PARALELISMO Y SISTEMAS DISTRIBUIDOS

GRADO EN CIENCIA E INGENIERÍA DE DATOS

FACULTAT D'INFORMÀTICA DE BARCELONA (FIB)

UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC)

PROFESORAS: JULITA CORBALAN (JULI@AC.UPC.EDU), YOLANDA BECERRA (YOLANDAB@AC.UPC.EDU)

SESIÓN 1: RECORDANDO LINUX

Durante esta sesión se recordarán los comandos principales para trabajar en un entorno Linux. Aunque lógicamente se realizarán en un entorno concreto, los comandos que veremos son básicos y podéis utilizarlos en cualquier sistema con Linux. Al final del documento encontraréis una descripción de la documentación que debéis generar como resultado de esta sesión, leedla antes de empezar.

CONEXIÓN AL SISTEMA

En las aulas donde realizaréis las prácticas tenéis unos PCs con las siguientes características: Intel Core i5 2400 3.10 Ghz 8GB RAM, DVD-ROM, Nvidia Geforce 210 1Gb, Monitor HP L1940T. Sistema operativo Windows y OpenSuSe. Desde estos PCs podemos conectarnos mediante ssh (programa de conexión remota) a Marenostrum4 (MN4) que es donde realizaremos las prácticas.

Al inicio de clase os daremos los nombres de usuario y contraseñas para conectaros a MN4). En el siguiente enlace podéis encontrar los manuales de usuario de MareNostrum con más detalles de los que os damos en esta documentación.

- https://www.bsc.es/user-support/mn4.php

MARENOSTRUM 4

Acceso remoto a MareNostrum

Para acceder a MareNostrum es necesario conectarse mediante ssh. MareNostrum tiene 3 nodos de login a los que os conectáis y 3.456 nodos de cómputo solamente accesibles mediante el sistema de colas.

Para acceder a los nodos de login utilizaremos principalmente ssh. También se puede acceder mediante Putty, que es un cliente de SSH gratuito (http://www.putty.org). En la documentación encontrareis mas detalles para conectaros dese Windows etc.

Para conectar haremos:

```
% ssh username@mn1.bsc.es (puede ser mn1, mn2 o mn3)
```

Para cambiar la contraseña debéis conectar a una máquina diferente

```
% ssh -l username dt01.bsc.es
username@dtransfer1:~> passwd
    Changing password for username.
    Old Password:
    New Password:
    Reenter New Password:
    Password changed.
```

Transferencia de ficheros desde/a MareNostrum

Si queréis copiar ficheros desde Boada a vuestra zona de disco en el PC del laboratorio o a vuestro PC o portátil privado podéis utilizar el comando scp para hacer copias remotas. El formato de scp es “scp origen destino” . en el caso de Marenostrum siempre debéis ejecutar el comando desde vuestra máquina local, nunca desde MareNostrum. También podéis utilizar sftp.

```
localsystem$ scp localfile username@mn1.bsc.es:
    username's password:
localsystem$ sftp username@mn1.bsc.es
    username's password:
    sftp> put localfile
```

Arquitectura

MareNostrum es un superordenador basado en procesadores Intel Xeon Platinum, racks de cómputo Lenovo SD530, sistema operativo Linux y una red de interconexión Intel Omni-Path.

A continuación, se muestran las características del sistema del clúster de propósito general (https://www.bsc.es/es/marenostrum/marenostrum/informacion-tecnica) :

- Rendimiento pico de 11.15 Petaflops
- 384.75 TB de memoria principal
- 3,456 nodos:

  - 2x Intel Xeon Platinum 8160 24C a 2.1 GHz
  - 216 nodos con 12x32 GB DDR4-2667 DIMMS (8GB/core)
  - 3240 nodos con 12x8 GB DDR4-2667 DIMMS (2GB/core)
- Redes de interconexión:
  - 100Gb Intel Omni-Path Full-Fat Tree
  - 10Gb Ethernet
- Sistema Operativo: SUSE Linux Enterprise Server 12 SP2

COMANDOS PRINCIPALES: MANIPULAR DIRECTORIOS Y FICHEROS

Los ficheros que vayamos generando los iremos organizando en carpetas o directorios. Un directorio en Linux es un fichero más de "tipo" directorio. Al conectarnos al sistema lo hacemos a nuestro directorio principal (asociado a nuestro usuario al crear la cuenta), y que se conoce como "*home directory*" o directorio raíz. No confundir con el directorio raíz del sistema, que corresponde con el punto de entrada común para todo nuestro sistema de ficheros.

¿Dónde estamos?

Para saber en cualquier momento donde estamos podemos usar el comando **pwd**. Por ejemplo.

```
nct00004@login1:~> pwd
/home/nct00/nct00004
```

En este caso es /home/nct00/nct00004, que es el directorio raíz del usuario nct00004.

Todos los ficheros de las sesiones os los copiaremos en la carpeta /gpfs/projects/nct00/nct00004. Deberéis copiarlos en vuestro directorio de trabajo y trabajar con vuestra copia.

¿Qué hay en mi directorio?

Para ver los ficheros que hay en un directorio usamos el comando ls. Generalmente todos los comandos Linux tienen 2-3 letras y suelen ser muy configurables. Para ver las opciones que ofrecen tenemos tres opciones básicas:

- Consultar el manual online: "man ls"
- Mirar la ayuda del propio comando: "ls -h" (a veces –help)

- Buscar información adicional en internet

Por ejemplo, con la opción -l tenemos detalles adicionales como los permisos de acceso, fechas, etc:

```
nct00004@login1:~/S1> ls -l
total 6
-rw-r--r-- 1 nct00004 nct00 553 feb  5 15:10 add.c
-rw-r--r-- 1 nct00004 nct00  75 feb  5 15:10 add_function.c
-rw-r--r-- 1 nct00004 nct00 101 feb  5 15:10 add_function.h
-rw-r--r-- 1 nct00004 nct00 464 feb  5 15:10 hello.c
-rw-r--r-- 1 nct00004 nct00 265 feb  5 15:10 Makefile
-rw-r--r-- 1 nct00004 nct00 731 feb  5 15:10 suma.c
```

En este caso vemos un directorio que contiene los ejemplo del T1 de la asignatura. Podemos observar varias columnas con datos: tipo de fichero y permisos de acceso, número de *hard links*, propietario, grupo, tamaño, fecha de la ultima modificación y nombre del fichero.

Es importante saber interpretar los permisos que tiene un fichero ya que es una de las principales causas de pérdida de tiempo: por ejemplo, intentar leer un fichero que no tiene permisos de lectura. Existen tres tipos de permisos asociados a tres dominios:

- Permisos de lectura (r=read), escritura (w=write), y ejecución (x=execute)
- Dominio: propietario (por defecto, el que ha creado el fichero), grupo al que pertenece el propietario, y resto de usuarios.

Para cada fichero, encontramos 10 letras en la primera columna. La primera indica el tipo de fichero que es (d=directorio, -=fichero datos, l=soft link, p=pipe, c=dispositivo tipo carácter, b=dispositivo tipo bloque). Luego encontramos 3 grupos de 3 caracteres. Cada grupo se refiere a un dominio (el orden es propietario, grupo y resto de usuarios), y cada una de las tres letras a los permisos de lectura, escritura y ejecución (por este orden). Si está la letra significa que el permiso está activado y si en esa posición aparece un – significa que no está activado. Por ejemplo "-rw-r--r—-" indica que es un fichero normal, que el propietario tiene permisos de lectura y escritura (rw-) y que el grupo y el resto sólo de lectura (r--).

Podéis obtener estos ficheros copiando de la carpeta /gpfs/projects/nct00/nct00004/S1 el fichero psd_s1.tar.gz. Ejecutad los siguientes comandos para copiar el paquete y descomprimirlo.

```
nct00004@login1:~$ cp /gpfs/projects/nct00/nct00004/S1/* .
nct00004@login1:~$ tar zxvf psd_s1.tar.gz
```

Creamos un directorio de trabajo y copiamos ficheros

Al extraer los ficheros del paquete se crea un directorio llamado S1. Dentro del directorio S1 crearemos un sub-directorio nuevo llamado sesion1_psd. Una vez dentro de la nueva carpeta, ejecutad el comando ls (opción -la).

```
nct00004@login1:~/S1> mkdir sesion1_psd
nct00004@login1:~/S1> cd sesion1_psd
nct00004@login1:~/S1/sesion1_psd> ls –la
total 2
drwxr–xr–x 2 nct00004 nct00 4096 feb  5 15:42 .
drwxr–xr–x 3 nct00004 nct00 4096 feb  5 15:42 ..
nct00004@login1:~/S1/sesion1_psd>
```

Las opciones de los comandos en Linux típicamente se acumulan. Las opciones -la incluyen la opción l (mostrar detalles) y a (mostrar ficheros ocultos).

Podemos observar los ficheros "." y "..", que son ficheros que se crean automáticamente al crear el directorio. Estos ficheros hacen referencia al propio directorio (fichero ".") y al directorio superior (fichero "..")

Copiad los ficheros del directorio *padre (..)* al actual

```
nct00004@login1:~/S1/sesion1_psd> cp ../* .
```

Siguiendo los comentarios de clase, arreglad los ficheros de forma que no den ningún error al compilarlos. Debéis poder compilarlos con el comando "make".

Comprobamos y modificamos nuestro entorno

Al conectarnos a la máquina, se nos asigna automáticamente un directorio, tenemos un usuario definido, una serie de valores por defecto, etc. Todos estos datos es lo que entendemos por nuestro entorno de trabajo. Esta configuración se puede modificar de forma permanente o puntual para una sesión de trabajo. El sistema utiliza unas variables de configuración que se llaman variables de entorno. Para modificarlas de forma temporal las modificaremos directamente en nuestra sesión de trabajo. Por ejemplo, la variable PATH indica los directorios donde el sistema buscará los ejecutables automáticamente. El comando echo muestra el valor de la variable y el comando export lo modifica. Con el siguiente ejemplo concatenamos el fichero "." al PATH (cada directorio está separado por ":".

```
nct00004@login1:~/S1/sesion1_psd> echo $PATH
/apps/modules/bsc/bin:/apps/INTEL/2017.4/impi/2017.3.196/bin64:/apps/I
NTEL/2017.4/bin:/home/nct00/nct00004/bin:/usr/local/bin:/usr/bin:/bin:
/usr/games:/usr/lpp/mmfs/bin:/home/nct00/nct00004/bin
nct00004@login1:~/S1/sesion1_psd> export PATH=.:$PATH
nct00004@login1:~/S1/sesion1_psd> echo $PATH
.:/apps/modules/bsc/bin:/apps/INTEL/2017.4/impi/2017.3.196/bin64:/apps
/INTEL/2017.4/bin:/home/nct00/nct00004/bin:/usr/local/bin:/usr/bin:/bi
n:/usr/games:/usr/lpp/mmfs/bin:/home/nct00/nct00004/bin
```

Si abres otra sesión y compruebas el valor de la variable verás que la modificación solo afecta a la sesión donde has hecho el cambio.

Consulta el valor de las variables de entorno siguientes: PATH, HOME, LD_LIBRARY_PATH, HOSTNAME, TMP, SHELL. Dependiendo del sistema, alguna de estas variables podría no estar definida.

Para hacer que el cambio afecte a todas tus sesiones y sea permanente, lo añadiremos al fichero de inicio de sesión. En bash existen varios ficheros que se interpretan al iniciar una sesión. Esos ficheros pueden contener todos los comandos que quieras que se ejecuten automáticamente cuando inicias la sesión. El fichero .bash_profile (si existe) se interpreta cuando se inicia un nuevo login shell (shell de entrada en la máquina) y el fichero .bashrc se interpreta (si existe) cuando se inicia un shell que no es de login. Habitualmente, se ponen en el .bash_profile las configuraciones necesarias sólo para el inicio de sesión y en el .bashrc las cosas comunes. Y entonces al final del .bash_profile se lanza la interpretación del .bashrc.

Edita ahora el fichero $HOME/.bashrc y añade al final del fichero la línea para extender el PATH con el directorio “.”. Comprueba cómo ahora la modificación se aplica para todas las nuevas sesiones. Abre una nueva sesión y comprueba el valor de la variable de entorno.

Para visualizar de forma rápida un fichero sin tener que abrirlo con el editor podéis utilizar el comando cat. (cat nombrefichero).

Meta-caracteres

Para manipular de forma mas sencilla los ficheros existen los metacaracteres, caracteres que se substituyen por otros (1 o N). Los dos más utilizados son:

- "*” (asterisco): Substituye a 0..N caracteres. Muy utilizado para manipular ficheros que tengan una parte común. Por ejemplo, si queremos que un comando concreto aplique solo a ficheros que acaban en .c (fuentes de programas en C), podemos ejecutar: comando *.c
- “?” (interrogante): Substituye a 0 o 1 carácter.

Borrado de ficheros, manipulación de permisos y alias.

Para borrar un fichero tenemos el comando rm. No hay opción para recuperar un fichero una vez borrado, así que es conveniente utilizar opciones como por ejemplo quitar los permisos de escritura de aquellos ficheros que puedan ser críticos o usar la opción “-i” para que el comando rm nos pida confirmación antes de borrar cada fichero. Esta segunda opción es bastante cargante si hemos de borrar muchos ficheros de golpe, pero puede ser recomendable en algunos casos.

Para cambiar los permisos de un fichero existe el comando chmod. El comando chmod acepta diferentes formatos pero uno bastante sencillo es indicar a que contexto queremos aplicar (u=propietario , g=group, o=others) y que permiso queremos añadir (+) o eliminar (-). Por ejemplo:

```
nct00004@login1:~/S1/sesion1_psd $ echo "esto es una prueba" >
prueba.txt
nct00004@login1:~/S1/sesion1_psd $ ls –l prueba.txt
–rw–r––r–– 1 nct00004 nct00 19 feb  5 15:50 prueba.txt
nct00004@login1:~/S1/sesion1_psd $ chmod g+w prueba.txt
nct00004@login1:~/S1/sesion1_psd $ chmod o–r prueba.txt
nct00004@login1:~/S1/sesion1_psd $ ls –l prueba.txt
–rw–rw––– 1 nct00004 nct00 19 feb  5 15:50 prueba.txt
nct00004@login1:~/S1/sesion1_psd $
```

Inicialmente el fichero prueba.txt tenia los permisos que pone el sistema por defecto: lectura y escritura para el propietario, y lectura para grupo y otros. Después de nuestras modificaciones, el fichero tiene permisos de lectura y escritura para el usuario y grupo y ninguno para el resto. Si eliminamos los permisos de escritura y intentamos borrarlo veremos que el sistema nos pregunta antes de borrarlo (al ser el propietario si lo deseas puedes borrarlo).

```
nct00004@login1:~/S1/sesion1_psd> chmod ug–w prueba.txt
nct00004@login1:~/S1/sesion1_psd> ls –l prueba.txt
–r––r––––– 1 nct00004 nct00 19 feb  5 15:50 prueba.txt
nct00004@login1:~/S1/sesion1_psd> rm prueba.txt
rm: ¿borrar el fichero regular 'prueba.txt'  protegido contra
escritura? (s/n) s
nct00004@login1:~/S1/sesion1_psd> ls –l
nct00004@login1:~/S1/sesion1_psd> ls –l prueba.txt
ls: no se puede acceder a 'prueba.txt': No existe el fichero o el
directorio
nct00004@login1:~/S1/sesion1_psd>
```

Otros comandos relacionados con chmod son chown y chgrp para cambiar el propietario y el grupo de un fichero respectivamente.

En Linux podemos definir alias de comandos: nuevos comandos que no son tal, sino que el sistema los substituye automáticamente por lo que hayamos indicado. Por ejemplo:

```
nct00004@login1:~/S3> alias rmi='rm –i'
nct00004@login1:~/S3> alias
alias rmi='rm –i'
```

A partir de ese momento, si escribimos rmi el sistema lo convertirá en “rm -i”. Para eliminar un alias previamente definido podemos usar el comando “unalias”. Por ejemplo, “unalias rmi”. Si queremos conservar un alias que nos interese podemos añadirlo al fichero $HOME/.bashrc.

```
nct00004@login1:~$ unalias rmi
```

Redirección de comandos

Aparte de los caracteres que hemos visto previamente, el intérprete de comandos tiene una serie de caracteres especiales que son interpretados antes de ejecutar la orden que estamos introduciendo. Especialmente útiles son los caracteres para conectar dos comandos y para modificar la entrada o la salida de un programa (es lo que se conoce como redireccionar).

Linux permite conectar fácilmente la salida de un programa con la entrada de otra, de forma que se pueden crear secuencias de filtros o de procesados fácilmente. Para conectar dos programas (o crear una secuencia) tenemos el carácter "|". También tenemos la opción de enviar (redirigir) la salida y/o entrada de un programa a un fichero mediante los caracteres ">" y "<". El primero redirige la salida y el segundo la entrada.

Por ejemplo, si queremos filtrar la salida de un programa y ver sólo las líneas que contentan un texto concreto, tenemos dos posibles opciones:

- Enviar la salida del programa a la entrada del comando grep (para buscar un texto en una

```
nct00004@login1:~/S1> ./add 1 2
hello world!
This program add two numbers
I'm ./add and my arguments are 1 and 2
The result is 3
nct00004@login1:~/S1> ./add 1 2|grep result
The result is 3
nct00004@login1:~/S1>
```

- Almacenar la salida del programa en un fichero de datos y buscar luego el texto en ese fichero utilizando el comando grep. La diferencia con la opción anterior es que no utilizamos espacio en disco ya que los datos van directos del primer programa al segundo.

```
nct00004@login1:~/S1> ./add 1 2 > salida
nct00004@login1:~/S1> grep result salida
The result is 3
nct00004@login1:~/S1>
```

Análisis de aplicaciones

Durante el curso tendremos que medir el tiempo de ejecución de nuestros programas de diferentes formas, con más o menos detalles. Hay un par de comandos que nos dan el tiempo y algún dato adicional. Por ejemplo, el comando time nos da el tiempo de usuario(user), de sistema (sys) y el total del programa (real) que estemos ejecutando.

```
nct00004@login1:~/S1> time sleep 1

real 0m1.012s
user 0m0.005s
sys  0m0.000s
nct00004@login1:~/S1>
```

En el ejemplo anterior hemos ejecutado el comando sleep que recibe como parámetro el número de segundos a esperar. El comando time no tiene una precisión muy alta, máximo mili segundos, por lo que si necesitamos medir algo más rápido habrá que utilizar otro comando o realizar N ejecuciones y después calcular la media.

Para obtener más información sobre la ejecución de nuestros programas mientras se están ejecutando, podemos consultar los datos que gestiona el sistema bien usando el comando ps (para un proceso concreto) o bien consultando en los ficheros disponibles bajo la carpeta /proc/[PID], donde [PID] es el identificador del proceso que queremos consultar. Por ejemplo, si ejecutamos a modo de ejemplo el comando “sleep 100”, y sabemos que su pid es 60947, podemos obtener información básica del programa mientras se ejecuta con la opción ps -p 60947.

```
nct00004@login1:~/S1> sleep 100 &
[1] 60947
nct00004@login1:~/S1> ps –p 60947
   PID TTY          TIME CMD
 60947 pts/18   00:00:00 sleep
nct00004@login1:~/S1> cat /proc/60947/status
Name:    sleep
State:   S (sleeping)
Tgid:    60947
Ngid:    0
Pid: 60947
PPid:    96955
TracerPid:    0
Uid: 9204 9204 9204 9204
Gid: 16440    16440    16440    16440
FDSize:  256
Groups:  905 909 16440
NStgid:  60947
```

También podemos consultar el fichero status (como en el ejemplo), stat o cualquiera de los que hay bajo /proc/[PID]. El comando ps obtiene la información de los ficheros que hay en ese directorio, que lo crea y gestiona el sistema operativo. Esos ficheros SOLO existen mientras existe el proceso, por lo que no sirven para consultar datos una vez termina. También existen SOLO en el sistema donde se están ejecutando, en la mayoría de sistemas remotos no se puede utilizar como mecanismo para obtener información.

Consulta las opciones del comando ps para ver qué posibilidades ofrece y busca también información que relacione esos datos con los ficheros existentes en /proc/[PID].

Monitorización del sistema

Algunos comandos que visteis en la asignatura COM como ps, top, vmstat, free etc son especialmente útiles para ver los programas (procesos) que están en ejecución desde el punto de vista de un nodo. Sin embargo, en los sistemas paralelos formados por N nodos interconectados, no sirven para tener una visión global del sistema.

Edición de programas

En los PCs del laboratorio hay instalado un SO OpenSuSe con un entorno gráfico, por lo que podréis encontrar diferentes editores de texto. Sin embargo, las máquinas a las que accedéis de forma remota no están pensadas para usar esas herramientas. Podéis utilizar el editor que queráis, pero aquí os dejamos una explicación básica del editor vi que podéis encontrar instalado en cualquier máquina Unix/Linux.

El editor vi es un editor tipo texto muy potente pero diferentes a lo que estáis acostumbrados. Cuando editáis un fichero "vi nombre_fichero" el fichero se abre en modo comando. El programa vi tiene dos modos:

- Modo comando: las teclas que apretáis no se escriben, sino que son comandos que el editor interpreta como copiar líneas, buscar texto, borrar líneas, etc. Sería el equivalente a usar los menús de un editor gráfico.
- Modo inserción: las teclas se insertan, no se interpretan.

Para pasar de modo inserción a modo comando hay que apretar la tecla "ESC". Las letras/órdenes básicas que podemos usar en modo comando son:

- Pasar a modo inserción
  - 'i' → en el punto donde está situado el cursor
  - 'a' → en el justo añadiendo en la posición detrás de donde está el cursos
  - 'I' → al inicio de línea
  - 'A' → al final de línea
  - 'o' → inserta una línea debajo de donde estamos
  - 'O' → inserta una línea encima de donde estamos
- Busquedas
  - '/texto_a_buscar'
- Copiar y pegar
  - 'ynumlineasy' → Copia "numlineas". Por ejemplo, y10y copia 10 líneas
  - 'p' → pega las líneas que tenemos copiadas
  - 'dnumlineasd' → Borra numlineas". Por ejemplo, d10d borra 10 líneas
- Ir a:

    - ‘: numlinea’ → por ejemplo :100 va a la línea 100, :$ va al final del fichero
- Deshacer:
    - ‘u’ → deshace el ultimo cambio
- Guardar, salir, etc
    - ‘:q!’ → sale sin guardar
    - ‘:wq’ → sale guardando los cambios
    - ‘:w nombre fichero’ → guarda con otro nombre

Resultado de esta sesión

Para cada sesión generaremos algún tipo de información adicional resultado de los ejercicios que habremos ido haciendo. No debéis entregarlo, pero se os podría pedir a final de curso como información complementaria para valorar la asignatura.

En este caso debéis crear un documento (formato libre) con una lista de todos los comandos que hemos trabajado en esta sesión con una breve descripción de su funcionalidad, así como las opciones más útiles. Si habéis buscado información en internet o ejemplos, adjuntad los enlaces.

SESIÓN 2: ENTORNOS DE EJECUCIÓN PARA HPC

Los entornos de ejecución para HPC tienen características que es necesario conocer.

Configuración de tu entorno de usuario

MareNostrum utiliza un sistema de configuración del entorno de usuario basado en módulos de usuario. Los módulos de usuario permiten tener un número limitado de paquetes configurados por defecto y que cada usuario elija que librerías necesita, versiones concretas, etc. Los módulos básicamente configuran variables de entorno como PATH o LD_LIBRARY_PATH que indican la lista de directorios por defecto para binarios y librerías. Para ver la lista de módulos disponibles podéis el ejecutar el comando:

```
nct00018@login2:~> module avail
-------------------- /apps/modules/modulefiles/applications
3DNA/2.3     gate/8.1.p01  (D)    plumed/2.3.2   (D)ARNOLD/mtoa_2.0.1
gdl/0.9.7    plumed/2.3.3_libmatheval /1.7.2 geant4/9.5
plumed/2.4.1_devel CDO/1.8.2_ts geant4/9.5.p01 plumed/2.4.1
CDO/1.8.2 geant4/9.6.p01 plumed/2.4.2 CDO/1.9.3 (D) geant4/10.03.p01
plumed/2.5.0
…
---------------------- /apps/modules/modulefiles/compilers
gcc/4.8.5 gcc/7.2.0 (D)intel/2017.4 (L,D) intel/2018.1 java/latest
java/8u144
   gcc/4.9.4    gcc/8.1.0    intel/2017.6          intel/2018.2    java/6u45
java/8u151
…  ------------------------ /apps/modules/modulefiles/libraries
   HYPRE/2.11.2 libgif/5.1.4                  petsc/3.1-p8-real
   arpack/96  libjpeg-turbo/1.5.2           petsc/3.4.3-complex
   atlas/3.10.3-gcc8.1          libpng/1.5.13        (D)      petsc/3.4.3-real
---------------------- /apps/modules/modulefiles/libraries
   HYPRE/2.11.2  libgif/5.1.4                   petsc/3.1-p8-real
   arpack/96      libjpeg-turbo/1.5.2           petsc/3.4.3-complex
   atlas/3.10.3-gcc8.1  libpng/1.5.13        (D)      petsc/3.4.3-real
```

Para ver los módulos que tenemos cargados en un determinado momento podemos usar “module list” y para eliminar de nuestro entorno todos los módulos podemos usar “module purgue”.

En MareNostrum utilizaremos el entorno Intel tanto de compilador como de librería. Para ello cargaremos los siguientes módulos. Podéis consultar el valor de las variable LD_LIBRARY_PATH y PATH para ver como se ha modificado.

```
bsc19620@login1:~> module list
Currently Loaded Modules:
  1) intel/2017.4   2) impi/2017.4   3) mkl/2017.4   4) bsc/1.0
bsc19620@login1:~> module load intel/2018.4
Set INTEL compilers as MPI wrappers backend
bsc19620@login1:~> module load impi/2018.4
load impi/2018.4 (PATH, MANPATH, LD_LIBRARY_PATH)
bsc19620@login1:~> module load mkl/2018.4
load mkl/2018.4 (LD_LIBRARY_PATH)
bsc19620@login1:~> echo $LD_LIBRARY_PATH
```

```
/apps/INTEL/2018.4.057/mkl/lib/intel64:/apps/INTEL/2018.4.057/impi/201
8.4.274/lib64:/apps/INTEL/2018.4.057/lib/intel64:/home/bsc19/bsc19620/
NEST/lib64:/home/bsc19/bsc19620/NEST/lib64:/usr/lib:/usr/lib/gtk-
2.0::/usr/lib:/usr/lib/gtk-2.0::
```

Estos tres módulos nos puede interesar tenerlos cargados por defecto, así que añadidlos al fichero .bashrc en vuestro directorio raiz.

Ejecución de programas

MareNostrum utiliza el sistema de colas SLURM (https://slurm.schedmd.com/quickstart.html), uno de los sistemas más utilizados en centros HPC. En cuanto a las colas, tenéis asignadas debug, interactive y training. Podéis consultar los detalles mediante el comando.

```
bsc_queues
```

Copia los ficheros que puedes encontrar en /gpfs/projects/nct00/nct00004/S2. Compila el fichero hello_world_mpip.c para tener el fichero ejecutable `hello_world_mpi` (`utiliza el Makefile`). MN utiliza el sistema de colas de SLURM. Para ejecutar un trabajo en los nodos de computación utilizaremos comandos que indican a SLURM lo que necesitamos y lo que queremos hacer. El comando que utilizaremos en nuestro case es sbatch (*sbatch fichero_trabajo*).

Sbatch recibe una lista de requerimientos bien sea por parámetro o definidos dentro del propio script. Podéis ejecutar 'man sbatch' o 'sbatch –help' para tener más información sobre sbatch.

Un ejemplo sencillo de script para un ejecutar un programa MPI (solo MPI) sería el siguiente. Crea un fichero mpi_test.sh con el siguiente contenido y ejecútalo usando sbatch.

```
#!/bin/bash
#SBATCH --job-name=mpi_test
#SBATCH --output=mpi_%j.out
#SBATCH --error=mpi_%j.err
#SBATCH --nodes=1
#SBATCH --ntasks=4
#SBATCH --qos=debug

srun ./hello_world_mpi
```

```
nct00018@login2:~> sbatch ./mpi_test.sh
```

Consulta en el manual que significan cada una de las opciones que hemos utilizado. En este ejemplo, estamos pidiendo 4 procesos MPI. ¿Cuál es el significado del símbolo "%j" como parte del nombre del fichero de salida y de error?

Consulta la utilidad de los comandos squeue y scancel (“man squeue” o “squeue -help”, y “man scancel” o “scancel --help”).

Modifica el programa mpi_test.sh de forma que puedas rellenar la siguiente tabla. El objetivo es entender la asignación de nodos que hace SLURM

Nodos pedidos	Procesos creados (––ntasks **Y**)	Lista nodos asignados	Rango procesos -nodo
1	1		
1	8		
1	16		
1	32		
1	64		
2	16		
2	32		

¿Qué ha pasado al ejecutar la combinación 1 nodo y 64 procesos? Razona el mensaje obtenido.

¿Qué script deberíamos ejecutar si no tuviéramos la información técnica de los nodos de coómputo de MareNostrum IV y quisiéramos obtener nosotros mismos los detalles de la arquitectura?

- El comando ‘cat /proc/cpuinfo’ nos da esta información, pero se ha de ejecutar en un nodo de computación

En el mismo directorio podéis encontrar un ejemplo de multiplicación de matrices matrix_mul.c. Generad el binario (usando el Makefile) y cread un fichero matrix_mul.sh similar al anterior pero que ejecute el binario matrix_mul. Este programa puede utilizar también diferentes números de procesos. El programa matrix_mul genera el tiempo de ejecución que ha utilizado. Rellena la siguiente tabla (tened en cuenta que matrix_mul genera debe ser ejecutado con workers+1 procesos:

Nodos pedidos	Procesos workers creados	Tiempo ejecución	Speedup	Eficiencia
1	1			
1	2			
1	4			
1	8			
2	2			
2	4			
2	8			
2	16			

Ejecución de kernels

Para familiarizaros con el entorno HPC, coged el fichero NPB3.4-MZ.tar.gz que contiene diferentes carpetas con kernels típicamente utilizados en HPC. Estos kernels se puede configurar con diferentes clases (tamaño de datos) y número de procesos. Además, incluyen diferentes versiones: MPI, MPI+OpenMP, etc.

Para que veáis el impacto de algunas opciones, compilaremos y ejecutaremos algunos kernels de la versión MPI-OpenMP que podemos encontrar en la carpeta NPB3.4-MZ/NPB3.4-MZ-MPI. En este caso, ejecutaremos diferentes combincationes de MPI y OpenMP. En primer lugar compilaremos los kernels.

- Copiad el fichero config/suite.def.template en config/suite.def. Este fichero define lo que queremos compilar por defecto al hacer "make suite". Modificad los casos que se generan por defecto para que se compilen los tres kernels (bt-mz,sp,mz y lu-mz) con la clase C. El contenido del fichero debe ser como este:

```
sp-mz   C
bt-mz   C
lu-mz   C
```

- Copiad el fichero config/make.def.template en config/make.def. Este fichero nos permitirá cambiar el compilador, opciones de compilación etc. Cambiad la definición de F77, FFLAGS, FLINKFLAGS, CC, CFLAGS y CLINKFLGAS por los siguientes valores:

```
FC = mpiifort
FFLAGS  = -O3 -qopenmp
CC = mpiicc
CFLAGS  = -O3 -qopenmp
```

- Ejecutad "make suite" para generar un primer conjunto de kernels por defecto. Al acabar, deberíais tener los siguientes kernels en la carpeta bin

```
bt-mz.C.x  lu-mz.C.x  sp-mz.C.x
```

- El fichero bt.sh es un ejemplo de cómo ejecutar el kernel bt-mz.C con 1 nodo, 12 procesos, y 1 thread. Las opciones "nodes, ntasks, y cpus-per-task" permiten especificar al sistema de colas los recursos que queremos. La variable de entorno "OMP_NUM_THREADS" permite especificar cuantos threads por proceso queremos utilizar. Esta variable de entorno es leída por la librería de OpenMP, si la definimos con un valor incorrecto, podemos crear un número de threads diferente al que hemos pedido y crear un conflicto que influya en rendimiento de nuestro programa. El comando srun es un comando de slurm para poner en ejecución programas paralelos. Si no especificamos nada, utilizará las mismas opciones que hemos especificado en el script de sbatch.

- Este script (fichero con comandos) se ejecuta utilizando el comando sbatch. Recordad que no se puede ejecutar directamente, hay que "enviarlo" a la cola con sbatch.

```
nct00018@login1:~/sesion2/psd/sesion2> cat bt.sh
#!/bin/bash
#SBATCH --job-name=bt-mz
#SBATCH --output=bt-mz_%j.out
#SBATCH --error=bt-mz_%j.err
#SBTACH --nodes=1
#SBATCH --ntasks=12
#SBATCH --cpus-per-task=1
#SBATCH --qos=debug

# Modificad esta linea con vuestro PATH a los ejecutables
export NAS_PATH=VUESTRO_PATH
export OMP_NUM_THREADS=1
srun $NAS_PATH/bt-mz.C.x


nct00018@login1:~/sesion2/psd/sesion2>sbatch bt.sh
```

- Modificad (o cread uno nuevo) el script bt.sh para poder realizar las siguientes ejecuciones y rellenar la siguiente tabla para bt-mz , sp-mz y lu-mz:

Total cores	Nodos pedidos	Procesos (tasks)	Threads	Tiempo ejecución	Speedup	Eficiencia
1	1	1	1			
48	1	12	4			
48	1	48	1			
96	2	12	8			
96	2	48	2			

Documentación a generar como resultado de esta sesión

El objetivo de esta sesión y la previa es familiarizarse con el entorno de MareNostrum IV, especialmente con los comandos que necesitáis para ejecutar vuestros trabajos. También empezar a tener una intuición del impacto en el tiempo de ejecución de algunas configuraciones.

Como resultado de esta sesión deberíais tener una mini guía de los comandos básicos en ambas máquinas.

Enlaces

[1] Top500. https://www.top500.org/list/2018/11

[2] Descripción MN4 en Top500. https://www.top500.org/site/49748

[3] MPICH https://www.mpich.org/

[4] MareNostrum IV. User documentation https://www.bsc.es/user-support/mn4.php