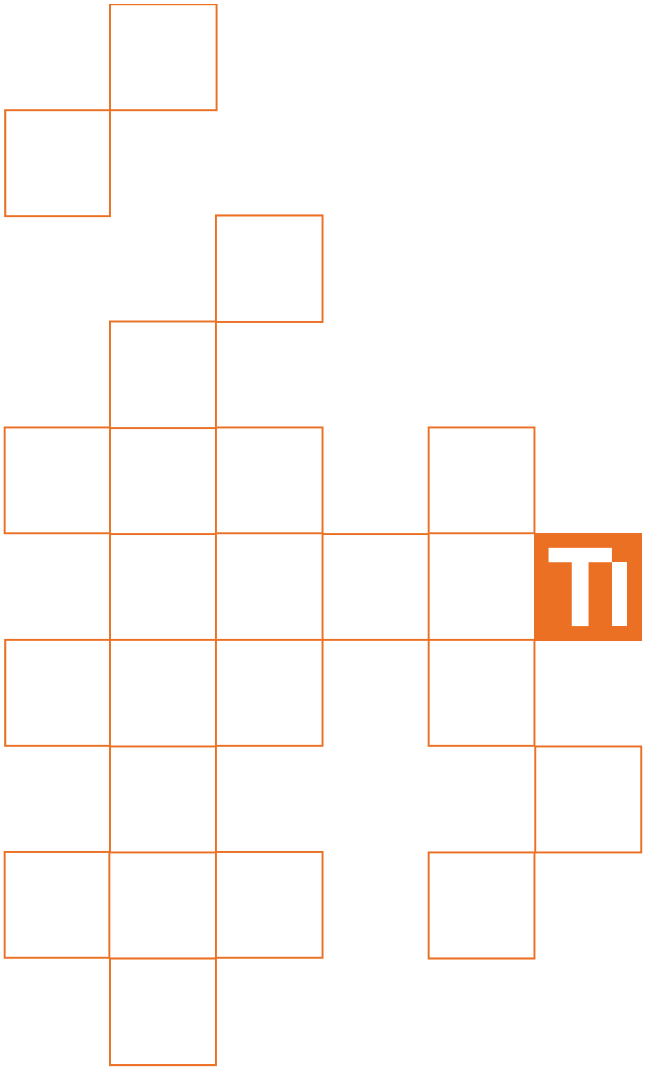


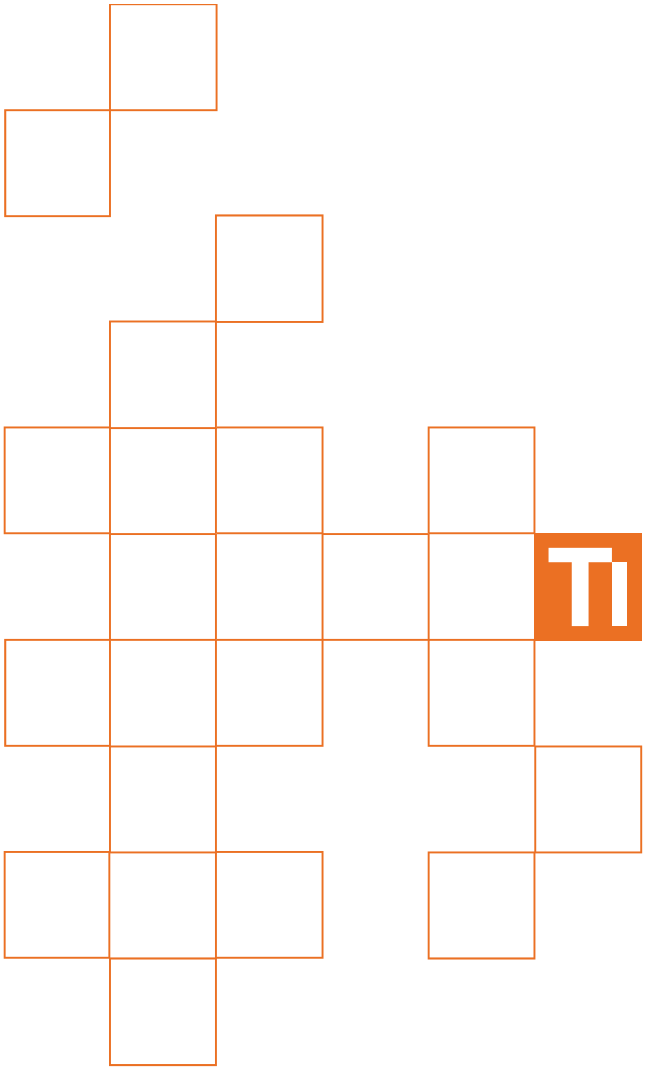
CiTIUS

Centro Singular de Investigación
en **Tecnoloxías** da
Información

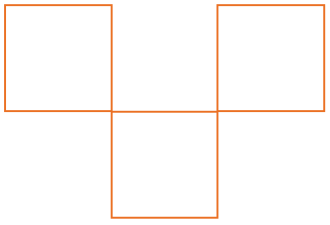
Guía de usuario del clúster `ctcomp2`

Diego R. Martínez

v1.2 [Septiembre 2013]



Contenidos



Introducción

El clúster `ctcomp2` es un sistema de computación de altas prestaciones (CAP) que proporciona a los usuarios del CITIUS la posibilidad de ejecutar programas computacionalmente exigentes.

En su forma más simple, la utilización de los recursos computacionales de este clúster se puede resumir en los siguientes pasos:

1. Acceder, mediante `ssh`, al clúster. Los ficheros necesarios para la ejecución de los trabajos se pueden importar al espacio de usuario del clúster mediante `scp`.

2. Si utilizamos lenguajes compilados (C, C++, Fortran...) es necesario compilar adecuadamente el código fuente.

3. Escribir un script que indique al gestor de colas los parámetros de la ejecución: el directorio de trabajo, el comando de ejecución, el número de nodos/núcleos necesarios....

4. Enviar, mediante `qsub`, el script al gestor de colas para que lo incluya en la cola de planificación de trabajos, a la espera de ser emitido para su ejecución en los nodos computacionales.

5. La emisión del trabajo a los nodos computacionales se realizará cuando el gestor de colas encuentre un hueco temporal y espacial adecuado para la ejecución del trabajo, en función de los parámetros de ejecución indicados en el script. Es posible utilizar una opción en el script para indicar una dirección de correo electrónico en la que se notificará la finalización de las tareas indicadas en el script.

Descripcion del sistema

El clúster `ctcomp2` es un clúster heterogéneo, formado por 8 nodos computacionales HP Proliant BL685c G7, 5 nodos Dell PowerEdge M910 y 5 nodos Dell PowerEdge M620. Cada nodo HP Proliant dispone de 4 procesadores AMD Opteron 6262 HE (16 núcleos) y 128 GB de RAM. Cada nodo Dell PowerEdge M910 está equipado con 2 procesadores Intel Xeon L7555 (8 núcleos, 16 hilos) y 64 GB de RAM. Cada nodo Dell PowerEdge M620 está equipado con 2 procesadores Intel Xeon E5-2650L (8 núcleos, 16 hilos) y 64 GB de RAM. Los nodos compationales están conectados entre sí a través de varias redes 10 GbE.

El acceso de los usuarios al clúster se realiza a través de una máquina independiente, que denominamos `frontend`. Este `frontend` tiene unas

¡!

prestaciones limitadas, y únicamente está preparado para gestionar ficheros, compilar código y enviar trabajos al sistema de colas del clúster. No está permitida la ejecución de código paralelo en esta máquina.

El usuario dispone de los siguientes espacios, dentro del sistema de ficheros del clúster, para ubicar los ficheros relacionados con los trabajos realizados en clúster:

— `/home/local/nome.apelido/`
Este directorio es el `$HOME` del usuario `<nome.apelidos>` y es accesible en todos los nodos del clúster, incluido en el `frontend`. Por defecto, será el directorio de referencia en las ejecuciones de los códigos en los nodos computacionales. La cuota de usuario es limitada, por lo que los usuarios que necesiten gestionar ficheros muy grandes deberán hacerlo a través del directorio `/sfs/`.[1]

— `/sfs/`
Este directorio también está compartido entre todos los nodos de computación y el `frontend`. Este directorio podrá utilizarse como espacio auxiliar durante la ejecución de trabajos, para el almacenamiento de ficheros temporales que deban estar accesibles en todos los nodos o para almacenar los ficheros resultado de una ejecución. En este directorio no se garantiza la conservación permanente de los ficheros que no hayan sido accedidos en los últimos 30 días. Se recomienda utilizar nombres de ficheros/directorios que identifiquen claramente al binomio usuario/programa, para evitar potenciales conflictos entre usuarios.

— `/scratch/`
Cada nodo computacional del clúster dispone de un directorio *scratch* local que puede ser utilizado para almacenamiento temporal local durante la ejecución de una tarea. El contenido de un directorio *scratch* no es visible desde el resto de nodos. El contenido de este directorio no estará accesible desde el "frontend", por lo que no es un lugar adecuado para guardar ficheros con resultados. En cualquier caso, no se garantiza la conservación de ficheros de manera permanente en este directorio. Se recomienda utilizar nombres de ficheros/directorios que identifiquen claramente al binomio usuario/programa, para evitar potenciales conflictos entre usuarios.

[1] En cualquier caso, los usuarios con necesidades de almacenamiento singulares deberán solicitar esos requerimientos a través del sistema de incidencias del CITIUS.

1 Acceso al clúster ctcomp2

Los servicios del clúster son accesibles a través del frontend (ctcomp2.inv.usc.es). El acceso remoto a este servidor se realiza mediante ssh (*secure shell*). Se utiliza el sistema de autentificación centralizado del CITIUS.

ssh
```
local$ ssh [-X] [-p<port>] nome.apelido@ctcomp2.inv.usc.es
Password:
ct$
```

Los argumentos de este comando son:

`-p<port>` (*opcional*) Indica el puerto por el cual se conecta el comando scp.

`-X` (*opcional*) Activa la redirección de X. Es un requisito imprescindible para ejecutar aplicación que requieren modo gráfico.

1.1 Importación/exportación

El espacio de usuario del clúster es independiente de otros sistemas en el CITIUS, por lo que es necesario importar al espacio de usuario del clúster todos los ficheros necesarios para la ejecución de nuestros programas (por ejemplo, el código fuente o los ficheros de entrada del programa). El comando scp permite el intercambio de ficheros con otros sistemas conectados en red. La sintaxis del comando scp es la siguiente:

scp
```
scp [-P<port>] [-r] <direccion_origen> <direccion_copia>
```

Los argumentos de este comando son:

`-P<port>` (*opcional*) Indica el puerto por el cual se conecta el comando scp.

`-r` (*opcional*) Es un argumento que se utiliza cuando direccion_origen es un directorio, e indica que se debe copiar de manera recursiva el contenido del directorio.

`<direccion_origen>` Indica la ruta completa del fichero/directorio que se copiará.

`<direccion_copia>` Indica la ruta completa donde queremos ubicar la copia del fichero/directorio.

Ejemplos `scp` en `ctcomp2`

A modo de ejemplo, se muestran varios ejemplos de importación de ficheros. En estos ejemplos, se supone que el puesto de trabajo habitual del usuario `nome.apelido` es `ctXXX.inv.usc.es`. El fichero/directorio que queremos importar está situado en `ctXXX.inv.usc.es`, en el directorio `/datos/work/`, y queremos hacer una copia en el espacio de usuario del clúster en el directorio `work` del `$HOME` del usuario.

Si ejecutamos `scp` en el propio clúster:

scp (@ctcomp2)

```
ct$ scp -P<port> \
  nome.apelido@ctXXX.inv.usc.es:/datos/work/un.fichero \
  ~/work/
ct$ scp -P<port> -r \
  nome.apelido@ctXXX.inv.usc.es:/datos/work/directorio/ \
  ~/work/
```

Si ejecutamos `scp` en nuestro sistema local (externo al clúster):

scp (@local)

```
local$ scp -P<port>       \
  /datos/work/un.fichero  \
  nome.apelido@ctcomp2.inv.usc.es:~/work/
local$ scp -P<port> -r    \
  /datos/work/directorio/ \
  nome.apelido@ctcomp2.inv.usc.es:~/work/
```

2 Gestión de software con `modules`

El comando `modules` permite gestionar, de manera eficaz y consistente, múltiples versiones de librerías y sofware para que el usuario utilice la versión adecuada en función de sus requerimientos. Su funcionamiento se basa en el encapsulamiento, dentro de un módulo, de las variables de entorno relacionadas con una versión de software determinada. De este modo, es el propio usuario quien gestiona la utilización de las diferentes versiones de software disponibles en el sistema.

La gestion, a nivel de usuario, de los módulos se realiza con el comando `modules` :

module

```
ct$ module avail
ct$ module list
ct$ module load module_name
ct$ module unload module_name
ct$ module purge
```

Las opciones son:

`avail` Muestra todos los módulos disponibles en el sistema.

`list` Muestra todos los módulos que están siendo utilizados en la sesión actual.

`load` Activa el módulo `module_name`

`unload` Desactiva el módulo `module_name`

`purge` Desactiva todos los los módulos de la sesión actual.

El comando `modules` manipula las variables de entorno relacionadas con los *path* del sistema (`PATH`, `LD_LIBRARY_PATH`, etc.), por lo que se recomienda a los usuarios no modificar estas variables de modo arbitrario.

Se recomienda utilizar este comando de manera interactiva. Su uso dentro de `.bashrc` para cargar automáticamente *módulos* habituales no está recomendado, ya que todos los scripts que se ejecuten leen este fichero.

Se recomienda utilizar las versiones por defecto de los difentes módulos. En cualquier caso, el comando `module avail` porporciona una lista completa de todos los los módulos y versiones disponibles.

3 Compilación

Compilación C/C++/Fortran

La colección de compiladores GNU (GNU Compiler Collection, GCC) es accesible en el clúster a través de sus comandos y opciones habituales. Por defecto, los compiladores instalados en el sistema pertenecen a la versión 4.7.2 de GCC (versión por defecto del SO).[2]

GCC

```
ct$ module load gcc
ct$ gcc      -O exemplo.c   -o exemplo
ct$ g++      -O exemplo.cpp -o exemplo
ct$ gfortran -O exemplo.f   -o exemplo
```

Las opciones recomendadas son:

`-O` Genera código optimizado para obtener un mayor rendimiento. Es equivalente a `-O1`. Alternativamente, se pueden utilizar las opciones `-O0`, `-O2` o `-O3`. El número indica el nivel de optimización, siendo 0 el nivel sin ningún tipo de optimización y 3 el nivel con el que se optiene un mayor rendimiento (la opción `-O3` realiza algunas optimizaciones agresivas que pueden generar resultados imprecisos).

`-o <name>` Establece el nombre del fichero ejecutable.

Compilación OpenMP

La colección de compiladores GCC permite la compilación de código OpenMP, indicándolo mediante la opción `-fopenmp`.

GCC + OpenMP

```
ct$ gcc      -O -fopenmp exemplo.c
ct$ g++      -O -fopenmp exemplo.cpp
ct$ gfortran -O -fopenmp exemplo.f
```

[2] Esta versión de los compiladores disponen de una opción de optimización (`-march=bdver1`) para generar código específico para la arquitectura de los nodos del clúster (procesadores Opteron 6200 series, 15th Family *Bulldozer* Interlagos). Esta opción de compilación no garantiza el cumplimiento del estándar matemático definido en GCC, por lo que no se recomienda su uso, salvo en aquellos casos en los que se conozca en profundidad el comportamiento de las opciones de compilación.

Compilación MPI

Para compilar código MPI es necesario cargar un módulo MPI (como, por ejemplo, el módulo `openmpi`), que proporcione los scripts de compilación de código MPI (`mpicc`, `mpicxx`, `mpif77`). Estos scripts hacen llamadas al compilador del lenguaje correspondiente.

GCC+MPI

```
ct$ module load openmpi
ct$ mpicc  -O exemplo.c
ct$ mpicxx -O exemplo.cpp
ct$ mpif77 -O exemplo.f
```

4 Envío de trabajos al sistema de colas Torque/PBS

Los usuarios no pueden conectarse a los nodos computacionales, por lo que la ejecución de trabajos en los nodos computacionales tiene que realizarse, obligatoriamente, a través del sistema de colas Torque/PBS. El sistema de colas registrará por orden cada una de las solicitudes enviadas y las emitirá, para su ejecución en los nodos computacionales, cuando estén disponibles los recursos requeridos.

El envío de trabajos se realiza a través del comando `qsub`, cuyo argumento obligatorio es el nombre de un script de *shell*. El script tiene que disponer de permisos de ejecución. Dentro del script, el usuario debe indicar la acciones que se realizarán en los nodos, una vez que los recursos requeridos estén disponibles (La §4.2: *Ejemplos de scripts* contiene diferentes ejemplos relacionados con los módulos instalados en el clúster).

qsub
```
ct$ chmod u+x script.sh
ct$ qsub script.sh
```

Los comandos `qstat` y `pbsnodes` permiten al usuario consultar el estado de la cola PBS.

qstat
```
ct$ qstat    # Información de los trabajos de usuario
ct$ qstat -q # Información global de las colas
ct$ pbsnodes # Información del estado de los nodos
```

El comando `qdel` permite al usuario eliminar un trabajo de la cola PBS, antes de que sea emitido a los nodos computacionales para su ejecución. Este comando necesita como argumento el identificador que PBS le asigna cuando se registra un nuevo trabajo, y que se puede consultar con `qstat`.

qdel
```
ct$ qdel job_id
```

4.1 Opciones de configuración Torque/PBS

El sistema de colas PBS permite a los usuarios configurar diferentes aspectos de la ejecución de los trabajos. Las instrucciones de configuración Torque/PBS se indican en los scripts a través de líneas que comienzan (sin espacios) con `#PBS`.[3] También es posible indicar estas opciones directamente como argumentos de `qsub` en la línea de comandos.

[3] De algún modo, estas órdenes son comentarios *especiales* del script, ya que el inicio de los comentarios se indica con el símbolo #.

Algunas de las opciones más comunes son:

`-N` Indica el nombre de referencia de nuestro trabajo en el sistema de colas. Por defecto sería el nombre del ejecutable.
Ej: `#PBS -N myjob`

`-l` Indica los recursos que se solicitan para la ejecución de nuestro trabajo, como el número de núcleos computacionales y el tiempo de ejecución. Los diferentes tipos de recursos se separan por comas.
Ej: `#PBS -l nodes=1:ppn=1,walltime=1:00:00`

- `nodes=`ν`:ppn=`κ: solicitamos ν nodos computacionales, y κ núcleos en cada nodo.[4]
- `walltime=HH:MM:SS`: solicitamos la exclusividad de los recursos durante un tiempo máximo de HH horas, MM minutos y SS segundos. El límite máximo de tiempo permitido es de 168 horas (1 semana).

`-e` Indica el fichero en el que se redireccionará la salida estándar de error de nuestro ejecutable. Por defecto, la salida estándar de error se redirecciona a un fichero con extensión `.eXXX` (donde `XXX` representa el identificador PBS del trabajo).
Ej: `#PBS -e mySTD.err`

`-o` Indica el fichero en el que se redireccionará la salida estándar de nuestro ejecutable. Por defecto, la salida estándar se redirecciona a un fichero con extensión `.oXXX` (donde `XXX` representa el identificador PBS del trabajo).
Ej: `#PBS -o mySTD.out`

`-m` Indica el tipo de eventos que serán notificados por correo electrónico. Los argumentos posibles de esta opción son: `b` cuando el trabajo se emita a los nodos, `a` en caso de que se aborte la ejecución del trabajo inexperadamente y/o `e` cuando el trabajo termine su ejecución sin ningún incidente. Estos argumentos no son excluyentes y se pueden combinar.
Ej: `#PBS -m ae`

[4] No se garantiza la ejecución en exclusividad de los nodos, si no se solicitan los 64 núcleos de un nodo.

`-M` Indica la dirección de correo en la que se notificarán los eventos indicados con la opción `-m`.
Ej: `#PBS -M nombre.usuario@usc.es`

Por defecto, el entorno de ejecución del sistema Torque/PBS define algunas variables de entorno que pueden ser utilizadas dentro de los scripts:

— `$PBS_O_WORKDIR`: contiene el *path* del directorio de trabajo (`$PWD`) desde donde se ha ejecutado el comando `qsub`. Es útil para establer un directorio de referencia durante la ejecución de los trabajos indicados.

Colas de usuario

El sistema PBS del clúster `ctcomp2` tiene cuatro colas de usuario y ocho colas de sistema. Las colas de usuario son en realidad colas *routing* que determinan, en función del número de núcleos computacionales solicitados, la cola de sistema en la que debe ejecutarse cada trabajo. Los usuarios deben enviar sus trabajos a las colas de usuario, ya que no pueden enviar trabajos directamente a las colas de sistema.

Independientemente del tipo de cola que se utilice para el envío de trabajos, los únicos parámetros que puede solicitar el usuario son el **número de nodos**, el **número de núcleos (procesos) por nodo** y el **tiempo de ejecución**. La memoria asignada y el tiempo máximo de ejecución de un trabajo están determinados por la cola de sistema en la que se ejecute el trabajo. Los trabajos que superen estos límites durante su ejecución serán cancelados. Por lo tanto, aquellos trabajos en los que tanto la memoria como el tiempo de ejecución son críticos, se recomienda a los usuarios modificar el número de procesos solicitados (aunque realmente no se utilicen todos durante su ejecución) para que se garanticen los requisitos del trabajo. El límite máximo de trabajos por usuario y la prioridad de los trabajos es también dependiente de la cola de sistema. Se permite que los usuarios determinen el tiempo de ejecución de los trabajos ya que una estimación precisa de los tiempos de ejecución permite al sistema de colas hacer un uso eficiente de los recursos sin perturbar las prioridades establecidas. En cualquier caso, es recomendable establecer un margen de tiempo suficiente para garantizar la correcta ejecución del trabajo y evitar la cancelación del mismo. Para ejecutar trabajos que no se ajusten a los parámetros de las colas PBS será necesario ponerse en contacto con el responsable del clúster.

Las colas de usuario son batch, short, bigmem e interactive.

— batch. Es la cola por defecto.[5] Admite hasta 10 trabajos por usuario. Los trabajos enviados a esta cola podrán ejecutarse en cualquier cola de sistema.

— short. Es una cola diseñada para dismunir el tiempo de espera de trabajos que no sean muy costosos temporalmente (máximo 12 horas) y que no consuman muchos recursos (menos de 16 núcleos computacionales). La cola short tiene una mayor prioriodad que la cola batch y admite hasta 40 trabajos por usuario. Los trabajos enviados a esta cola solo pueden ejecutarse en un subconjunto de las colas de sistema: np16, np8, np4, np2 y np1. Para solicitar la ejecución de un trabajo en la cola short es necesario indicarlo explícitamente con la opción -q del comando qsub:

```
ct$ qsub -q short script.sh
```

— bigmem. Es una cola diseñada para trabajos que consuman mucha memoria. Esta cola reservará un nodo de 64 núcleos completo para el trabajo del usuario, por lo que se deben indicar nodes=1:ppn=64 en la opción -l de qsub. Esta cola tiene una mayor prioriodad que la cola batch y está limitada a dos trabajos por usuario. Para solicitar la ejecución de un trabajo en la cola bigmem es necesario indicarlo explícitamente con la opción -q del comando qsub:

```
ct$ qsub -q bigmem script.sh
```

— interactive. Esta cola es la única que admite sesiones interactivas en los nodos computacionales. En esta cola solo se admite un trabajo por usuario, con un tiempo máximo de ejecución de 1 hora y se tendrá acceso a un único núcleo de un nodo computacional. La utilización de la cola interactive no requiere un *script*, pero es necesario indicar la interactividad de nuestro trabajo a través de la opción -I:

```
ct$ qsub -q interactive -I
```

[5] Si no se especifica una cola particular, mediante la opción -q del comando qsub, el trabajo se asignará a la cola batch.

Las colas de sistema son `np1`, `np2`, `np4`, `np8`, `np16`, `np32`, `np64` y `parallel`.

— `np1`. Trabajos que requieren 1 proceso y 1 nodo. La memoria asociada a los trabajos de esta cola es 1,99 GB y el tiempo máximo de ejecución 672 horas.

— `np2`. Trabajos que requieren 2 procesos. La memoria asociada a los trabajos de esta cola es 3,75 GB y el tiempo máximo de ejecución 192 horas.

— `np4`. Trabajos que requieren 4 procesos. La memoria asociada a los trabajos de esta cola es 7,5 GB y el tiempo máximo de ejecución 192 horas.

— `np8`. Trabajos que requieren 8 procesos y, como máximo, 5 nodos. La memoria asociada a los trabajos de esta cola es 15 GB y el tiempo máximo de ejecución 192 horas.

— `np16`. Trabajos que requieren 16 procesos y, como máximo, 5 nodos. La memoria asociada a los trabajos de esta cola es 31 GB y el tiempo máximo de ejecución 192 horas.

— `np32`. Trabajos que requieren 32 procesos y, como máximo, 5 nodos. La memoria asociada a los trabajos de esta cola es 63 GB y el tiempo máximo de ejecución 288 horas.

— `np64`. Trabajos que requieren 64 procesos y, como máximo, 5 nodos. La memoria asociada a los trabajos de esta cola es 127 GB y el tiempo máximo de ejecución 384 horas.

— `parallel`. Trabajos que requieren más de 32 procesos en, al menos, 2 nodos diferentes. La memoria asociada a los trabajos de esta cola es 64 GB y el tiempo máximo de ejecución 192 horas.

La siguiente tabla resume las características de las colas de usuario y de sistema del clúster `ctcomp2` (Pr ≡ *procesos*, No ≡ *nodos*, Me ≡ *memoria máxima del trabajo*, Tr ≡ *trabajo*, Us ≡ *usuario*, Hm ≡ *Tiempo máximo en horas*, π ≡ *prioridad*):

cola	Pr	No	Me (GB)	Tr/Us	Hm	π
`batch`	1-64	-	-	128	-	1
`short`	1-16	-	-	256	-	3
`bigmem`	64	-	-	8	-	2
`interactive`	1	1	2	1	1	7
`np1`	1	1	1,99	120	672	6
`np2`	2	2	3,75	120	192	5
`np4`	4	4	7,5	60	192	4
`np8`	8	5	15	60	192	4
`np16`	16	5	31	15	192	3
`np32`	32	5	63	15	288	2
`np64`	64	5	127	4	384	1
`parallel`	32-160	5	64	15	192	3

Solicitud de recursos específicos

Cada nodo computacional está asociado con diferentes etiquetas que identifican sus características. Las etiquetas de cada nodo se puede consultar con el comando `pbsnodes`, en el campo `propiedades`. La función básica de estas etiquetas en `ctcomp2` es crear conjuntos de nodos homogéneos, lo que permite a los usuarios solicitar la ejecución de sus trabajos en un tipo de nodo computacional específico. En particular, los nodos HP Proliant tienen asociada la etiqueta `amd` y los nodos Dell PowerEdge la etiqueta `intel`.

Para solicitar la ejecución de un trabajo en un nodo asociado a una etiqueta particular es necesario indicarlo explícitamente en el comando `qsub`, dentro el campo `nodes` de la opción `-l`. Por ejemplo, si queremos ejecutar un trabajo (1 núcleo, 1 nodo, 1 hora) en un nodo Dell PowerEdge 910 (procesadores Intel Xeon L755) la correspondiente línea `#PBS` del scrip debería ser:

```
#PBS -l nodes=1:ppn=1:intel:xeonl
```

Si queremos que nuestro trabajo se ejecute en un nodo Dell PowerEdge 620 (procesadores Intel Xeon E5-2650L), la línea debería ser:

```
#PBS -l nodes=1:ppn=1:intel:xeone5
```

Si por el contrario, queremos que se ejecute en un nodo HP Proliant, la línea debería ser:

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

Si no se indica ninguna etiqueta el trabajo se ejecutará indiferentemente en cualquier tipo de nodo, en función de los recursos disponibles en ese instante. Si únicamente se indica la etiqueta `intel`, el trabajo se ejecutará indiferentemente cualquiera de los nodos Dell PowerEdge.

Sesión interactiva con gráficos

Por defecto, las colas de usuario no permiten la ejecución de aplicaciones en modo gráfico. Además, en el frontend la ejecución de este tipo de aplicaciones está restringida. En general, los usuarios que necesiten ejecutar en el clúster aplicaciones en modo gráfico de manera interactiva, deberán utilizar el comando `graphicX_session`:

```
ct$ graphicX_session NPROC
```

`NPROC` es un parámetro obligatorio que indica el número de núcleos computacionales que queremos asociar a nuestra sesión interactiva. En realidad, mediante este parámetro también estamos asociando el tamaño de memoria asignada a la sesión, ya que la memoria asignada a un trabajo (en GB) es dos veces el número de núcleos asignado.

Este comando solicita una sesión interactiva en un nodo del clúster y, automáticamente, se conecta por SSH a ese nodo con la redirección de las X activada. Para que este comando funcione correctamente es necesario que la conexión SSH inicial al frontend tenga también la redirección de X activada (`ssh -X ...`). El tiempo de ejecución de esta sesión interactiva está limitado a 4 horas.

La disponibilidad de este tipo de sesiones está limitada por la carga de trabajo del clúster y el número de sesiones gráficas activas de todos los usuarios.

4.2 Ejemplos de scripts

A continuación se muestran varios ejemplos de scripts que muestran el conjunto básico de intrucciones para el envio de diferentes tipos de trabajos al gestor de colas.

Ejemplo de trabajo secuencial:

```
#!/bin/bash
#PBS -l nodes=1:ppn=1,walltime=1:00:00
#PBS -N ejemplo
#PBS -e salida.out
#PBS -o salida.err
#PBS -m ae -M direccion_correo@usc.es
cd $PBS_O_WORKDIR
./executable
```

Ejemplo de trabajos secuenciales en paralelo:

```
#!/bin/bash
#PBS -l nodes=1:ppn=4,walltime=1:00:00
#PBS -N ejemplo
cd $PBS_O_WORKDIR
(cd dir1; ./executable1) &
(cd dir2; ./executable2) &
(cd dir3; ./executable3) &
(cd dir4; ./executable4) &
wait
```

Ejemplo de trabajo Java:

Java consume, por defecto, toda la memoria del sistema. En un sistema compartido por diferentes usuarios es necesario limitar la cantidad de memoria asignada a cada proceso, por lo que el tamaño del *heap* de Java en `ctcomp2` está limitado al 25% del límite de memoria de la cola asignada al proceso y, cualquier caso, con un tamaño máximo de 8 GB.

Los usuarios pueden utilizar otros tamaños de *heap* en sus trabajos si modifican, antes de ejecutar JAVA, el valor de la opción -Xmx de JAVA a través de la variable `_JAVA_OPTIONS`. Por ejemplo, si queremos que el *heap* tenga 16 GB, el comando sería:

```
export _JAVA_OPTIONS=-Xmx16777216K
```

Al modificar el tamaño del *heap* el usuario debe asegurarse, bajo su responsabilidad, que el conjunto de procesos que se estén ejecutando concurrentemente en su trabajo no sobrepase el límite de memoria establecido en la correspondiente cola, ya que en ese caso el trabajo será cancelado automáticamente.

Los usuarios que ejecuten en sus trabajos una sola instancia de java (independientemente de los threads que ejecute) podrán aumentar el tamaño

del *heap*, pero se recomienda que no sea un valor cercano al límite de memoria de la correspondiente cola. Si se ejecutan varias instancias de java, se recomienda ajustar adecuadamente el tamaño del *heap* para evitar cancelaciones.

```
#!/bin/bash
#PBS -l nodes=1:ppn=64,walltime=1:00:00
#PBS -N ej-java
cd $PBS_O_WORKDIR
module load jdk
export _JAVA_OPTIONS=-Xmx16777216K
java executable
```

Ejemplo de trabajo OpenMP:

```
#!/bin/bash
#PBS -l nodes=1:ppn=64,walltime=1:00:00
#PBS -N ej-openmp
cd $PBS_O_WORKDIR
export OMP_NUM_THREADS=64
./executable
```

Ejemplo de trabajo MPI:

```
#!/bin/bash
#PBS -l nodes=8:ppn=64,walltime=1:00:00
#PBS -N ej-mpi
cd $PBS_O_WORKDIR
module load openmpi
mpirun -np 512 ./executable
```

Ejemplo de trabajo R:

No es posible realizar una sesión R interactiva, por lo que es necesario escribir previamente los comandos en un script.

```
#!/bin/bash
#PBS -l nodes=1:ppn=1,walltime=1:00:00
#PBS -N ej-R
cd $PBS_O_WORKDIR
module load R
R --no-save < test.R
```

Ejemplo de trabajo MATLAB:

No es posible realizar una sesión MATLAB interactiva, por lo que es necesario escribir previamente los comandos en un script. MATLAB emplea internamente, y de manera transparente al usuario, threads para paralelizar ciertas operaciones, por lo que se puede utilizar reservando varios núcleos computacionales dentro de un mismo nodo. Sin embargo, en las versiones instaladas no es posible controlar esta característica, por lo que MATLAB utiliza todos los recursos del nodo asignado, independientemente de los recursos solicitados. Para evitar posibles cancelaciones debido a un uso indebido de los recursos asignados, se recomienda a los usuarios utilizar el paralelismo implícito de MATLAB solo cuando se reserve en exclusividad un nodo del clúster. En cualquier otro caso, se recomienda solicitar únicamente un núcleo computacional y utilizar la opción `-singleCompThread`, que desactiva el paralelismo implícito de MATLAB para realizar una ejecución secuencial. En este ejemplo, el nombre del script es `test.m`.

¡!

```
#!/bin/bash
#PBS -l nodes=1:ppn=1,walltime=1:00:00
#PBS -N ej-MATLAB
cd $PBS_O_WORKDIR
module load matlab
matlab -r test -nodisplay -nojvm -singleCompThread
# -r:          indicar el fichero a ejecutar (sin .m)
# IMPORTANTE: incluir la orden quit al final del fichero
# -nodisplay: sin display X.
# -nosplash:  sin pantalla inicial (OPCIONAL)
# -nojvm:     sin entorno java (OPCIONAL)
# -singleCompThread: ejecuta MATLAB secuencialmente
```

5 Preguntas frecuentes

¿Qué es un clúster de computación?

Es un conjunto de nodos computacionales interconectados mediante una red dedicada y que pueden actuar como un único elemento computacional

En la práctica, esto se traduce en:

— potencia computacional (ejecución de un trabajo paralelo muy grande o muchas ejecuciones pequeñas concurrentemente)...

— ...en una infraestructura compartida entre varios usuarios

¿Qué es un sistema de gestión de colas?

Un sistema de gestión de colas (SGC) es un software que planifica la ejecución de trabajos entre los recursos computacionales disponibles. Es un software habitual en los sistemas de computación de altas prestaciones ya que permite una gestión eficiente de los recursos computacionales en un sistema con múltiples usuarios. En el clúster está instalado PBS/TORQUE. Para mayor información sobre el sistema de colas de ctcomp2: Envío de trabajos al sistema de colas Torque/PBS.

La dinámica de funcionamiento de estos sistemas es:

1. El usuario solicita al SGC unos determinados recursos para realizar una tarea computacional. Esta tarea estará formada por un conjunto de instrucciones, que deben estar almacenadas en un script.

2. El SGC registra la solicitud en una de sus colas.

3. Cuando estén disponibles los requisitos solicitados, y en función de las prioridades establecidas en el sistema, el SGC se encarga de ejecutar la tarea en los nodos computacionales y devolver la salida generada.

Es importante destacar que, al contrario que en un PC, la solicitud y la ejecución de una tarea son acciones independientes que no tienen por que realizarse de manera indisoluble. De hecho, lo más habitual es que la ejecución tenga que esperar en la cola durante un tiempo indefinido hasta que haya recursos disponibles. Por otro lado, una consecuencia directa de

desligar estas dos acciones es la imposibilidad de realizar ejecuciones de manera interactiva.[6]

¿Por qué no se ejecuta inmediatamente el trabajo que he enviado con qsub?

El sistema de colas permite desacoplar la ejecución de un trabajo en dos fases claramente diferenciadas. La primera acción consiste en una solicitud, a través de qsub, para ejecutar un código en los nodos. La segunda acción es transparente al usuario y consiste en el envío del trabajo a los nodos computacionales. Este envío solo se produce cuando se cumplen las condiciones de ejecución establecidas por el usuario en su solicitud.

Por lo tanto, si el trabajo no se ejecuta inmediatamente, puede deberse a diversas situaciones:

— Existe un pequeño retardo entre la solicitud y el envío. No es una cantidad de tiempo determinada, pero suele ser menor de 1 minuto.

— Si existen muchos trabajos encolados (se puede consultar con `qstat -q`) puede que nuestro trabajo no tenga disponibles los recursos que solicitamos. Cuando existan los recursos disponibles, y no existan trabajos en espera con mayor prioridad, nuestro trabajo será enviado automáticamente para su ejecución. (En el clúster `ctcomp2` es posible crear una alerta para conocer cuando se envía un trabajo, y cuando termina: Envío de trabajos al sistema de colas Torque/PBS.)

— El clúster dispone de un sistema de gestión de energía que apaga los nodos computacionales cuando no han sido utilizados durante cierto tiempo. Puede suceder que, aunque no haya trabajos en la cola, nuestra solicitud no sea enviada a los nodos porque los recursos estaban apagados en el momento de la solicitud. En este caso, la ejecución del trabajo deberá esperar a que los recursos estén activos (es también un tiempo variable, pero suele ser menor de 10 minutos).

[6] Existen colas especiales, con tiempo y recursos limitados, que permite realizar sesiones interactivas.

Referencia @ctcomp2

CiTIUS
Centro Singular de Investigación en Tecnoloxías da Información

```
# Acceso
ssh -p <port> nome.apelido@ctcomp2.inv.usc.es

# importar
scp -p <port> \
  fich_orixe  \
  nome.apelido@ctcomp2.inv.usc.es:~/fich_destino
scp -p <port> -r \
  dir_orixe      \
  nome.apelido@ctcomp2.inv.usc.es:~/dir_destino

# exportar
scp -p <port>                                  \
  nome.apelido@ctcomp2.inv.usc.es:~/fich_orixe \
  fich_destino
scp -p <port> -r                               \
  nome.apelido@ctcomp2.inv.usc.es:~/dir_orixe \
  dir_destino
```

```
# Gestión de módulos
module avail
module list
module load <module_name>
module unload <module_name>
module purge
```

```
# Compilación GCC
gcc       -O exemplo.c   -o exe
g++       -O exemplo.cpp -o exe
gfortran -O exemplo.f   -o exe
```

```
# Compilación OpenMP (+GCC)
gcc       -O -fopenmp exemplo.c   -o exe-omp
g++       -O -fopenmp exemplo.cpp -o exe-omp
gfortran -O -fopenmp exemplo.f   -o exe-omp
```

```
# Compilación OpenMPI (+GCC)
module load openmpi
mpicc  -O exemplo.c    -o exe-mpi
mpicxx -O exemplo.cpp  -o exe-mpi
mpif77 -O exemplo.f    -o exe-mpi
```

```
# Manejo de la cola de trabajos
chmod u+x myjob.sh
qsub myjob.sh
qstat
qstat -q
pbsnodes
qdel <job_id>
qsub -q short myjob.sh
qsub -q bigmem myjob_64ppn.sh
qsub -I -q interactive
```

```
#!/bin/bash
# --- Ejemplo de configuración PBS --- #
#PBS -l nodes=1:ppn=1,walltime=1:00:00
#PBS -N proba-PBS
#PBS -e mySTD.err
#PBS -o mySTD.out
#PBS -m ae -M nome.apelido@usc.es
cd $PBS_O_WORKDIR
./executable
```

```
#!/bin/bash
# --- Ejemplo de ejecución: nodos Intel M910 --- #
#PBS -l nodes=1:ppn=1:intel:xeonl,walltime=1:00:00
cd $PBS_O_WORKDIR
./executable_intel
```

```
#!/bin/bash
# --- Ejemplo de ejecución: nodos Intel M620 --- #
#PBS -l nodes=1:ppn=1:intel:xeone5,walltime=1:00:00
cd $PBS_O_WORKDIR
./executable_intel
```

```
#!/bin/bash
# --- Ejemplo de ejecución: nodos AMD --- #
#PBS -l nodes=1:ppn=1:amd,walltime=1:00:00
cd $PBS_O_WORKDIR
./executable_amd
```

```
#!/bin/bash
# --- Ejemplo trabajo secuencial --- #
#PBS -l nodes=1:ppn=4,walltime=1:00:00
#PBS -N ej-secuencial
cd $PBS_O_WORKDIR
(cd dir1; ./exec1) &
(cd dir2; ./exec2) &
(cd dir3; ./exec3) &
(cd dir4; ./exec4) &
wait
```

```
#!/bin/bash
# --- Ejemplo trabajo JAVA --- #
#PBS -l nodes=1:ppn=64,walltime=1:00:00
#PBS -N ej-java
cd $PBS_O_WORKDIR
module load java
export _JAVA_OPTIONS=-Xmx16777216K
java executable
```

```
#!/bin/bash
# --- Ejemplo trabajo OpenMP --- #
#PBS -l nodes=1:ppn=64,walltime=1:00:00
#PBS -N ej-openmp
cd $PBS_O_WORKDIR
export OMP_NUM_THREADS=64
./executable
```

```
#!/bin/bash
# --- Ejemplo trabajo MPI --- #
#PBS -l nodes=2:ppn=64,walltime=1:00:00
#PBS -N ej-mpi
cd $PBS_O_WORKDIR
module load openmpi
mpirun -np 128 ./executable
```

```
#/bin/bash
# --- Ejemplo trabajo R --- #
#PBS -l nodes=1:ppn=1,walltime=1:00:00
#PBS -N ej-R
cd $PBS_O_WORKDIR
module load R
R --no-save < test.R
```

MATLAB secuencial:

```
#!/bin/bash
# --- Ejemplo trabajo MATLAB (SECUENCIAL) --- #
#PBS -l nodes=1:ppn=1,walltime=1:00:00
#PBS -N ej-MATLAB
cd $PBS_O_WORKDIR
module load matlab
matlab -r test -nodisplay -nosplash -nojvm -singleCompThread
# -r:         indicar el fichero a ejecutar (sin .m)
# (!) Incluir la orden quit al final del fichero
# -nodisplay: sin display X.
# -nosplash: sin pantalla inicial (OPCIONAL)
# -nojvm: sin entorno java (OPCIONAL)
# -singleCompThread: ejecuta MATLAB secuencialmente
```

MATLAB paralelo, exclusivo nodos AMD (64 threads implícitos):

```
#!/bin/bash
# --- Ejemplo trabajo MATLAB (PARALELO nodos amd: 64 threads implícitos) --- #
#PBS -l nodes=1:ppn=64:amd,walltime=1:00:00
#PBS -N ej-MATLAB
cd $PBS_O_WORKDIR
module load matlab
matlab -r test -nodisplay -nojvm -nosplash
# -r:         indicar el fichero a ejecutar (sin .m)
# (!) Incluir la orden quit al final del fichero
# -nodisplay: sin display X.
# -nosplash: sin pantalla inicial (OPCIONAL)
# -nojvm: sin entorno java (OPCIONAL)
```

MATLAB paralelo, exclusivo nodos INTEL (32 threads implícitos):

```
#!/bin/bash
# --- Ejemplo trabajo MATLAB (PARALELO nodos intel: 32 threads implícitos) --- #
#PBS -l nodes=1:ppn=32:intel,walltime=1:00:00
#PBS -N ej-MATLAB
cd $PBS_O_WORKDIR
module load matlab
matlab -r test -nodisplay -nojvm -nosplash
# -r:         indicar el fichero a ejecutar (sin .m)
# (!) Incluir la orden quit al final del fichero
# -nodisplay: sin display X.
# -nosplash: sin pantalla inicial (OPCIONAL)
# -nojvm: sin entorno java (OPCIONAL)
```