Transformation System report

Luisa Arrabito[1], Federico Stagni[2]

1) LUPM CNRS/IN2P3, France

2) CERN

7th DIRAC User Workshop 29th – 31st May 2017, Warsaw

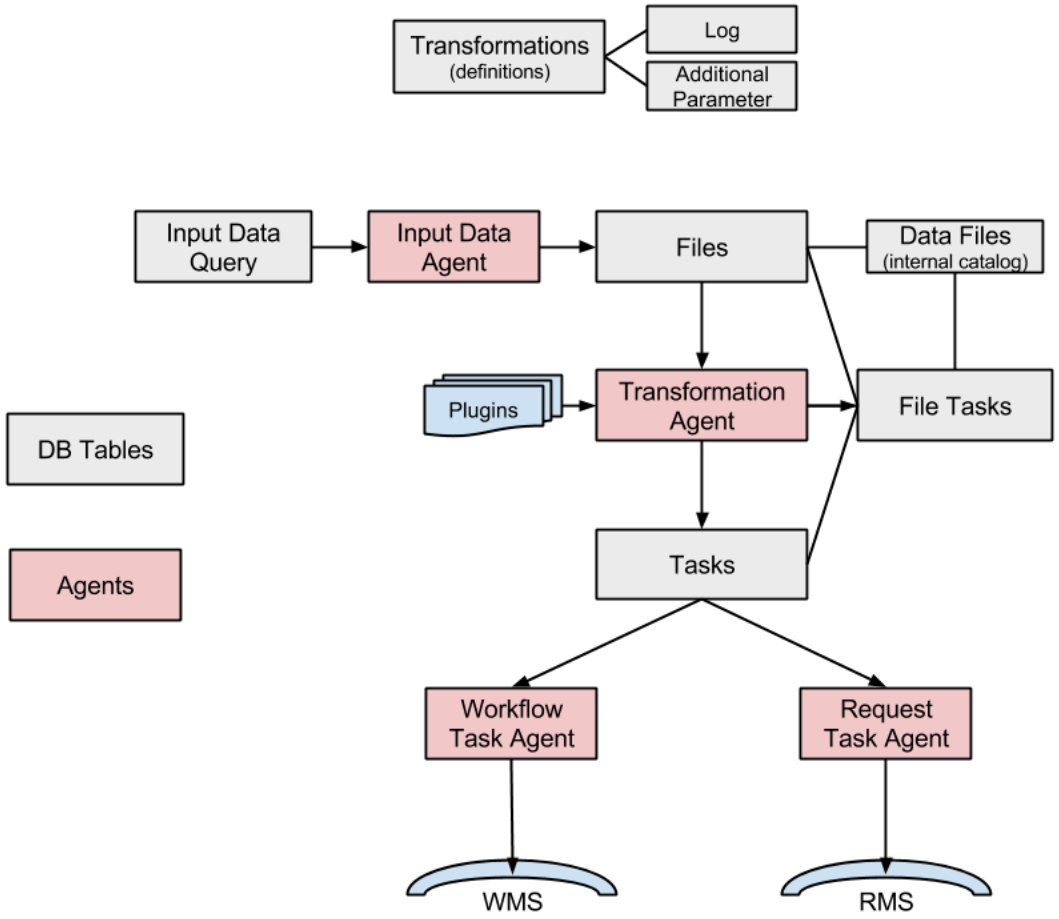
Plan

- What's the Transformation System?
- Evolutions since last year
- Future plans

What's the Transformation System?

- A DIRAC System as usually comprising:
  - MySQL DB, Services, Agents, Clients, Scripts and *Plugins*
- A system for handling "repetitive work", i.e. many identical tasks with a varying parameter
- 2 main usages:
  - Productions: the "same" job – i.e. the same workflow - is executed
    - Client for the Workload Management System
  - Data handling: replications, removal
    - Client for the Request Management System
- It handles input datasets (if present)
  - It interacts with Replica and Metadata catalogs (*e.g.* DFC or external catalogs)
  - Use of 'Plugins' to group tasks input files and set tasks destinations
- It does not support multi-VO installations
- LHCb 'Production System' is built on top of it. Also CTA, ILC and Belle II use it for their productions

Transformation System architecture



- Production Manager defines the transformations
- TransformationAgent processes the transformations and creates tasks given a Transformation Plugin
- InputDataAgent queries the Catalog to obtain files to be 'transformed'
- WorkflowTaskAgent transforms tasks into job workflows, given a TaskManager Plugin
- RequestTaskAgent transforms tasks into requests

Plugins

- Transformation Plugins
  - Group input files of the tasks according to different criteria
    - Standard
      - Group files according to replica location
    - BySize
      - Group files until they reach a certain size (input size in Gb)
    - ByShare
      - Groups files given the share (specified in the CS) and location

    For replication

    - Broadcast
      - Take files at the source SE and broadcast to a given number of locations

Plugins

- TaskManager Plugins
  - Used to specify tasks destination
    - BySE
      - Default plugin
      - Set jobs destination depending on the input data location
    - ByJobType
      - It allows to implement any distributed computing model by simple configuration in the CS
        - By default, all sites are allowed to run every job
        - Different rules for site destination can be specified in the CS for each JobType

ByJobType Plugin: how it works?

- Configuration
  - Set Operations/Transformations/DestinationPlugin = ByJobType
  - Define the rules for each JobType in Operation/JobTypeMapping, *e.g.*:

```
JobTypeMapping
{
   AutoAddedSites = LCG.CERN.ch
   AutoAddedSites += LCG.IN2P3.fr
   AutoAddedSites += LCG.CNAF.it
   AutoAddedSites += LCG.PIC.es
   AutoAddedSites += LCG.GRIDKA.de
   AutoAddedSites += LCG.RAL.uk
   AutoAddedSites += LCG.SARA.nl
   AutoAddedSites += LCG.RRCKI.ru
   DataReconstruction
   {
      Exclude = ALL
      Allow
      {
         LCG.NIKHEF.nl = LCG.SARA.nl
         LCG.UKI-LT2-QMUL.uk = LCG.RAL.uk
         LCG.CPPM.fr = LCG.SARA.nl
         LCG.USC.es = LCG.PIC.es
         LCG.LAL.fr = LCG.CERN.ch
         LCG.LAL.fr += LCG.IN2P3.fr
         LCG.BariRECAS.it = LCG.CNAF.it
         LCG.CBPF.br = LCG.CERN.ch
         VAC.Manchester.uk = LCG.RAL.uk
      }
   }
   Merge
   {
      Exclude = ALL
      Allow
      {
         LCG.NIKHEF.nl = LCG.SARA.nl
      }
```

JobType

AutoAddedSites: sites allowed to run jobs with files in their local SEs

Exclude: sites that will be removed as destination sites

Allow: sites allowed to run jobs with input data at another site

  - Here ‘Merge’ jobs having input data at LCG.SARA.nl can run both at LCG.SARA.nl and LCG.NIKHEF.nl

7 Set ‘Merge’ JobType in the job workflow

Support for parametric jobs

- Support for parametric jobs
  - Improvement of job submission
  - TaskManager prepares and submits a bunch of jobs in one go
  - It’s activated by Transformations/BulkSubmission flag in CS

How it works in practice (I)?

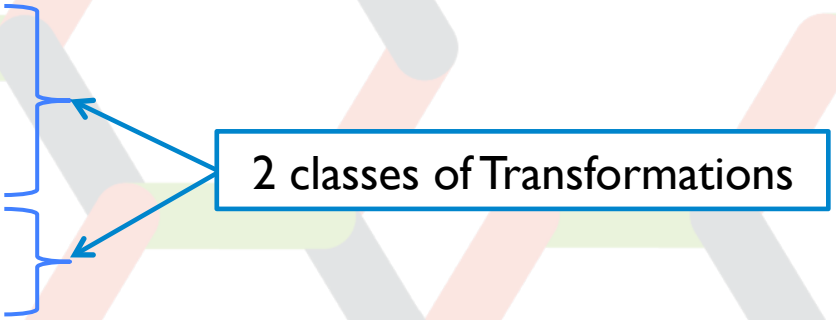
- See documentation at:
  - http://dirac.readthedocs.io/en/stable/AdministratorGuide/Systems/Transformation/index.html

- Installation
  - Need to have the Transformation System installed and running. The minimum is:
    - **Service:** TransformationManagerHandler
    - **Database:** TransformationDB
    - **Agents:**
      - TransformationAgent
      - WorkflowTaskAgent
      - RequestTaskAgent
      - InputDataAgent
      - TransformationCleaningAgent

How it works in practice (II)?

- Configuration
  - Add the transformation types in the Operations/[VO]/Transformations section, *e.g.*:

```
Transformations
{
  DataProcessing = MCSimulation
  DataProcessing += Merge
  DataProcessing += Analysis
  DataProcessing += DataReprocessing
  DataManipulation = Removal
  DataManipulation += Replication
}
```



  - Eventually configure the WorkflowTaskAgent and the RequestTaskAgent to treat a particular transformation type

Use cases examples (I)

- MC Simulation
  - You want to generate many identical jobs with a varying parameter (and no input files)
  - The varying parameter should be built from *@{JOB_ID}*, which corresponds to the *TaskID*, and it's used in the job workflow, *e.g.*:

```
job.setExecutable( './dirac_prod3_corsika', arguments = '@{JOB_ID}' )
```

  - Create a MC transformation

```
from DIRAC.TransformationSystem.Client.Transformation import Transformation
from DIRAC.Interfaces.API.Job import Job
j = myJob()
...
t = Transformation( )
t.setTransformationName("MCProd") # This must
t.setTransformationGroup("Group1")
t.setType("MCSimulation")
t.setDescription("MC prod example")
t.setLongDescription( "This is the long description of my production" ) #mandatory
t.setBody ( j.workflow.toXML() )
t.addTransformation() #transformation is created here
t.setStatus("Active")
t.setAgentType("Automatic")
```

set Type

Use cases examples (II)

- Data analysis, i.e. process a large number of files with the same program
  - You want to create many identical jobs with varying input files
  - Create a transformation with a valid type (see slide on TS configuration), *e.g.*:
    - setType(”Analysis")
  - Add files to the transformation using the TransformationClient
    - Add a list of existing files
      - addFilesToTransformation(transID,infileList)
    - Add files which are the result of a DFC query
      - createTransformationInputDataQuery(transID, {’site': ’Paranal’,'particle': 'proton','analysis_prog=evndisp’})
      - In this way files are added as soon as they are registered in the Catalog (InputDataAgent)
      - They are most likely the result of another on-going transformation
    - Set the number of input files per job, *e.g.*:
      - setGroupSize(10)
    - Define how files should be grouped, *e.g.*:
      - setPlugin("Standard”)

Use cases examples (III)

- Data handling
  - Bulk data replication, i.e. replicate many files to a list of Target SEs
    - You want to create many identical replication requests with varying input files
    - Create a Replication transformation
      - Define the type of requests to be executed
        - setBody('ReplicateAndRegister')
      - Set a valid type (see slide on TS configuration)
        - setType("Replication")
      - Set the source and the target SEs for replication
        - setSourceSE(['CYF-STORM-Disk','DESY-ZN-Disk'])
        - setTargetSE(['CEA-Disk'])
        - setPlugin("Broadcast")
  - Bulk data removal (see details in documentation)

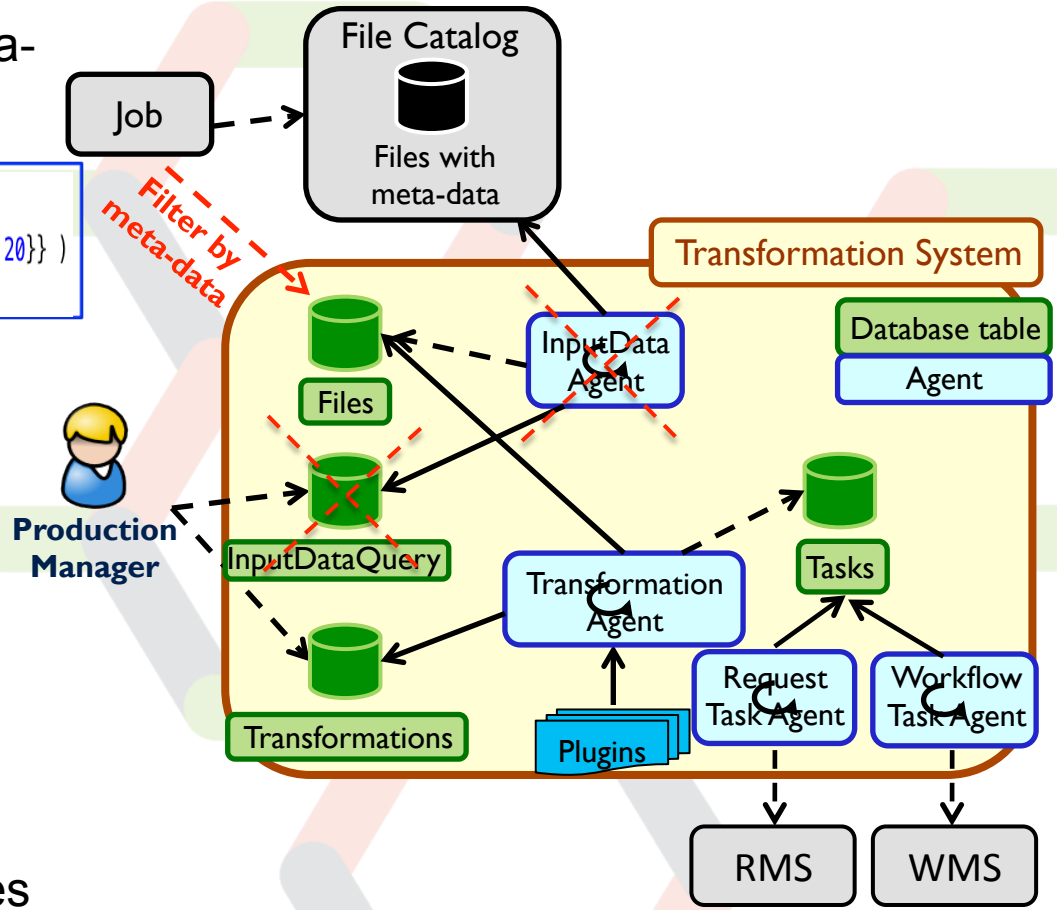
Evolutions since last year

- Meta-filters introduced as Technology Preview in v6r17 (see RFC #21)

- Define transformations with 'meta-data filters', *e.g.*:

```
t = Transformation()
filter = json.dumps( {'particle':'gamma_diffuse', 'zenith':{"<=": 20}} )
t.setFileMask( filter )
```

- When new files are registered in the File Catalog, they are evaluated against these filters
  - If they pass a filter, they are attached to the corresponding transformation
  - Need to activate the TS Catalog together with the standard File Catalog (DFC or external)
- Avoids 'large' File Catalog queries by the InputData Agent



Evolutions since last year

- Meta-filters introduced as Technology Preview in v6r17
  - See doc: http://dirac.readthedocs.io/en/stable/AdministratorGuide/Systems/Transformation/index.html
  - Already used in CTA production
  - Some comments gathered during BilD meeting for improvement
    - When creating a new transformation having files matching the filter, files are attached on the fly to the transformation
      - Suggestion to handle this asynchronously
    - Improve the logic of the MetaQuery utility which evaluates the files against the filters
    - Other?

Future plans

- Already discussed last year, see RFC #21:
  1. Implement meta-filters
     - done in v6r17 -> need to be ‘certified’
  2. Support for chained transformations
     - Example: Re-processing -> Merging -> Removal, in LHCb these chained transformations are handled by a dedicated Production System
     - Proposal to extend the TS to support chained transformations as basis for each community to build its own 'Production System’
  3. Use MQ complementary to polling
     - Agents in the TS work in ‘polling’ mode
     - Proposal to use a Message Queuing System complementary to polling