TP4 Grille de Calcul $3^{ème}$ année F2

Gestion de fichiers et Distribution de traitements sur grille de calcul

Éléments à rendre avant le prochain TP :

À envoyer dans un tarball en pièce-jointe à `passerat@isima.fr` *:*

- le projet Maven répondant à l'exercice 3 (nettoyé = **sans les binaires** → .class, .jar)

1 Le LFC (Logical File Catalog)

Le LFC est un élément atomique dans une VO. Il implémente la fonction de catalogue de fichiers, offrant un espace d'alias intelligibles pour désigner de manière unique un fichier et ses répliques sur les SE d'une VO. Nous allons découvrir les commandes de base pour le manipuler dans cette partie.

1. La commande `lcg-infosites -vo biomed lfc` vous donnera l'adresse du LFC utilisé sur la VO biomed. Vous pouvez à présent remplir correctement la variable d'environnement LFC_HOST.
2. Créez un répertoire propre à votre binôme dans le dossier `/grid/biomed/mip/jopasserat`.
3. Vérifiez la présence de votre répertoire et les droits qui lui ont été attribués.
4. Testez la suppression de ce répertoire, puis créez le de nouveau pour la suite du TP.
5. Ce répertoire sera la racine de votre répertoire personnel. Fixez la variable d'environnement LFC_HOME en conséquence.

2 Commandes de bases

Les commandes précédents faisant directement intervenir le LFC ne sont que rarement utilisées, si ce n'est pour créer des répertoires. On leur préfère les outils connus sous le nom de *lcg_utils*, capables de créer des répliques sur les SE mais également de les inscrire dans le LFC. Ces commandes manipulent donc différentes formes de dénominations de fichiers, comme les identifiants uniques, les noms logiques du LFN ou les surl.

1. Créez un fichier texte et copiez le sur le SE de votre choix. Notez le retour de la commande, de quoi s'agit-il ?
2. Affichez des informations à propos de ce fichier.
3. Répliquez le fichier sur un autre SE associé à la VO *biomed*.
4. Retrouvez les adresses `surl` des répliques à partir de l'identifiant unique de votre fichier.
5. Réalisez l'opération inverse à partir d'une des *surl* obtenues.

6. Supprimer l'une des répliques et vérifiez le résultat.
7. Même chose pour la réplique restante.
8. Créez un nouveau fichier sur la grille en essayant de lui associer le nom logique précédent dans le LFN. Cela fonctionne-t-il ? Qu'en concluez-vous ?
9. Après avoir créé plusieurs répliques de ce fichiers, détruisez l'entrée le désignant dans le LFN ainsi que toutes les répliques à l'aide d'une seule et unique commande.

3 Application concrète - PovRay

Dans le but d'étudier la distribution de calculs sur grille avec une application concrète, nous allons utiliser l'application de ray-tracing *PovRay*[1]. Cette application génère une suite d'images à partir des paramètres qu'elles reçoit en entrée. Pour le TP, les paramètres vous sont fournis et n'ont pas besoin d'être retouchés, l'idée nétant pas de découvrir les dessous des algorithmes de ray-tracing, mais plutôt d'exploiter leur parallélisme intrinsèque. En effet, chaque image peut être générée indépendamment des autres.

Indications :
- Le logicel PovRay est disponible sous forme d'un tarball, de même que les fichiers de configurations le concernant, à l'emplacement `http://fc.isima.fr/~passerat/enseignement/grille/tp4/povray.tgz`
- Les paramètres utilisés dans notre cas conduisent à la génération de 60 images.
- Les données en entrée/sortie étant d'une part d'une taille conséquente, et d'autre part utilisées par plusieurs jobs, il semble judicieux de les stocker en amont sur des SE.
- PovRay génère un fichier au format PNG par image de la séquence dans le repertoire courant. Pensez à les stocker pour les récupérer ultérieurement.
- Pour générer les 20 premières images de la séquence, on utilise la syntaxe suivante :

```
./povray +A +W640 +H480 +Lshare/povray-3.6/include/ +SF1 \
+EF20 glsbng.ini
```

Post-traitement :
Les images de la séquence peuvent être rassemblées pour former une animation. La suite ImageMagick, installée sur l'UI du CRRI propose la commande `convert`, capable de réaliser ce montage :

1. `http://www.povray.org/`

1. Une fois l'ensemble des PNG récupérés, lancez l'une des commandes suivantes pour obtenir respectivement une séquence aux formats privateurs GIF ou MPEG-2 :

```
convert glsbng*.png -delay 6 -quality 100 glsbng.gif
convert glsbng*.png -delay 6 -quality 100 glsbng.m2v
```

2. Enjoy :)

Consignes : En vous inspirant du code de la classe d'exemple `SagaJob` et de la classe `NamespaceCopy` (`http://fc.isima.fr/~passerat/enseignement/grille/tp4/NamespaceCopy.java`), proposez un code JSAGA permettant de distribuer l'application *PovRay*.

Le code étant plus agréable à développer avec un IDE digne de ce nom, vous utiliserez l'éditeur de votre choix sur le serveur *etud* de l'ISIMA, avant de **construire** et **déployer** votre solution sur l'UI du CRII à l'aide de Maven. La partie suivante vous aiguillera quant à l'utilisation de cet outil (même les utilisateurs chevronnés devraient y jeter un œil, ne serait-ce que pour récupérer quelques fichiers...

Note : *La classe `NamespaceCopy` est utilisée par l'API en ligne de commande de JSAGA. En effet, les appels à jsaga-cp.sh se contentent de lancer le `main` contenu dans cette classe...*

Introduction à un outil de construction Java : Maven

Pour compenser les faiblesses de la machine servant d'UI, nous allons déporter l'édition du code et la compilation sur votre serveur de travail habituel : *etud*. Pour ce faire nous allons utiliser Netbeans et un outil Apache : Maven.

Maven [2] est l'outil de construction de projets Java le plus prometteur des dernières années. Il dispose en effet de nombreuses caractéristiques telles que la gestion automatique des dépendances, la compilation de plusieurs langages via l'utilisation de plugins, ou encore le déploiement de paquets JAR sur des serveurs de gestions d'artefacts. La philosophie de cet outil consiste à plébisciter les conventions plutôt que configuration pour gérer un projet (*Convention Over Configuration*). Ainsi, un projet respectant les directives d'organisation de Maven ne demandera que peu de configuration pour être construit.

2. `http://maven.apache.org`

Vous pouvez trouver une archive de Maven à l'adresse suivante : `http://fc.isima.fr/~passerat/enseignement/grille/tp4/apache-maven-3.0.tar.bz2`

1. Décompressez l'archive de Maven et suivez la procédure d'installation (*la variable d'environnement JAVA_HOME doit être fixée à :* `/usr/local/java/jdk1.6.0_18`).
2. Récupérez l'archive du projet à l'adresse suivante : `http://fc.isima.fr/~passerat/enseignement/grille/tp4/tp4.tgz` et décompressez-la.
3. Les IDE Java reconnaissent pour la plupart les projets Maven. Lancez Netbeans sur *etud* et ouvrez le projet que vous venez de télécharger.
4. Prenez connaissance de l'arborescence mise en œuvre pour manipuler une classe que vous devez reconnaître.
5. Construisez le projet à l'aide de la commande `mvn package` depuis le dossier contenant le fichier `pom.xml`. Qu'a fait Maven de manière automatique ?
6. L'exécutable créé nécessite une connexion Internet accédant librement aux divers services de grille (VOMS, WMS, ...), ce qui n'est pas le cas sur *etud*. Nous allons donc demander à Maven de **déployer** ce fichier JAR sur l'UI :
   (a) Modifiez le dernier bloc du fichier `pom.xml` présent dans le projet pour modifier la balise URL comme suit :
   `<url>scp://193.55.252.156/home/votre_compte_sur_UI/deployed</url>`
   (b) Copiez le fichier `http://fc.isima.fr/~passerat/enseignement/grille/tp4/settings.xml` dans votre répertoire `~/.m2/` sur *etud* et modifiez la balise `<username>passerat</username>` qu'il contient pour qu'elle indique votre login sur l'UI.
   (c) De retour dans le dossier du projet, tapez la commande `mvn deploy`
7. Vous pouvez finalement vous connecter sur l'UI et lancer le JAR exécutable situé dans le dossier deployed de votre compte avec la commande :
   `java -jar pov-ray-grid-0.1.jar`