Exercices Grille Marseille 22/10/2010

- Les exercices se font sur la machine marui.in2p3.fr . Vous y avez chacun un compte à votre nom.
- **S'enregistrer dans une vo:**

  La vo utilisée est vo.formation.idgrilles.fr qui est dédiée aux formations.

  L'enregistrement se fait à partir de votre pc sur:

  https://cclcgvomsli01.in2p3.fr:8443/voms/vo.formation.idgrilles.fr/

  avec votre certificat présent dans votre browser, cliquez sur le lien “New user registration” dans la liste à gauche

  Remplissez le formulaire et cliquez sur le bouton "I have read and agree to the VO’s Usage Rules".

  Vous allez recevoir un message pour confirmer votre adresse électronique.Cliquez sur l' URL de confirmation.

  Le VO Manager doit valider votre demande. Vous recevrez un message.

Pré-requis (suite)

- Transférer votre certificat par ssh sur marui

  Pour l'exporter du navigateur:

- Firefox : Sélectionnez le menu FireFox/Preferences . Cliquez sur "Avancer" Ã droite et

l’onglet "Chiffrement". Cliquez sur le bouton "Afficher les resultats", sélectionnez votre certificat et

utilisez la fonction "importer".

Mozilla : Dans le menu Edit/Preferences allez dans l’onglet "Privacy \& Security" puis dans

"certificates". Cliquez sur "Manage Certificates", et utilisez la fonction "backup".

• Internet Explorer : Dans le menu Outils/Options Internet, allez dans l’onglet "contenu" et

cliquez sur "certificats". Cliquez ensuite sur "exporter". Sélectionner exportation de la clé

privée.

Il vous sera demandé un mot de passe que vous choisissez.

Pré-requis: certificat

Sur marui.in2p3.fr:

Créer un repertoire .globus : *mkdir .globus*

Y copier le certificat avec le nom *usercred.p12*

Définir les droits d'accés en lecture seule uniquement pour vous :

chmod 400 usercred.p12

Exercices: Repérez les différents champs du certificat (sujet, validité ...) en lisant son contenu avec la commande openssl:

openssl pkcs12 -in ~/.globus/usercred.p12 -nokeys -clcerts | *openssl x509 -text*

Quel est votre Subject ?

Quelle est la date de fin de validité de votre certificat?

S'identifiez sur la grille

- Création d'un Proxy:

Pour permettre à vos jobs d’accéder aux services de la grille, vous devez avoir un proxy. Un proxy est un un "clone" de votre certificat contenant des informations supplémentaires, telles que la VO à laquelle vous appartenez (ces extensions sont souvent appelées extensions VOMS). Les proxies sont valables pour une durée limitée, normalement 12 à 24 heures.

Les principales commandes permettant d'obtenir un proxy et de le gérer sont :

voms-proxy-init - -voms VO : permet de créer un proxy à partir du certificat. Cette commande demande le mot de passe protégeant la clé privée créée à l'étape précédente. Il est très important d'utiliser l'option --voms. VO est le nom de votre VO.

Note: le message Cannot find file or dir: ... /.glite/vomses est normal.

voms-proxy-info : affiche les informations sur le proxy existant. Avec l'option -all, permet d'afficher la liste des groupes et roles activés.

voms-proxy-destroy : supprime un proxy encore valide.

Il existe aussi des commandes myproxy-*. Elles permettent le renouvellement automatique des proxies pour des jobs de très longue durée. Elles seront abordées avec la soumission de jobs.

Exercices proxy

1) Demander un proxy à l'aide de la commande voms-proxy-init.

Par exemple:

voms-proxy-init --voms vo.formations.idgrilles.fr

2) Afficher les informations du proxy créé. Les informations intéressantes sont le "subject", l'équivalent de votre "username" dans la grille, le "timeleft", le temps durant lequel le certificat est encore valable, et le chemin du fichier pour le proxy.

3) Supprimer explicitement le proxy. Cette commande supprime le proxy dans l'ordinateur local. Les copies éventuellement envoyées avec vos jobs ne sont pas affectées. Vérifiez que le proxy a bien été supprimé avec la commande voms-proxy-info.

4) Regarder les autres options disponibles avec la commande voms-proxy-init. (Utilisez man ou l'option --help.) On peut changer, par exemple, la durée de vie et la taille (nombre de « bits ») du proxy. Normalement les valeurs par défauts sont correctes.

Système d'information

gLite fournit la commande lcg-infosites pour interroger le BDII. Elle cache la complexité de la syntaxe de la ldapsearch. Le but principal de la commande lcg-infosites est d'identifier les ressources disponibles pour une VO particulière. Pour cela on utilise l'option --vo voname. Le principal paramètre de cette commande est le type de ressource cherchée : ce, se, wms, lfc...

Utiliser l'option --help pour obtenir la liste de tous les types de service.

Note: l'utilisation de la commande lcg-infosites ne nécessite pas d'avoir un proxy.

Exercices :

1) Identifier la liste de tous les *computing elements* (CE) accessibles à la VO vo.formation.idgrilles.fr.

2) Identifier la liste de tous les *storage elements* (SE) accessibles à la VO vo.formation.idgrilles.fr

3) Identifier la liste de tous les workload management system (WMS) accessibles à la VO vo.formation.idgrilles.fr

4)Identifier la liste de tous les serveurs catalogue LFC accessibles à la VO vo.formation.idgrilles

Soumission de jobs

Les commandes permettant de soumettre et gérer un job commencent toutes par glite-wms-job- et utilise les services d'un gLite WMS.

Les principales commandes sont :

glite-wms-job-submit : soumission d'un job. Cette commande nécessite un *delegation proxy* qui peut être créé automatiquement si on utilise l'option -a ou être créé avec la commande glite-wms-job-delegation-proxy et être spécifié avec l'option -d identifieur. L'utilisation de -d doit être préférée si on soumet plusieurs jobs car elle est beaucoup plus efficace.

glite-wms-job-status : affichage de l'état d'un job (en attente, en cours d'exécution, terminé...). L'option --all permet d'afficher l'état de tous les jobs de l'utilisateur, si le LB (*Logging and Bookkeeping*, l'un des services du WMS) utilisé fait partie des LBs configurés par défaut sur l'UI.

glite-wms-job-output : récupération de l'output du job. Le répertoire utilisé pour stocker les résultats dépend de la configuration du site. L'option –dir permet de fixer le répertoire

glite-wms-job-logging-info : affichage d'informations détaillées sur l'exécution du job et les éventuelles erreurs rencontrées.

glite-wms-job-delegate-proxy -d identifieur : obtention d'un delegation proxy pour soumettre plusieurs jobs. identifieur est un nom arbitraire qui sera utilisé pour désigner le *delegation proxy* et l'utiliser dans avec les commandes comme glite-wms-job-submit.

Soumission de job

Toutes les commandes ont une aide en ligne accessible en utilisant l'option --help ou la commande man.

La commande glite-wms-job-submit retourne un *jobid* qui est un URL. Ce *jobid* devra être utilisé pour désigner le job dans les autres commandes. Il doit être conservé sans quoi il n'est plus possible d'interagir avec le job (connaitre son état ou récupérer son output). Quand on soumet plusieurs jobs, il peut être plus pratique d'utiliser l'option -o fichier lors du submit : dans ce cas, le *jobid* sera écrit dans le fichier. Le même fichier pourra être utilisé à la place du paramètre *jobid* dans les autres commandes en utilisant l'option -i.

Les commandes glite-wms-job-xxx nécessitant d'avoir un proxy valide, il n'y a pas besoin de spécifier la VO utilisée pour ces commandes. Elle est obtenue à partir du proxy.

Soumission de job:Exercices

Cette exercice consiste à soumettre un job très simple qui écrira "Hello World" dans le fichier d'output. Le job en lui-même n'a pas d'intérêt particulier mais permet d'illuster la soumission et le suivi de l'exécution d'un job. Un job grille est décrit dans un langage particulier appelé JDL décrit plus tard.

1) Si ce n'est pas déjà fait, créez un proxy à l'aide de la fonction voms-proxy-init.

2) Soumettez le job HelloWorld.jdl se trouvant dans le matériel du tutorial récupéré précédemment, en utilisant la commande glite-wms-job-submit. Sauver le *jobid* retourné ou utiliser l'option -o.

3) Vérifiez le statut du job en utilisant la commande glite-wms-job-status.

- Suivez les modifications de l'état du job jusqu'à l'état "Done(Success)"
- Combien d'états différents pouvez-vous distinguer ?
- Si le job se termine dans l'état "Aborted", c'est qu'il y a eu une erreur. On peut trouver plus d'informations avec la commande glite-wms-job-logging-info.

4) Lorsque le job est terminé (Status: Done (Success)), récupérez les données générées à l'aide de la commande glite-wms-job-output.

5) Vérifiez que tout s'est déroulé correctement en consultant les fichiers std.out et std.err. Le fichier std.err doit être vide et std.out doit contenir "Hello World". Cette procédure simple est utilisée pour le suivi de tous les jobs.

Gestion des données

Les services de grille impliquées dans la gestion des données sont le **LFC** (**file catalogue**) et les **SE storage element**.

LFC: Ce service gère les noms des fichiers donnés par les utilisateurs et leurs réplicas. Les noms de fichier sont comme des fichiers Unix. Un fichier peut se trouver à plusieurs endroits (réplica).

SE: C'est le service qui gère les données. Il y a différentes implémentation de ce service qui peuvent s'interfacer avec différents médias (disques, bandes etc..)

Le service « File catalog » (aussi appelé replica catalogue) est implémenté dans glite en utilisant LFC. Le service LFC a des clients implémentés via des commandes lfc- Ces commandes sont disponibles sur les ui et les worker nodes.

En général les commandes lcg-utils sont plutôt utilisées. Une exception est la commande lfc-mkdir qui doit être utilisées pour créer des répertoires avant d'y créer des LFN.

Gestion des données:LFC

Les commandes LFC principales sont:

lfc-mkdir [-]: crée un répertoire dans LFC

lfc-ls [-l]: similaire à la commande Unix ls. Donne le contenu du répertoire et peut donner d'autres informations.

lfc-getacl / **lfc-setac**l: l'espace de nom LFC supporte des ACL de type Posix. Voir le man pour la syntaxe. Un fichier a un ACL mais un répertoire a 2 ACL différents: celui controllant l'accès au répertoire et l'ACL par défaut pour les nouveaux fichiers. Les sous répertoires héritent des ACL de leur parent.

Exercices:

1) Trouvez le lfc de la vo vo.formation.idgrilles.fr avec la commande lcg-infosites

2) Pour l'utiliser export LFC_HOST=lfc-egee.in2p3.fr

3) Lister les fichiers pour cette vo. lfc-ls /grid/vo.formation.idgrilles.fr

4) Créer un répertoire à votre nom avec lfc-mkdir dans /grid/vo.formation.idgrilles.fr/cppm2010

Gestion des données lcg-utils

Un ensemble de commandes sous le nom de lcg_utils sont disponibles pour l'utilisateur, elles sont indépendantes de l'implémentation (donc valable pour tous les SE).

La plupart des commandes peuvent utiliser pour un fichier différents type d'identifiant.

- Logical File Name (LFN): nom dans le catalogue de fichier, syntaxe Unix like et commencant par /grid/*voname*. A LFN doit être préfixé lfn: Dans notre cas /grid/vo.formation.idgrilles.fr/
- GUID: identifiant unique associé à un fichier (ne change pas si LFN est changé ou renommé). Le GUID doit être préfixé guid:
- SURL: nom du replica sur un SE particulier Il a un format URL like srm://se.host.name/path/to/file.

Gestion des données:lcg-utils

Les commandes principales sont:

- lcg-cr (copy and register): copie un fichier local ou sur un SE et crée un nouveau fichier sur un SE en l'enregistreant dans le catalogue. La destination doit être un SE. Même si le fichier vient d'un autre SE cela créera un nouveau fichier et pas un replica. Par défaut le nom logique sur le LFC est généré mais il peut être fixé de manière explicite avec l'option -l.Les répertoires doivent être créés avec la commande lfc-mkdir. Le SE où le fichier va être copié peut être spécifié avec l'option -d , le nom du fichier est généré si seulement le hostname est présent. Si cette option n'est pas spécifiée le défaut sera pris dans VO_VONAME_DEFAULT_SE dans la VO utilisée. L'option -v (verbose) est recommandée pour suivi en cas d'erreur.
- lcg-cp (copy) similaire à la commande cp de UNIX. La source et la destination peuvent être un fichier local ou un fichier sur un SE. La destination n'est pas enregistrée dans le catalogue et ne peux être un LFN. La source peut être identifiée par un LFN ou un GUID si elle est sur un SE. Commande principalement utilisée pour avoir une copie locale sur un UI ou un WN d'un fichier résidant sur un SE.

Gestion des données lcg-utils

- lcg-rep: commande pour ajouter un nouveau replica à un fichier déjà enregistré dans le catalogue. Syntaxe similaire à lcg-cr mais la source doit être un LFN. Contrairement à lcg-cr cette commande mets à jour une entrée mais ne la crée pas.
- lcg-lr:(list replica) liste tous les replicas associés à un LFN ou un GUID
- lcg-ls: caractéristique et syntaxe similaire à ls de UNIX
- lcg-del: efface un fichier replica d'un SE et optionnellement efface l'entrée dans le catalogue LFC après avoir effacé tous les replicas d'un fichier
- lcg-lg: retourne le GUID associé à un LFN
- lcg-gt: retourne le TURL (hostname et filesystème) pour un LFN ou GUID donné

  Toutes ces commande reposent sur le BDII pour utiliser les défauts selon les VO. Si la BDII n'est pas disponible on peut utiliser l'option -nobdii mais il faut alors préciser tous les paramètres.

Gestion des données: exercices

1. Afficher les SE disponibles pour la VO vo.formation.idgrilles.fr en utilisant la commande lcg-infosites
    1. Combien de Ses possibles pour vo.formations.idgrilles.fr ?
    2. Trouver la même chose pour la vo dteam. Combien de SE?
2. Créer un répertoire à votre nom avec lfc-mkdir dans cppm2010
3. Créer un fichier texte et le copier sur un SE en utilisant la commande lcg-cr, utiliser l'option -l pour définir un nom logique. Essayer de définir le SURL (replica name on SE) aussi
4. Vérifier que le fichier est présent sur le catalogue et lister ses replicas avec la commande lcg-lr. Cela doit vous donner le SURL pour tous les replicas. Un SURL commence avec le préfixe srm:
5. Afficher toutes les informations de ce fichier avec la commande lcg-ls
6. Trouver le GUID associé aux fichiers précédents
7. Répliquer le fichier sur un autre SE
8. Vérifier que le nouveau replica a été ajouté au catalogue avec la commande lcg-lr. Comparer les résultats

Gestion des données exercices

- Recopier localement le fichier à partir du SE en utilisant la commande lcg-cp, le faire en utilisant le GUID, le LFN et un replica SURL.
- Supprimer un replica avec la commande lcg-del et vérifier les résultats avec lcg-lr
- Supprimer l'autre réplica avec la commande lcg-del et vérifier le résultat avec lcg-lr
- Recréer un fichier avec 2 réplicas et essayer de les supprimer en une fois. Vérifier avec lcg-ls et lcg-lr.

JOBS: JDL

Le fichier de description des jobs (JDL) utilise un langage particulier pour décrire les attributs et besoins du job. Ces informations sont utilisées par le wms pour sélectionner le CE approprié et par le CE pour lancer l'application de l'utilisateur.

Le format de ce fichier est un ensemble de définition attribut/valeur en utilisant le format suivant:

Attribut: Valeur;

Les principaux attributs sont

- Executable (obligatoire): définit la commande à exécuter. S'il s'agit d'un shell script, le shell utilisé par le script (indiqué dans la ligne #!) doit exister dans le worker node.
- Arguments (facultatif): une chaine de caractère passée comme argument de la commande, en utilisant la syntaxe attendue par la commande.
- InputSandbox (facultatif) : liste des fichiers locaux à transférer avec le job.
- OutputSandbox (obligatoire) : liste des fichiers produits par le job et devant être retournés par la commande glite-wms-job-output. Il doit y avoir au moins stdout et stderr.

JOB:Exercices

Utilisation des Sandbox:

1) Modifier le fichier HelloWorld.jdl pour qu'il n'utilise 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 définir le paramètre InputSandbox pour transférer le script du job.

   InputSandbox = {"HelloWorldScript.sh"};

   Le job HelloWorld-2.jdl vous donne la solution. A quoi sert l' input Sandbox ? Exécuter le job et vérifier que tout fonctionne.

2) Pour voir un exemple d'utilisation d'output sandbox, utiliser le job Output.jdl et le script Output.sh

   Remarque: Vous pouvez utiliser un script de votre choix dans le langage de votre choix (perl,python,bash ...). Mais shell utilisé doit être présent sur le worker.

3) Envoyer un job avec un script de votre choix.

Requirement et Rank

L'attribut Requirement permet de passer au WMS un ensemble de conditions que rempliront les sites sélectionnés.

Pour voir les sites correspondants sans lancer le job on utilise la commande:

```
glite-wms-job-list-match  helloWorld.jdl
```

Ajouter l'expression suivante dans le fichier HelloWorld.jdl pour sélectionner les sites qui permettent d'avoir plus de 1h de CPU.

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

L'attribut requirement 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 des sites in2p3, changez la valeur de Requirements comme suit:

Requirements = RegExp(“.*\.in2p3.fr:.*”,other.GlueCEUniqueID);

Combien de sites avez vous trouvé?

Modifier la ligne Requirements pour avoir un site in2p3 et plus de 1h de CPU.

Vous pouvez aussi modifier le Requirement pour tourner au CPPM.(ou sur le site de votre choix).

Jobs JDL

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

Par exemple pour utiliser la ressource avec le plus de CPU libre, on utilise

Rank = other.GlueCEStateFreeCPUs

Controller la resoumission par le WMS

Quand un job est soumis au WMS, celui ci passe par plusieurs phases jusqu'à la soumission finale à un CE et son exécution. Pour des raisons variées, des erreurs peuvent arriver à différents moment et l'utilisateur peut controller le nombre d'essais que le WMS doit faire avec 2 attributs:

* ShallowRetryCount: défini le nombre maximum de fois que le WMS resoumettra le job au CE sélectionné en cas d'une erreur pendant la soumission avant que le job ai démarré. Cela s'appelle “shallow resubmission”, à chaque essai un CE différent est sélectionné. Le défaut dépend de la configuration de l'UI.

* RetryCount: définit le maximum de fois que le WMS resoumet le job dans le cas ou une erreur arrive après le démarrage du job. Cela s'appelle “deep resubmission”, des actions spécifiques sont demandées pour nettoyer les fichiers laissés par les essais précédents.

Remarque: en mode debug il est préférable de mettre ces variables à 0

Environnement d'exécution

Chaque utilisateur de la grille est associé à un compte local, spécifique à chaque site. L'accès aux ressources locales est contrôlé par les droits de ce compte. Si on soumet plusieurs jobs en utilisant des proxies différents (VO et/ou groupe/rôle), on n'est pas associé au même compte sur le worker node.

Exercices :

1. Visualiser le contenu du fichier JDL whoami.jdl. Lancez le job et récupérez l'output. Visualisez le fichier std.out. Sur quel compte êtes-vous mappé? Essayer d'autres commandes comme hostname,pwd ...

2. Visualiser le contenu du script envvar.jdl. Soumettez un job qui lance ce script sur la grille. Regardez la liste des variables. Combien de variables concernent la grille?

3. Ecriver un job qui liste les versions des logiciels disponibles dans le Worker Node. On peut utiliser la commande rpm pour le faire.

Renouveler un proxy en cours de job

Par défaut, la validité d'un proxy est relativement courte, généralement entre 12 et 96h. La durée maximale est fixée par le serveur VOMS de la VO. Si le proxy expire avant la fin du job, il ne sera pas possible de récupérer les résultats. S'il expire avant le début du job, le job échouera.

Pour permettre à un job de s'exécuter sans problème quelque soit son temps d'attente et sa durée, il faut utiliser un service de renouvellement de proxy, appelé MyProxy. Son utilisation est très simple. Il faut d'abord ajouter la ligne suivante dans le JDL du job (la valeur du paramètre doit être un serveur MyProxy acceptant la VO utilisée pour le voms-proxy-init et le resource broker utilisé pour soumettre le job, myproxy.grif.fr est le serveur MyProxy de GRIF) :

MyProxyServer = "myproxy.grif.fr";

Après avoir fait le voms-proxy-init et avant de soumettre le job, il faut exécuter la commande myproxy-init, comme suit :

myproxy-init -d -n -s myproxy.grif.fr

La configuration d'un serveur MyProxy détermine les WMS autorisés à utiliser le service pour renouveler des proxies. Le serveur MyProxy de GRIF, myproxy.grif.fr, accepte les demandes de renouvellement en provenance du WMS du cppm, marwms.in2p3.fr.

myproxy

On peut voir la liste des proxies valides avec la commande myproxy-info et on peut mettre fin au renouvellement du proxy (avant ou pendant l'exécution du job) avec la commande myproxy-destroy. L'option -d doit être toujours être utilisée avec l'ensemble des commandes myproxy-*.

Note : les commandes myproxy-xxx utilisent un nom de fichier différent des autres commandes gLite si on utilise un fichier unique pour le certificat et la clé privée (extension .p12). Pour résoudre le problème, il faut créer le lien symbolique suivant :

cd ~/.globus

ln -s usercert.p12 usercred.p12

Exercices :

1. Utiliser la commande myproxy-init pour vérifier que vous pouvez créer un proxy dans le serveur.

2. Utiliser aussi les commandes myproxy-info et myproxy-destroy

Job avec données sur SE

Les Sandbox sont limitées en taille, il est souvent utile de mettre des données ou un soft sur un SE et de les utiliser dans le job. Cette exercice montre un exemple en entrée et en sortie.

1) Copier vos scripts sur le se avec la commande lcg-cr
2) Faire un script pour faire le copy sur le worker node
3) Ecrire un jdl qui utilise le script 2)
4) Faire la même chose pour l'output.

Les fichiers copytest2se.sh, run_test.jdl, run_test.sh donnent une solution possible

Job Compilation

- Modifier de nouveau HelloWorld.jdl de manière à ce qu'il appelle cette fois l'exécutable myhostname. Vous pouvez visualiser la source de cet exécutable, qui est un programme C : myhostname.c. Vous n'avez cette fois pas besoin de définir d'argument. Il faut modifier la ligne InputSandbox. Exécutez le job et vérifiez que tout fonctionne. Sur quel ordinateur a tourné votre job ?
- L'exécution d'un programme en C compilé n'est pas forcément pratique : l'exécutable peut être d'une grande taille, dépendre de plusieurs fichiers, ou dépendre d'un environnement d'exécution particulier. Une solution consiste à compiler le programme directement sur le CE. Modifier une nouvelle fois HelloWorld.jdl de manière à ce qu'il appelle le script buildandrun.sh, avec pour argument myhostname.

  * Testez ce script seul pour comprendre l'argument nécessaire.

  * Exécutez le job et vérifiez qu'il fonctionne toujours.

Job MPI

A faire uniquement pour ceux qui utilisent MPI

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

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

Il y a 2 paramètres spéciaux pour les jobs MPI: JobType et Node Number.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.

Vérifiez combien de sites sont susceptibles de répondre

Collection de jobs

Ne faire que si intérêt particulier et/ou il reste du temps

Objectif: soumettre plusieurs jobs en les gérant comme un seul. Regardez les jobs qui se trouvent dans le répertoire Jobcollection, on retrouve le job HelloWorld, un job qui excecute hostname et un dernier qui fait pwd.

Depuis le répertoire Jobcollection;

glite-wms-job-submit --collection . -o jobid -d delID

avec delID obtenu avec glite-wms-job-delegate-proxy -d delID

glite-wms-job-status -i jobid

Quand le job est fini on récupère les sorties.

glite-wms-job-output –dir result -i jobid

Notez la présence des 3 répertoires correspondant au 3 jobs.

On peut aussi construire un jdl plutot que de mettre les jobs dans un répertoire. Voir JobCollection.jdl

Chainage de jobs (DAG jobs)

Parfois seul le jeu de 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 --o jobid -d delID JobPara.jdl –dir result