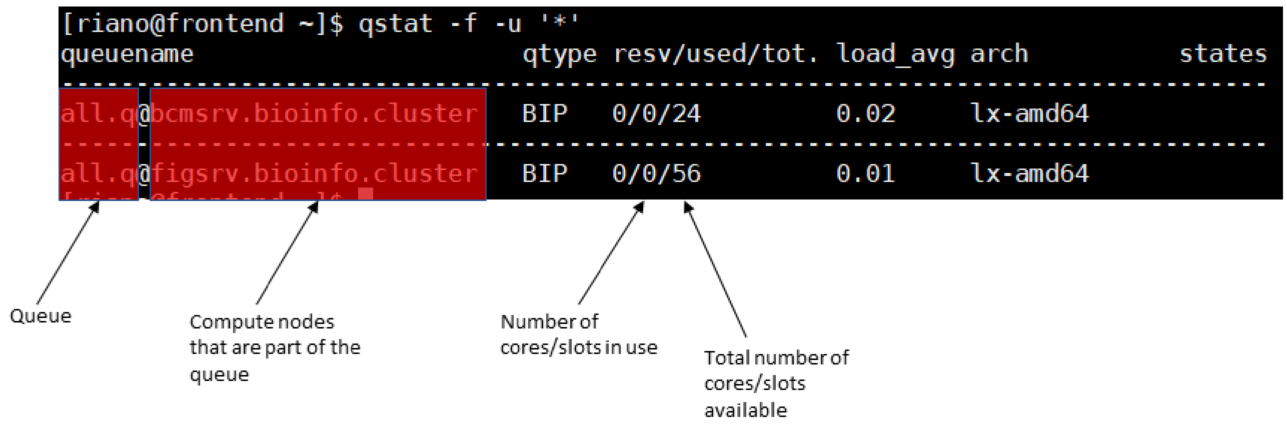
-f: Full format display

-u "*": All users

So you can execute this command to get a full picture of how the cluster is being used:

```
qstat -f -u '*'
```

You will get something to what is show in the following image:



```
[riano@frontend ~]$ qstat -f -u '*'
queuename                          qtype resv/used/tot. load_avg arch          states
---------------------------------------------------------------------------------
all.q@bcmsrv.bioinfo.cluster       BIP   0/0/24         0.02     lx-amd64
---------------------------------------------------------------------------------
all.q@figsrv.bioinfo.cluster       BIP   0/0/56         0.01     lx-amd64
```

Queue
Compute nodes that are part of the queue
Number of cores/slots in use
Total number of cores/slots available

Knowing what is currently running on the cluster and what resources are available at any time will help you to plan your scripts, so go ahead and use the **`qstat`** command, explore what other options you could use by reading its manual page with:

```
man qstat
```

Now we will see how to submit a job to the queue. As you noticed in the figure above, we have a single queue that is called **all.q,** and both compute nodes (**figsrv** and **neotera**) are assigned to that queue, which means that jobs submitted to the queue **all.q**, will be run on any of these nodes. Later we will show how to run your scripts on a specified node.

Bash scripts for SGE

Let's start with a simple script, that prints the name of the host, the date in the system, sleeps for 20 seconds and prints again the date of the system. That will be accomplished with the linux commands: `hostname`, `date` and `sleep`. You must remember that the script must have all the commands that you would run on the command line.

The scripts that you will be doing are bash scripts. Bash is one of the most widespreads shells in linux and a programming language on itself, and you can exploit that in your own scripts (http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html).

Before starting describing a typical script with its parts, please remember:

Never run any computation command on the frontend node

All your scripts will be divided into three sections.
The first section, is the single first line.The first line in the script is mandatory and it is always the same, it is called the shebang, and it tell the system that what follows is written in bash. The line is:

```
#!/bin/bash
```

Please note that there are no spaces in that line. You MUST have that line as the first line in all of your scripts.

The second section of the script, is characterized because it is a set of lines that all start with the two symbols

```
#$
```

These two symbols (hashtag and dollar sign) are recognized specifically by SGE, and will instruct it to assign computational resources for you.
There are some line that start with #$ that are mandatory. You must inform SGE which queue you are gonna use. As currently (this may change in the future) we have a single queue: all.q, you must tell SGE the following to use that queue:

```
#$ -q all.q
```

It is also highly advised to pass all your environmental variables to your job and make sure your output files are stored in the same folder where your script is located, for that you will use two lines:

```
#$ -V
#$ -cwd
```

That is the minimum that you script should have, with such instructions you can go for the third section of the script.

The third section of the script, is the part where you would the command that you want SGE to run on your behalf and that will led you to interesting biological discovery and important publications. In our toy example we will just use command to print the name of the host name, the date of the system, make the script sleep for 20 seconds and print the date of the system again. The linux commands for that are:

```
echo `hostname`
date
sleep 20
date
```

All together the script should look like:

```
#!/bin/bash

#$ -q all.q
#$ -V
#$ -cwd

echo `hostname`
date
sleep 20
date
```

We recommend that you create such scripts directly on the frontend using the text editor **`nano`** (https://www.howtogeek.com/howto/42980/the-beginners-guide-to-nano-the-linux-command-line-text-editor/). Another option is to create the script in notepad in your windows-based computer and then copy the text, and then paste it in a **`nano`** screen from the frontend.

We will create the script **test2.sh** with the contents above. The extension .sh, indicates that this is a bash script, it is not required but it is strongly advised to use it, so that you can easily identify your scripts in your folders.

Job/Script submission and deletion

Now we will see how to make SGE run the script on your behalf on the compute nodes. Remember, you are located in the frontend, but your script (job) should run on any of the compute nodes (figsrv or neotera), which are the ones that have real power for your computations. This step is called submitting your job to the queue, and is is achieved with the command

```
qsub
```

You can try to use **`man qsub`** to get more information about how it works.7

In order to submit the script we created before (test2.sh) we need to run:

```
qsub test2.sh
```

As you submit your job you will receive a message like the following:

Your job 153 ("test2.sh") has been submitted

The number 153 is an identifier specific for your job. You can use it to alter or stop your job.

Let's try again the command **`qstat`** from before:

```
qstat -f -u "*"

queuename                      qtype resv/used/tot. load_avg arch          states
---------------------------------------------------------------------------------
all.q@neotera.bioinfo.cluster  BIP   0/0/24         0.01     lx-amd64
---------------------------------------------------------------------------------
all.q@figsrv.bioinfo.cluster   BIP   0/0/56         0.01     lx-amd64

############################################################################
 - PENDING JOBS - PENDING JOBS - PENDING JOBS - PENDING JOBS - PENDING JOBS
############################################################################
    153 0.00000 test2.sh   riano        qw    12/26/2018 21:29:40     1
```

If you are fast enough, or the system is quite busy, you will see that as soon as you submit your job it will appear in a list of PENDING JOBS, with the job id (153 in this case), the name of the job (the name of the script file), the name of the user who submitted the job, the characters **qw**, the date and time of submission and the number of computing cores the job requested.

The characters **qw**, represent the status of the job. **q**, mean queued, and **w**, waiting. Other states are **t**, transferring and **r**, running. After some time, and if enough computational resources are available, your job will run. Let's try running **`qstat`** again:

```
qstat -f -u "*"
queuename                      qtype resv/used/tot. load_avg arch          states
---------------------------------------------------------------------------------
all.q@neotera.bioinfo.cluster  BIP   0/0/24         0.01     lx-amd64
---------------------------------------------------------------------------------
all.q@figsrv.bioinfo.cluster   BIP   0/1/56         0.01     lx-amd64
    153 0.55500 test2.sh   riano        r     12/26/2018 21:29:43     1
```

Now you can see that your job has moved from the list of PENDING JOBS and it is now running on the compute node: figsrv, with status **r**, running. The date and time also changed, and they now show the time when the job started running.

In the case you ever need to stop one of your submitted jobs, there is the command qdel. In order to use you must first get the job identifier of the job you want to remove from the queue. In our toy example, that number is 153, so to stop it you would run:

```
qdel 153
```

You will only be allowed to stop jobs that belong to you.

IMPORTANT
Never run any computation command on the frontend node

Running jobs with multiple cores

Some of the software that you will use would perhaps accept multiple cores, which in most cases reduce the time for computation. Check the manual of the software you want to use and look for arguments such as **--threads**, **--cpu**, **--nproc** or the like. So far we had only seen scripts that use 1 core at the most, we will now see how to use multiple cores.
Keep in mind that we have two servers, one with 24 cores and another with 56 cores. So the maximum number of cores that you can request and use is 56. Also you are strongly advised to use up to 60% of the cluster resources (cores) at any one time.
Pay attention that there are two different layers here, one is how you want your scientific software to run, and the other what you tell SGE that you will be using. They are separated, but you must make any effort to make it consistent. So, if you want to run your software with 10 cores, you must inform SGE that you will be using 10 cores. **Make sure that whatever you tell SGE that you will be using, is actually what you use, not less, not more.**
SGE will only reserve computational resources for your to use. If you reserve for instance 10 cores, put only actually use 2 in your script, 8 cores will still be reserved and no one else could use them, so it is a waste of resources. On the other hand, if you tell SGE to reserve 10 cores for you and you actually use 20, you are tricking SGE and putting the cluster at risk, as SGE will just trust you, and will “think” you are using only the requested 10 cores. So, again, **make sure that whatever you tell SGE that you will be using, is actually what you use, not less, not more.** If we notice any inconsistency between what you are reserving and what you are actually using we will contact you to fix this, which could lead to stop your job and fix the script to submit again.
In order to tell SGE that you will use multiple cores, you should add a new line in your script, something like:

```
#$ -pe smp 10
```

Such a line will request the reservation of 10 cores for your job. When you use that line, a variable is available for you to us in your script, $NSLOTS, which holds the number of cores you have requested. We strongly advise you to use such variable in the body of your script. I will show an example. Let's create a script to run the compression of a read file using multiple cores/threads:

```
#!/bin/bash

#$ -q all.q
#$ -cwd
#$ -V
#$ -pe smp 10

echo $NSLOTS
```

pigz is a software package that compress target files using multiple cores. The argument -p of pigz, tell the number of cores/threads that will be used. In this case we have used the variable $NSLOTS, that is holding the number 10, which was requested in the line -pe smp 10.

Running jobs in a specific hostname

If you find yourself needing to run a job on a specific compute node (figsrv or neotera) you can ask SGE to reserve it for you. Use an instruction as the following:

```
#$ -l hostname=NNNN
```

Where NNNN is the name of the compute node you want to submit your job to, for instance **figsrv** or **neotera**.

Further information about SGE

SGE is a very elaborate and capable system to administer shared computational resources. You can find further information in the following links, we particularly encourage you to read the documentation about Job Arrays (last link below):

http://bioinformatics.mdc-berlin.de/intro2UnixandSGE/sun_grid_engine_for_beginners/README.html

https://www.uibk.ac.at/zid/systeme/hpc-systeme/common/tutorials/sge-howto.html

http://docs.hpc.shef.ac.uk/en/latest/parallel/JobArray.html

Note about resource sequestration

Avoid sequestering all computational resources for long periods of time (more than a few days), remember that this is a shared resource and we want to give a chance to run jobs to a large number of users. If you sequester a large fraction of computational resources, your job might be killed by the cluster administrator. **For large jobs, use, at the most 60%** of the computational resources. If in doubt contact diego.riano@cena.usp.br

What Software is installed in the cluster?

We are using environment modules (http://modules.sourceforge.net/) in the cluster to install and manage the diverse scientific software packages that you have requested.
If you want to know what is installed in the cluster, just run:

```
module avail
```

That command will show you all scientific software packages available. If you need a software package that is not installed, please make the single-point-of-contact from you lab contact the STI to request the installation.
If the package you need to use is installed, then you must load it. For instance, let's say you want to run **`fastqc`**, a package to evaluate the quality of a sequencing run. If you type:

```
fastqc --help
```

you will get, a command not found message. You first need to load the module. After checking the output of module avail, you will notice that there is a module called FastQC/0.11.8, we need to load it with:

```
module load FastQC/0.11.8
```

Now if you run:

```
fastqc --help
```

Yo will get the help page for FastQC.
To unload a single module you can run:

```
module unload FastQC/0.11.8
```

Or to unload all modules at once, then run:

```
module purge
```

Do not forget to load the modules you need in your SGE scripts

Where to run your stuff?

There are five partitions, highlighted in red below, that are available in all computers that make part of the cluster. Have a look with the command **`df -h`**

```
df -h
Sist. Arq.                         Tam. Usado Disp. Uso% Montado em
/dev/mapper/centos_frontend-root    27G  3,9G   24G  15% /
devtmpfs                           904M     0  904M   0% /dev
tmpfs                              916M     0  916M   0% /dev/shm
tmpfs                              916M  8,9M  907M   1% /run
tmpfs                              916M     0  916M   0% /sys/fs/cgroup
/dev/sda1                         1014M  204M  811M  21% /boot
192.168.6.7:/nfsshare/home         917G  2,6G  868G   1% /home
192.168.6.7:/nfsshare/progs        1,4T   18G  1,3T   2% /Storage/progs
192.168.6.7:/nfsshare/databases    2,8T  332G  2,5T  12% /Storage/databases
192.168.6.7:/nfsshare/data2         15T   18M   14T   1% /Storage/data2
192.168.6.7:/nfsshare/data1         46T  951G   45T   3% /Storage/data1
tmpfs                              184M     0  184M   0% /run/user/0
tmpfs                              184M     0  184M   0% /run/user/1002
```

Software is installed, by the system administrators, under `/Storage/progs`. Databases of common use are under `/Storage/databases`. User accounts are available under `/home`. As you can notice `/home` is a small partition of a little bit less than 1TB, please DO NOT store input files or result files under this partition, do not RUN any computation under this partition. Just store configuration files, testing software or keep copies of your SGE scripts in there.

As you can see in the output from **`df`**, the two largest partitions are **`/Storage/data1`** and **`/Storage/data2`**, with 46TB and 15TB respectively. **You will find a folder with your username and that you own under these two paths. You MUST put all your data in any of these two paths and run all your scripts from there.**

None of the data in the cluster has backups, so it is your sole responsibility to keep safe copies of your data and results.

Databases available

Databases of common use are under `/Storage/databases`. Currently we have:

1. NCBI's nt (nucleotide)
2. NCBI's nr (peptide)
3. NCBI's taxonomy

4. UniProt reference proteomes
5. UniProt SwissProt
6. UniProt TrEMBL
7. UniProt UniRef 100
8. UniProt UniRef 90
9. BUSCO sets of conserved genes in different taxonomy groups.
10. EggNOG (euk bact arch viruses virNOG poaNOG fuNOG)
11. Pfam