ATLAS Data Management - Concepts and Implications for Federations

Graeme Stewart
for the ATLAS DDM team

Creating Federated Data Stores For The LHC, CC-IN2P3, November 2011

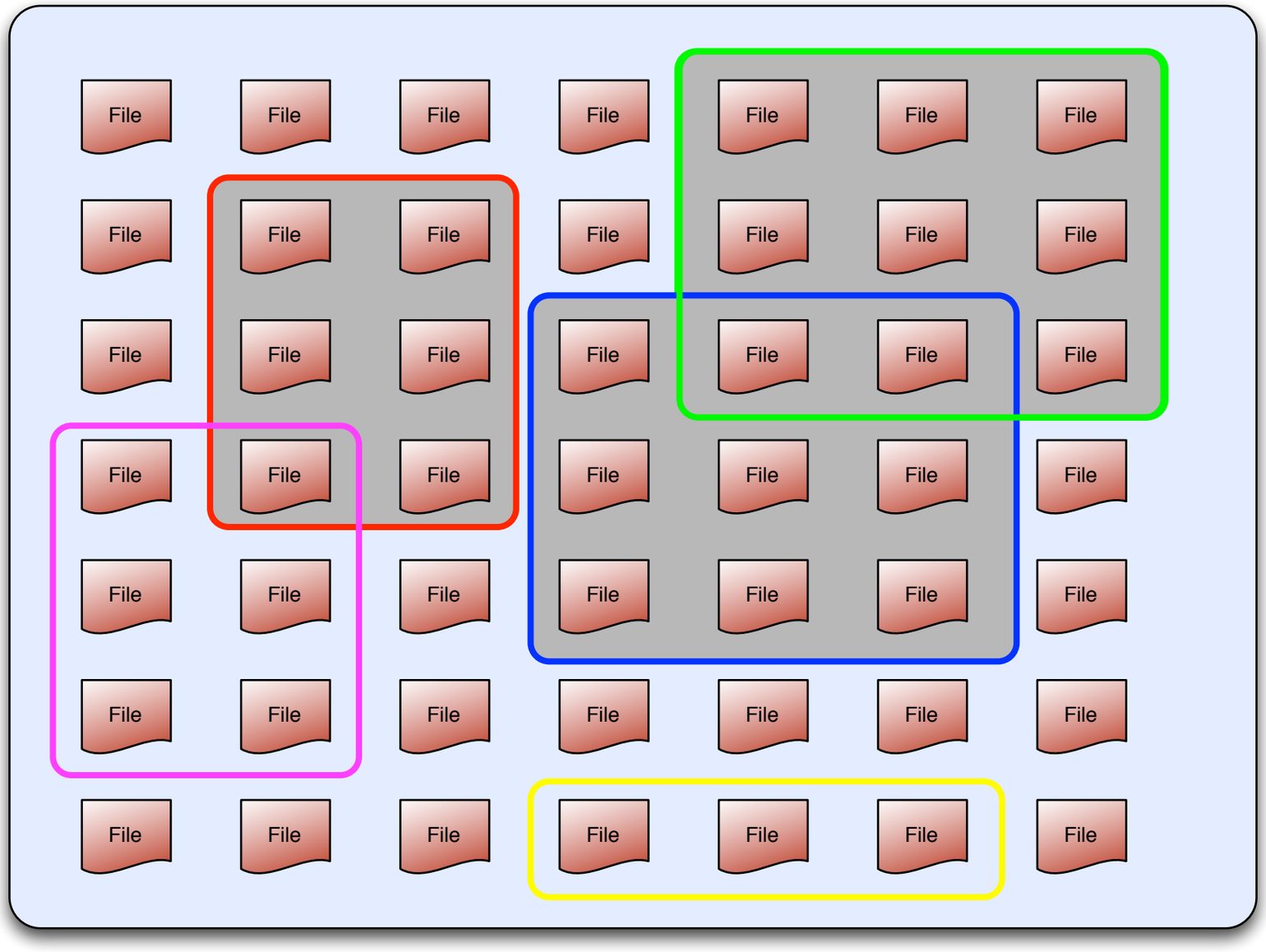
Data Management Project

- The ATLAS Distributed Data Management project is charged with managing ATLAS data on the grid
- All for the purpose of helping the collaboration store, manage and process LHC data in a heterogeneous distributed environment
- Requirements:
  - Register and catalog data
  - Transfer data to/from sites
  - Delete data from sites
  - Ensure data consistency at sites
  - Enforce ATLAS computing model requirements

ATLAS Data Concepts

- At the heart of everything is a file, of course
- Files in ATLAS are collected into datasets
  - Datasets live in a flat namespace
    - Naming convention: e.g., data11_7TeV.00184130.physics_Muons.recon.ESD.r2603_tid491184_00
    - All files must be in at least one dataset
    - But overlapping datasets are supported, i.e., files may be in multiple datasets
- Datasets are the units of replication
- Note in particular that ATLAS central catalogs do not organise data in a filesystem-like way
- Datasets can be aggregated into container objects

Data Model



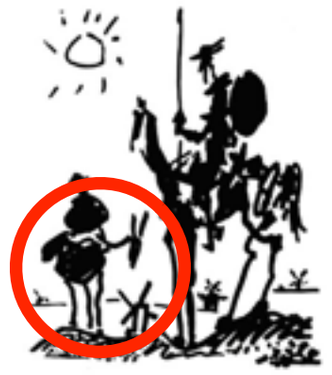
Dataset U Dataset U Dataset = Container/Dataset

DQ2 Rucio

DQ2

- Current implementation of this data model
- Some specific aspects are not very helpful for federations:
  - No concept of uncataloged endpoint
    - If we put it there we expect it to be there (~forever)
  - Files are written into storage in a non-unique way
    - Dataset name is in the path - but it's not unique
    - Transfer failures complicate things even more - no SRM overwrite means an 'attempt number' gets written
    - All of which means an *external* service (LFC) needs to be called to find out the *local* name of any file
      - Slow and painful

Rucio



- DQ2 working, but non-negligible problems with the design and scaling
- New system proposed: Rucio
  - Improve functionality where needed and simplify the system where possible
  - Take advantage of advances in other fields (databases, storage interfaces, etc.)

Rucio Concepts

- Separate logical view of data from physical one
  - Datasets become purely logical, not used as the unit of replication
- Deal with sites in a more flexible way
  - Data sinks are possible
  - Other storage protocols can be used
  - Do not build the system around any one technology

In A Nutshell...

- Transfer data but do not catalog it ✔
  - Feed Tier-3s and federated sites with data as needed
- Deterministic local file names ✔
  - Unique path and name (not finalised!):
    - prod:data11_7TeV.00187543.physics_MinBias.merge.AOD.f396_m945._lb0007-lb0026
      - N.B. This file name **is** the unique identifier for the file
    - Hash name: 2c0a794c61624781 92dcc40ac70e823c
    - Generate path stub: 2c/0a/79
    - File path is: 2c/0a/79/prod:data11_7TeV.00187543.physics_MinBias.merge.AOD.f396_m945._lb0007-lb0026
    - Prefix determined by site + access protocol, e.g,
    - https://atlas.t2.ac.uk/atlas/2c/0a/79/prod:data11_7TeV.00187543.physics_MinBias.merge.AOD.f396_m945._lb0007-lb0026
- Can easily generate manifest files for any set of data
  - You **do** have to call Rucio to resolve dataset ⇒ files

More on Self-Managed Sites

- Rucio will only catalog data it controls
  - Important to optimise system globally through a very fast internal catalog system
- Any autonomous data system can only be a sink
  - In a federation should we know?
    1. The top level redirector
    2. The individual sites
    3. All of the above
  - Probably 3 to allow most flexible feeding of data into the federation
- Should we source data from the federation?
  - Seems entirely feasible to do so - should be as flexible as we can to use all possible data sources