

Table of Contents

Submitting machine o puerta de entrada al cluster SMG	2
Sistema de ficheros	2
Software	2
Como usar el comando use	2
pbsnodes: Componentes del clúster	3
Job	4
Los requerimientos de un job en una cola PBS	4
Comandos cola PBS	6
Otras wikis	6
Ejemplos	7
Lanzar un job interactivo	7
Lanzar un job con dependencias	7
Array de jobs	7
Trabajo con variable	9
Monitorización del clúster	10
Recomendaciones	10

Información actualizada en la [?wiki del cluster del Sdel](#)

Submitting machine o puerta de entrada al cluster SMG

El acceso al clúster se debe hacer desde la maquina `ui.macc.unican.es` mediante [?SSH](#):

```
[user@localmachine]$ ssh user@ui.macc.unican.es
```

O bien con la aplicación [?putty](#) desde Windows.

Sistema de ficheros

Los directorios a continuación indicados son accesibles desde cualquier maquina del cluster:

`/oceano/gmeteo/users/[usuario]`: home del usuario. Cada usuario tiene asignado una cuota de almacenamiento de 50GB. Para comprobar su estado utilice el comando **quota**:

```
[user@ui ~]$ quota -s
Disk quotas for user user (uid XXXXX):
   Filesystem  blocks   quota  limit  grace  files   quota  limit  grace
  abedul:/meteo 16655M 40960M 51200M          195k    0    0
```

`/oceano/gmeteo/DATA`: almacenamiento de todos los datos (observaciones, GCMS, RCMs, ...). Para comprobar el espacio libre disponible utilice el comando **df**:

```
[user@ui ~]$ df -h /oceano/gmeteo/DATA
Filesystem      Size  Used Avail Use% Mounted on
-                127T   34T   94T  27% /vols/seal/oceano/gmeteo/DATA
```

`/oceano/gmeteo/WORK`: directorio de trabajo. Para comprobar el espacio libre disponible utilice el comando **df**:

```
[user@ui ~]$ df -h /oceano/gmeteo/WORK
Filesystem      Size  Used Avail Use% Mounted on
-                13T   8,6T  4,5T  66% /vols/seal/oceano/gmeteo/WORK
```

- `/oceano/gmeteo/SOFTWARE`: directorio con las aplicaciones disponibles.

Software

La gestión del software, tanto compiladores como programas, se realiza mediante el comando **use**.

Como usar el comando use

Para el uso del comando **use** es necesario realizar un `source` del fichero `load_use`:

```
[user@ui ~]$ source /software/meteo/use/load_use
```

- Es recomendable copiar esta instrucción en el fichero `.bashrc`, disponible en cada home de usuario, para no repetir su uso.
- Listado de de software disponible:

```
[user@ui ~]$ use

Programs available for "use":

cdo                gmt
grib_api           grib_api1.10
hdf5              intel
intell1           java
matlab2009a       matlab2013a
```

```
nco                ncview
netcdf             netcdfintel
netcdf_viejo      openmpi4intel
openmpi18intel    pgi
p_interp          python
python2.7.2       python2.7.2b
```

- Configuración de un programa:

```
[user@ui ~]$ use python2.7.2b
```

pbsnodes: Componentes del clúster

El comando `pbsnodes` nos permite examinar todos los componentes que se encuentran en el clúster, sus características y cuál es su estado. Esta información pormenorizada para cada *nodo* (*workind node*) tiene la siguiente estructura:

```
[nodo]
  state = [estado]
  np = [ncpus]
  properties = [cola asignada]
  ntype = [tipo]
  jobs = [lista de jobs que tiene ejecutándose]
  status = [propiedades físicas de la unidad]
```

Los valores para el nodo:

- [estado]: "active", "all", "busy", "down", "free", "offline", "unknown", "up"
- [propiedades físicas de la unidad]: opsys, uname, sessions, nsessions, nusers, idletime, totmem, availmem, phymem, ncpus, loadave, netload, state, jobs, varattr, rectime

De las más importantes tenemos:

- state: estado del nodo
- ncpus: Número máximo de cpus de cálculo del nodo (equivalente al número máximo de trabajos permitidos)
- phymem: memoria total de la unidad

Se muestra una parte de su ejecución para mostrar su contenido (se muestra parcialmente):

```
[user@ui ~]$ pbsnodes
wn002.macc.unican.es
  state = job-exclusive
  np = 2
  properties = ensembles,macc,lcgpro,estadistica
  ntype = cluster
  jobs = 0/308594.ce01.macc.unican.es, 1/308606.ce01.macc.unican.es
  status = opsys=linux,uname=Linux wn002.macc.unican.es 2.6.32.14-5-16-18-33-36-38-41-60-74-89-103-104 #18 SMP
Sat May 29 06:31:42 EDT 2010 x86_64,sessions=10248, 18567, nsessions=2,nusers=1, idletime=26392, totmem=5137992kb,
availmem=4784508kb, phymem=2057808kb, ncpus=2, loadave=1.03, netload=2685739215248,state=free, jobs=308594.ce01.
macc.unican.es 308606.ce01.macc.unican.es, varattr=, rectime=1292346952
```

El nodo `wn002.macc.unican.es` (con 2 cpus y 2GB de memoria, asignado a las colas: 'ensembles', 'macc', 'lcgpro', 'estadistica') está totalmente ocupada con dos trabajos (308594.encia y 308606.encia) de un mismo usuario

```
wn002.macc.unican.es
  state = free
  np = 2
  properties = ensembles,macc,lcgpro,estadistica
  ntype = cluster
  status = opsys=linux,uname=Linux wn002.macc.unican.es 2.6.18-194.26.1.e15xen #1 SMP Tue Nov 9 13:35:30 EST
```

```
2010 x86_64, sessions=? 0, nsessions=? 0, nusers=0, idletime=111779, totmem=4866040kb, availmem=4694372kb, physmem=1785856kb, ncpus=2, loadave=0.00, netload=8284314822645, state=free, jobs=, varattr=, rectime=1292346961
```

La unidad `wn002.macc.unican.es` (con 2 unidades de cálculo y 2GB de memoria, asignado a las colas: 'ensembles', 'macc', 'lccpro', 'estadística') está libre para admitir trabajos

```
wn042.macc.unican.es
state = job-exclusive,busy
np = 16
properties = oper
ntype = cluster
jobs = 0/308629.ce01.macc.unican.es, 1/308629.ce01.macc.unican.es, 2/308629.encia, 3/308629.encia, 4/308629.encia, 7/308629.encia, 8/308630.encia, 9/308630.encia, 10/308630.encia, 11/308630.encia, 12/308630.encia, 13/308630.encia, 30.encia, 15/308630.encia
status = opsys=linux, uname=Linux wn042.macc.unican.es 2.6.18-164.15.1.el5 #1 SMP Wed Mar 17 11:30:06 EDT
2010 x86_64, sessions=25810 25872, nsessions=2, nusers=1, idletime=2324174, totmem=19507632kb, availmem=14199848 kb, physmem=16427448kb, ncpus=16, loadave=17.10, netload=640484970867, state=busy, jobs=308629.ce01.macc.unican.es 308630.ce01.macc.unican.es, varattr=, rectime=1292346966
```

La unidad `wn042.macc.unican.es` (con 16 unidades de cálculo y 16 GB de memoria, asignada a la cola 'oper') está ocupada con 16 trabajos (el job `308629.encia` está en las cpus de la 0 a 7 y el job `308630.encia` de la 8 a la 15) de un único usuario.

Job

Un job del clúster tiene unos cuantos requerimientos básicos tales como:

- **número de procesos:** el número de cpus que se requieren para la ejecución del job.
- **cola de ejecución:** etiqueta que se le da a un grupo de nodos. Determina los nodos que acogeran todos los jobs que se manden a una cola. Es muy común que un clúster se compartimente en distintas colas a modo de intentar maximizar el rendimiento de un clúster. Estas colas pueden compartir nodos entre ellas y estar asignadas a proyectos y/o grupos de usuarios distintos.
- **wall-time:** determina el tiempo de ejecución del job dentro del clúster. Una vez pasado este tiempo el job y su aplicación serán detenidos forzosamente. Es un muy común que en los clústers se ejecuten antes los jobs con un wall-time pequeño que con uno de grande.

La pericia de `l@s usuari@s` en saber escoger adecuadamente los requerimientos de sus jobs, determinará la rapidez con la que se ejecutaran.

Los requerimientos de un job en una cola PBS

En esencia los gestores colas son muy parecidos entre sí. Sólo cambia parte de su semántica.

En la gestión de colas PBS existen dos términos para referirse a la unidad cálculo: el *node* y el *cpp*. Un *node* equivalen a las unidades físicas de cálculo (los ordenadores) que integran distintas cantidades de cpus *cpp* (2, 8, 16,...)

Las peticiones se hacen por un conjunto de *flags*. El envío del job al gestor de la cola se puede hacer añadiendo todas las directrices en la línea de comandos de la aplicación, o por medio de una script (archivo de texto con instrucciones de sistema). En según que circunstancias nos va interesar hacerlo de una manera u otra, pero las dos son equivalentes. La estructura tipo de un script (llamado por ejemplo `job.pbs`) de colas tiene la siguiente estructura (se añaden todas las opciones a modo explicativo):

```
#!/bin/bash
### Job name
#PBS -N [jobname]
### Max run time
#PBS -l walltime=[HH]:[MM]:[SS]
### Max memory
#PBS -l mem=[MM][kb/mb/gb/tb]
### Queue name
#PBS -q [queueNAME]
### Job mail
#PBS -m [flags]
#PBS -M [emailCOUNT]
### Standard error output
```

```
#PBS -e [rutaArchivo]
### Standard output
#PBS -o [rutaArchivo]
### Dependecy
#PBS -W depend=afterany:[jobid]
### Array request
#PBS -t [array]
### System variable
#PBS -v [variable]
### Number of nodes and processors per node
#PBS -l nodes=[N]:ppn=[M]

cd ${PBS_O_WORKDIR}
source /software/ScientificLinux/4.6/etc/bashrc

[aplicacion]
```

En esta plantilla se muestra la sintaxis de los siguientes requerimientos/especificaciones. Se tienen que cambiar los argumentos que estén entre [].

- *-N [jobname]*: nombre del job
- *-l walltime=[HH]:[MM]:[SS]*: duración del job (en [horas]:[minutos]:[segundos]).
- *-l mem=[MM][kb/mb/gb/tb]*: memoria requerida y límite de la memoria (número entero) en kb: kilobytes, mb: megas, gb: gigas, tb: teras (NOTA: la memoria del nodo se irá consumiendo con los valores de 'mem' de cada job. El job que requiera mas memoria que la que le queda libre al nodo, se quedará a la espera hasta que se libere la memoria necesaria. Si el job al correr supera el límite, el sistema de colas lo matará)
- *-q [queue]*: nombre de la cola a la cual se manda el job
 - *-m [flags]*: indica cuando se tiene que mandar un correo. Si no se pone este requerimiento si el job es abortado por el sistema se manda un correo al usuario (variable MAIL). Hay las siguientes opciones (son combinables):
 - *n*: sin correo
 - *a*: el job es abortado por el sistema
 - *b*: se inicia la ejecución del job
 - *e*: el job a terminado
- *-M [emailCOUNT]*: dirección de correo donde se quiere que mande un job
- *-e [rutaArchivo]*: ruta del archivo en donde se quiere almacenar la salida estandard del error del job (si no se indica se construye en el directorio donde está el script el archivo \${PBS_JOBNAME}.e\${PBS_JOBID})
- *-o [rutaArchivo]*: ruta del archivo en donde se quiere almacenar la salida estandard del job (si no se indica se construye en el directorio donde está el script el archivo \${PBS_JOBNAME}.o\${PBS_JOBID})
- *-W depend=afterany:[jobid]*: el job se mandará a la cola cuando otro job anterior (el número [jobid]) termine su ejecución (no importa como termine)
 - *-t [array]*: permite tratar un conjunto de jobs cómo una array. Cada job es identificado con un único valor dado por la [array] que se construye como combinación de valores. Por ejemplo:
 - 1-100: 100 jobs del 1 al 100
 - 1-5,15,35: 5 jobs del 1 al 5, el 15 y el 35
- *-v [variable]*: para mandar un job que tome el valor [variable]
- *-l nodes=[N]:ppn:[M]*: número de nodos ([N]) y número de cpus a coger de cada nodo ([M]) la aplicación se repartirá en [N]x[M] cpus

Para editar un script desde una ventana de mar se puede utilizar nano y/o vi :

```
[user@ui ~]$ nano job.pbs
[user@ui ~]$ vi job.pbs
```

El job se mandaría a la cola con la instrucción:

```
[user@ui ~]$ qsub job.pbs
```

o bien por la línea de comandos en una línea como sigue:

```
qsub -N [jobname] -l walltime=[HH]:[MM]:[SS] -l mem=[MM] -q [queueNAME] -m [flags] -M [emailCOUNT] -e [rutaArchivo]
-o [rutaArchivo] -W afterany:[jobid] -t [array] -v [variable] -l nodes=[N]:ppn=[M] [aplicacion]
```

Todos los jobs tienen un número que los identifica que se llama [JOB ID]. En el clúster del GMS tiene el siguiente formato: [jobid].encina. Cada job tiene asociado los siguientes valores PBS:

- **Job ID:** identificador del trabajo (asignado por PBS).
- **Username:** propietario del trabajo (usuario que lo envió).
- **Queue:** cola en la que reside el trabajo.
- **Jobname:** nombre del trabajo dado por el usuario o establecido por PBS en caso de no especificarse.
- **NDS:** número de nodos solicitados.
- **TSK:** número de cpus solicitadas.
- **Req'd Memory:** cantidad de memoria solicitada.
- **Req'd Time:** tiempo de reloj (wallclock) solicitado.

Comandos cola PBS

Hay distintas instrucciones para gestionar los jobs de un usuario. Si se quiere aunar en el significado y las opciones de estos comandos abrir el manual con la instrucción `man [comando]`:

- **qsub [archivo].pbs** envía del job [archivo].pbs a la cola
- **qdel [jobid]** termina forzosamente el job con id [jobid]
- **qstat** permite ver el estado de los jobs gestionados. Tiene también distintos flags (se muestran los más usuales):
 - `-n1` muestra todos los nodos a los cuales se ha mandado un job (con tantas veces el nombre del nodo, cómo cores cogidos)
 - `-u [usuario]` muestra todos los jobs de [usuario] Por lo general muestra en lista los trabajos que concidan con los criterios escogidos con la siguiente estructura:

```
[jobid] [usuario] [queue] [jobname] [SessID] [NDS] [TSK] [Req' Memory] [Req' time] [S] [Time]
```

- Los estados [S] que puede tener un job en la cola son:
 - E el trabajo está saliendo después de finalizar su ejecución.
 - H el trabajo está capturado (*Hold*, ej. esperando que termine otro)
 - Q el trabajo está en cola, elegible para su ejecución.
 - R el trabajo está ejecutándose.
- **qalter** permite alterar algunos de los parámetros de los jobs mientras está en espera

Un ejemplo de salida de `qstat -n1`:

```
[user@ui ~]$ qstat -n1
encina:
-----
```

Job ID	Username	Queue	Jobname	SessID	NDS	Req'd TSK	Req'd Memory	Elap Time	S	Time
476038.encina	gutierjm	estadist	STDIN	17506	--	--	6gb	--	R	1664: wn010/0
539894.encina	solanajr	gtfe	CPSE-Girifalco	22778	1	1	--	--	R	146:0 wn061/0
541194.encina	maru	asna_b3	JID_7757	3032	1	16	--	--	R	10:48 wn031/15+wn031/14+wn
541197.encina	maru	asna_b3	JID_8153	13318	1	16	--	--	R	10:00 wn035/15+wn035/14+wn
541210.encina	maxtuni	amd	JID_284	7537	1	8	--	--	R	14:31 wn037/7+wn037/6+wn03
541211.encina	maxtuni	amd	JID_285	7724	1	8	--	--	R	14:30 wn037/15+wn037/14+wn
541217.encina	maxtuni	amd	JID_287	8671	1	8	--	--	R	13:25 wn038/7+wn038/6+wn03
541221.encina	maxtuni	amd	JID_288	11002	1	8	--	--	R	12:55 wn038/15+wn038/14+wn
541234.encina	solanajr	gtfe	CPSE-Girifalco	14756	1	1	--	--	R	09:58 wn061/1
541238.encina	maru	asna_b3	JID_7559	22682	1	16	--	--	R	09:17 wn033/15+wn033/14+wn
541239.encina	maru	asna_b3	JID_7823	3491	1	16	--	--	R	09:01 wn043/15+wn043/14+wn
541240.encina	maru	asna_b3	JID_7889	17117	1	16	--	--	R	08:53 wn036/15+wn036/14+wn
541241.encina	maru	asna_b3	JID_8087	21891	1	16	--	--	R	08:32 wn046/15+wn046/14+wn
541242.encina	maru	asna_b3	JID_8351	9854	1	16	--	--	R	08:08 wn034/15+wn034/14+wn
541244.encina	jjunquer	citimac	100.0Bohr_rs10.0	22174	1	1	--	--	R	09:12 citimac05/0
541245.encina	jjunquer	citimac	100.0Bohr_rs10.0	22185	1	1	--	--	R	09:12 citimac05/1

Otras wikis

- Hay una shell script específica para mandar trabajos de matlab [wiki:Enviamatlab](#)

Ejemplos

Lanzar un job interactivo

A la hora de hacer pruebas es muy útil abrir una sesión interactiva en un nodo del clúster. Esto se hace mandando un job interactivo. La sesión que se abra durará todo el tiempo que se quiera hasta que no se le mande la instrucción de salida 'exit'. La sintaxis es (la cola asume 'ppn=1'):

```
qsub -I -l nodes=1 -q [queue]
```

NOTA: A la hora de lanzar este tipo de jobs se tiene que ser muy consciente de que se está ocupando un nodo del clúster. Tendría que utilizarse sólo para realizar pruebas que no sean muy largas. Para lanzar jobs muy largos mejor prepararse una script de shell.

Lanzar un job con dependencias

En este caso, no se lanzará una script [attachment:listar.bash?](#) hasta que no termine la espera de 60 segundos (una vez corregido el walltime)

```
[user@ui ~]$ date
Mon Dec 13 17:53:57 CET 2010
[user@ui ~]$ qsub prueba.pbs
307079.ce01.macc.unican.es
[user@ui ~]$ qsub -N listado -q grid -W depend=afterany:307079 -l nodes=1 ./listar.bash
307080.encina
```

La script 'listar.bash' no se ejecutará hasta que termine 'prueba.pbs' con pid 307079. Se sigue:

```
[user@ui ~]$ qstat -nl | grep prueba
307079.encina user    grid    prueba    27714    1  --    --    48:00 R    --    wn002
[user@ui ~]$ qstat -nl | grep listado
307080.encina user    grid    listado    --        1  --    --    48:00 H    --    --
```

El trabajo 307079 está ejecutándose en el nodo wn002, pero el trabajo 307080 está a la espera. Más tarde.

```
[user@ui ~]$ date
Mon Dec 13 17:54:43 CET 2010
[user@ui ~]$ ls
espera.bash  listar.bash  prueba.pbs
[user@ui ~]$ date
Mon Dec 13 17:55:02 CET 2010
[user@ui ~]$ ls
espera.bash      prueba.pbs  listar.bash  prueba.e307079  prueba.o307079
```

El primer job ya ha finalizado. Pasados unos segundos el segundo también termina

```
[user@ui ~]$ date
Mon Dec 13 17:55:12 CET 2010
[user@ui ~]$ ls
espera.bash listado.e307080 listado.o307080 listar.bash prueba.e307079 prueba.o307079 prueba.pbs
[user@ui ~]$ cat prueba.o307079
Hola mundo
[user@ui ~]$ cat prueba.e307079
```

Array de jobs

La cola PBS permite hacer un conjunto de jobs tratados de manera vectorial. A cada uno de estos jobs la cola PBS le asigna un único valor determinado por las directrices del flag `-t [array]`. Esto nos puede resultar útil para ejecutar múltiples veces un programa/aplicación en la que va variando un parámetro. Un ejemplo muy sencillo se puede construir con la función [?genenv](#) de Matlab que nos permite coger variables del entorno. Para ilustrarlo se crea el archivo de matlab [attachment:imprimir_numero.m?](#), el cuál imprime por pantalla unos valores del sistema. Este programa lo ejecutaremos cinco veces independientemente con la siguiente script de colas (llamado matlab_pruebas.pbs):

```
#!/bin/bash
### Job name
#PBS -N MATprueba
### Max run time
#PBS -l walltime=00:01:00
### Queue name
#PBS -q grid
### Job mail
#PBS -m ea
#PBS -M user@unican.es
### Array request
#PBS -t 1-5
### Number of nodes and processors per node
#PBS -l nodes=1:ppn=1

cd ${PBS_O_WORKDIR}

export LD_ASSUME_KERNEL=2.6.18
export PATH=/software/Matlab_R2009a/bin:$PATH
matlab -logfile ${PBS_O_WORKDIR}/log_lluis.err -r "cd ${PBS_O_WORKDIR};imprimir_numero;" -nosplash -nodesktop
```

Lanzamos el job y aparecen:

```
[user@ui ~]$ qstat -nl | grep MAT
308635-1.encina user    grid    MATprueba- 19710    1  --  --  48:00 R  --  wn002
308635-2.encina user    grid    MATprueba- 19731    1  --  --  48:00 R  --  wn002
308635-3.encina user    grid    MATprueba- 1060     1  --  --  48:00 R  --  wn003
308635-4.encina user    grid    MATprueba- 1079     1  --  --  48:00 R  --  wn003
308635-5.encina user    grid    MATprueba- 23120    1  --  --  48:00 R  --  wn004
```

Una vez terminado aparecen los siguientes ficheros en el directorio de trabajo:

```
[user@ui ~]$ ls -l
-rw-r--r-- 1 user gmeteo 1030 Dec 14 17:43 log_user.err
-rwxr-xr-x 1 user gmeteo  526 Dec 14 17:42 matlab_prueba.pbs
-rw----- 1 user gmeteo  374 Dec 14 17:43 MATprueba.e308635-1
-rw----- 1 user gmeteo  374 Dec 14 17:43 MATprueba.e308635-2
-rw----- 1 user gmeteo  374 Dec 14 17:43 MATprueba.e308635-3
-rw----- 1 user gmeteo  374 Dec 14 17:43 MATprueba.e308635-4
-rw----- 1 user gmeteo  374 Dec 14 17:43 MATprueba.e308635-5
-rw----- 1 user gmeteo 1121 Dec 14 17:43 MATprueba.o308635-1
-rw----- 1 user gmeteo 1121 Dec 14 17:43 MATprueba.o308635-2
-rw----- 1 user gmeteo 1121 Dec 14 17:43 MATprueba.o308635-3
-rw----- 1 user gmeteo 1121 Dec 14 17:43 MATprueba.o308635-4
-rw----- 1 user gmeteo 1121 Dec 14 17:43 MATprueba.o308635-5
```

Aparecen los ficheros de salida de los cinco jobs. Miramos el contenido del tercero:

```
[user@ui ~]$ cat MATprueba.e308635-3
which: no shopt in (/software/Matlab_R2009a/bin:/software/CentOS/5.2/netcdf/4.1.1/gcc-gfortran4.1.2/bin:/opt/d-cache/srm/bin:/opt/d-cache/dcap/bin:/opt/edg/bin:/opt/glite/bin:/opt/globus/bin:/opt/lcg/bin:/bin:/usr/bin:/ocean/o/gmeteo/users/lluis/bin)
[user@ui ~]$ cat MATprueba.o308635-3
Warning: No display specified.  You will not be able to display graphics on the screen.
Warning: No window system found.  Java option 'MWT' ignored

      < M A T L A B (R) >
      Copyright 1984-2009 The MathWorks, Inc.
      Version 7.8.0.347 (R2009a) 64-bit (glnxa64)
      February 12, 2009
```


To get started, type one of these: helpwin, helpdesk, or demo.
For product information, visit www.mathworks.com.

```
shellpath =
```

```
/software/Matlab_R2009a/bin:/software/CentOS/5.2/netcdf/4.1.1/gcc-gfortran4.1.2/bin:/opt/d-cache/srm/bin:/opt/d-cache/dcap/bin:/opt/edg/bin:/opt/glite/bin:/opt/globus/bin:/opt/lcg/bin:/bin:/usr/bin:/oceanogmeteo/users/lluis/bin
```

```
Este es el path: /software/Matlab_R2009a/bin:/software/CentOS/5.2/netcdf/4.1.1/gcc-gfortran4.1.2/bin:/opt/d-cache/srm/bin:/opt/d-cache/dcap/bin:/opt/edg/bin:/opt/glite/bin:/opt/globus/bin:/opt/lcg/bin:/bin:/usr/bin:/oceanogmeteo/users/lluis/bin
```

```
pbsjob =
```

```
308635-3.encina
```

```
Esta es la realizacion: "308635-3.encina"
```

```
num =
```

```
3
```

```
num =
```

```
3
```

```
Este es el numero: 3
```

Mientras que en el correo (llegando por orden de finalización), me aparecen 5 correos con el título:PBS JOB 308635-[N].ce01.macc.unican.es siendo [N]=(1,5). El contenido del correo #3

```
PBS Job Id: 308635-3.encina
Job Name: MATprueba-3
Exec host: wn003.macc.unican.es/0
Execution terminated
Exit_status=0
resources_used.cput=00:00:02
resources_used.mem=0kb
resources_used.vmem=0kb
resources_used.walltime=00:00:18
```

Trabajo con variable

Tomamos el job sencillo de ejemplo. En este caso el tiempo de espera para la escript se lo pasaremos como variable del job.

```
#!/bin/bash
### Job name
#PBS -N prueba
### Max run time
#PBS -l walltime=00:00:10
### Queue name
#PBS -q grid
### Number of nodes and processors per node
#PBS -l nodes=1:ppn=1

cd ${PBS_O_WORKDIR}
```

```
./espera.bash ${waitTIME}
```

Lanzando el job:

```
[user@ui ~]$ qsub -v waitTIME='60' prueba.pbs
```

Monitorización del clúster

Por entorno web está instalado el [?ganglia](#). En este entorno se muestra toda la actividad de los nodos (consumo de cpu, memoria, discos duros, etc...) con una interfaz gráfica.

Recomendaciones

- Cualquier trabajo que implique manejo de datos, archivos, etc... se tiene que hacer mediante un job de colas.
- **mar SOLO sirve para mandar trabajos a la cola**
- No utilizar jobs interactivos para tareas muy largas. Cuanto mas larga se la tarea mas peligro que se termine por algún problema del nodo
- Realizar jobs con *checking points*, que en caso de caída del nodo no se tenga que volver a empezar desde el principio
- Ajustar el *walltime* de los trabajos y aprender a reconocer los requerimientos de los trabajos
- Aprender a hacer shells y mandarlas a las colas. Se gana en eficiencia y permite poder recuperar tareas.
- El clúster es un recurso compartido, actúa en consecuencia
- Es muy distinto que se cuelgue un nodo a que se cuelgue una máquina que dé servicio a tod@s
- Trabajar masivamente en el directorio `/oceano/gmeteo/WORK`, dejar en el HOME, `/oceano/gmeteo/users/[usuario]` sólo el material sensible (scripts, programas, etc...)