DIRAC Transformation System

un coups d'oeil

S. Poss

CERN, LAPP

11 avril 2013

Transformation System

Transformation: Opération sur un lot de fichiers : création et/ou modification

Tâche: Element d'une transformation: associe un/plusieurs fichier(s) à une opération

Objectif

1. Obtenir un système permettant l'exécution de tâches similaires sur un lot de données
2. Automatisation de la procédure
3. Enchaînement simple de tâches
4. Limiter les actions manuelles
5. Ne pas re-traiter les fichiers déjà utilisés

Comment ?

Le Transformation System permet ceci:

- Avec un “template” de job (workflow)
- avec un lot de fichier (input des jobs, peut être un chemin ou autre)

Il crée les tâches qui conviennent:

TransformationAgent créé les tâches: associe un fichier à une tâche (plusieurs règles possible, voir plus loin)

Workflow/RequestTaskAgent envoie les tâches au système correspondant: WMS pour les jobs, Request Manager pour les requêtes (réplication par exemple)

DataRecoveryAgent recupère les jobs failed et change le status des fichiers en input pour que l’agent 1 recréé une/plusieurs tâche(s) (n’existe pas par defaut)

InputDataAgent met à jour la liste d’input par transformation suivant des requêtes de meta data dans le catalogue (data-driven procedure).

TransformationCleaningAgent nettoie les transformations lorsque necessaire: nettoie les tables, retire les fichiers si necessaire, nettoie les catalogues

Comment ?

Utilise le status des fichiers:

Unused: Le fichier n'est pas encore utilisé

Assigned: le fichier est associé à une tâche: il n'est pas considéré lors de la création d'une nouvelle tâche

Processed: Le fichier a été traité: généralement, ce satus est mis à jour à la fin d'un job (en appelant le *FileReport*). Si le job est 'failed' et la raison comprise alors le fichier peut être marqué comme 'Unused'.

MaxReset: Lorsqu'un fichier échoue trop souvent, il est marqué de manière à ne pas recréer de nouvelle tâche. Peut être ignoré par le DataRecoveryAgent s'il est utilisé.

Définition du template

- Techniquement, un template peut être n'importe quoi: c'est un BLOB dans la TransformationDB.
- Ajustement des agents nécessaire si ce n'est pas un DIRAC Workflow: reimplémentation du WorkflowTaskAgent

(*t.setBody()*)

Règles de création des tâches (TransformationPlugin)

Tous les fichiers disponibles et non associés à une tâche sont utilisés pour créer de nouvelles tâches en utilisant les règles suivantes:

Standard	Groupe les fichiers par site, avec la taille du groupe choisie lors de la création de la transformation (*t.setGroupSize()*).
Broadcast	Utilisé pour les replications: assigne les destinations des fichiers (*t.setTargetSE()*)
ByShare	S'assure que tout les sites ont une utilisation équivalente
BySize	Groupe les fichiers d'input de manière à avoir une utilisation du disque équivalente (pas souvent utilisé)

Possibilite d'étendre ces règles simplement (subclass).

(*t.setPlugin()*)

Exemple

```
from DIRAC.TransformationSystem.Client.Transformation import Transformation
from DIRAC.TransformationSystem.Client.TransformationClient import Transformat
from DIRAC.Interfaces.API.Job import Job

j = Job()
tc = TransformationClient()

t = Transformation()
t.setTransformationName("Un exemple") #<- unique
t.setTransformationGroup("Un groupe") #<- for monitoring
t.setType("MCSimulation")#type must be among known types
t.setDescription("Ceci est un exemple")
t.setLongDescription("C'est un bel exemple")
t.setBody(j.workflow._toXML())
t.setGroupSize(1)
t.setPlugin("Standard")
t.addTransformation() #<-- transformation is created here
t.setStatus("Active") #<-- make it start
t.setAgentType("Automatic") #<-- should be by default
transfid = t.getTransformationID()['Value'] #<- unique
tc.createTransformationInputDataQuery(transfid,
                                       {'meta1':val1,"meta2":{">":34}})
```

Monitoring

Depuis le portail DIRAC (Matvey):

- Suivi de l'évolution de chaque transformation
- Opérations: start, stop, clean, complete (, flush)
- Obtension des propriétés de chaque transformation
- Possibilité d'inspecter chaque job
- Extension de certaines transformations

Conclusion

Si un ensemble de tâches identiques doivent être appliquées sur un lot de données, alors le TS est le bon choix

L'enchainement des tâches est possible en utilisant l'aspect data-driven

L'extension des fonctionnalités est possible en créant de nouvelles classes/services/agents

Backup Slides