Metadata and Distributed Analysis

Johannes Elmsheuser

Ludwig-Maximilians-Universität München, Germany

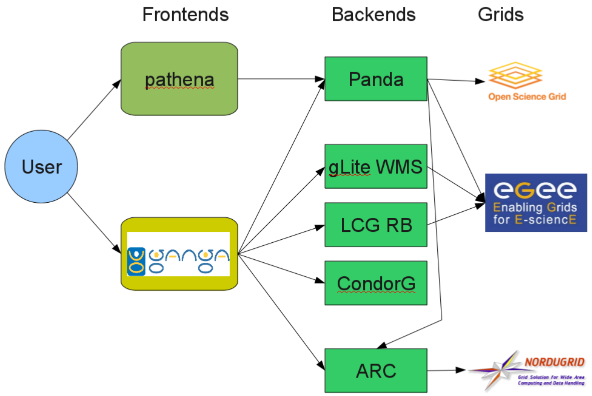
30 August 2010
ATLAS Metadata workshop

LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

Introduction

- The following is a loose collection of different meta-data aspects that matters in distributed analysis

ATLAS Distributed Analysis



Data is centrally being distributed by DQ2 - Jobs go to data

Different kind of inputs and outputs to e.g. monitoring and job repositories

What is or could be Meta-data in DA ?

- Meta-data as input to athena job:
  - GRL, generator settings
- Meta-data as input data-set:
  - run period
- Meta-data for output data-set:
  - data-set name
- Meta-data for jobs or job-set:
  - job configuration (athena version, source code tarball), data-set, num files or events to process, job splitting, site
- Job brokering:
  - many small different information sources, BDII, scheconfig, DQ2 catalog
- Meta-data for job statistics:
  - call backs to DQ2 popularity service, job performance, error rate, event rate, CPU/Walltime

Where do we store Meta-data in DA ?

- Input:
  - xml files, input parameters to TRFs, flat files transferred with input-sandbox, AMI query
- Input data-sets:
  - DQ2
- Output data-sets:
  - DQ2, some users try to code meta-data in output datasetname
- Job:
  - Ganga job repository or PandaMon
- Job Statistics:
  - extracted by log parsing and stored in job repository

Example 1: Specifying Input Data from AMI Searches

- Ganga can query AMI with logical data-set pattern and GoodRunList

```
...
j.inputdata = AMIDataset()
j.inputdata.logicalDatasetName= "%physics_MinBias.merge.AOD.r%"
j.inputdata.goodRunListXML = File('/path/to/MyLBCollection.xml')
...
```

- Query during submission time via pyAMI

Example 2: Event picking

- pAthena queries ELSSI for run & event number to resolve DQ2 dataset and file

```
pathena AnalysisSkeleton_topOptions.py
   --eventPickEvtList rrr.txt
   --eventPickDataType AOD
   --eventPickStreamName physics_CosmicCaloEM
   --outDS user...
```

Example 3: Event based job splitting with AMI

- Usually job splitting based on file level (info from DQ2)
- Ganga queries AMI for event based job splitting

```
...
j.inputdata = AMIDataset()
j.inputdata.logicalDatasetName= "%physics_MinBias.merge.AOD.r%"
j.splitter=DQ2JobSplitter()
j.splitter.numevtsperjob=100
...
```

- Opportunities we currently see:
  - inconsistency in the naming (tid, containers)
  - user data-sets not in AMI – so no event based splitting is possible (without manual user intervention).
  - Should (some) user output added to AMI ?

Future ideas for Meta-data in DA ?

- Job brokering:
  - combine all infos into AGIS ? DB with API access, but also flat file needed (similar to ToA)
- Bookkeeping for inputdataset:
  - cf. SAM at DØ /CDF, DB stores what file have been already processed by a job, if data-set grows only added files are processed in 2nd turn.
- Job verification:
  - Did the job-set process all available files ?
- Job error classification
  - improved error reporting for error classification
  - Very tedious to implement...
- Job repository
  - Store all infos locally or globally in a monitoring instance ?