

Transformation System report

Luisa Arrabito¹, Federico Stagni²

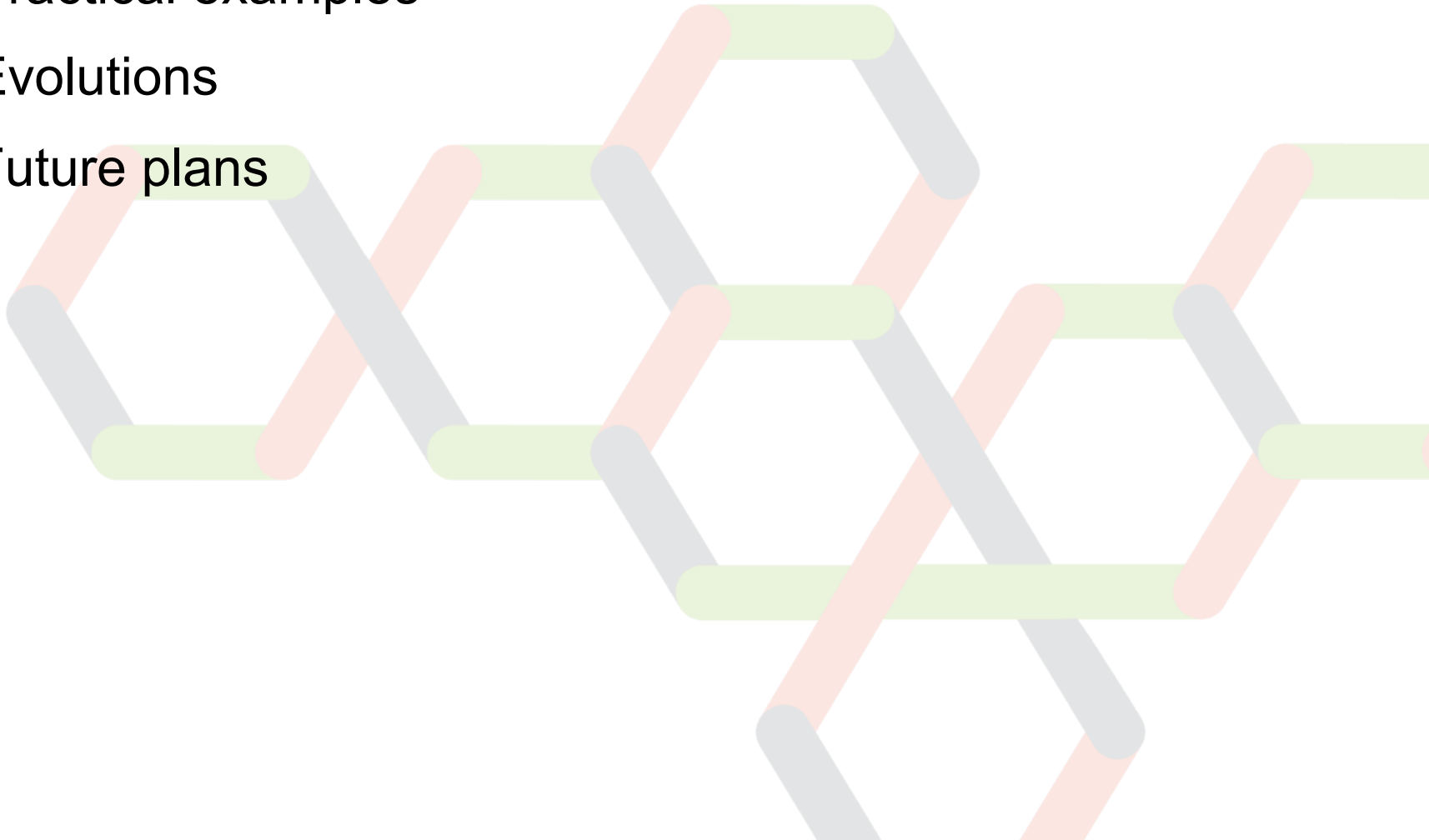
1) *LUPM CNRS/IN2P3, France*

2) *CERN*

8th DIRAC User Workshop 22nd –
25th May 2018, Lyon

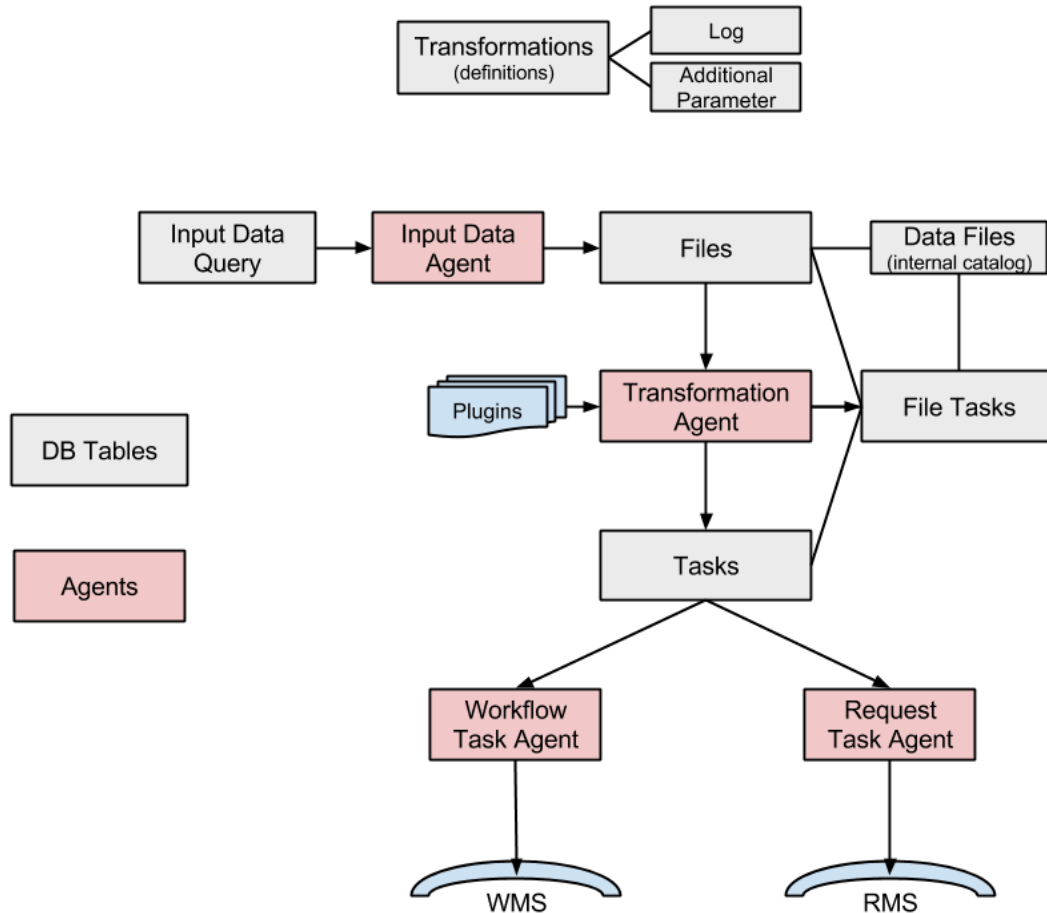



-
- ▶ What's the Transformation System?
 - ▶ Practical examples
 - ▶ Evolutions
 - ▶ Future plans



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’ as well ‘Distributed Data Management System’ is built on top of it. Also CTA, ILC and Belle II use it for their productions



- **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

▶ 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

▶ TaskManager Plugins

▶ Used to specify tasks destination

▶ BySE

- ❑ Default plugin
- ❑ Set jobs destination depending on the input data location

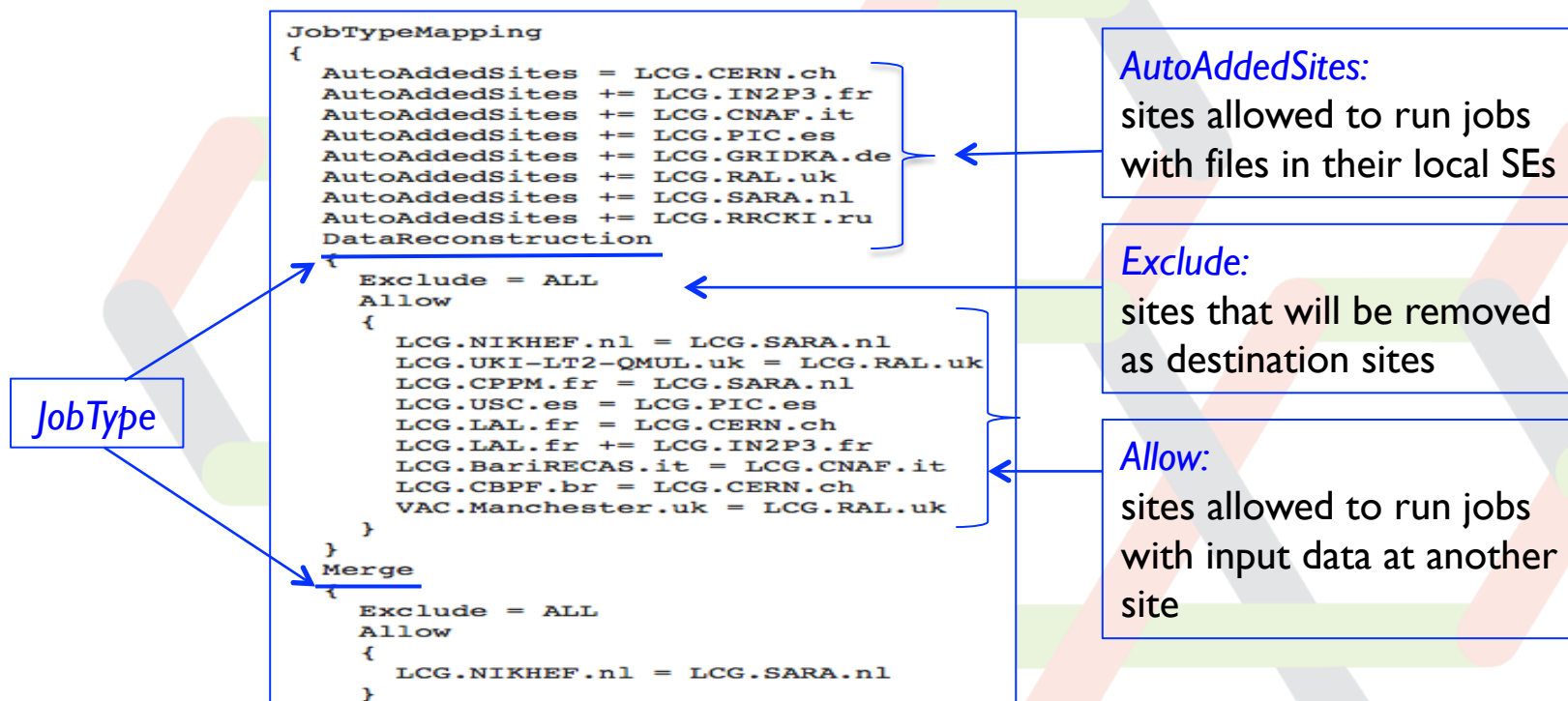
▶ ByJobType

- ❑ It implements the *mesh processing*, i.e. the possibility to run jobs at 'distant' sites with respect to data location
- ❑ 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.:



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

- ▶ 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

▶ 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

► 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
}
```



2 classes of Transformations

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

▶ 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

- ▶ **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**
 - Using the InputDataQuery Agent
 - Using the TSCatalog interface (since v6r17)
 - Files are added as soon as they are registered in the Catalog
 - They are most likely the result of another on-going transformation
 - Query example: {'site': 'Paranal','particle': 'proton','analysis_prog=evndisp'}
 - **Set the number of input files per job, e.g.:**
 - setGroupSize(10)
 - **Define how files should be grouped, e.g.:**
 - setPlugin("Standard")

▶ 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)

- ▶ No significant evolutions since last year
 - ▶ Meta-filters certified (see RFC #21)
- ▶ Documentation
 - ▶ <http://dirac.readthedocs.io/en/stable/AdministratorGuide/Systems/Transformation/index.html>
- ▶ Tutorial session on thursday

▶ See RFC #21:

1. Improve the logic of the MetaQuery utility which evaluates the files against the meta-filters
2. Use MQ complementary to polling
 - Agents in the TS work in 'polling' mode
 - Proposal to use a Message Queuing System complementary to polling
3. Support for chained transformations
 - Proposal to extend the TS to support chained transformations as basis for each community to build its own 'Production System'
 - Topic of the 'Production Management' session



Backup

- 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

