Atelier travaux pratiques : Grille de calcul et applications

Connecter via putty à l'interface utilisateur (UI) de magrid : **ui1.magrid.ma**

Créer un proxy de 12H : **voms-proxy-init --voms magridschool**

Interrogation du system d'information sur les ressources disponible dans un VO

Connaitre les ressources de calcul (CE) disponibles : **lcg-infosites --vo magridschool ce**

Connaitre les ressources de stockage (SE) disponibles : **lcg-infosites --vo magridschool se**

Connaitre les applications (Les tags) disponibles : **lcg-infosites --vo magridschool tag**

Exemple de soumission de jobs sur la grille

http://wiki.magrid.ma/index.php/Gestion_des_jobs

Commandes pour soumettre un job, suivre son statut, l'annuler ou bien récupérer les résultats :

glite-wms-job-submit –a –o <id qlconq à attribuer au job et à changer à chq soumission> <le fichier du job, i.e. le .jdl>

```
exemple : glite-wms-job-submit -a -o id001  myjob.jdl
```

glite-wms-job-status –i < id du job>

```
exemple : glite-wms-job-submit –i id001
```

glite-wms-job-cancel –i < id du job>

```
exemple : glite-wms-job-cancel –i id001
```

glite-wms-job-output –i < id du job> --dir <nom du dossier qui va contenir les outputs: mettre « . » simplement pour les mettre dans le dossier en cours>

```
exemple : glite-wms-job-submit –i id001 –-dir .
```

Remarque: Après la soumission d'un job, à chaque fois que quelque chose semble ne pas marcher, il faut attendre un peu, si ça dure bizarrement longtemps, annuler le job et le ressoumettre à nouveau avec un autre Id. Quand le statut est devenu Running c'est un bon signe (mais pas toujours si ça dure très longtemps).

Exercice1 : Job simple pour Compiler et exécuter un programme C++ sur la grille

Dans cet exemple, on suppose que vous avez un program.C à compiler et à exécuter sur la grille.
Alors ce qu'il faut faire:

1- Ecrire un fichier text de description de job, par exemple : myjob.jdl

2- Ecrire un script par exemple : myscript.sh qui va contenir les commandes pour compiler et exécuter myprogram.C sur le Worker Node.
3- Mettre les trois fichiers ensemble (myprogram.C, myscript.sh, myjob.jdl) dans un même répertoire
4- Soumettre le job : myjob.jdl
5- Suivre l'état d'exécution
6- Une fois le statut est Done : récupérer le résultat.

Exercice2 : Job pour utiliser Octave (comme Matlab)

http://wiki.magrid.ma/index.php/OCTAVE

Dans cet exemple, on suppose que vous avez un script Octave test.m à exécuter sur la grille. Alors ce qu'il faut faire:

1- Ecrire un fichier text de description de job, par exemple : octave.jdl
2- Ecrire un script par exemple : octave.sh qui va charger l'environnement d'Octave sur le Worker Node ensuite exécuter Octave avec l'argument test.m.
3- Mettre les trois fichiers ensemble (test.m, octave.sh, octave.jdl) dans un même répertoire
4- Soumettre le job : octave.jdl
5- Suivre l'état d'exécution
6- Une fois le statut est Done : récupérer le résultat.

Exercice3 : Job pour Utiliser ROOT

http://wiki.magrid.ma/index.php/ROOT

Même principe qu'Octave, les fichiers à préparer : exemple.C, root.jdl, root .sh

Exercice4 : Job pour utiliser Geant4

http://wiki.magrid.ma/index.php/GEANT4

nanobeam est un projet parmi les projets qui viennent avec geant4 (regarder dans le dossier exemples de g4). On se propose de le compiler sur la grille.

1- Ecrire un fichier text de description de job, par exemple : geant4.jdl
2- Ecrire un script par exemple : geant4.sh qui va contenir les commandes pour charger l'environnement de geant4 sur le Worker Node ensuite décompresser et compiler l'exemple nanobeam (l'exécutable sera produite dans le dossier /bin qu'on va compresser et récupérer avec /tmp dans le fichier : output_nanobeam.tar.gz).
3- Mettre les trois fichiers ensemble (geant4.jdl, geant4.sh, nanobeam.tar.gz) dans un même répertoire
4- Soumettre le job : geant4.jdl
5- Suivre l'état d'exécution
6- Une fois le statut est Done : récupérer le résultat.