DQ2 - Data distribution with DQ2 in Atlas

DQ2 - A data handling tool

Kai Leffhalm

DESY

March 19, 2008
Technisches Seminar Zeuthen

Outline

1. Introduction
   - Motivation
2. Introduction to DDM
   - What is DDM
   - Atlas Data Model
   - Dataset Subscriptions and Versions
3. DQ2
   - DQ2 - overview
   - DQ2 core and design
   - DQ2 - Agents
   - DQ2 Site services
4. DQ2 commands - Overview
   - DQ2 Enduser commands, CLI, Python API and examples
5. When not to use DQ2
6. Future Development
7. Problems

Motivation

Atlas detector will produce lots of data

RAW 1.6 MB/event, $2 * 10^9$ ev/year, 3.2 PB/year $\Rightarrow$ one copy

ESD $\sim$ 1 MB/event, 2 PB/year $\Rightarrow$ two copies

AOD 0.1 MB/event, 180 TB/year $\Rightarrow$ more than 20 copies

TAG 1 kB/event, 2 TB/year $\Rightarrow$ more than 50 copies

DPD 0.1kB/event, 2TB/year $\Rightarrow$ more than 20 copies

- $\Rightarrow$ round about 11PB of data to be distributed from CERN

MC Production

- . . .will produce 30% of the real data again
- Production is distributed all over the world

Additional data transfers for calibration

- . . .will be done between CERN and several sites (T1 or T2)

What is Distributed Data Management

Duties of DDM

- Accounting of all ATLAS file-based experiment and user data.
- Managing data distribution across sites.
- Ensuring the distribution according to the computing model.

Provided by DDM

- Automatic system for distributing data from MC simulation and detector.
- Data monitoring for replicas and transfers.
- Tools for users to find data.
- Tools for users to request data distribution.
- Command line tools to automatize user procedures.

DDM services

Services provided by DDM:

- Shifters for monitoring data transfers and job execution.
- Savannah-Bug-List for resolving software or data problems.
- Hypernews for user problems.
- Mailing list for questions and announcements.
- Weekly meetings for discussion with T1s.
- Wiki pages with manuals and tutorials.
- Tutorials and talks during software weeks.
- Scripts made available in CVS

Atlas Data model

Data organization

- Data is organized in files, that belong to a dataset.
- Files that don't belong to a dataset can't be accessed through the system
- Metadata is stored with the dataset.
- Different versions from a dataset can exist.
- Datasets can be closed, open or frozen.

Data transfers

- Atlas moves data in organized bulk transfers (centrally organized according to Atlas Computing Model)
- Data can also be transferred according to user requests (within a cloud).
- All data transfers are (better should be) organized in subscriptions.
- The transfers are done in a pull mode.
- Sites are subscribed to a dataset(version).

Atlas Computing Model

Computing model concerning the data transfers

- Data is transferred strictly hierarchically from T0 to T1 to T2/T3
- Transfers between T1 and T2 are just done within a cloud
- Transfers between T2s are just done within a cloud
- Each cloud should have at least 2 copies of all AODs (DPDs)
- Jobs should go where the data is.
- If data is requested at a T2 and only available from a T2 in another cloud, both T1s should get the data

A word on Subscriptions and Versions

Dataset Subscriptions

- Subscriptions have two main components:
  - Fetcher: Search for new and deleted subscriptions in the databases.
  - Transfer agents: Find the files to be transferred and transfer them.
- Subscriptions belong to the user requesting it.
- Subscriptions to dataset versions will stop, when the version is fully replicated.
- Subscriptions to datasets will stop when the dataset is frozen (no new versions possible) and fully replicated.

Dataset versions

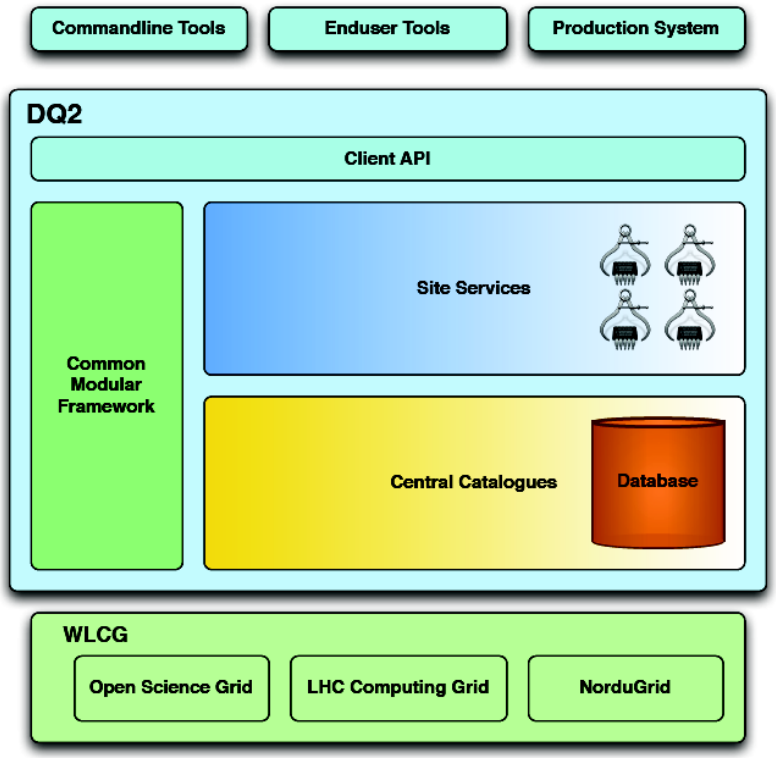
- Datasets are open or frozen.
- Dataset version are open or closed.
- Files are not added to datasets but to dataset versions (mostly to version 1 as most datasets now have just one version and are not closed).

DQ2 - Overview

What is DQ2

- DQ2 is an ATLAS tool for defining and handling datasets
- The name was inspired by Don Quixote (which now seems to be a good analogue)
- The core consists of several databases storing information about datasets
- The CLI provides functions to:
  - Handle datasets: create, delete and add files to datasets
  - Find and monitor replicas at sites
  - Create, delete and manage subscriptions for transfers to sites
- The Python API provides:
  - The same features as the CLI
  - Many functions to write useful scripts to do specific processes required by DDM and users

DQ2 - system



Schematic structure of the DQ2 data system. The framework is written in Python (2.3 compatible).

See talk from M.Lasnig on Chep 07

The DQ2 core

DQ2 holds all information in several databases:

Content Catalog which files belong to the dataset

Subscription Catalog Subscriptions and their status for all datasets

Location Catalog Information about the master-site and all replicas

Repository Catalog Meta-data of each dataset (version, state, timing,...)

Further databases will be added in the next version

Container Catalog Change in naming convention of physical datasets

Extended Location Catalog Information for consistency check

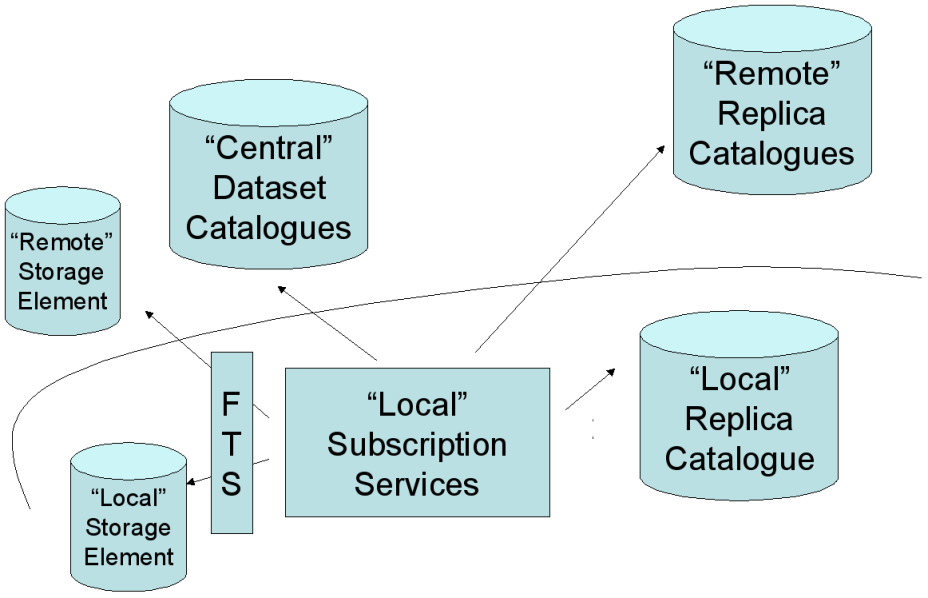
Deletion Catalog Information about the status of deletion of (files in) datasets

Database Backend

Oracle Database

- The databases are partitioned by date (is in plan at least)
- Multiple databases are used.
- DNS load-balancing is used.
- Changes in the schema happens with new versions
- ...due to new requirements of DDM and production

Design



Site services Tutorial

DQ2 Agents

- It exists a set of agents to process the subscriptions.
- Agents "belong" to the database tables
- Agents are organized in an agent table
  - Each time an agent starts, a row in the table is created
- Central agent handler takes care of
  - starting agents
  - controlling number of running agents
  - updating the database
  - stops the agents
  - checking sanity

DQ2 - list of the agents

Different agents handle the tasks of the system

Fetcher
- Does not rely on agent framework
- Find subscriptions to a site from catalog
- Compare to running subscriptions and fill/update agents database

Subscription Resolver Agent
- looks for a subscription

Splitter
- looks up files on the destination replica catalog.

Replica Resolver
- Finds and chooses a source from possible source replica catalogs
- Choose from the list of possible replicas with smallest replica attempt number, the computing model “replica”
- Does not check the state of that source replica, or rank replicas whether they are on disk/tape
- It does not look everywhere to find the best source, just for replicas on sites matching the subscription policy.

DQ2 - list of the agents cont.

Different agents handle the tasks of the system

Partitioner
- Dimensions requests (now also checks if channel is full/not full and dimensions request appropriately)
- It checks the state of the FTS-channel and dimensions a single submitter request to fill the the channel.

Submitter
- Submits request without further analysis.

Pending Handler
- checks state of files within request

DQ2 - list of the agents cont.

Different agents handle the tasks of the system

Verifier

- Verifies the result of transfers and decides when to retry - sends back to Replica Resolver
- Each file has a “pickup_date”: the earliest date it will be picked up for processing within a request.
- Verifier sets this value before sending the file back to Replica Resolver.
- Replica Resolver will actually ignore the pickup_date: it will immediately restart with a source.
- If it knows all sources it just chooses from the replicas that have been tried less.
- Then the file, after moving through the Replica Resolver (and given an actual source to use), will take into account the pickup_date and will only consider submitting the request after that date.

Replica Register

- registers the replica in the catalog.

DQ2 site services

- Site services are installed on VO-Boxes at Cern
- One VO-box per cloud
- Can be installed locally:
  - Makes fine tuning of data transfer possible
  - You can define dataset-patterns that will make transfers of matching datasets go to special paths and pools
  - Logging information is available, when transfers does not work good enough

DQ2 Enduser Commands

Getting information

`dq2_ls -ltfgp [-s Site] <dataset-name with wildcards>`

Lists all datasets which matches the given dataset-name, "$*$" can be used as a wildcard.

Getting datasets

`dq2_get -r -d <SE-directory> -s <source-site> <dataset-name> <file-names>`

Copies the given dataset or the given files from the dataset to the local directory, if no destination is given.

DQ2 Enduser Commands cont.

Creating datasets

`dq2_put -d <directory on SE> <dataset-name>`

Registers a dataset with the files in the directory in DQ2 catalog and in LFC catalog. The files must be visible to DQ2, this means they should be on a SE. Names should follow the naming convention:
`user.KaiLeffhalm.test.t01.`
Pool files have already guids associated, these should be given to the `dq2_put`-command with a `PoolFile.xml` file.

DQ2 Command Line Interface

Commands for more advanced users

- Some fundamental functions of dq2 are not covered with the enduser-tools, like subscriptions.
- Most of these functions are not needed in daily use of the system.
- These commands are used for handling:
  - Subscriptions
  - Replicas
  - Dataset-versions

CLI cont.

Overview of currently usable command line tools

```
dq2-close-dataset
dq2-delete-files
dq2-delete-replicas
dq2-delete-subscription
dq2-destinations dq2-erase
dq2-freeze-dataset
dq2-get-metadata
dq2-get-number-files
dq2-list-dataset
dq2-list-dataset-by-creationdate
dq2-list-dataset-replicas
dq2-list-dataset-site
dq2-list-erased-datasets
dq2-list-files
dq2-list-subscription
dq2-list-subscription-info
dq2-list-subscription-site
dq2-metadata dq2-ping
dq2-register-dataset
dq2-register-files
dq2-register-location
dq2-register-subscription
dq2-register-version
dq2-reset-subscription
dq2-reset-subscription-site
dq2-sample dq2-set-metadata
dq2-sources
```

Python API

- DQ2 is written in python.
- All functions and classes can be used via the python api.
- They are (of course) not documented anywhere
- . . .except publishing the code via CVS is documentation enough.
- Best way to start writing tools is looking in existing tools (not DQ2 tools)
- Many scripts written by DDM people are in the CVS too
  - Framework to check consistency and to clean up (Cedric Serfon)
  - Scripts to delete dataset replicas (Stephane Jezequel, Kai Leffhalm)
- these scripts are good to find out which functions can be used
  - . . .**but** they shouldn't be just taken and run
  - Deletion for example can't be done by anyone

Examples for the CLI

dq2-list-dataset-replicas

```
dq2-list-dataset-replicas
trig1_misal1_mc12.006204.ttbar190_mcatnlo.recon.AOD.v12000
```

dq2-list-subscription

```
dq2-list-subscription
trig1_misal1_mc12.006204.ttbar190_mcatnlo.recon.AOD.v12000
```

dq2-list-subscription-site

```
dq2-list-subscription-site DESY-ZN
```

DQ2 should not be used for everything

- Subscriptions should be made via the web interface
  - This ensures there is enough place on the sites.
  - The procedure is quite new and needs some more testing and automation.
  - Subscriptions are centrally managed.
    - Subscriptions will be handled as DDM subscriptions.
    - Good in case of site problems.
    - Good in case of general transfer problems.
  - Monitoring should be done via the dashboard.
    - If functionality is not available, ask developers for it.
  - Deletion has to be handled carefully for consistency reasons.
  - Consistency checks are on the way to be done centrally.

Future Development

DQ2

- Regularly new versions will be published, version 1.0 is under development.
- Consistency check catalogs should be available for endusers to track completeness of datasets at sites with “user-friendly” commands.
- Deletion Catalog will be integrated.
- Merging of small files is under investigation.
- Automatic check-sum test with adler-32 algorithm.

Monitoring

- Tools for monitoring deletion will be developed.
- Information about deleted datasets might be available in the future.

Existing Problems

Software

- Still there are transfers that fails (partly)
- $\Rightarrow$ not cleaned up properly
  - Zombies (zero length files) and orphans (files not registered in any catalog) are created on storage element
- Agents control fails, algorithms of agents are under constantly under investigation and change from version to version

Design

- Computing Model can't be enforced to be followed
  - Has to be discussed if this is wished or not.