Table of Contents

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.

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, physmem, 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)
- physmem: memoria total de la unidad

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

```
[lluis@mar CursilloPBS]$ pbsnodes
wn001.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 wn001.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, physmem=2057808kb, ncpus=2, loadave=1.03, netload=2685739215248,state=free, jobs=308594.ce01.
macc.unican.es  308606.ce01.macc.unican.es, varattr=, rectime=1292346952
```

La unidad `wn001.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.ce01.macc.unican.es` y `308606.ce01.macc.unican.es`) 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.el5xen #1 SMP Tue Nov 9 13:35:30 EST
2010 x86_64,sessions=? 0,nsessions=? 0, nusers=0, idletime=111779, totmem=4866040kb, availmem=4694372kb, physme
m=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', 'lcgpro', 'estadistica') 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.ce01.macc.unican.es, 3/308629.
ce01.macc.unican.es, 4/308629.ce01.macc.unican.es, 5/308629.ce01.macc.unican.es, 6/308629.ce01.macc.unican.es,
7/308629.ce01.macc.unican.es, 8/308630.ce01.macc.unican.es, 9/308630.ce01.macc.unican.es, 10/308630.ce01.macc.
unican.es, 11/308630.ce01.macc.unican.es, 12/308630.ce01.macc.unican.es, 13/308630.ce01.macc.unican.es, 14/3086
30.ce01.macc.unican.es, 15/308630.ce01.macc.unican.es
    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.ce01.macc.unican.es` está en las cpus de la 0 a 7 y el job `308630.ce01.macc.unican.es` de la 8 a la 15) de un único usuario.

Job

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.

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

Infraestructura del GMS

La infraestructura del *Grupo de Meteorología de Santander* (GMS) es la siguiente (diciembre 2010).

NOTA IMPORTANTE: la ejecución de jobs se hace sólo desde la máquina llamada *mar* (`ssh mar`) El clúster está formado por distintas máquinas. Los únicos sitios que son accesibles des de todos los ordenadores son los que se encuentran desde `/oceano/gmeteo`

Almacenamiento: visible desde todos los nodos con:

- `/oceano/gmeteo/users/[usario]`: *HOME* del usuario llamado [usuario]
- `/oceano/gmeteo/DATA`: almacenamiento de todos los datos (observaciones, GCMs, RCMs, ...)
- `/oceano/gmeteo/WORK`: directorio de trabajo

  `/software`: directorio con todas las aplicaciones. Por ejemplo:
  - /software/Matlab_R2009a/bin/matlab
  - /software/CentOS/5.2/R/2.11.1/bin/R
  - ejectuando `source /software/ScientificLinux/4.6/etc/bashrc` se exporta en memoria el acceso (y su configuración) de muchas aplicaciones instaladas en `/software`

colas (no accesibles para tod@s, diciembre 2010):

- `grid`: nodos `[001]-[009]`
- `estadistica`: nodos `[010]-[015]`
- `dinamica`: nodos `[012]-[025]`
- `blade`: nodos `[031]-[034]`
- `oper`: nodos `[042]`
- `puerca`: nodos `[041], [043], [044], [035], [036], [045], [046]`

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 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 (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` (más intuitivo) y/o `vi` (otro mundo):

```
[lluis@mar]$ nano job.pbs
[lluis@mar]$ vi job.pbs
```

El job se mandaría a la cola con la instrucción:

```
[lluis@mar]$ 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].ce01.macc.unican.es. 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.

Comandos cola PBS

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

- **qsub [archivo].pbs** 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):
  - *-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 mestra 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` (recortada y maquetada)

```
[lluis@mar CursilloPBS]$ qstat -n1
ce01.macc.unican.es:
                                                            Req'd  Req'd   Elap
Job ID               Username Queue    Jobname    SessID NDS   TSK Memory Time  S Time
-------------------- -------- -------- ---------- ------ ----- --- ------ ----- - -----
303117.ce01.macc.uni lluis    blade    scne2a      10618     1  --    --  24:00 R 03:31  wn033+wn033+wn033+wn033
3+wn033+wn033+wn033+wn033+wn033+wn033+wn033+wn033+wn033+wn033+wn033+wn033
303118.ce01.macc.uni lluis    blade    scne2a        --      1  --    --  24:00 H   --     --
303240.ce01.macc.uni lluis    dinamica scne5a      22145     1  --    --  24:00 R 07:43   wn023+wn023+wn023+wn0
23+wn023+wn023+wn023+wn023
303241.ce01.macc.uni lluis    dinamica scne5a        --      1  --    --  24:00 H   --     --
303242.ce01.macc.uni lluis    dinamica scne5a        --      1  --    --  24:00 H   --     --
304465.ce01.macc.uni lluis    dinamica scne3a      21060     2  --    --  24:00 R 01:25   wn013+wn013+wn013+wn0
13+wn013+wn013+wn013+wn013+wn018+wn018+wn018+wn018+wn018+wn018+wn018+wn018
306291.ce01.macc.uni lluis    blade    scne1b       8739     1  --    --  24:00 R 02:43   wn032+wn032+wn032+wn0
32+wn032+wn032+wn032+wn032+wn032+wn032+wn032+wn032+wn032+wn032+wn032+wn032
306292.ce01.macc.uni lluis    blade    scne1b        --      1  --    --  24:00 H   --     --
307057.ce01.macc.uni sixto    estadist papers      15343     1  --    --    --  R 28:20   wn012
307059.ce01.macc.uni sixto    estadist papers      24344     1  --    --    --  R 28:15   wn010
307060.ce01.macc.uni sixto    estadist papers      17582     1  --    --    --  R 28:13   wn012
307591.ce01.macc.uni sixto    estadist papers      24762     1  --    --    --  R 26:32   wn010
308550.ce01.macc.uni lluis    dinamica postWRF6    12269     1  --    --    --  R 11:45   wn025+wn025+wn025+wn02
5+wn025+wn025+wn025+wn025
308594.ce01.macc.uni markel   grid     STDIN       10248    --   --    --  48:00 R 02:32   wn001
308629.ce01.macc.uni markel   oper     oper_gfs20  25810     1  --    --    --  R 03:31   wn042+wn042+wn042+wn04
2+wn042+wn042+wn042+wn042
308630.ce01.macc.uni markel   oper     oper_gfs20  25872     1  --    --    --  R 03:30   wn042+wn042+wn042+wn04
2+wn042+wn042+wn042+wn042
308631.ce01.macc.uni markel   grid     post_opera    --     --   --    --  48:00 H   --     --
```

Otras wikis

- Hay una shell script específica para mandar trabajos de matlab wiki:Enviamatlab
- Hay una breve introducción al shell scripting en attachment:dgms_2009_1.pdf?

ejemplos

Wall time insuficiente

Tomamos un job sencillo de ejemplo escrito en un fichero llamado attachment:prueba.pbs?. Este job lanza el script attachment:espera.bash? el cual se espera los segundos dados por el primer argumento y escribe por pantalla el mensaje "Hola mundo".

```
#!/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 60
```

Lanzando el job obtendríamos:

```
[lluis@mar CursilloPBS]$ qsub prueba.pbs
307065.ce01.macc.unican.es
[lluis@mar CursilloPBS]$ qstat | grep prueba
307065.ce01               prueba           lluis                   0 R grid
[lluis@mar CursilloPBS]$ qstat -n1 | grep prueba
307065.ce01.macc.uni lluis    grid     prueba      23057     1  --    --  48:00 R   --    wn001
[lluis@mar CursilloPBS]$ ls
prueba.pbs
```

Pasados los 60 segundos...

```
[lluis@mar CursilloPBS]$ ls
prueba.e307065  prueba.o307065  prueba.pbs
```

La ejecución del job ha generado dos ficheros de salida en los cuales se almacena información de la ejecución del job y de la propia aplicación (ficheros `prueba.[e/o]307065`). Al mirar el contenido de estos ficheros:

```
[lluis@mar CursilloPBS]$ cat prueba.o307065
[lluis@mar CursilloPBS]$ cat prueba.e307065
=>> PBS: job killed: walltime 43 exceeded limit 10
```

El tiempo dado al job ha diso insuficiente.

Memoria insuficiente

Para evitar generar problemas en los nodos por exceso de uso de memoria se tiene el flag 'mem'. A modo de ejemplo se tiene un programa de Fortran que ocupa en memoria 160 MB. Mandamos un job con 150MB de límite de memoria.

```
#!/bin/bash
### Job name
#PBS -N prueba
### Max memory
#PBS -l mem=150mb
### Queue name
#PBS -q grid
### Number of nodes and processors per node
#PBS -l nodes=1:ppn=1

cd ${PBS_O_WORKDIR}

./Fortran.out
```

El job se lanza a la cola, y pasados unos segundos nos sale este mensaje en el fichero de salida 'prueba.e477378':

```
[lluis@mar Fortran]$ cat prueba.e477378
```

De momento (30 Sept. 2011) no sale mensaje de error. Se está trabajando para que mande algún mensaje

Ocupar toda la memoria de un nodo con un único job

Se pueden tener aplicaciones que requieran mucha memoria. Para evitar que el job se mande en un nodo donde tiene parte de la memoria ocupada, y/o para no entorpecer los otros jobs, es posbile quedarse con toda la memoria de un nodo. Para hacer esto nos tenemos que asegurar que no entre ningún otro job al nodo. Para hacer esto, mandaremos el job requiriendo todas las unidades de cálculo del nodo. Así, si queremos toda la memoria de los nodos de la cola `'dinamica'` (con 8 cpus), requeriríamos:

```
#PBS -l nodes=1:ppn=8
```

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)

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

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

```
[lluis@mar CursilloPBS]$ qstat -n1 | grep prueba
307079.ce01.macc.uni lluis    grid     prueba      27714     1  --    --  48:00 R   --    wn001
[lluis@mar CursilloPBS]$ qstat -n1 | grep listado
307080.ce01.macc.uni lluis    grid     listado       --      1  --    --  48:00 H   --     --
```

El trabajo `307079` está ejecutándose en la unidad `wn001`, pero el trabajo `307080` está a la espera. Más tarde.

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

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

```
[lluis@mar CursilloPBS]$ date
Mon Dec 13 17:55:12 CET 2010
[lluis@mar CursilloPBS]$ ls
espera.bash listado.e307080 listado.o307080 listar.bash prueba.e307079 prueba.o307079 prueba.pbs
[lluis@mar CursilloPBS]$ cat prueba.o307079
Hola mundo
[lluis@mar CursilloPBS]$ 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 assigna 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 lluis.fita@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:

```
[lluis@mar CursilloPBS]$ qstat -n1 | grep MAT
308635-1.ce01.macc.u lluis    grid     MATprueba-  19710     1  --    --  48:00 R   --    wn002
308635-2.ce01.macc.u lluis    grid     MATprueba-  19731     1  --    --  48:00 R   --    wn002
308635-3.ce01.macc.u lluis    grid     MATprueba-   1060     1  --    --  48:00 R   --    wn003
308635-4.ce01.macc.u lluis    grid     MATprueba-   1079     1  --    --  48:00 R   --    wn003
308635-5.ce01.macc.u lluis    grid     MATprueba-  23120     1  --    --  48:00 R   --    wn004
```

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

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

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

```
[lluis@mar CursilloPBS]$ 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-ca
che/srm/bin:/opt/d-cache/dcap/bin:/opt/edg/bin:/opt/glite/bin:/opt/globus/bin:/opt/lcg/bin:/bin:/usr/bin:/oceano/
gmeteo/users/lluis/bin)
[lluis@mar CursilloPBS]$ 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-ca
che/dcap/bin:/opt/edg/bin:/opt/glite/bin:/opt/globus/bin:/opt/lcg/bin:/bin:/usr/bin:/oceano/gmeteo/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:/oceano/gmete
o/users/lluis/bin

pbsjob =

308635-3.ce01.macc.unican.es

Esta es la realizacion: "308635-3.ce01.macc.unican.es"

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.ce01.macc.unican.es
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:

```
[lluis@mar CursilloPBS]$ 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, 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...)

Apendice

Componentes de un clúster

Un clúster es una agrupación de unidades de computación el cual permite trabajar con unidades individuales o cualquier subconjunto de ellas. Las unidades de computación pueden ser de muchos tipos y tener muchas características distintas, por lo general con una unidad de cálculo (*core*) y una memoria (particular del core o compartida con otros). Un clúster suele tener much@s usuari@s que querrán ejecutar distintas aplicaciones. Un gestor de colas es el software encargado de manejar las aplicaciones que se quieren ejecutar en un determinado clúster.

Las unidades básicas que constituyen un clúster son:

- **nodos:** computadoras que contienen una estructura parecida a un ordenador común. En estas máquinas un conjunto de cpus se conectan por la placa base sin necesidad de una red externa. Además tienen memória y un disco duro propio.

  cpu: unidad de cáclulo. El nombre es ambiguo puesto que no tiene que coincidir exactamente con una cpu física. Ya que ho en día hay cpus que pueden contener 2, 4, 8, 16 o más cores (estos cores pueden ser reales o que permitan 'multithreading'). Así que cpu es la unidad mínima de cálculo independientemente de su condición. Así que la situación física de los nodos es:
  - **nodos viejos:** 1 cpu con dos cores
  - **nodos menos viejos:** 26 nodos de 2 cpus con 4 cores
  - **nodos nuevos:** 12 nodos de 2 cpus con 8 cores
- **switch:** unidad de interconnexión entre nodos. Permite ver todas las cpus de distintos nodos como una única máquina.

  unidad de almacenamiento: máquina constituida por un conjunto de discos duros. Suele tener tres espacios básicos:
  - `HOME`: Es un directorio particular de cada usuario al cual se accede al iniciar una sesión/job en el clúster
  - datos: datos necesarios para trabajar con las aplicaciones de los usuarios del clúster
  - trabajo: espacio en donde se almacenan los resultados de los jobs
  - Lo mas común es que todas las unidades de almacenamiento sean visibles des de todos los nodos de un clúster

Gestor de colas

Para que un clúster funcione adecuadamente para tod@s us usuari@s y el gestor de colas pueda hacer su trabajo, l@s usuari@s sólo tienen que interactuar con el gestor por medio de *jobs*. Estos *jobs* son peticiones de recursos del clúster (número de cores, tiempo de ejecución...) necesarios poder lanzar la aplicación del usuari@. El gestor de colas será el encargado de permitir la ejecución de la petición si hay suficientes recursos disponibles en el clúster. En caso contrario, el trabajo se quedará *encolado*, a la espera de que haya los recursos suficientes requeridos en el *job*. De esta manera, todas las peticiones de l@s usuari@s son centralizadas y manejadas automáticamente por el gestor de colas. Este gestor de colas puede a su vez, estar siguiendo unas directrices de prioridades de ejecución determinadas por otros softwares buscando maximizar el rendimiento del clúster.

El sistema de colas del clúster del *Grupo de Meteorología de Santander* (Diciembre 2010) es el sistema de colas PBS (*Portable Batch System*).

Otros ejemplos

Matar todos los jobs

Si se quiere matar todos los jobs que cumplan un requisito (en el ejemplo un nombre de usuario), se podría utilizar el `qstat`:

```
[lluis@mar $] qdel `qstat | grep lluis | awk '{print $1}'`
```

Se ha explicado que el '`qdel [jobid]`' finaliza el job con número '`[jobid]`'. La instrucción '`qstat | grep lluis`' sólo listará ls jobs que contengan la palabra '`lluis`'. Al añadir al final otra instrucción '`| awk '{print $1}'`', sólo se lista el primer valor (que coincide con el número de job). Así que '`qdel`', va a ejecutarse con la salida de todo lo que está entre ` `, es decir, todos los número de jobs listados con '`qstat`' que contengan la palabra '`lluis`'.

Mandar el job a un nodo en concreto

Si se quiere mandar el job a un nodo en concreto se substituye la línea `'PBS -l nodes=1:ppn=1'` por:

```
#PBS -l nodes=wn[nnn].macc.unican.es
```

Donde [nnn] es el nodo requerido. Este job carece un poco de sentido, puesto que si se hacen las cosas bien, el trabajo tendría que ser ejecutable igualmente en cualquier nodo.

APENDICE: Correos

Correo 30 septiembre 2011: Requerimientos de memoria trabjos en la cola 'estadistica'

Buenos días,

Esto es una actualización del correo que os mandé hace una semana...

Últimamente hemos tenido problemas con el clúster. Estos problemas acaecían por jobs que superaban la capacidad de memoria de los nodos. Después de mirárnoslo, hemos encontrado la manera para evitarlo. Se trata de mandar los jobs al clúster (envia_matlab, envia_R) con un requerimiento de memória incluído. Con este requerimiento de memoria se establecen dos cosas en los jobs:

1.- Los trabajos que se manden sólo se van a ejecutar si ecuentran un nodo que tenga disponible la memoria requerida 2.- Si algún elemento del trabajo mandado supera este límite de memoria, la ejecución del job se parará.

Para hacer este requerimiento se tiene que añadir a la instrucción 'qsub' -l mem=[cantidad_de_memoria kb/mb/gb/tb]

o si es en un script de pbs #PBS mem=[cantidad_de_memoria kb/mb/gb/tb]

La [cantidad_de_memoria] tiene que ser un número entero

Cada job irá reservando memoira del nodo. Cuando un trabajo no encuentre suficiente memoria se quedará a la espera hasta que la consiga. Esto puede hacer que por ejemplo con un nodo con 8 cores y 8 GB de memoria, sólo pueden correr en el 3 jobs de 2250mb, un cuarto ya no entraría. Así quedarían 5 cores libres que no harían nada (a no ser que le mandáramos 5 jobs que no llegasen a consumir los 250mb restantes)

Recordad que tenéis a vuestra disposición una wiki con toda esta información

?https://www.meteo.unican.es/trac/meteo/wiki/ColasPBS

La cola 'estadística' tiene por defecto un valor de 'mem=750mb'. Si tenéis que mandar un trabajo al clúster que sepáis que vaya a ocupar mucha memoria, por favor, augmentad el valor de 'mem' a lo que estimeis, pero nunca superando los '7gb', ya que los nodos de la cola estadística sólo tienen 8gb de memoria (el sistema ocupa una parte).

Por ahora si el job muere por exceso de memoria no habrá manera de saberlo, puesto que de momento no sabemos como hacer que el sistema mande un mensaje diciendo algo. Estamos en ello.

Hasta ahora,

Lluís