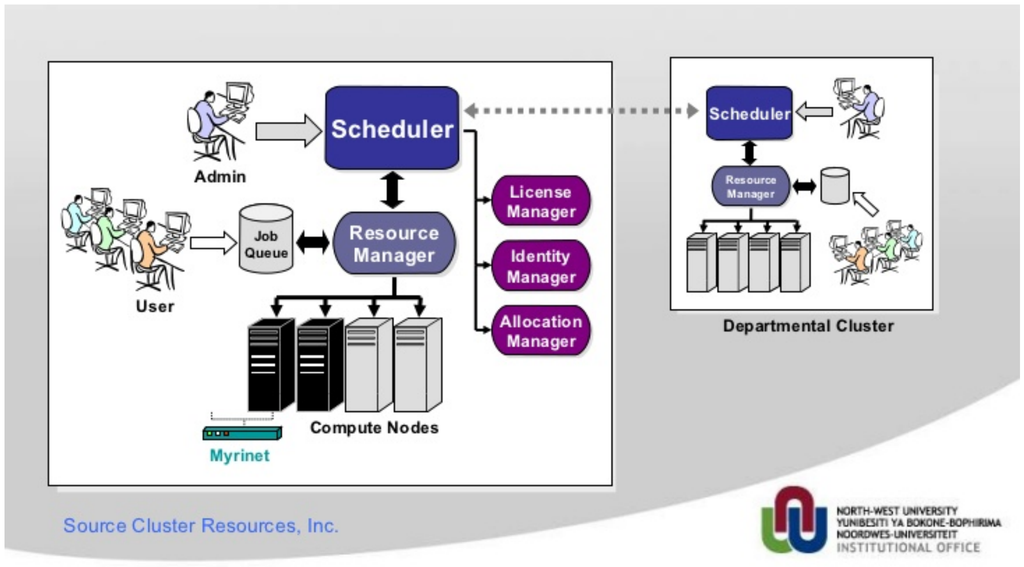
HIGH PERFORMANCE COMPUTING

Introduction and Goals

High Performance Computing facilities are managed by Job Schedulers and Resource Managers. As depicted in the following figure, they are responsible to manage hardware and software resources exposed by the HPC data center and process requests from users to execute applications on top of those resources.



The goal of this assignment is to understand what the schedulers and resource managers are, what they do and how to use them. In the context of this course the scheduler used will be SGE. However, other schedulers and resource managers are widely used in other facilities such as: LSF, Slurm, MAUI etc.

Preliminary Quiz

In this semester we will introduce concepts related to the use of high performance computing and related technologies for scientific use cases. However, before starting with hand-on aspects, we have some preliminary questions that will allow us to explore some topics that we will discuss during the semester that are looking-forward as opposed to traditional use of high performance computing systems. This initial section complements the core of the assignment and will be considered for the grading.

Q1. *Before searching for information (please provide a frank response):* Do you know what is cyberinfrastructure and what does it mean?

Q2. *After exploring online:* Please describe the concept of cyberinfrastructure and provide one example.

Q3. How do you think you can interface with a data-centric cyberinfrastructure?

Q4. How do you think cyberinfrastructure can be useful for science and engineering? Please provide an example.

Assignment Description

The assignments will be performed in a cluster based on GNU / Linux. You can find in the virtual campus a manual for the basic use of the queuing system, please use this as a first reference. You can also find more information on the web, for example on the following link you can find more details queuing system SGE (Sun Grid Engine) that is installed on the UOC's cluster:

http://gridscheduler.sourceforge.net/htmlman/htmlman1/qsub.html

It is important to request a user account by sending an email to acasys@uoc.edu as soon as possible.

You will connect to the cluster through a secure connection using ssh. If you work in GNU / Linux or Mac OS X environments can use "ssh" directly from the command line, however, if you use Windows you will need a tool, e.g., Putty. You can download Putty (for example) in the following URL:

http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

You can also find online tutorials, e.g., in youtube:

https://www.youtube.com/watch?v=9CZphjhQxIQ

The access to the system through ssh is:

```
Ssh -p55000 user_name@eimtarqso.uoc.edu
```
The username will be provided upon request of your account.

If you need to transfer files to the UOC's cluster there exist some tools that can be helpful such as scp / sftp clients, e.g., winscp:

http://winscp.net/eng/docs/lang:es

If you have any issues or problems with your user account you should report them to the email address provided above.

Description of the system:

Eimtarqso is the head (or login) node in a cluster of 10 compute nodes. Eimtarqso is the only node that you should connect to directly and from where will be able to run jobs to/across other nodes via the SGE queue system. It is very important that you respect this policy because you will share resources with other colleagues (e.g., do not ssh compute nodes).

Details of the system can be shown below using the "qstat –f" SGE command (the result is also shown below):

```
[ivan@eimtarqso ~]$ qstat -f
queuename                      qtype resv/used/tot. load_avg arch          states
---------------------------------------------------------------------------------
all.q@compute-0-0.local        BIP   0/0/4          0.00     linux-x64
---------------------------------------------------------------------------------
all.q@compute-0-1.local        BIP   0/0/4          0.00     linux-x64
---------------------------------------------------------------------------------
all.q@compute-0-2.local        BIP   0/0/4          0.00     linux-x64
---------------------------------------------------------------------------------
all.q@compute-0-3.local        BIP   0/0/4          0.00     linux-x64
---------------------------------------------------------------------------------
all.q@compute-0-4.local        BIP   0/0/4          0.00     linux-x64
---------------------------------------------------------------------------------
all.q@compute-0-5.local        BIP   0/0/4          0.00     linux-x64
---------------------------------------------------------------------------------
all.q@compute-0-6.local        BIP   0/0/4          0.00     linux-x64
---------------------------------------------------------------------------------
all.q@compute-0-7.local        BIP   0/0/4          0.00     linux-x64
---------------------------------------------------------------------------------
all.q@compute-0-8.local        BIP   0/0/4          0.00     linux-x64
---------------------------------------------------------------------------------
all.q@compute-0-9.local        BIP   0/0/4          0.00     linux-x64
```

The output shows ten compute nodes (compute-0-0.local ... compute-0-9.local), each node has 4 cores in total (in the column resv / used / **tot**) and load average of 0.00.

Note: Please do not connect to the compute nodes directly.

Traditional execution vs. cluster:

To exemplify the use of SGE queue system, we will use two system commands that you have in your path:

- `hostname`, provides the name of the server/node
- `sleep`, the process remains idle for the number of seconds indicated

The usual way to run the "hostname" command is:

```
[ivan@eimtarqso ~]$ hostname eimtarqso.uoc.edu
```

However, we can execute this command in other nodes of the cluster through a SGE script. A simple sample script is shown below:

```
[ivan@eimtarqso ~]$ cat hostname_example.sge #!/bin/bash
#$ -cwd
#$ -S /bin/bash
#$ -N hostname_jobname
#$ -o hostname.out
#$ -e hostname.err
hostname
```

Note that the last line of the scripts calls the command/s that will be run in the assigned (by the SGE queuing system/schedule) node.

You need to use "qsub" to submit your script to the SGE queuing system. An example is shown below:

```
[ivan@eimtarqso ~]$ qsub hostname_example.sge
Your job 121597 ("hostname_jobname") has been submitted
```

Note that 121597 is the job ID assigned to the submitted job.

QUESTIONS:

1. What is the result of the hostname command? Where is the output of the execution? Check if new files have been created.

2. Execute the sample script several times (e.g., 3-4 times). What are the results? It is always the same? Why?

3. If you execute the example script (exactly the same file) multiple times at the same time, do you have any problem to identify/keep the output of each of the executions? How you can save the results of all executions?

Once you have submitted a job to the queue using qsub, you can cancel it, if necessary (for example, if you have realized that there is a mistake or you need to change parameters). This is done through the SGE command "**qdel**".

In order to exercise the "qdel" command you can run the example script shown below, which runs a sleep command for 100 seconds.

```
[ivan@eimtarqso ~]$ cat sleep.sge
#!/bin/bash
#$ -cwd
#$ -S /bin/bash
#$ -N sleep_job
#$ -o sleep.out
sleep 100
```

An example of the commands discussed above and the results obtained are provided as follows (note that comments in red are included manually).

```
[ivan@eimtarqso ~]$ qsub sleep.sge
Your job 121598 ("sleep_job") has been submitted
[we submit the job the SGE queue]

[ivan@eimtarqso ~]$ qstat
job-ID prior     name             user state submit/start at       queue                     slots ja-task-ID
-----------------------------------------------------------------------------------------------------------------
121598 0.00000   sleep_job        ivan qw    09/21/2015 01:37:34                             1
[with qstat we can see that the job is submitted and it is waiting in the queue to be dispatched/executed]

[ivan@eimtarqso ~]$ qstat
job-ID prior     name             user state submit/start at       queue                     slots ja-task-ID
-----------------------------------------------------------------------------------------------------------------
121598 0.55500   sleep_job        ivan r     09/21/2015 01:37:37   all.q@compute-0-2.local   1
[after a few seconds we can see that the job is in state "r", i.e., running]
```

```
[ivan@eimtarqso ~]$ qstat -f
queuename                 qtype   resv/used/tot.   load_avg arch    states
---------------------------------------------------------------------------------
all.q@compute-0-0.local   BIP     0/0/4            0.00     linux-x64
---------------------------------------------------------------------------------
all.q@compute-0-1.local   BIP     0/0/4            0.00     linux-x64
---------------------------------------------------------------------------------
all.q@compute-0-2.local   BIP     0/1/4            0.00     linux-x64
 121598 0.55500 sleep_job ivan       r 0.     9/21/2015    01:37:37    1
---------------------------------------------------------------------------------
all.q@compute-0-3.local   BIP     0/0/4            0.00     linux-x64
---------------------------------------------------------------------------------
(...)
[with qstat –f we can see that our job is running in node compute-0-2.local]

[ivan@eimtarqso ~]$ qdel 121598
ivan has registered the job 121598 for deletion [with qdel and the JobID we can cancel it]

[ivan@eimtarqso ~]$ qstat
[once cancelled, the job is not shown in the queue anymore]

[ivan@eimtarqso ~]$ qstat -f
queuename                 qtype   resv/used/tot.   load_avg arch    states
---------------------------------------------------------------------------------
all.q@compute-0-0.local   BIP     0/0/4            0.00     linux-x64
---------------------------------------------------------------------------------
all.q@compute-0-1.local   BIP     0/0/4            0.00     linux-x64
---------------------------------------------------------------------------------
all.q@compute-0-2.local   BIP     0/0/4            0.00     linux-x64
---------------------------------------------------------------------------------
all.q@compute-0-3.local   BIP     0/0/4            0.00     linux-x64
(...)
[ivan@eimtarqso ~]$
```

Large jobs vs. multiple small jobs:

Sometimes we need to make parametric or statistical studies that involve running a program multiple times. In this scenario we have two basic options:

1. Use a script that performs SGE different executions as part of a single job

2. Send multiple jobs, one for each (or a subset) of the executions.

In this course you must use the second option because it is the way to fairly share the resources among the students. For example, if a few students execute very long jobs in all resources, this can make the rest of students have to wait for hours or even days before their jobs can be executed.

Sample parametric study

The steps required to compile and run a sequential naive matrix multiplication program (without optimizations) is provided below (the sources are found in the virtual campus):

- Compile the program `mm.c` through: `gcc -O3 mm.c -o mm`
- Submit the job to the SGE queuing system. Note that the size of the array is passed to the program by a parameter

  IMPORTANT: do not execute the matrix multiplication program from the command line, it could overload the login node. Also the measurements won't be relevant due to resource sharing.

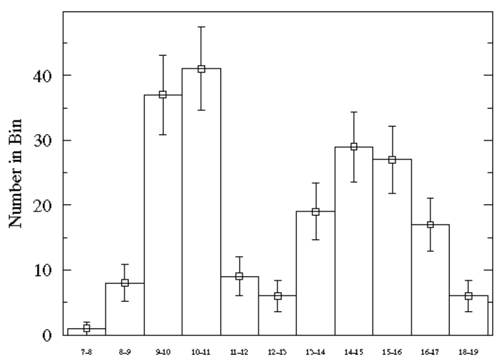
QUESTIONS:

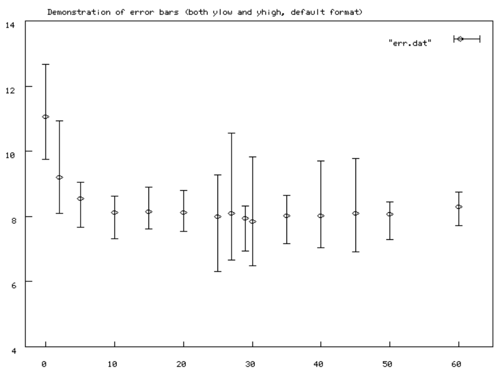
4. How did you copy the file `mm.c` from your computer to the server `eimtarqso.uoc.edu`?

5. Provide the SGE script that you have created to execute the program.

You are requested to conduct a parametric study using the matrix multiplication program discussed above. You need to provide the execution time of the matrix multiplication for different problem sizes (dimension of matrices, in this case square).

To mitigate the variability in executions (e.g., due to memory contention issues, cache effects, randomness, etc.) you are requested to conduct a statistical study, i.e., provide average, standard deviation, quartiles and other metrics across several executions. An example using error bars is provided below using gnuplot. Note that gnuplot is NOT mandatory, you can use your preferred tool for plotting.





As you will run multiple executions for each configuration (e.g., 4), you should consider running these executions systematically/programmatically. We suggest using scripts to perform this task (for example, shell scripts can call the qsub command for each of the configurations).

The following studies are requested:

- Execution time for different problem sizes (e.g. execution time obtain from "time" command). You can run matrix multiplication with the following sizes 100, 500, 1000, 1500, for relatively short (and convenient) executions.

QUESTIONS:

6. How did you obtained the execution time for the different executions?

7. Provide the scripts (actual scripts or methodology) that you used to conduct the executions systematically.

8. Provide a plot of execution time for different problem (matrix) sizes.

Scheduling: QUESTIONS

9. How does the cluster to access the files in your $HOME from any compute node of the cluster? What filesystem do you think is used in the cluster?

10. Provide the scheduling outcome of the jobs provided in the following table (assuming a system with **10 CPUs**) using the scheduling policies listed below:

(a) FCFS

(b) EASY-backfilling (backfill does not allow delaying an existing queued job)

It is also requested to provide:

(c) Resource utilization (%)

Resource utilization is defined as follows:

Utilization = used resources (CPUs)/total resources (CPUs)

Job #	Arrival Time	Runtime	#CPUs
1	1	8	8
2	2	2	2
3	2	8	1
4	3	4	4
5	6	2	1
6	8	3	9
7	8	4	1
8	9	6	4
9	9	4	4
10	10	2	10
11	11	4	2
12	12	4	2
13	12	4	1
14	16	3	1
15	16	6	1
16	20	2	5
17	24	3	8

An example at smaller scale (6 CPUs) and using the FCFS policy is shown as follows:

Job #	Arrival Time	Runtime	#CPUs
1	2	2	4
2	2	1	1
3	3	1	6
4	6	2	4

Scheduling outcome:

time / CPU#	1	2	3	4	5	6	7
CPU 1		J1	J1	J3		J4	J4
CPU 2		J1	J1	J3		J4	J4
CPU 3		J1	J1	J3		J4	J4
CPU 4		J1	J1	J3		J4	J4
CPU 5		J2		J3			
CPU 6				J3			

Utilization = 54.76%

Grading Policy

This assignment has 10 questions and a preliminary quiz. The use of scripting and providing plots with statistical values will be appreciated.

Submission Format and Due Date

Please submit your assignment as a single PDF containing all the responses including the scripts, graphs etc. Do not provide any other source. Brevity and concretion will be positively considered to grade the assignment.

Assignments should be submitted by 15 March 2020.