Soumission de jobs

1. Présentation du tutoriel :

Ce tutoriel présente les différentes manières de soumettre des jobs sur la grille. Il est important de comprendre que la soumission de jobs sur la grille se fait grâce à un ordonnanceur (Ressource Broker : RB ou WMS) dont le rôle est de distribuer, selon certaines contraintes, le job sur un site.

Nous allons utiliser les deux types d'ordonnanceur qui existent, celui nommé Ressource broker, présentant moins de fonctionnalités et moins stable mais qui reste encore le plus répandu et le WMS présentant de nouvelles fonctionnalités et une plus grande stabilité

Tous les exemples qui suivent sont disponibles dans un fichier tar.
Copiez et décompressez le fichier /tmp/EXO-TUTORIAL.tar dans votre HOME.

```
$ cp /tmp/EXO-TUTORIAL.tar .
$ tar –cvf  EXO-TUTORIAL.tar
```

Créer un répertoire dans votre home sous le nom de JobOutput qui sera par défaut le répertoire de réception des sorties de job.

```
$ mkdir ~/JobOutput
```

Vous appartiendrez à la VO vo.lal.in2p3.fr pour ce tutoriel.

2. Initialisation du proxy :

Si cela n'est pas déjà fait, initialisez votre proxy en précisant le nom de la VO à laquelle vous appartenez (vo.lal.in2p3.fr). Utilisez la commande suivante :

```
$ voms-proxy-init –-voms vo.lal.in2p3.fr
```

3. Soumission d'un job :

Pour cette soumission, nous allons utiliser le fichier `HelloWorld.jdl` fourni. Les fichiers JDL sont des fichiers de description de job (voir présentation sur la soumission de job). Pour commencer, ouvrez le fichier HelloWorld.jdl pour voir ce qu'il contient.

a) Placez-vous dans le répertoire de ce fichier (cd EXO-TUTORIAL) et exécutez la commande suivante :

```
$ edg-job-submit -o jobid HelloWorld.jdl
```

L'option –o permet de préciser le nom du fichier dans lequel sera stocké l'identifiant du job. Cet identifiant se présente sous la forme http://.....

b) Vous pouvez vérifier le statut du job en utilisant la commande suivante :

```
$ edg-job-status -i jobid
```

Ce statut vous permet de voir si votre job est en cours d'exécution, en attente, annulé etc… Vérifier le statut de ce job jusqu'à ce que celui-ci soit égal à "Done (Success)".

Vous pouvez ne pas faire appel au fichier qui stocke les identifiants et directement passer l'identifiant de job à la commande.

```
$ edg-job-status http://.....
```

A noter que si pour plusieurs jobs vous utilisez toujours le même nom de fichier pour stocker vos identifiants de job, ceux-ci seront concaténés et dans ce cas les commandes utilisées avec l'option –i s'appliqueront à l'ensemble des jobs dont les identifiants sont dans le fichier.

Sur quel CE (Computing Element) votre job a-t-il été exécuté ? Comparez avec vos voisins. Quels sont les différents états que votre job a eu ?

c) Si l'état final de votre job est "Aborted", cela signifie qu'une erreur a eu lieu sur l'un des éléments de la chaine. Pour obtenir des détails, exécutez la commande suivante :

```
$ edg-job-get-logging-info –i jobid –v 1
```

Vous pouvez faire varier l'option –v de 0 à 2 pour avoir plus ou moins de détails.

d) En revanche, si votre job s'est terminé correctement ("Done (Success)"), vous pouvez récupérer les fichiers de sortie que vous avez définie dans le fichier jdl. Utilisez la commande suivante :

```
$ edg-job-get-output -i jobid --dir .
```

L'option --dir précise l'emplacement dans lequel les fichiers de sortie seront copiés (sous un répertoire identifiant le job). Deux fichiers devront être recopiés :

`stdout.log` et `stderr.log`. Vérifiez leur contenu. Vous devriez avoir un fichier `stderr.log` vide et un fichier `stdout.log` contenant la chaine "Hello World".

A ce niveau il est bon de rappeler qu'il existe également des APIs (java, c, c++) qui permettent la gestion des jobs depuis un programme compilé. Ce tutorial ne couvre pas ces APIs mais on retrouve exactement la même approche que celle que nous venons de voir.

Il est également possible de demander au Resource Broker de nous donner la liste des sites susceptibles de faire tourner notre job sans réellement l'envoyer pour exécution sur le site.

```
$ edg-job-list_match HelloWorld.jdl
```

Combien de sites sont susceptibles de faire tourner votre job ? Comparez avec le site où votre job a tourné.

4. Modification et édition des fichiers JDL

Nous allons dans ce chapitre aborder l'utilisation des sandbox, qui permettent notamment d'emporter et de récupérer des data avec le job.

a) Modifiez le fichier HelloWorld.jdl de manière à ce qu'il n'appelle plus /bin/echo mais le script HelloWorldScript.sh. pour cela :

La ligne Executable doit être HelloWorldScript.sh,

La ligne Argument peut rester avec Hello World

Vous devez de plus définir le paramètre InputSandbox afin de transférer le script avec le job.

La syntaxe de cet attribut JDL est :

```
$ InputSandbox = {"HelloWorldScript.sh"};
```

Le job HelloWorld-2.jdl vous donne la solution.

Comment comprenez-vous le job ainsi défini ? A quoi sert l'inputsandbox ?

Exécutez le job et vérifiez que tout fonctionne.

On peut utiliser n'importe quel script (perl, python, bash,…), cependant le shell utilisé par le script (indiqué dans la ligne ``\#!") doit

exister dans le Worker Node (ce qui est le cas quasi systématique pour le bash).

b) Nous allons à présent voir un exemple de récupération de fichier via l'output sandbox.

Pour cela nous utiliserons le job Output.jdl et le script Output.sh.
Ouvrez le job JDL et essayez de comprendre ce qui se passe, lancez le job.

5. Comprendre les Requirements et Rank

a) Nous allons basculer sur la VO dteam afin de disposer de plus de ressources (la voms vo.lal.in2p3.fr est supportée sur un nombre limité de sites).

Réinitialisez votre proxy en détruisant votre proxy puis en demandant un proxy « simple » avec la commande.

```
$ grid-proxy-init
```

Ce type de proxy ne contient pas d'information sur votre VO d'appartenance il faudra donc la préciser au moment des requêtes.

Regardons à présent la liste des sites susceptibles de faire tourner le job HelloWorld.jdl sous la VO dteam.

```
$ glite-job-list_match –vo dteam HelloWorld.jdl
```

Comparez avec le nombre de sites qui vous étaient proposés sous la VO vo.lal.in2p3.fr.

b) L'attribut *Requirements* permet de passer au Resource Broker un ensemble de conditions que rempliront les sites qui seront sélectionnés.

Par exemple, ajoutez l'expression suivante dans le fichier HelloWorld.jdl pour choisir tous les sites qui permettent à un job d'utiliser plus de une heure de temps CPU.

```
Requirements = (other.GlueCEPolicyMaxCPUTime > 60);
```

Combien de ressources autorisent les jobs qui utilisent plus d'une heure de temps CPU? Plus de deux heures? Plus de 10000 minutes?

L'attribut *Requierement*s peut être précisé pour l'ensemble des informations que publie un site (Version OS, librairie disponible,…). Par exemple pour choisir des

sites spécifiques on peut utiliser le nom de la ressource comme pré-requis. Pour choisir tous les sites en France, changez la valeur de *Requirements* comme suit :

```
Requirements = RegExp(".*\.fr:.*",other.GlueCEUniqueID);
```

Combien y a-t-il de sites en France?

Modifiez le requirement afin de compter combien de sites en France autorisent les jobs qui utilisent plus d'une heure de temps CPUs?

Modifiez la ligne Requirements pour choisir une ressource précise.

> *Une option ``-r'' existe pour la commande edg-job-submit qui permet de choisir une ressource spécifique. Cependant cette option évite tout le processus MatchMaking de Resource Broker et n'ajoute pas le fichier nécessaire (le fichier .BrokerInfo voir plus loin dans le tutorial) pour la gestion de données. La technique avec other.GlueCEUniqueID est plus flexible et plus sûre.*

c) L'attribut *Rank* permet de définir la règle qui sera appliquée pour classer les sites qui répondent aux conditions des pré-requis.

Ajoutez les lignes suivantes pour utiliser la ressource avec le plus grand nombre de CPUs libres.

```
Rank = other.GlueCEStateFreeCPUs
```

Utilisez la commande edg-job-list-match pour visualiser le résultat. (La ressource la plus intéressante est la première).

d) Question annexe :
Pourquoi dans cet exercice ne soumettons-nous pas les jobs ?

6. L'environnement d'exécution sur le Worker Node

Chaque utilisateur de la grille est mappé dans un compte local sur chaque site.

```
Réinitialisez votre proxy avec l'option --voms vo.lal.in2p3.fr
```

a) Générer un jdl permettant de voir sous quelle machine tourne le job (solution Hostname.jdl).

Lancez le job et récupérez l'output. Visualisez le fichier std.out. Sur quel compte êtes vous mappé? Comparer avec vos voisins ?

b) Visualisez le contenu du script Printenv.jdl.
Soumettez un job qui lance ce script dans la grille. Regardez la liste des variables. Des variables concernent-elles la grille ?

c) Lancez le job Brokerinfo.jdl. Regarder le résultat ? commentez ?

7. Soumission d'une job MPI

Beaucoup de disciplines utilisent des jobs parallèles. MPI (Message Passing Interface) est un protocole qui permet la communication entres les tâches parallèles.

Regardez dans les fichier MPI.jdl MPI.sh (les adeptes de MPI peuvent également regarder le hello.c)

Il y deux paramètres spéciaux pour les jobs MPI : JobType et NodeNumber. Le NodeNumber est le nombre de CPUs réellement utilisés par le job.

Regardez le script MPI.sh. Le script permet la compilation du code MPI, la vérification de l'existence de la liste des machines et le lancement du job en parallèle.

Vérifier combien de sites sont susceptibles de répondre.

8. Utilisation du nouveau ordonnanceur (RB) de tâches gLite WMS

Nous allons voir ici quelques exemples des nouvelles fonctionnalités qu'apporte la nouvelle version de l'ordonnanceur de tâches.

Pour utiliser cet ordonnanceur et non plus celui par défaut il est nécessaire d'utiliser des fichiers de configuration qui pointent vers cet autre service.
Ce sont les fichiers glite_wmsui_LAL.conf et glite_wms.conf..

Le mode de fonctionnement de cet ordonnanceur, bien que compatible avec la version précédente, utilise une autre approche sécuritaire à travers un service appelé : délégation de proxy.

```
$ glite-wms-job-delegate-proxy -–config glite_wms.conf  -a
```

La commande ci-dessus, vous permet d'acquérir une délégation de proxy générée de façon aléatoire (à conserver si on souhaite l'utiliser).

Notez l'argument --config qui permet d'utiliser le fichier de config pointant sur le nouveau service d'ordonnancement.

```
$ glite-wms-job-delegate-proxy –-config glite_wms.conf  –d delID
```

Vous permet d'acquérir une délégation de proxy identifiée par *delID* , ce qui est plus pratique car on définit delID et on ne récupère pas un nom aléatoire

a) Soumission d'un job

```
glite-wms-job-submit  --config  glite_wms.conf  -o  jobid  -d  delID
Helloworld.jdl
```

Notez la commande glite-wms au lieu de edg. Quelle différence voyez-vous avec la façon de soumettre précédente ?

Pour suivre le status et récupérer les outputs :

```
glite-wms-job-status  --config glite_wms.conf  -i jobid
```

```
$ glite-wms-job-output  --dir result --config glite_wms.conf  -i jobid -d
delID
```

b) Soumission de collection de jobs

Objectif ; soumettre plusieurs jobs en les gérant/contrôlant comme un seul
Regarder les jobs qui se trouvent dans le répertoire Jobcollection, on retrouve le job Helloworld, un job qui exécute 'hostname' et un dernier qui fait un simple 'pwd'.
Depuis le répertoire Jobcollection :

```
glite-wms-job-submit --config ../glite_wms.conf --collection . -o jobid -d
delID
```

```
glite-wms-job-status  --config ../glite_wms.conf  -i jobid
```

Notez les différences avec les exemples précédents de récupération de job status.

A présent si le job est fini on peut récupérer les sorties.

```
$ glite-wms-job-output  --dir result --config ../glite_wms.conf  -i jobid
-d delID
```

Notez la présence des 3 répertoires contenant les stdout/stderr des trois jobs.

Plutôt que de mettre les jobs dans un répertoire il est possible de construire un JDL qui les définit explicitement : voit le fichier JobCollection.jdl.

Sortir du répertoire JobCollection.

```
$ glite-wms-job-submit --config glite_wms.conf -o jobid -d delID
JobCollection.jdl
```

Noter l'absence de l'argument collection qui est à présent explicité dans le JDL.

Il est possible de conditionner/séquencer l'exécution des jobs d'une collection, c'est ce que l'on appelle des jobs de type DAG (Direct Acyclic Graph) .

c) Soumissions de jobs multi paramètres

Parfois seul le jeu des paramètres d'un job change, plutôt que d'envoyer x jobs pour couvrir x jeux de paramètres et donc gérer x jobs, on peut envoyer/gérer un seul job.

Ouvrez le job JobPara.jdl et essayez de comprendre ce qu'il doit faire.

Soumettez-le.

```
$ glite-wms-job-submit -config ../glite_wms.conf -o jobid -d delID
JobPara.jdl --dir result
```

NON ABORDE DANS CETTE PLANCHE EXERCICE

- Le renouvellement de proxy : existe uniquement avec l'ordonnanceur RB
- Le suivie des outputs d'un job qui s'exécute ; existe uniquement avec le WMS et reste un très gros consommateur de ressources