Soumission de job sur MUST

Jobs batch - LAPP & LAPTH uniquement

- Quelques exemple de jobs
  - Simple
  - Avec compilation
  - Avec rapatriement de données
  - De type MPI

- L’ensemble des exemple peut être récupéré depuis lappsl ici:

  /home1/elles/public/batch-LAPP-LAPTH-MUST/MUST-qsub.tar

Soumission de base : testPrintenv.sh

- Simplement lance les commandes printenv, df –h et visualise /proc/cpuinfo
- Attend 2 minutes avant de sortir.
- Intéressant de voir dans le résultat les variables d'environnements (notamment MPI) qui existent.

```
#!/bin/sh
##testpbs
echo This is a test
echo Today is `date`
echo This is `hostname`

echo "*************** Printenv **************"
printenv

echo "*************** Disk df **************"
df -h

echo "*************** cat /proc/cpuinfo
      **************"
cat /proc/cpuinfo

#so you have time to see it in the queue
sleep 2m
```

Soumission de base : testPrintenv.sh

Lappsl > qsub

Envoie le job sur le jobmanager (lapp-ce01)

Lapp-ce01> ordonnance le job et lorsqu'il est éligible l'affecte a une worker node (lapp-wnxxx)

Lapp-wnxxx> exécute le job et retorune les sdt output et error.

Commande de contrôles

Showq

Showq –r

Showq –i

Showq –b

Permet de voir la position des jobs au niveau du scheduler et leur priorité

Soumission : testbench.sh

- Un vrai bench
- Le répertoire unixbench-4.1.0 contient les sources du benchmark
- Copie les sources du bench (depuis le home utilisateur partagé par tout le cluster) vers une zone disque propre au worker sur lequel le job est arrivé
- Compile le code
- Lance le code

Ce bench prend un certain temps

```
#!/bin/sh
echo Today is `date`
echo This is `hostname`

# copy the source of the bench from the
      sharing filesystem to the the local disk (
      usage of the TMPDIR)

cp -rf $PBS_O_HOME/unixbench-4.1.0/
      $TMPDIR

# Compile the program

cd $TMPDIR/unixbench-4.1.0
gmake

# run the program ( can take long time)
Run
```

Soumission : testbench.sh

- Même schema que job #1
- Ici on pourrait se passer de copier les sources sur le disque local de la machine.
- Mais les disques locaux sont beaucoup plus performants que le système de fichier partagé.

Option de qsub

Qsub –m aeb permet de recevoir un mail lorsque le job :

Abord (a)

Commence (b)

Fini (e)

Le mail contient des info sur la consommations des ressources.

Le mail ne peut pas être défini par l'utilisateur c'est celui rattaché au compte unix.

Soumission : testData.sh

- Répertoire recupdata
- Ce job récupère un fichier distant ainsi que son checksum
- Recalcule le checksum du fichier récupéré
- La copie se fait par scp via un échange de clefs
- Cela nécessite que la machine servant les data est configurée pour accepter la clef qu'on lui propose

```
#!/bin/sh

# Make a directory on the tmpdir of the worker

MYTMP=$TMPDIR/fede
mkdir $MYTMP

# retrieve input file from remote host via scp and key sharing

echo "";
echo "Retrieve input file from remote host ";

KEYPATH=$HOME/recupdata/id_rsa

scp -i $KEYPATH
        fede@marseilleui.mrs.grid.cnrs.fr:TEST/test4uds
        $MYTMP
scp -i $KEYPATH
        fede@marseilleui.mrs.grid.cnrs.fr:TEST/test4uds_md5
        $MYTMP

# Show the md5sum of the file

echo "MD5SUM given by remote site is : "
cat $MYTMP/test4uds_md5
echo "";

# compute md5 of the file retrieving

echo "Compute MD5SUM of file"
echo "MD5SUM of retrieving file is :"
md5sum $MYTMP/test4uds

rm -rf $MYTMP
```

Soumission : testData.sh

- Même schéma que job #1
- Dans ce cas de figure il est avantageux de copier les data sur le disque local ou tourne le job plutôt que sur la zone partagé.

  Car le bench est susceptible de d'accéder de façon importante aux données.

Commande de contrôle

Diagnose –f

Diagnose –g

Diagnose –j #job

….

Commande utiles pour acquérir des informations sur le système d'ordonnancement en particulier sur les paramètre permettant de définir les priorités

Soumission : Job MPI (MPIhostname.sh)

- Répertoire mpi
- Fait un simple bonjour depuis chacun de processeur mis en jeu
- Compile le code (ici *hostname.c*)
- Le lance via mpirun
- Pour lancer une telle commande il faut faire

  qsub –lnodes=8

  Afin de requérir 8 « cpu »

```
#!/bin/sh
#
# this parameter is the binary to be executed

EXE="$PBS_O_HOME/mpi/hostname"

# this parameter is the number of CPU's to be reserved for
        parallel execution
CPU_NEEDED=8

$MPI_OPENMPI_PATH/bin/mpicc -o $EXE
        $PBS_O_HOME/mpi/hostname.c

cat $PBS_NODEFILE >> $PBS_O_HOME/mpi/info

chmod 755 $EXE


echo "**********************************" >>
        $PBS_O_HOME/mpi/info
$MPI_OPENMPI_PATH/bin/mpirun -np $CPU_NEEDED
        $EXE

echo "**********************************" >>
        $PBS_O_HOME/mpi/info
```

Soumission : Job MPI

- Deux autres jobs de type MPI sont disponibles.
- Les exemples génèrent également deux fichiers nodefile et/ou info qui contient des informations supplémentaires.
- A noter que le binaire est généré dans zone partagé (dans le home de l’utilisateur), pour être partagé par l’ensemble du cluster.