Guía uso Finis Terrae

Santiago de Compostela, Abril 2008

El sistema Finis Terrae está formado por 142 nodos de computación, cada uno de ellos con 16 cores Itanium2 a 1.6GHz y 128GB de memoria principal, y 1 nodo con 128 cores Itanium2 y 1024GB de memoria. El acceso al sistema se realiza a través de dos nodos (*nodo de login*) con 4 cores Xeon y 8GB de memoria principal cada uno.

1. CONEXIÓN

Para acceder a Finis Terrae, utilizar un cliente ssh (como putty para windows) y conectarse a ft.cesga.es

Introducir los datos de login y password

El sistema al que se habrá conectado (*nodo de login*) tiene acceso directo a las colas y a los directorios de usuario. Sin embargo, la configuración hardware de estos nodos (procesadores) no es compatible con los nodos de cálculo.

En este nodo de login se pueden realizar entre otras las siguientes operaciones:

- Chequear el estado de los trabajos
- Enviar nuevos trabajos a cola
- Borrar trabajos de la cola
- Crear y editar ficheros
- Realizar transferencias de ficheros a/desde Finis Terrae
- Realizar algunas acciones de preprocesado y postprocesado

Las *aplicaciones* disponibles en Finis Terrae no se podrán ejecutar desde este sistema, a menos que se especifique lo contrario (consultar la página web www.cesga.es para documentación más reciente al respecto). Tampoco esta operativo en estos nodos el sistema para la carga de entornos *modules*. Para utilizar las aplicaciones y el sistema de cálculo deberán seguirse las instrucciones descritas en los puntos 2 y 3 de esta guía. Podrán abrirse hasta 10 conexiones por usuario a este sistema.

2. COMPILACION Y USO INTERACTIVO DE APLICACIONES

Para poder compilar y ejecutar aplicaciones en interactivo, es necesario conectarse a un *nodo de computación* de Finis Terrae. Los nodos de computación disponen de procesadores Itanium2 (16 procesadores por nodo y 142 nodos). Para ello, desde el nodo de login hay que ejecutar el comando:

compute ––memory xx (xx indica la memoria máxima para la sesión, en GB). Si no se especifica la opción memory, reservará 1 GB.
Este comando abrirá una sesión interactiva con un nodo de Finis Terrae, con las siguientes limitaciones:

- Máximo una hora de tiempo de CPU
- Máximo de 8 horas de sesión (tiempo real)
- Máximo de Memoria xx GB. Limitado a 64 GB
- Tamaño de fichero limitado por el sistema de ficheros en uso

Sólo será posible abrir dos sesiones simultáneas por usuario. En caso de que en ese momento no pueda establecerse una conexión con la cantidad de memoria requerida, se informará al usuario de la máxima memoria que se puede utilizar.

Durante la sesión interactiva, estarán disponibles los siguientes sistemas de ficheros (véase el apartado 7 para más información sobre los sistemas de ficheros):
Home: referenciado por $HOME
Scratch: referenciado por $TMPDIR
Scratch paralelo: referenciado por $HOMESFS

Para cerrar la sesión es suficiente ejecutar el comando `exit` o pulsar `control+d`

Problemas en interactivo

Durante una sesión interactiva es posible que el servicio o *nodo de login* pueda fallar por diversos motivos técnicos que se irán solucionando. Sin embargo, el sistema está preparado para admitir nuevas conexiones en caso de fallo. Por ello, se aconseja cerrar o salir de la sesión activa y reintentar la conexión un minuto más tarde.

3. ENTORNOS DE EJECUCION. MODULES

Modules es una utilidad que permite configurar el entorno de usuario de forma dinámica para el uso de diferentes paquetes y aplicaciones.

Se ha desarrollado un fichero de módulo para cada aplicación soportada por el CESGA, incluídos los compiladores y algunas librerías.

El comando básico para su utilización es "module".

`module avail` (o simplemente `module av`)
Muestra la lista de módulos disponibles.

```
module load | add  <modulefile>
```
Carga el módulo "modulefile". Por ej:

```
module load ifort
```
Carga la versión por defecto del compilador del fortran de intel

```
module load ifort/10.1.012
```
Carga la versión 10.1.012 del compilador del fortran de intel

Inicialmente los principales módulos disponibles son:

<table>
<tr><td>intel/9(default)</td><td rowspan="2">module load intel
Carga el entorno de uso de los compiladores de intel icc/ifort/mkl (por defecto versión 9)</td></tr>
<tr><td>intel/10</td></tr>
<tr><td>ifort/9.1.052(default)</td><td rowspan="2">module load ifort
Carga el entorno de uso de los compiladores de fortran de intel (por defecto versión 9)</td></tr>
<tr><td>ifort/10.1.012 (no recomendado)</td></tr>
</table>

<table>
<tr><td>mkl/9.1(default)</td><td rowspan="2">module load mkl
Carga el entorno de uso de las librerías mkl (por defecto versión 9)</td></tr>
<tr><td>mkl/10.0.011</td></tr>
<tr><td>icc/9.1.052(default)</td><td rowspan="2">module load icc
Carga el entorno de uso de los compiladores de C/C++ de intel (por defecto versión 9)</td></tr>
<tr><td>icc/10.1.012</td></tr>
<tr><td>hp-mpi</td><td>module load hp-mpi
Carga el entorno de uso del MPI de HP</td></tr>
<tr><td>impi/3.0(default)</td><td rowspan="2">module load impi
Carga el entorno de uso del MPI de intel (por defecto versión 3.0)</td></tr>
<tr><td>impi/3.1</td></tr>
<tr><td>g03/c1</td><td rowspan="3">module load g03
Carga el entorno de uso de Gaussian 03 (por defecto versión E.01)</td></tr>
<tr><td>g03/d2</td></tr>
<tr><td>g03/e1(default)</td></tr>
</table>

module unload <modulefile>
Descarga el módulo "modulefile".

module list
Muestra una lista de los módulos que el usuario tiene cargados.

module initadd <modulefile>
Añade el modulo "modulefile" a los archivos de inicio de sesión del usuario, de forma que siempre que se conecte al servidor tenga dicho módulo precargado. Para que este comando funcione, en el script de arranque de la sesión de usuario (.bashrc en el caso de utilizar bash, .profile en el caso de ksh), debe contener la línea module load modules

module initrm <modulefile>
Elimina el módulo "modulefile" de los archivos de inicio de sesión de usuario.

4. SISTEMA DE COLAS

En caso de necesitar más recursos que los disponibles para las sesiones interactivas, es necesario la utilización del sistema de colas. El modo de uso de este sistema es similar al existente en el resto de los servidores de cálculo del centro:

qsub -l num_proc=nproc, s_rt=hh:mm:ss, s_vmem=memoria, h_fsize=disco –pe mpi slots job.sh

En donde:

- slots indica el número de procesos MPI a utilizar. En caso de no especificar la opción –pe mpi slots, sólo se utilizará un nodo. Éste es el único parámetro opcional, el resto de parámetros son obligatorios para todos los trabajos. Los procesos MPI podrán ir a un mismo nodo o a nodos distintos. La política de ocupación intentará llenar el máximo número de procesos MPI por nodo.
- num_proc indica el número mínimo de procesadores por nodo. El número de procesadores total del trabajo será num_proc*slots (sólo deberá ser mayor que 1 en el caso de aplicaciones de memoria compartida como OpenMP)
- s_rt: es el tiempo máximo de ejecución del trabajo
- s_vmem la memoria máxima del trabajo (por nodo)
- h_fsize: la ocupación máxima en disco scratch (por nodo)

Los límites inicialmente serán los siguientes (información actualizada sobre los límites puede encontrarse en la página web del CESGA, www.cesga.es):

- Máximo de procesadores: 160 (el valor máximo de num_proc no podrá ser mayor de 16 y el de slots no podrá superar 160)
- Tiempo de ejecución (s_rt):
  trabajos secuenciales hasta 1000 horas.
  Trabajos paralelos 2 a 8 procesadores: 100 horas.
  De 9 a 16 procesadores: 48 horas.
  De 17 a 64 procesadores: 24 horas.
  De 65 a 128 procesadores: 10 horas.
  De 129 a 160 procesadores: 8 horas.
- Memoria: 112 GB (por nodo)
- H_fsize: 500GB (scratch local)

Los trabajos que necesiten tiempo superior a 200 horas, deben cumplir las siguientes reglas:

1. Deben escribir los ficheros de scratch en el scratch paralelo (véase el apartado 7 para su utilización)
2. Deben disponer de un mecanismo de checkpoint, que permitan detener el trabajo y continuar su ejecución desde el punto en el que se encontraba en ese momento.

3. Deben ser por tanto checkpointables por el sistema de colas (es decir, permitir la parada completa del programa y continuar su ejecución más adelante)

Sólo estos trabajos serán admitidos, y en caso necesario, recibirán un checkpoint pero serán puestos nuevamente en ejecución de forma automática por el sistema de colas. Para consultas sobre cómo implementar estas reglas y cómo utilizarlas en las aplicaciones disponibles, pueden dirigirse a aplicaciones@cesga.es.

En caso de necesitar recursos superiores a los límites anteriores, deberá solicitarse el acceso a recursos especiales, siguiendo el procedimiento descrito en la página web del CESGA, www.cesga.es o dirigiéndose a sistemas@cesga.es

El script que se envía a cola job.sh deberá encargarse de cargar también los modulos necesarios para la ejecución del trabajo. Este script se ejecturá con shell bash, a menos que dentro del propio script se especifique lo contrario.

Nota importante: por razones de puesta a punto del sistema, no se admitirán trabajos de más de 100 horas hasta el día 30 de Abril.

5. Aplicaciones Disponibles

Inicialmente las aplicaciones disponibles son:

Gaussian, Gromacs, Namd y Siesta

Un listado actualizado de todas las aplicaciones disponibles se encuentra en la propia página web, en el apartado de aplicaciones.

6. Compiladores y librerías matemáticas

Se recomienda la utilización de los compiladores de Intel frente a los de GNU (debido a la diferencia de rendimiento obtenido). Para utilizar los compiladores de Intel deben cargarse los módulos correspondientes:

`module load icc`: Carga el compilador de C/C++
`module load ifort`: Carga el compilador de FORTRAN

Para compilar códigos, deberá utilizarse los siguientes comandos:

`icc fichero.c` (códigos en C)
`icpc fichero.C` (códigos en C++)
`ifort fichero.f` (códigos en FORTRAN)

Algunas opciones de compilación importantes son:

-O0	Sin optimización. Se debe usar en las etapas de desarrollo y debugging del programa
-O1	Optimización en tamaño. Omite optimizaciones que tienden a incrementar el tamaño del objeto.
-O2	Maximiza velocidad. Es la optimización por defecto.
-O3	Optimización a nivel 2 más optimizaciones de bucles y memoria mucho más agresivas. Este nivel es el recomendado para aplicaciones con bucles que usan operaciones de coma flotante intensivamente o procesan grandes conjuntos de datos.
-g	Genera información para el debugging. Esta opción fija -O0 la opción por defecto.
-openmp	Habilita las opciones de compilación con OpenMP
-parallel	Genera código paralelizado automáticamente para su utilización dentro de un único nodo (memoria compartida)
-fast	Optimiza el código de forma agresiva (utilizar con precaución)

Información más detallada sobre la utilización de los compiladores de Intel y la optimización de las aplicaciones se puede encontrar en:

Quick-Reference Guide to Optimization with Intel® Compilers
Optimizing Applications with Intel® C++ and Fortran Compilers

Utilización de librerías matemáticas MKL (BLAS/LAPACK/ScaLAPACK):

MKL contiene versiones robustas y optimizadas para Itanium2 de los siguientes conjuntos de funciones:

- BLAS
- Sparse BLAS
- LAPACK
- ScaLAPACK
- Sparse Solver routines
- Vector Mathematical Library functions
- Vector Statistical Library functions
- Fourier Transform functions (FFT)
- Cluster FFT
- Interval Solver routines
- Trigonometric Transform routines
- Poisson, Laplace, and Helmholtz Solver routines
- Optimization (Trust-Region) Solver routines

Existen 2 versiones disponibles de las MKL en el Finisterrae:

1. 9.1 (versión recomendada) : module load mkl
2. 10.0.011: module load mkl/10.0.011

No recomendamos la utilización de la versión 10 de las mkl debido al profundo cambio en el esquema de linkado y problemas detectados en varias aplicaciones.

Para linkar un programa que necesite estas librerías (versión 9.1) se ha de seguir la siguiente fórmula general:

```
-L<MKL path> -I<MKL include> [-lmkl_solver] [-lmkl_lapack95] [-lmkl_blas95] [-lmkl_lapack] {-lmkl_ipf, [-lmkl] [-lvml]} -lguide -lpthread [-lm]
```

Por ejemplo, el comando:

```
ifort -o test-dgemv test-dgemv.f -lmkl_ipf -lguide -lvml -pthread
```

generará un ejecutable test-dgemv que cargará a las librerías (si se utilizan en el programa, claro está),
/opt/cesga/intel/intel9.0/ict/3.0.1/cmkl/9.1/lib/64/libmkl.so
/opt/cesga/intel/intel9.0/ict/3.0.1/cmkl/9.1/lib/64/libguide.so
/opt/cesga/intel/intel9.0/ict/3.0.1/cmkl/9.1/lib/64/libvml.so

NOTA: sólo cuando los enteros son de 64 bits - Integer*8 en Fortran y long long en C -, se deben usar la versión de la libreria MKL ilp64 instaladas en /opt/cesga/intel/intel9.0/ict/3.0.1/cmkl/9.1/lib_ilp64/64 (versión 9.1).

Compilación con pase de mensajes MPI

MPI es una librería de pase de mensajes para programación paralela. Existen dos distribuciones de MPI instaladas en la máquina: HP MPI e Intel MPI (de este último están las versiones 3.0 y 3.1, siendo la 3.0 la versión por defecto).

El módulo del MPI de HP se carga de este modo:

```
module load hp-mpi
```

y el de MPI de Intel:

```
module load impi
```

Los comandos son básicamente los mismos en ambas distribuciones

1. HP MPI (recomendado)

Utiliza los compiladores de GNU, salvo que tengamos cargados los módulos de los compiladores de Intel.

mpicc, compila código MPI en C
mpiCC, compila código MPI en C++
mpif77, compila código MPI en Fortran 77
mpif90, compila código MPI en Fortran 90

De todos ellos existe versión con sufijo .mpich, que permite código MPI con sintaxis de la distribución mpich de MPI

Para ejecutar la aplicación se pueden usar los comandos:

```
mpirun
mpiexec
```

El comando mpiexec necesita crear previamente unos daemons mpi en los nodos que se desea ejecutar y es un poco más complicado de utilizar, por lo que no centraremos en el mpirun.

A este comando se le debe indicar el numero de procesos a utilizar (-np), y en el caso de necesitar más de un nodo para la ejecución, también se le debe indicar el fichero de nodos a utilizar (-hostfile) y el ejecutable con sus parámetros. Cuando las nodos de ejecución son asignados por el fichero de colas, el fichero de nodos a utilizar se encuentra en $TMPDIR/machines.

2. Intel MPI

En este caso, varia el comando de compilación dependiendo del compilador que se quiera utilizar. Si se desea utilizar un compilador de Intel debe estar cargado previamente su módulo correspondiente

mpicc, compila código MPI en C, utilizando el compilador de GNU gcc
`mpicxx`, compila código MPI en C++, utilizando el compilador GNU gcc
`mpif90`, compila código MPI en Fortran 90, utilizando el compilador GNU gfortran
`mpiicc`, compila código MPI en C, utilizando el compilador de Intel icc
`mpiicpc`, compila código MPI en C++, utilizando el compilador de Intel icpc
`mpifort`, compila código MPI en Fortran 90, utiilzando el compilador de Intel ifort

Para ejecutar la aplicación se pueden usar los comandos:

```
mpirun
mpiexec
```

Al igual que en el caso de la distribución de HP se describe y recomienda el uso de mpirun por su mayor simplicidad. A este comando se le debe indicar el numero de procesos a utilizar (-np), y en el caso de necesitar más de un nodo para la ejecución, también se le debe indicar el fichero de nodos a utilizar (-machines) y el ejecutable con sus parámetros. El fichero de nodos del asignado por el sistema de colas su ubicación es igual que en el caso anterior.
Ejemplo:

```
mpirun -np 4 -machines file.hosts mpi_program
```

Compilación y ejecución con OpenMP
Tanto el compilador de Fortran como de C soportan la versión 2 de OpenMP. Para compilar un programa que necesite utilizar OpenMP se ha de utilizar la opción -openmp. Por ejemplo, para compilar el programa programa.F con OpenMP utilizar el comando:

```
ifort -openmp -o programa programa.F
```

Por defecto, el número de hilos que se arrancan es igual al número de procesadores. Para controlar el número de hilos que se ejecutan, utilizar la variable OMP_NUM_THREADS=<número de hilos>. Por ejemplo, en la shell ksh o bash el siguiente comando limitará el número de hilos a 16:

```
export OMP_NUM_THREADS=16
```

Si se ejecuta en batch, esa variable tendrá un valor igual al número de procesadores seleccionados en el qsub con la opción -l num_proc.

7. Dispositivos de almacenamiento

Existen los siguientes dispositivos y directorios de almacenamiento en Finis Terrae:

Directorio home: se hacen backups, máximo de 10GB por usuario y máximo tamaño de fichero de 1GB. Referenciado por $HOME está accesible desde todos los nodos. Es el lugar recomendado para almacenar los códigos fuentes y binarios de aplicaciones, ficheros pequeños (o de texto) con parametros de entrada y ficheros de salida pequeños (o de texto).
Directorio scratch: no se hace backup y los datos sólo se guardan durante la ejecución de los trabajos. Límite de 500GB por trabajo. Sólo accesible desde el nodo que ejecuta el trabajo. Referenciado por la variable de entorno $TMPDIR.

Directorio scratch paralelo: no se hace backup y no se pueden almacenar ficheros durante más de 1 mes (pasado el cuál serán borrados automáticamente). Limitado a 1 Terabyte y 1000 ficheros por usuario. Referenciado por $HOMESFS. Accesible desde todos los nodos. Recomendado para almacenar ficheros de entrada y salida de simulaciones grandes (a partir de 10Mbytes)
Directorios sistemas de cálculo SVG y Superdome: estarán accesibles desde los nodos de conexión los directorios $HOMESVGD y $HOMESD con los datos disponibles en el SVG y Superdome.
Directorio COMPARTIDO: disponible desde todos los sistemas del CESGA y referenciado por $COMPARTIDO (en Finis Terrae sólo es accesible desde los nodos de conexión).

Para ampliar estos límites, así como para poder realizar otras operaciones sobre estos sistemas de almacenamiento (tales como volcado de datos a cinta), deberá solicitarse el servicio de almacenamiento tal y como se describe en la página web de almacenamiento.