Ejercicio 1: Envío de trabajos

- Editar y enviar la tarea simple.sh a ejecutar en el clúster de docencia (moore.udl.net)
  1. Editar el trabajo: simple.sh

```
#!/bin/sh
#
echo "Este código se ejecuta en “‘/bin/hostname`
/bin/date
/bin/sleep 20
/bin/date
```

  2. Copiar el trabajo en vuestra cuenta del front-end del cluster de docencia
     - `scp simple.sh usuario@moore.udl.net:.`
  3. Lanzar el trabajo utilizando qsub:
     - `ssh usuario@moore.udl.net`
     - `qsub simple.sh`
  4. Lanzar el trabajo utilizando qsub y qmon
     - `ssh -X usuario@moore.udl.net`
     - `qmon &`
- Entrega: Script trabajo, ficheros de salida y captura pantalla qmon

Ejercicio 2: Opciones Trabajos

- Editar y enviar la tarea avanzado.sh a ejecutar en el clúster de docencia (moore.udl.net)
  1. Crear el script del trabajo (avanzado.sh) para que realice un listado recursivo de vuestro directorio actual.
  2. Especificar las siguientes opciones
     - Nombre Trabajo: Avanzado_**NombreAlumno**
     - Cola de trabajos: all.q
     - Juntar stdout y stderr
     - Mínimo memoria virtual: 256 MegaBytes
     - Enviar una notificación a vuestro correo cuando el trabajo arranque , finalice o aborte
     - Ejecutar el trabajo desde el directorio actual
  3. Copiar el trabajo en vuestra cuenta del front-end del cluster de docencia
     - `scp avanzado.sh usuario@moore.udl.net:.`
  4. Lanzar el trabajo utilizando qsub ó qmon
- Entrega:
  - Script del trabajo.
  - Ficheros de salida del trabajo
  - Fichero verificación del lanzamiento de vuestro scripts:
    - `qsub –verify avanzado.sh > TuNombre_job.txt`

Ejercicio 2b: Opciones Trabajos

- Editar y enviar la tarea avanzado2.sh a ejecutar en el clúster de docencia (moore.udl.net)
  5. Modificar el trabajo para que solicite 5 horas de tiempo de ejecución.
  6. Lanzar el trabajo utilizando qsub ó qmon
  7. Si no se ejecuta el trabajo, analizar cual puede ser la causas.

     Podéis utilizar el siguiente comando para obtener información útil que os ayude a detectar el problema:
     - `qconf -sql`
     - `qconf -sq all.q`
- Entrega:
  - Justificar cuales son las razones que no permiten que el trabajo modificado se pueda ejecutar y como lo solventaríais.

Ejercicio 3: Monitorización Trabajos

- Ejecutar de nuevo el script del ejercicio1 y monitorizar su ejecución
  1. Modificar el script para que tarde más tiempo en ejecutarse, cambiando el parámetro del sleep de 20 segs a 120 segs, por ejemplo.
  2. Lanzar el trabajo a ejecutar en el SGE
  3. Monitorizar la ejecución:
     a) De todos los trabajos del sistema (sin opciones)
     b) De vuestros trabajos (opción –u)
     c) Obtener información detallada del trabajo lanzado (opción –j)
  4. Repetir el paso 3 pero utilizando qmon.
  5. Después de la ejecución, consultar su utilización de recursos mediante qacct
     - `qacct -j nro_trabajo`
  6. Volver a lanzar el trabajo, esperar que inicie su ejecución, eliminarlo y repetir el paso 5
     - `qdel rro_trabajo`
- Entrega:
  - Ficheros volcado de los comandos de monitorización para los pasos 3, 5 y 6

Ejercicio 4: Matriz de Tareas

- Utilizar la aplicación *SumatorioPthreads.c* para calcular:

  For i=100.000.000 to 1000.000.000 step 100.000.000

  $$\sum_{j=1}^{i} j$$

  EndFor

  - Uso: SumatorioPthreads <max_number> <num_hilos>
  - Ejemplo:
    - `SumatorioPthreads 100 2 # Suma 100 primeros números con 2 hilos`
  - Utilizar el comando times para obtener el tiempo de ejecución.
  - Variar el número de hilos de 1 a 10 para cada prueba.
- Pasos:
  1. Compilar el programa SumatorioPthreads
  2. Definir el script del trabajo para la matriz de tareas
  3. Lanzar el trabajo utilizando qsub
  4. Monitorizar su ejecución.
- Entrega:
  - Script del trabajo
  - Fichero monitorización ejecución
  - Ficheros con los resultados (los resultados con diferentes hilos se guardan en el mismo fichero)

Ejercicio 5a: Trabajos Paralelos

- A partir de la matriz de trabajos del ejercicio anterior:
  1. Revisar los tiempos obtenidos por los trabajos de la matriz de tareas del ejercicio anterior.
     - ¿Son correctos/lógicos? ¿Se obtiene el mismo speed-up que en el front-end?
       - $ time ./SumatorioPthreads 1000000000 1
         real 0m2.380s
       - $ time ./SumatorioPthreads 1000000000 4
         real 0m0.639s
  2. Si no es así analizar cuales podrían ser las causas del menor de rendimiento.
  3. Modificar el script, con los atributos vistos en el último apartado, para asegurarse que se obtiene el rendimiento correcto.
- Entrega:
  - Justificación de la disparidad de resultados obtenidos.
  - Script del trabajo modificado
  - Ficheros con los resultados (los resultados con diferentes hilos se guardan en el mismo fichero)

Ejercicio 5b: Trabajos Paralelos

- Lanzar la ejecución de una aplicación paralela en el cluster de docencia:
  1. Compilar programa paralelo (en el cluster)
     - Descargar fuente programa paralelo cpi.c
     - Compilar programa paralelo:
       - `mpicc cpi.c –o cpi`
  2. Definir el script del trabajo paralelo:
     - Utilizar como guía alguno de los script paralelos que se os proporciona en la web del cluster: http://moore.udl.net/wordpress/?page_id=37
     - Modificar:
       - Nombre programa paralelo a ejecutar: cpi
         - `mpiexec -f $MPICH_MACHINES -n $NSLOTS <ruta_ejecutable>`
       - Cola de trabajos a utilizar: all.q,
       - Número de procesadores a utilizar: 1x4, 2x4, 4x4, y 8x4 (cola all.q) nodos
       - Utilizar el comando ***time*** para obtener el tiempo de ejecución de la aplicación paralela
  3. Lanzar el trabajo paralelo
  4. Monitorizar su ejecución y analizar los tiempos de ejecución finales.
- Entrega:
  - Script del trabajo
  - Fichero monitorización ejecución
  - Gráfica prestaciones