Table of Contents

Información actualizada en la ?wiki del cluster del Sdel

Submitting machine 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/[usario]`: 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**.

¿Cómo 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 software disponible:

```
[user@ui ~]$ use

Programs available for "use":

cdo                             gmt
grib_api                        grib_api1.10
hdf5                            intel
intel11                         java
matlab2009a                     matlab2013a
```

```
nco                             ncview
netcdf                          netcdfintel
netcdf_viejo                    openmpi14intel
openmpi18intel                  pgi
p_interp                        python
python2.7.2                     python2.7.2b
```

- Configuración de un software especifico:

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

Sistema de colas TORQUE/PBS

El cluster SMG dispone de un sistema de colas ?TORQUE/PBS para la gestión de jobs.

Comandos

Hay distintos comandos para gestionar los jobs de un usuario, para mas inforamción sbre ellos abrir el manual con la instrucción `man [comando]`:

- **?qsub [archivo]** envio del job [archivo].pbs a la cola
- **?qdel [jobid]** terminar forzosamente el job con id [jobid]
  ?qstat permite ver el estado de los jobs gestionados. Tiene también distintos flags (se muestran los mas usuales):
  - *-q* muestra las colas disponibles y sus principaes caracteristicas.
  - *-n1* muestra todos los nodos a los cuales se ha mandado un job.
  - *-u [usuario]* muestra todos los jobs de [usuario].
- **?qalter** permite modifocar parámetros de los jobs, pero `SOLO` cuando se encuentran en el estado **Q** (trabajo encolado).

pbsnodes: Componentes del clúster

El comando `pbsnodes` nos permite examinar los nodos que se encuentran en el clúster, sus características y cuál es su estado:

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

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).
- **physmem**: memoria total de la unidad.

Ejemplo de uso:

```
[user@ui ~]$ pbsnodes wn002
wn002
    state = job-exclusive
    np = 2
    properties = dell,grid
    ntype = cluster
    jobs = 0/308594.encina, 1/encina
    status = opsys=linux,uname=Linux wn002 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, physmem=2057808kb, ncpus=2, loadave=1.03, netload=2685739215248,state=free, jobs=308594.ce01.
macc.unican.es  308606.encina, varattr=, rectime=1292346952
```

El nodo `wn002` (con 2 cpus y 2GB de memoria y está totalmente ocupada con dos trabajos (`308594.encina` y `308606.encina`) de un mismo usuario.

Jobs

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

- **número de processos:** 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 usarios distintos.
- **wall-time:** determina el tiempo de ejecución del job dentro del clúster. Una vez pasado este tiempo el job y su aplicación seran detenidos forzosamente. Es un muy común que en los clústers se ejectuen antes los jobs con un wall-time pequeño que con uno de grande.

Los requerimientos de un job en una cola PBS

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

Las peticiones se hacen por un conjunto de *flags*. El envio 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 circumstancias 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:

```
#!/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}
use [software]

[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 associado 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.

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 habra 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 -n1 | grep prueba
307079.encina user    grid     prueba      27714     1  --    --  48:00 R   --    wn002
[user@ui ~]$ qstat -n1 | grep listado
307080.encina user    grid     listado       --      1  --    --  48:00 H   --     --
```

El trabajo `307079` está ejecutándose en le 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 ja 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
```

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
```

Otras wikis

- Hay una shell script específica para mandar trabajos de matlab wiki:Enviamatlab

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, disocs 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...)