Remarks on the first assignment (complementing the short videos).

- We have used different ways to move data to the UOC cluster such as scp, WinSCP, or FileZilla.

- The management of the job output is essential in batch systems. To specify SGE output files associated with specific job executions, we have used an SGE variable with the job ID. This mechanism prevents re-writing the output file. An example is provided below:

```
#!/bin/bash
#$ -cwd
#$ -S /bin/bash
#$ -N hostname
#$ -o hostname.out.$JOB_ID
#$ -e hostname.err.$JOB_ID

hostname
```

- To obtain the execution time of a program, you can use different mechanisms; however, the use of the "time" command is sufficient for the requested level of accuracy. An example is provided below:

```
#!/bin/bash
#$ -cwd
#$ -S /bin/bash
#$ -N app
#$ -o app.out.$JOB_ID
#$ -e app.err.$JOB_ID

echo "Size 500"
time ./app 500
```

- The "time" command shows the execution time in the standard error (stderr). You can obtain this information in the same output file by modifying the following line:

```
#$ -e app.out.$JOB_ID
```

so that the standard error is written to the standard output.

- Source of the library used (`lib.c`)

```
#include <stdio.h>
#include <stdlib.h>
#include <stdlib.h>
#include <time.h>

void func(int size) {
  int r;

  srand(time(NULL));

  r = rand()%size;
  usleep(r*10000);
}
```

- There is an issue related to running long executions using a single job, i.e., that the resource utilization is not optimized as it can be unfair as other users may need to wait in the queue for a long time. Please note that the cluster is configured with a limit of one hour job running time, so jobs are killed after one hour. It is recommended that your executions are balanced in such a way that they last minutes instead of seconds, but the workload is scheduled using multiple jobs. The script below shows an example for running different configurations within the SGE script:

```
#!/bin/bash
#$ -cwd
#$ -S /bin/bash
#$ -N app
#$ -o app.out.$JOB_ID
#$ -e app.out.$JOB_ID

for i in {5,10,20,50,100,200,500,1000}; do echo “Size $i; ./app $i; done;
```

In this assignment, it is necessary to run multiple times the program to avoid outliers that can be caused by external factors (e.g., system maintenance) and obtain meaningful results. While you are expected to balance your workload in various scripts, it is not expected to queue hundreds of scripts of very short duration. You can come up with multiple combinations within a single SGE script as long as it doesn't last very long.

- The following example shows a way of executing different combinations using multiple jobs (one job for each execution).

```
#!/bin/bash
#$ -cwd
#$ -S /bin/bash
#$ -N app
#$ -o app.out.$JOB_ID
#$ -e app.out.$JOB_ID

echo "Size ${size}”
time ./app ${size}
```

and from the command line, you can use a loop with the qsub calls. An example is shown as follows:

```
for i in {5,10,20,50,100,200,500,1000}; do qsub -v size=''$i'' app.sge; done;
```

Note that -v variable='VALUE', assigns the value indicated in VALUE to the variable ${variable} in the SGE script.

- There are other alternatives. The example below uses a keyword (KEY) in a script template. This template can be copied, and the keyword replaced with the desired value.

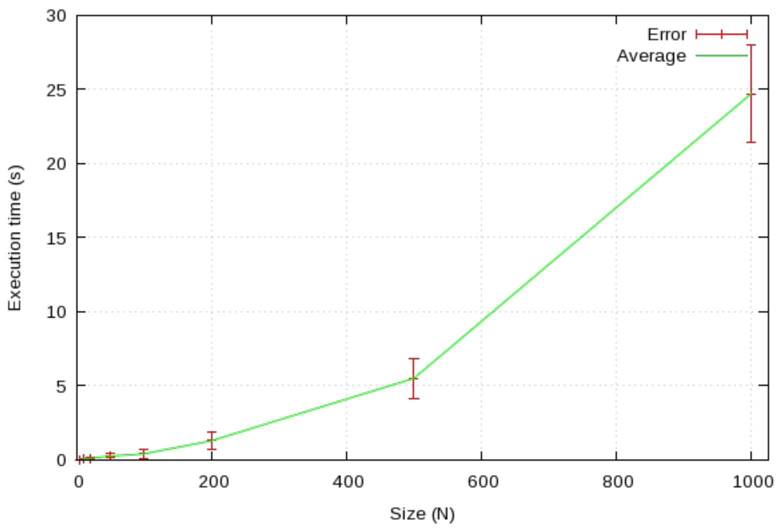
```
#!/bin/bash
#$ -cwd
#$ -S /bin/bash
#$ -N appKEY
#$ -o appKEY.out.$JOB_ID
#$ -e appKEY.out.$JOB_ID

echo "Size KEY"
time ./app KEY
```

and from the command line you can use the following loop, assuming that the template script is "app.sge".

```
for i in {5,10,20,50,100,200,500,1000}; do cp app.sge app$i.sge; sed -i
's/KEY/'$i'/g' app$i.sge; qsub app$i.sge; done;
```

The post-processing of the output data can be done in multiple ways (e.g., bash script, python script, gnuplot script).

- A reference sample plot is shown below.



- A summary of the scheduling outcomes using the FCFS and EASY-backfilling policies is provided in an attached file.