High Performance Computing - CET1

Introduction to HPC

Juan José Rodríguez Aldavero
April 3, 2020

Contents

1 Preliminary quiz

A. **Do you know what is cyberinfrastructure and what does it mean?**
I do not know very well what this term means, although I imagine that it is related to computing due to the cyber prefix, and that it has to do with the layoutof computing elements due to the term infrastructure.

B. **Please describe the concept of cyberinfrastructure and provide one example.**
After researching the term, we see how it refers to the integration into a single structure of a set of systems distributed over the Internet that deal with different tasks such as the acquisition, storage, and exploitation of data, and other computing techniques. As defined by J.Bottum, J.Davis and P. Siegel, "Cyberinfrastructure connects institutions, researchers, educators, and students with high-performance computing, remote sensors, large data sets, middleware, and sophisticated applications such as visualization tools and virtual environments. Allowing the sharing not only of tools and data but also of expertise, cyberinfrastructure merges technology, data, and human resources into a seamless whole." In brief, I think we can define it as a high-performance computer infrastructure.

C. **How do you think you can interface with a data-centric cyberinfrastructure?**
The interaction with a cyberinfrastructure will be through interfaces that communicate with high-performance computer systems. This interface will allow the performance of queries over the system and the extraction of information through high-speed connections, such as data visualization or analytics.

D. **How do you think cyberinfrastructure can be useful for science and engineering? Please provide an example.**
An interesting application of this type of computational structures can be the analysis of the large amount of information produced in different areas of science and technology. It can be useful, for example, to link the large amount of information produced in a scientific experiment with the group of scientists and specialists in charge of extracting knowledge from these data. A particular case that comes to mind is the analysis of data produced by the LHC particle accelerator, around 30 – 40tb per day.

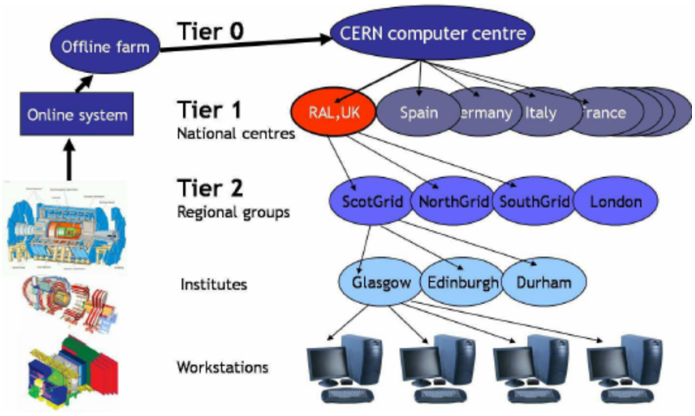

Figure 1.1: *CERN computing infrastructure*

2 Statistical analysis

In this exercise, we will learn the interface to control a cluster of computers through an SSH connection. The interaction with the cluster will be through a connection on the login node, using a terminal. This node will control and distribute the tasks and information to the different nodes of the cluster using a dedicated software called scheduler.

A. **What is the result of the hostname command? Where is the output of the execution? Check if new files have been created.**
The result of executing the hostname command is the name of the node we are connected to. In this case, because we have connected to the login node, we obtain the result eimtarqso.uoc.edu. The output of the execution is this message, print on the terminal and it does not create any new files on the working directory.

B. **Execute the sample script several times (e.g., 3-4 times). What are the results? It is always the same? Why?**
First, the sample script executed is the following:

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

hostname
```

Listing 2.1: *Hostname example script*

The result of executing the *hostname* command through the sample script is always the same: it writes out eimtarqso.uoc.edu in the output file. This is because we connect to the login node and this doesn't change, even though when we send a job to be executed it is assigned to any of the different computing nodes by the scheduling software. However, if we execute it several times, it comes out in different files of the same name, creating ambiguity.

C. **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?**
As we said before, if we execute the example script multiple times, there is a problem identifying each execution because we have configured the output files to be named the same way. This can be solved if we append the $JOB_ID parameter to the name as we will do later.

2.a Parametric study

With this exercise, we intend to find the execution times for a matrix multiplication script for several matrix sizes, and to perform a statistical analysis of these execution times to check the execution time variability of the cluster.

A. **How did you copy the file mm.c from your computer to the server eimtarqso.uoc.edu?** I used the software Filezilla as explained in the videotutorials for this exercise. I connected to the cluster using SFTP and transfered the file from my computer.
B. **Provide the SGE script that you have created to execute the program.** To perform a simple execution of the mm.c program, I created this simple script:

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

time ./mm 100
```

Listing 2.2: *Matrix multiplication script*

where the first six lines refer to the specification of parameters, and the last line refers to the execution of the compiled mm program for $L = 100$. As we can see, for clarity we have appended the $JOB_ID parameter to the file names.

In order to perform the statistic analysis, we run the script above for the following square matrix sizes: $L \in \{100, 500, 1000, 1500\}$. We do not examine the case for larger matrix sizes because the execution times are not linear and grow quickly with this parameter.

A. **How did you obtained the execution time for the different executions?** I decided to perform a for loop with 10 iterations for each matrix size calling the variable size for each execution. Then, the files containing the times are saved in the working directory with the names matmul.err$JOB_ID, where JOB_ID refers to the id assigned by the scheduling software to each job. Then, the times for the 10 executions for each matrix size are saved in the files matmul.err$JOB_ID and are easy to access.
B. **Provide the scripts (actual scripts or methodology) that you used to conduct the executions systematically.** The methodology followed was to find the execution times for each matrix size and for 10 iterations using the following script:

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

for i in {1,2,3,4,5,6,7,8,9,10}; do time ./mm ${size}; done;
```

Listing 2.3: *Matrix multiplication script*

it calls for the size variable and uses it to calculate the execution times for 10 consecutive iterations in the same computing job, saving the results in the same file matmul.err$JOB_ID. We then copied the results to a spreadsheet for simplicity in order to perform the statistical analysis. We could have created a gnuplot script that automates the result but given the small size of the data this process was simpler and more agile.

C. **Provide a plot of execution time for different problem (matrix) sizes.** In the following table we observe the different execution times:

$L = 100$	0,003	0,003	0,003	0,003	0,002	0,003	0,003	0,002	0,003	0,003
$L = 500$	0,498	0,496	0,498	0,495	0,496	0,497	0,498	0,499	0,498	0,499
$L = 1000$	4,834	4,83	4,831	4,831	4,829	4,832	4,832	4,832	4,829	4,829
$L = 1500$	16,510	16,505	16,508	16,506	16,503	16,506	16,501	16,500	16,509	16,505

If we perform a box plot for each matrix size, we obtain:

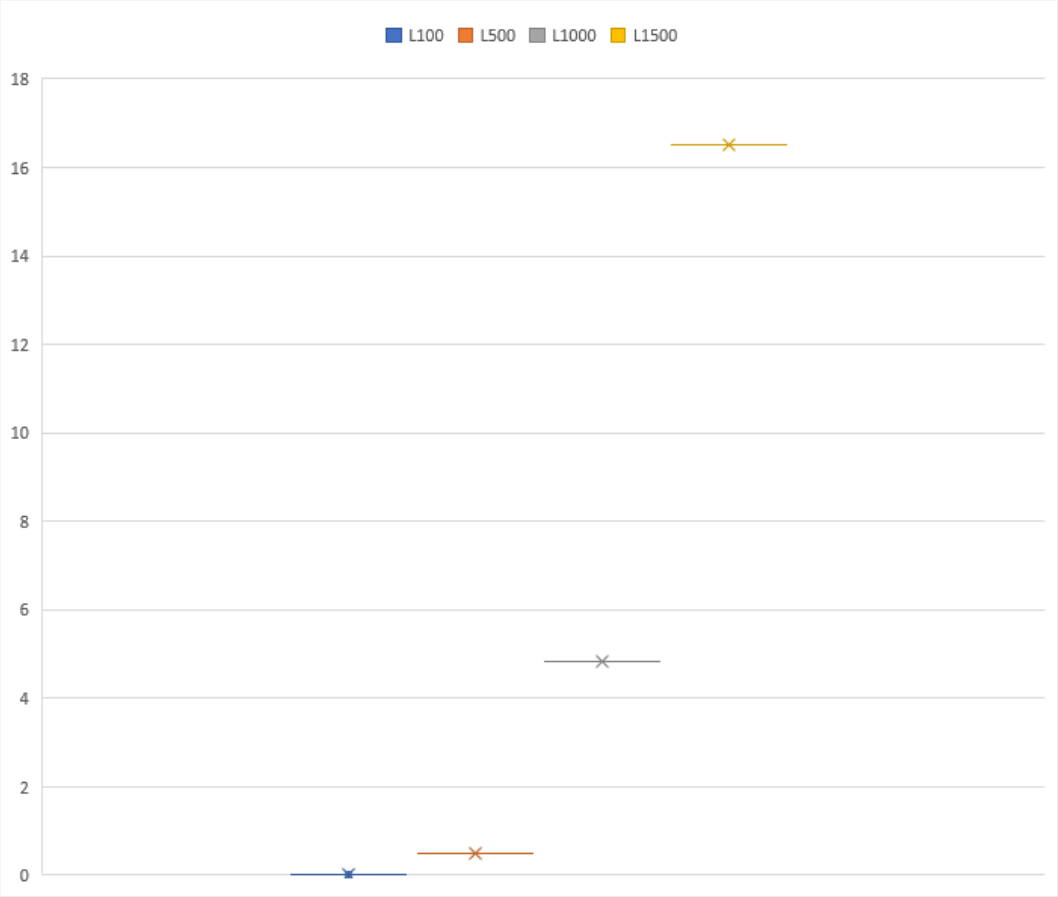

Figure 2.1: *Boxplot for the execution times*

We see how the variances for each instance can be seen properly because they are much smaller than the spread in execution times for all the instances. Therefore, we plot each instance individually to appreciate the variances:

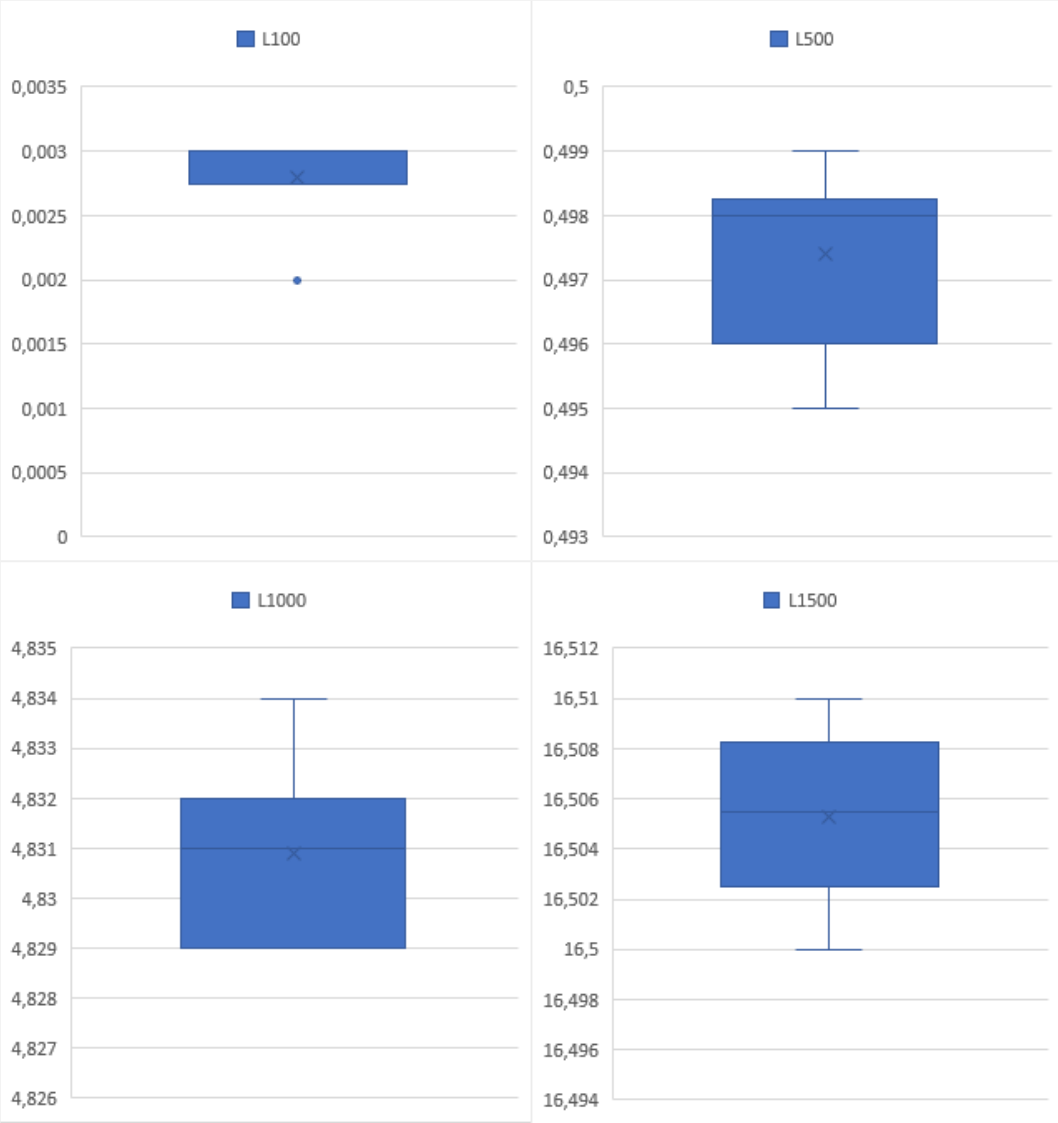

Figure 2.2: *Boxplot for the execution times*

In the following table we observe the mean and variance of each instance to better interpret the boxplots:

	L100	L500	L1000	L1500
Mean (s)	$2,80 \cdot 10^{-3}$	0,5	4,83	16,51
Variance (μs)	0,18	1,82	2,77	10,7

We see how the variances are in the order of microseconds, much lesser than the spread of execution times, in the order of seconds.

3 Scheduling

A. **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?**
The cluster uses the Open Grid Scheduler under the Sun Grid Engine (SGE). Researching the documentation of this scheduling software I found that the procedure used is FCFS for all jobs submitted from the same user or project. Therefore I assume that this is the procedure applied for this exercise.

B. **Provide the scheduling outcome of the jobs provided in the following table (assuming a system with 10 CPUs) using the scheduling policies listed below:**
For the First Come First Serve (FCFS) scheduling procedure we get the following scheduling outcome:

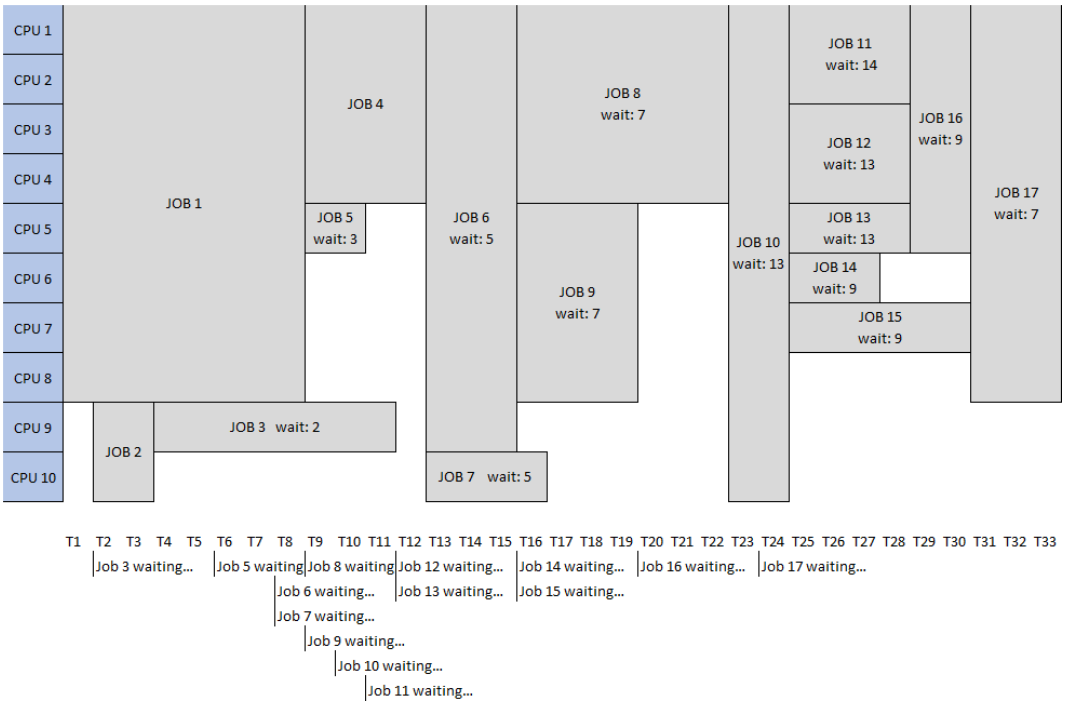

Figure 3.1: *Scheduling outcome for the FCFS algorithm*

with a total wait time of 116 units and a resource utilization around 76,1% of the total capacity of the cluster. We can see how this procedure is suboptimal because there are a lot of blank spaces and unused resources given by the fact that any job that comes before the others gets the resources first without exceptions. This is inefficient because no job can start before any previous one.

On the other hand, the EASY backfilling procedure is a simple scheduling algorithm that is much more efficient because, as its name implies, allows the backfilling of resources. With this procedure the scheduling outcome is:

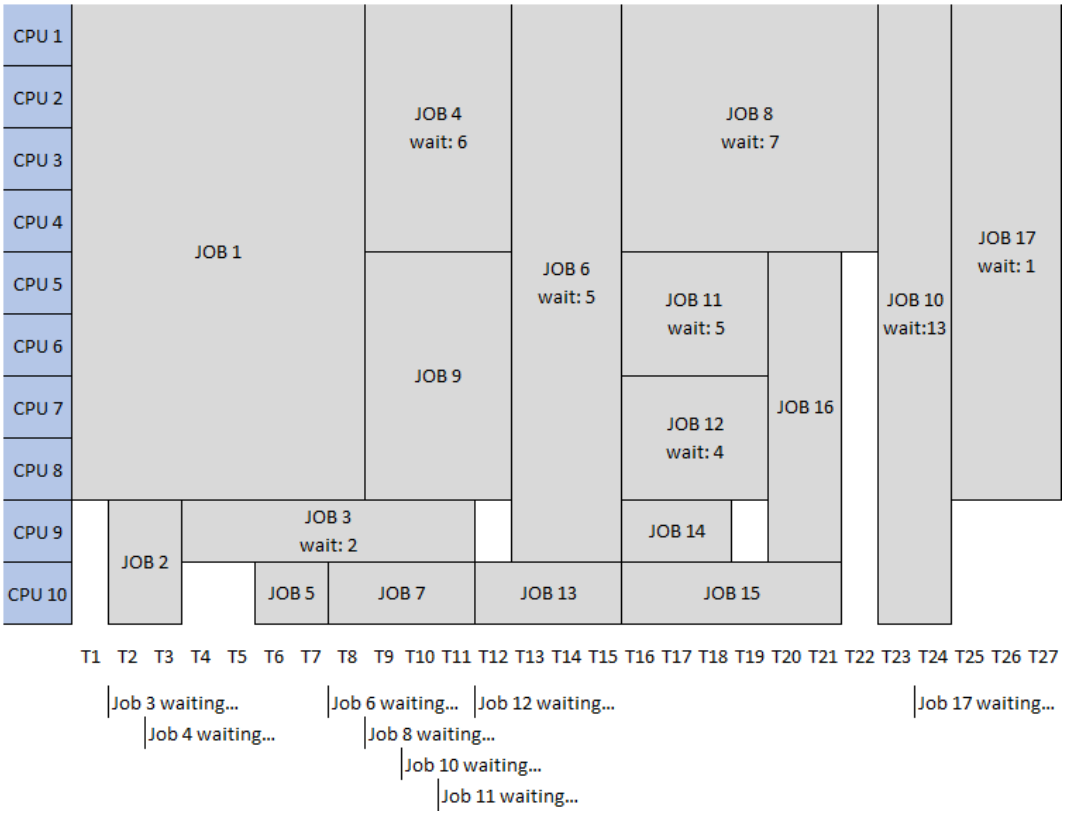

Figure 3.2: *Scheduling outcome for the EASY backfilling algorithm*

with this algorithm we obtain a total wait time of 43 units and a resource utilization around 93,3% of the total capacity of the cluster. We can see how it is much more efficient than the previous procedure because some jobs can start executing even if previous jobs haven been allocated yet (backfilling).