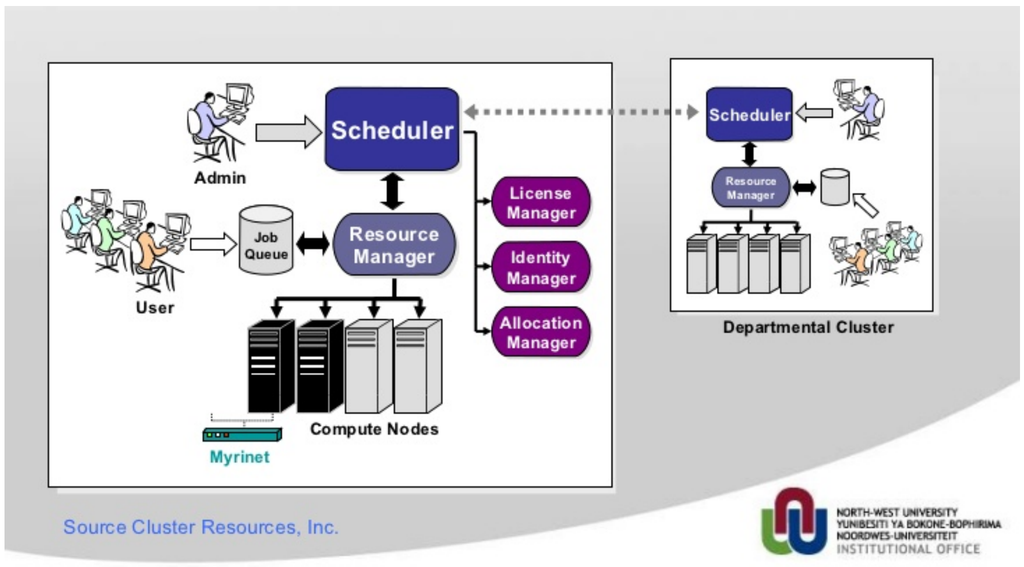
HIGH PERFORMANCE COMPUTING

Introduction and goals

High-Performance Computing facilities are managed by Job Schedulers and Resource Managers. As depicted in the following figure, they are responsible for managing hardware and software resources exposed by the HPC data center and processing user requests to execute applications on those resources.



This assignment aims 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.

Assignment description

You will conduct the assignments in a cluster based on GNU/Linux. In the virtual campus, you can find a manual for the queuing system's essential use; 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 on the SGE (Sun Grid Engine) queuing system installed in the UOC cluster:

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

You will receive the credentials to access the UOC cluster from acasys@uoc.edu.

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 may use PowerShell, optional features, or an ssh client, 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

Other ssh clients with X11-forwarding include MobaXterm (https://mobaxterm.mobatek.net).

The access to the system through ssh is on port 55000, so you need to use the following command or configure your ssh client accordingly:

```
ssh -p55000 YOUR_USERNAME@eimtarqso.uoc.edu
```

Your username will be provided with your credentials via email to your UOC address.

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).

If you have any issues or problems with your user account, you should report them to the support email address (**acasys@uoc.edu**).

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 you will run jobs to/across other nodes via the SGE queue system. It would be best to respect this policy because you will share resources with other colleagues.

Further information on the use of the cluster is provided in the cluster user guide provided along with this assignment description.

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

```
[user@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 framework 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:

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

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

```
[user@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 the `qsub` command to submit your script to the SGE queuing system. An example is shown below:

```
[user@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.

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**".

To utilize the "qdel" command, you can run the example script shown below, which runs a sleep command for 100 seconds.

```
[user@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).

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


[user@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]


[user@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]


[user@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]

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

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

[user@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
(...)
[user@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 when the jobs' execution time is significant because it allows fair sharing of the resources among the users. For example, if a few users execute very long jobs in all resources, the rest of them have to wait for hours or even days before their jobs can be executed.

Performance evaluation: Sample parametric study

The steps required to compile and run the program "`app.c`" are provided below. This program executes a (sequential) naive matrix multiplication and then calls a function "func" with the matrix size as the argument.

- Compile the program `app.c` executing the following command (note that it uses an object file that implements the "f" function used in `app.c`):

  Without optimizations: `gcc app.c lib.o -o app`

  With compiler optimizations (note that this option uses capital letter O and number 3):

```
gcc -O3 app.c lib.o -o app
```

- Submit the job to the SGE queuing system. Note that the size of the array is passed to the program by a parameter (e.g., "`./app 100`").

IMPORTANT: do not perform the executions directly from the command line; it could overload the login node. Also, the measurements may be misleading due to resource sharing.

You are requested to conduct a parametric study using the program discussed above. You need to provide the execution time of the program for different problem sizes (input parameter).

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 at least average and standard deviation across several executions. Two examples using error bars using Gnuplot are provided in `errorbars1.gnu` and `errorbars2.gnu`. Please note that these examples require `data1.txt` and `data2.txt` files, respectively. These codes are based on the examples provided online at http://gnuplot.sourceforge.net/demo/errorbars.html.

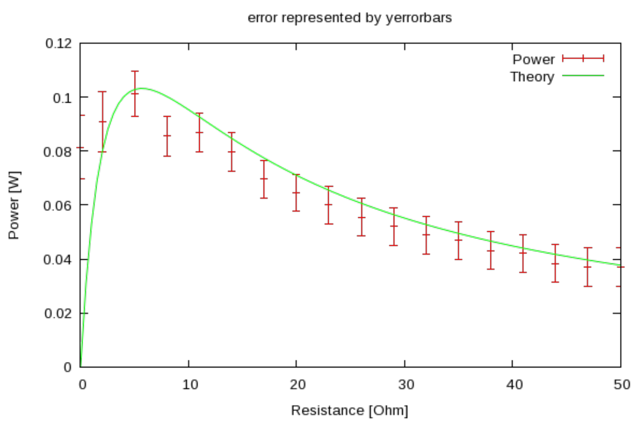
The gnuplot scripts can be run with the following commands

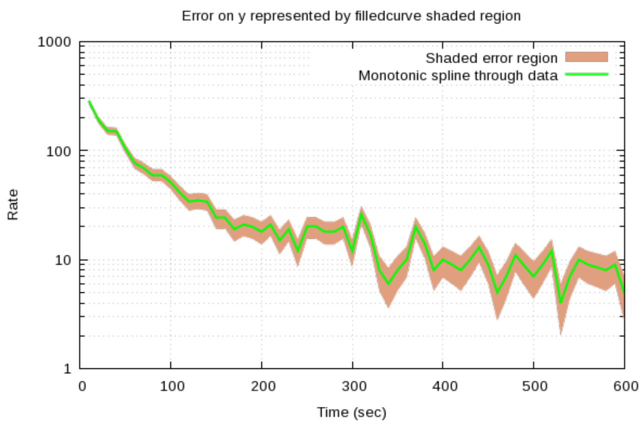
```
/share/apps/gnuplot/bin/gnuplot errorbars1.gnu
```

(generates `errorbars1.png` – figure below at the left)

```
/share/apps/gnuplot/bin/gnuplot errorbars2.gnu
```

(generates `errorbars2.png` – figure below at the right)





You are requested to execute the application discussed above (app.c) and measure its execution time with the following input parameters: 10, 100, 500, 1000, and 1500.

Ten (10) executions are required for each configuration to obtain the average and standard deviation. You may consider running these executions systematically/programmatically. We suggest using scripts to carry out this task.

The execution time can be obtained with the command "`time`" as shown in the example below:

```
[user@eimtarqso ~]$ time sleep 5

real 0m5.002s
user 0m0.001s
sys  0m0.001s
```

Use the time shown in "real" (5.002s in the example above).

QUESTIONS:

Q1 (3.5/10). Provide the **SGE script(s) and the shell scripts** (or methodology, e.g., program, command sequence) that you used to carry out the parametric study's executions systematically. Describe your choices (e.g., number of jobs launched vs. combinations in each job).

Q2 (3.5/10). Provide **a plot** of the execution time of the application with the requested input parameters. It is requested to include the average and standard deviation (or the percentiles/quartiles of your choice) of various (e.g., 5-10) executions. Note that Gnuplot is not mandatory for this question, and you can use your preferred tool for plotting; however, it is encouraged in preparation for the following assignments.

Q3 (0.5/10). Provide Based on the execution time results, can you guess what does function "`func`" do?

Automation of reports

Automating processes is an essential task in high-performance computing environments. For example, processing the output data and generating reports programmatically may be the most effective option for jobs that may take hours or even days to complete in a large facility.

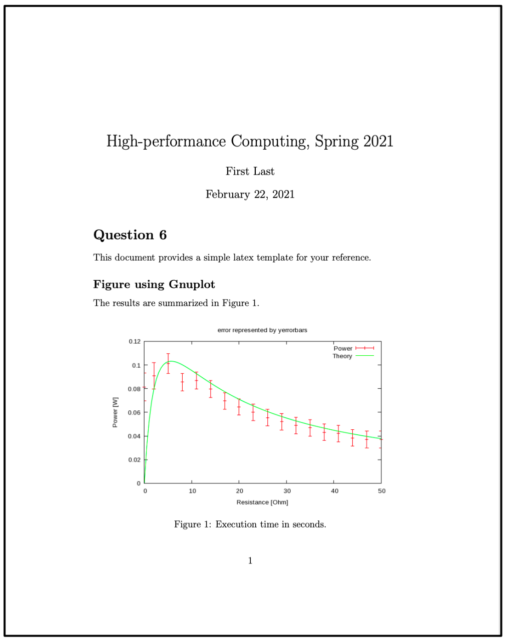
The previous section provides examples for using Gnuplot, which can be run programmatically upon the results obtained in your parametric study.

In addition to the plots, it is possible to automate the generation of report that combine multiple figures and information with Latex. A sample latex document (`report.tex`) is provided as a reference. This document provides some information as space holders and uses the sample Gnuplot figure `errorbars1.png`.

To compile the latex document you need to run:

```
[user@eimtarqso ~]$ pdflatex report.tex
```

It generates the PDF document `report.pdf` (see below).



High-performance Computing, Spring 2021

First Last

February 22, 2021

Question 6

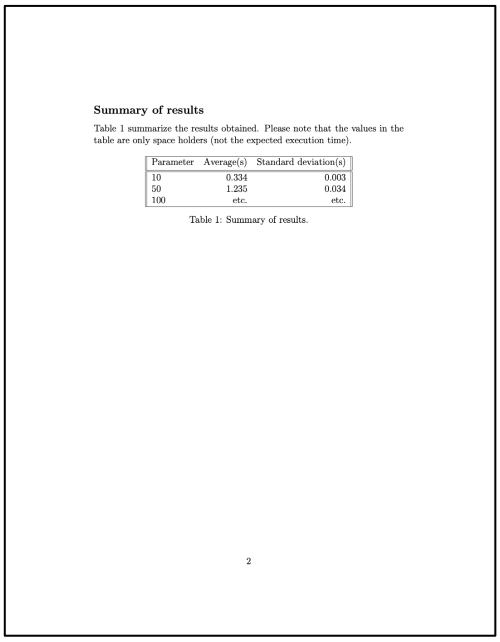
This document provides a simple latex template for your reference.

Figure using Gnuplot

The results are summarized in Figure 1.

Figure 1: Execution time in seconds.

1



Summary of results

Table 1 summarize the results obtained. Please note that the values in the table are only space holders (not the expected execution time).

Parameter	Average(s)	Standard deviation(s)
10	0.334	0.003
50	1.235	0.034
100	etc.	etc.

Table 1: Summary of results.

2

OPTIONAL: Provide the report and the scripts to automate its generation as described above with Gnuplot and Latex.

Queuing System and Scheduling

QUESTIONS:

Q4. (2.5/10). 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

c) Resource utilization (%) for FCFS and EASY-backfilling

Please note that 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 = 23 used slots / 42 total slots = 54.76%

Submission format

IMPORTANT: Your submission shall be a single file in PDF or ZIP format. Submissions in other formats will be returned without grading (e.g., RAR is not accepted).

The submitted file shall include:

- A single PDF file with the responses to the questions, including the scripts, plots, etc. **IMPORTANT: Plots as part of a spreadsheet are not accepted.**
- A directory named "files" with all the relevant materials (scripts, source code, Gnuplot scripts, latex document, latex report, etc.)

Brevity and concretion are a plus.