Wikiprint Book

Title: ColasPBS

Subject: TracMeteo - ColasPBS

Version: 85

Date: 01/24/2022 08:27:56 AM

Table of Contents

- Información actualizada en la ?wiki del cluster del SdeI
- HardwareCluster
- ChangePasswd

Conexión al cluster SMG

El acceso al cluster se debe hacer realizarse mediante la maquina `ui.macc.unican.es` usando ?SSH:

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

O bien con la aplicación ?putty desde Windows.

Trasferencia de ficheros

La transferencia de ficheros al cluster se puede realizarse desde una maquina remota a través de la maquina `ui.macc.unican.es`:

```
[user@localmachine]$ scp file user@ui.macc.unican.es:file
```

O directamente desde `ui.macc.unican.es`:

```
[user@ui]$ scp user@remote_machine:file file
```

Si la copia de ficheros implica ficheros superiores a Mega-bytes, es recomendable el uso de uno nodo del cluster mediante un job interactivo.

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**.

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

```
test -f /software/meteo/use/load_use && source /software/meteo/use/load_use
```

en el fichero `.bashrc`, disponible en cada home de usuario (`~/.bashrc`), para no repetir su uso. Por ejemplo:

```
echo "test -f /software/meteo/use/load_use && source /software/meteo/use/load_use" >> ~/.bashrc
```

- 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 modificar parámetros de los jobs, pero `SOLO` cuando se encuentran en el estado **Q** (trabajo encolado).

  ?pbsnodes examina 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]
```

  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

A continuación se muestra un aplantilla para la ejecución de jobs:

```
#!/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 directivas PBS/TORQUE:

- **-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
- **-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 utilizar los editores `nano` y/o `vi` :

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

Y para su envío se utilizaría el comando `qsub` seguido del script:

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

O haciendo uso de los los *flags* del comando:

```
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]
```

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]
```

A la hora de lanzar este tipo de jobs se tiene que ser muy consciente de que se está ocupando un nodo del clúster.

Lanzar un job con dependencias

En este caso, no se lanzará una script listar.bash? hasta que no termine la espera de 60 segundos:

```
[user@ui ~]$ date
Mon Dec 13 17:53:57 CET 2010
[user@ui ~]$ qsub prueba.pbs
307079.encina
[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 identificador 307079.

```
[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.

```
[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 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.
- La maquina **ui.macc.unican.es** SOLO sirve para mandar trabajos al sistema de colas.
- No utilizar jobs interactivos para tareas muy largas.
- 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.
- Trabajar masivamente en el directorio `/oceano/gmeteo/WORK`, dejar en el `HOME, /oceano/gmeteo/users/[usuario]` sólo el material sensible (scripts, programas, etc...).