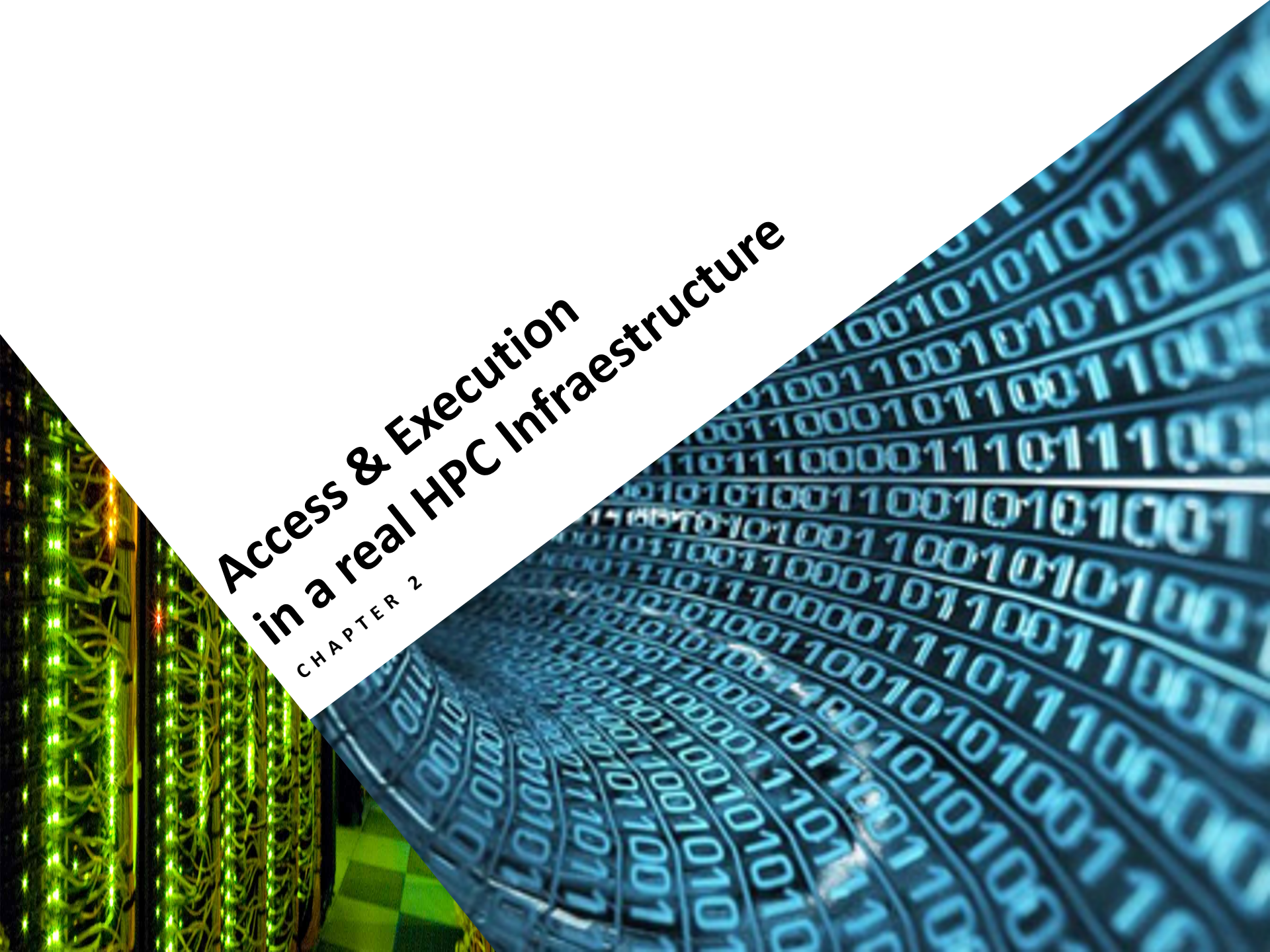
Access & Execution in a real HPC Infraestructure

CHAPTER 2

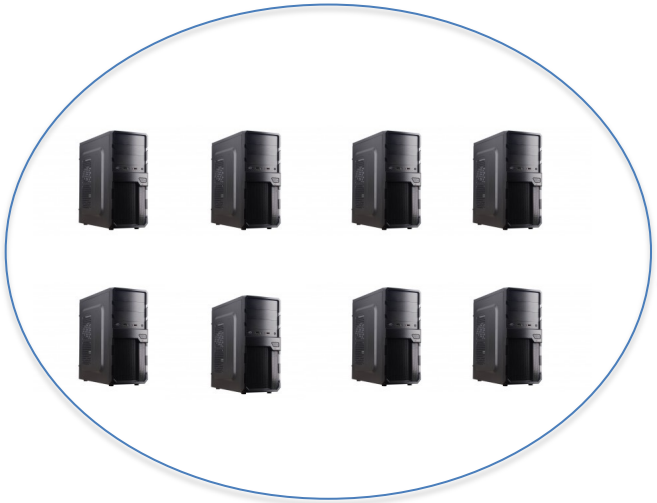


Access & Execution in a real HPC Infraestructure

- HPC Infrastructures at EPS
  - Educational Cluster Architecture
  - Front-end access
- SGE system queue management
  - Job Submission
  - Job Monitoring
  - Job Deletion
  - Frequent failures
  - Tasks Matrix

HPC Infrastructures at EPS

Moore: Educational Cluster



Front-end: moore.udl.cat

8 Nodes Intel Core I5
4cores/node x 4 GB of memory
Compute-0-0 to compute-0-7

For more information visit:
http://moore.udl.cat/

HPC Infrastructures at EPS

- Front-end access:

```
$ssh user@moore.udl.cat
```

(user: your id in the virtual campus)

- The password is your DNI without letter.
- Change the password by means of the Linux command passwd.
- Complete information of the cluster: http://moore.udl.cat/
  - Description of the cluster
  - Configuration MPICH 2
  - Execution of parallel jobs using the queues

Access & Execution in a real HPC Infraestructure

- HPC Infrastructures at EPS
- **SGE system queue management**
  - Job Submission

    Activity 1
  - Job Monitoring

    Activity 2
  - Job Deletion
  - Frequent failures
  - Tasks Matrix

    Activity 3

Job Submission - Sun Grid Engine (SGE)

- The ***qsub*** command allows to submit any task in the system queue by using a script

- The execution conditions for the applications and the required resources must be specified by the ***qsub command*** or by ***special SGE commands*** ($ #) **in the script** that is submitted.

- The meaning of the parameters are the same for both methods and lets you control things such as:
  - Required CPU.
  - Required Number of processors (for multi-processors work).
  - Destination of the output files.
  - Notification of the activity and / or job status

Configuration Options – script SGE

Example **1**

Execute the **Pi_serial** script in a node of the cluster moore, send the ***abort*** and ***end*** messages to your email address . In addition, the stderr and stdoutput must be stored on a file called “ouputs”.

- Script content (example.sh):

```
#!bin/bash
#$ -m ae
#$ -M francesc.gine@udl.cat
#$ -j y
#$ -cwd
#$ -o outoput
./PI_serial
```

- Command line:
  - Using script: $qsub example.sh
  - Using commands: $q*sub –m ae -M francesc.gine@udl.cat –j y –cwd –o output PI_serial.sh*

Configuration Options – script SGE

Example 2

```
#!/bin/bash
#  Force the shell to be the C-shell
#$ -S /bin/csh

#  Request 2 GBytes of virtual memory
#$ -l h_vmem=2G

#  Specify myresults as the output file
#$ -o myresults

#  Compile the program
gcc test.c -o mytestprog

#  Run the program and read the data that program
#  would have read from the keyboard from file mydata
mytestprog  < mydata
```

Configuration Options – script SGE

```
#!/bin/bash                          # Shell interpreter
#$ -V                                # get the current environment variables
#$ -cwd                              # Job execution in the current directory
#$ -N myJob                          # Job Name
#$ -j y                              # Combine stderr y stdout output
#$ -o $JOB_NAME.o$JOB_ID             # output filename
#$ -q high.q                         # Use the high.q
#$ -l h_rt=01:30:00                  # execution deadline (hh:mm:ss) - 1.5 hours
#$ -M pepe@alumnes.udl.cat           # Specify the e-mail address
#$ -m be                             # Notify at beginning and the end of the application
```

See the examples at moore.udl.cat:

http://moore.udl.cat/wordpress/?page_id=100

Configuration options – qsub command

Output options

-o pathname	Name of the file that will contain the main output
-e pathname	Name of the file that will contain error output messages
-j y *(highly recommended)*	(Join) Send both main ouput and error output messages into the same file.
-N jobname	Rather than the scriptfile use the supplied name as the jobname

Configuration option – qsub command

Notifications & Test

-M email_address	Email address for job notifications. Example: -M Pepe@gmail.com
-m b e a s	Send email(s) when the job **b**egins , **e**nds, **a**borted or **s**uspended Example: **–m be**
-now	Start running the job now or if can't be run exit with an error code.
-verify	do not submit the job but check and report on submission.

Configuration option – qsub command

Environment options

-cwd *directory*	Define the working directory. By default the job runs on the same directory it was submitted from. However, using this option you can specify a different directory.
-P *projectname*	Run the job under a special project account
-S shell	Use the specified shell to interpret the script rather than the default Bash shell. Example: **-S /bin/csh**
-v *variable[=value]*	Passes an environment variable to the executing jobs shell. Example **–v OMP_NUM_THREADS=4**
-V	Export all environment variables to the job.

Configuration option – qsub command

Resource options

-l h_rt=hh:mm:ss	The wall clock time. Example: **-l h_rt=01:00:00** (for one hour job)
-l h_vmem=memory or -l mem=memory	Sets the limit of virtual memory required (for parallel jobs per processor). Memory can be nnK, nnM or nnG Example: **-l mem=3G** (3 Gigabytes) **-l mem= 512M** (512 Mbytes)
-pe smp *n* -pe mpi *n* -pe mpich *n* -pe ompigige *n*	Specifies the parallel environment to be used for parallel jobs. n is the number of cpu's needed to run the parallel job. In general, to run an n-way parallel job you must specify n number of processors via this option. ***(Check the System Administrator Configuration)***
-pe *[env]* n-m	Rather than a fixed number of processors for any of the above parallel environments, a range of processors can also be specified to request minimum of (n) and maximum of (m) processors. Example: **–pe smp 4-8**
-help	Prints a list of options

Resources Request

- The resource requirements are specified using predefined resource requirement profiles (**-pe**) and queues (**-q**).
- The SGE seeks the resources that best fit the requirements and execute the jobs on selected nodes.
- Resources:
  - Storage space
  - CPU
  - Memory
  - Software
  - Operating System

Activity 1: SGE Job Submission (Optional)

- Create a shell script to execute in OMP the file executable EP.B.x, which has been previously combined .
- Configure the submission of this script with the following options:
  - Job name: ***EP_B***
  - Join the ***stdout*** and ***stderr***
  - Send a **email notification** when the job **ends** or **aborts**
  - Execute **with 2 and 4 threads**
  - Execute the job in the actual directory
- Submit the job script to the cluster system using ***qsub***

Job Monitoring

- It is possible to obtain information of a job in any of their states:
  - **Jobs waiting** in the queue:
    - We can use the ***qstat*** command to check if it has already started running
    - The ***qstat*** command provides information about all submitted jobs
  - **Jobs in Execution**:
    - Use ***qstat -j nro_trabajo*** for monitoring the job state, including time and memory consumption.
  - **Jobs Completed**:
    - ***qacct*** provides information about finished jobs by consulting the database of past executions.
      - `qacct -j nro_trabajo`

Job Monitoring - qstat

- The ***qstat*** command displays all jobs in the system queues that are waiting to be executed or in progress.
  - Complete list:
    - `qstat -f`
  - Jobs from a specific user:
    - `qstat -u nombre de usuario`
  - Detailed information from a specific user:
    - `qstat -f -u nombre de usuario`
  - Detailed information from a specific job:
    - `qstat -j id_job`

- The possible Job states are:

qw	waiting	**R**	Reset
r	in progress	**S**	suspended
h	held	**E**	error

Job Monitoring – qstat Ejemplo

- `qstat`

```
job-ID  prior   name       user         state submit/start at     queue                          slots ja-task-ID
-----------------------------------------------------------------------------------------------------------------
3064625 0.57695 grd80x120x cpp06kw      r     10/15/2010 02:35:43 long.q@node81.iceberg.shef.ac.    1
3064818 0.56896 M11_NiCl.3 chp09bwh     r     10/15/2010 15:44:44 parallel.q@node107.iceberg.she    4
3065270 0.56657 pythonjob  co1afh       r     10/16/2010 13:34:33 long.q@node59.iceberg.shef.ac.    1
3065336 0.56025 parallelNe elp05ws      r     10/16/2010 13:34:33 long.q@node86.iceberg.shef.ac.    1
3065340 0.56025 parallelNe elp05ws      r     10/16/2010 15:31:51 long.q@node23.iceberg.shef.ac.    1
3065558 0.55060 coaxialjet zzp09hw      r     10/17/2010 14:03:53 parallel.q@node105.iceberg.she   17
3065934 0.54558 periodichi mep09ww      r     10/18/2010 08:39:06 parallel.q@node96.iceberg.shef    8
3066207 0.53523 gav1       me1ccl       r     10/18/2010 20:00:16 long.q@node87.iceberg.shef.ac.    1
3066213 0.53510 ga2        me1ccl       r     10/18/2010 20:00:26 long.q@node87.iceberg.shef.ac.    1
3066224 0.53645 DDNaca0    mep09ww      r     10/18/2010 23:41:56 parallel.q@node112.iceberg.she   12
3066226 0.53493 ga3        me1ccl       r     10/18/2010 20:00:46 long.q@node33.iceberg.shef.ac.    1
3066231 0.53491 ga4        me1ccl       r     10/18/2010 20:00:46 long.q@node33.iceberg.shef.ac.    1
3066415 0.53078 job        elp07dc      r     10/19/2010 09:25:24 eenano.q@node124.iceberg.shef.   32
3066896 0.52323 fluent12jo fc1jz        qw    10/19/2010 05:32:01
3067083 0.52222 Oct_ATLAS  php09ajf     qw    10/19/2010 17:41:01
```

- `qstat -u cs1ds`

Job Deletion - qdel

- Using ***qdel*** can remove the jobs in the waiting queue and abort those already running.
  - Deleting a specific job:
    - `qdel 15112`
  - Deleting a list of jobs:
    - `qdel 15154 15923 15012`
  - Deleting all jobs from the same user
    - `qdel -u <user_name>`

Activity 2: SGE Job Monitoring (Optional)

- Submit again the script used in the ***activity1*** and monitor their execution.
  1. Modify the script to lengthen the execution time, if you consider necessary.
  2. Submit the job to the SGE system queues.
  3. Perform the following tasks of monitoring:
     - Obtain information about all the jobs in the queue
     - Obtain information about the jobs of a specific user
     - Obtain detailed information about the job previously submitted
  4. After the execution check the resources used by the executed job using ***qacct***.

     ```
     qacct -j job_id
     ```
  5. Submit the job again. Delete its execution and request again the resources used by the job.

     ```
     qdel -j job_id
     ```

- Deliveries

  Files with the output obtained in the steps 3, 4 and 5.

Frequent Errors

- The specified binary file in the job script is not found.
- The required input files are not in the home directory.
- The disk quota is exceeded and the job fails when it tries to write to a file.
- The maximum execution time defined is exceeded.
- Undefined environment variables.
- Hardware failure.

Task Matrix

- SGE allows the execution of a special type of jobs called **tasks matrix**.
  - By using a single ***qsub*** command a series of jobs using the same template can be submitted.
- Very useful functionality for execution of **Parameter Sweep Applications** (PSA).
  - High Performance Computing and Simulation
  - Calibration, Parameter Tuning, etc.
- SGE allows the efficient configuration and execution of a **tasks matrix**.
  - Manages the calculations as a set of independent tasks, but grouping the results as if it were a single task.
  - Allows monitoring and control as individual tasks or as a subset of tasks.

Task Matrix - Configuration

- To define a **tasks matrix** it is necessary:

  1) Indicate to the SGE how many tasks have to create from the submitted job.

     - Using ***qsub –t*** it is possible to define the number of tasks
       - `qsub mysim -t 1-10` `# Submmit mysim 10 times`
       - `qsub mysim -t 2-10:2` `# Submit mysim 5 times.`
         `Tasks: 2,4,6,8,10`

  2) Configuring each task with its own parameters, when necessary.

     - SGE provides a special environment variable called ***SGE_TASK_ID***, so that each of the tasks of the matrix receives a different task ***id***.
     - This ***id*** can be used by each task for controlling their functionality.

Task Matrix - Example

- The variable ***$SGE_TASK_ID*** can be used for parameterize the job. This variable can codify the set of operations to be done by the task or can be used to initialize a seed number generator, etc.

- It is also possible to use this variable for selecting the input file.

```
#!/bin/sh

# Defining a matrix of tasks in the range from 1 to 10.000
#$ -t 1-10000

# The SGE_TASK_ID variable identifies the number of task in each computational node.
if [ ! -e ~/results/output.$SGE_TASK_ID ]
then
  ~/program -i ~/data/input.$SGE_TASK_ID -o ~/results/output.$SGE_TASK_ID
fi
```

MPI Parallel Job Submision – Script Example

```
#!/bin/bash
## Specifies the interpreting shell for the job.
#$ -S /bin/bash
## Specifies that all environment variables active within the qsub utility be exported to the context of the job.
#$ -V
## Execute the job from the current working directory.
#$ -cwd
## Parallel programming environment (mpich) to instantiate and number of computing slots.
#$ -pe mpich 4
## The  name  of  the  job.
#$ -N job_name
## Send an email at the start and the end of the job.
#$ -m be
## The email to send the queue manager notifications.
#$ -M correu@alumnes.udl.cat
## The folders to save the standard and error outputs.
#$ -o $HOME
#$ -e $HOME
MPICH_MACHINES=$TMPDIR/mpich_machines
cat $PE_HOSTFILE | awk '{print $1":"$2}' > $MPICH_MACHINES
## In this line you have to write the command that will execute your application.
mpiexec -f $MPICH_MACHINES -n $NSLOTS <executable_file>
rm -rf $MPICH_MACHINES
```

Activity 3: SGE Tasks Matrix (Optional)

- Codify a program in C for doing the following calculation

```
For i=10 to 100000 step 1000
```

$$\sum_{j=1}^{i} j$$

```
EndFor
```

  - Identify the necessary parameters for using a tasks matrix where each taks execute one step calculation.
- Solution steps:
  1. Compile your program
  2. Define the Submit Script using the task matrix.
  3. Submit the job using *qsub*
  4. Monitor the execution
- Deliveries
  - Submission Script
  - Monitoring output files
  - Output files with the results