Gestor de colas

De Lab40

Para la instalación de un gestor de colas es necesario contar con una distribución linux y con un gestor de colas. En este caso utilizamos Rocks para el sistema base y Open Grid Scheduler (Grid Engine) como gestor de colas.

Índice

Configuración del frontal:

Pasos a seguir para instalar sistema base:

```
# Arrancamos desde CD con la distribución Rocks.
# En la consola de la página de incio de la distribución escribimos build y pulsamos Enter.
# Seleccionamos Introducir rolles desde CD.
# Marcamos los siguientes roles: base, hpc, java, kernel, os, sge y web-server. Pulsamos Submit.
# Pulsamos Next.
# Cubrimos la información del cluster y pulsamos Next.
# Pulsamos Next.
# Introducimos los datos de la red por la que el equipo saldrá a Internet (sólo en caso no no tener un servidor
y cambiamos la máscara de la subred para tráfico interno a 255.255.255.0. Pulsamos Next.
# Pulsamos Next.
# Introducimos la contraseña de root y pulsamos Next.
# Cambiamos la configuración horaria, en nuestro caso concreto seleccionamos Madrid. (MUY IMPORTANTE!!!!!!).
# Pulsamos Next.
# Seleccionamos Auto partitioning y pulsamos Next.
```

Pasos a seguir una vez instalado el sistema base:

```
# Cambiar la configuración del teclado.
# Crear imagen para los nodos.
```

Pasos a seguir para crear una imagen para los nodos:

```
# Abrir terminal.
# Introducir el comando rocks create distro y pulsar Enter.
```

Configuración de los nodos

Pasos a seguir para instalar un nodo de cálculo:

```
# En la consola del Front poner insert-ethers
# Luego seleccionar Compute
# Arrancamos el nodo que deseamos insertar (tiene que estar conectado en red privada con el Front)
# Ahora tenemos dos posibilidades, la primera es hacer la instalación del nodo nuevo mediante el CD o
  mediante PXE a través del Front.
# Luego en el Front cuando localice el nodo pulsar Enter y listo.Ya procede a la instalación del nodo de cálculo
```

Comandos

Comandos utilizados en las pruebas del gestor de colas:

Los comandos básicos para trabajar con el gestor de colas son:

```
# qstar: muestra los trabajos que hay en las colas.
# qhost: muestra los nodos disponibles en el gestor de colas.
# qconf -sql: muestra la lista de todas las colas.
# qconf -aq nombreCola: crea una cola de nombre nombreCola.
# qdel nombreTrabajo: borra el trabajo con nombre nombreTrabajo.
# qhold nombreTrabajo: pausa el trabajo con nombre nombreTrabajo.
# qrls: reactiva los trabajos pausados.
# qacct: muestra los trabajos terminados.
# qsub nombreTrabajo: envia el trabajo con nombre nombreTrabajo.
# qacct -j idTrabajo: muestra los datos del trabajo con id idTrabajo.
```

El resto de comando disponibles se pueden consultar en el siguiente enlace:
http://gridscheduler.sourceforge.net/htmlman/

Scripts de pruebas

Scripts sencillos:

Script Ejemplo1.sh trabajo que permanece durante 20 segundos ejecutándose:

```
# cat Ejemplo1.sh
# sleep(20);
# qsub Ejemplo1.sh
```

Script Ejemplo2.sh trabajo que vuelca en un fichero con nombre el mismo que el del propio script, la cadena "Hola soy " y su propio nombre.

```
# cat Ejemplo2.sh
# sleep 2
  echo "Hola soy " $0 > /var/compartida/`basename $0`.txt
# qsub Ejemplo2.sh
```

Script Ejemplo3.sh trabajo que ejecuta un programa escrito en C y guarda el resultado en un fichero de texto.

Programa en C para calcular el número PI:

```
#cat Ejemplo3.c
 #include <stdio.h>
 #include <stdlib.h>
 int main() {
  long int npts = 11000000;
  long int i;
  double f,sum;
  double xmin,xmax,x;
  xmin = 0.0;
  xmax = 1.0;
  for (i=0; i<npts; i++) {
    x = (double) rand()/RAND_MAX*(xmax-xmin) + xmin;
    sum += 4.0/(1.0 + x*x);
  }
  f = sum/npts;
  printf("PI calculated with %ld points = %f \n",npts,f);
 }
```

Script que compila y ejecuta el program en C y vuelca el resultado en un fichero de texto.

```
# cat Ejemplo3.sh
# gcc ejemplo6.c -o  pi
  ./pi >res.txt
#qsub Ejemplo3.sh
```

Script Ejemplo4.sh: trabajo que compila y ejecuta un programa MPI escrito en C.

Programa en C:

```
#cat Ejemplo4.c
 #include <stdio.h>
 #include <stdlib.h>
 #include "mpi.h"
   int main(int argc, char *argv[]) {
   int myid,nprocs;
   long int npts = 1e10;
   long int i,mynpts;
   double f,sum,mysum;
   double xmin,xmax,x;
   MPI_Init(&argc,&argv);
   MPI_Comm_size(MPI_COMM_WORLD,&nprocs);
   MPI_Comm_rank(MPI_COMM_WORLD,&myid);
   if (myid == 0) {
     mynpts = npts - (nprocs-1)*(npts/nprocs);
   } else {
     mynpts = npts/nprocs;
   }
   mysum = 0.0;
   xmin = 0.0;
   xmax = 1.0;
   srand(myid);
   for (i=0; i<mynpts; i++) {
     x = (double) rand()/RAND_MAX*(xmax-xmin) + xmin;
     mysum += 4.0/(1.0 + x*x);
   }
   MPI_Reduce(&mysum,&sum,1,MPI_DOUBLE,MPI_SUM,0,MPI_COMM_WORLD);
   if (myid == 0) {
     f = sum/npts;
     printf("PI calculated with %ld points = %f \n",npts,f);
   }
```

```
  MPI_Finalize();
}
```

Script que compila el programa en C para MPI:

```
# cat Ejemplo4.sh
 ##!/bin/bash
 #$ -q all.q
 #$ -pe orte.pe 3
     #
     # ...specify the SGE "parallel.q" queue and **also** the "orte.pe" PE...
     #
 #$ -cwd
 #$ -S /bin/bash
     #
     # ...ensure the OpenMPI-related executables and libaries can be found by
     #    the job...
     #
 mpirun -np $3 ./Ejemplo4.c
     #
     # ...start the job!  "$NSLOTS" is the number of cores/processes/slots the
     #    job will use --- the value is set by the orte.pe PE.
     #
     #    *** N.B. We have not specified a host/machinefile here --- OpenMPI picks
     #             this up automagically from the PE. ****
     #
#qsub Ejemplo4.sh
```

Traído desde "http://wiki.lab40.esei/index.php?title=Gestor_de_colas&oldid=132"
