Distributed Data Management
on the Grid

Mario Lassnig
Who am I?

- Mario Lassnig
- Computer scientist
  - main field of study was theoretical (algorithm design)
  - working on/with distributed and embedded systems since 2003 (Austrian Research Centers, Navigation)
- since 2006 at CERN, PH-ATLAS Computing
  - doing a PhD on distributed data management
  - working on DQ2 (Don Quijote 2)
    - the experiment’s distributed data management system
Outline

- Some basic definitions
- ATLAS Distributed Data Management (DDM)
  - What are the components?
  - How does it work?
  - How are we testing the system?
  - Where are the problems?
  - and how do we go about them

- Some graphics/texts courtesy S. Campana, thanks!
Data Resource Management is the development and execution of architectures, policies, practices and procedures that properly manage the full data lifecycle needs of an enterprise.

- by DAMA (Data Management Association)

Two teams
- DQ2 and DDM Operations

We (the DQ2 people) are concerned with
- development

The DDM Operations people are concerned with
- execution
- policies
- practices and procedures

Naturally, those are not mutually exclusive
- Operations people request features from us, based on needs
- We suggest best practices to them, based on technological limits
- And of course, users come directly to us to request features
Grid?

- Term coined in the late 90s, Ian Foster (Argonne)
  - massive distributed metacomputing

- Idea is to connect heterogeneous computing infrastructures together to solve a common goal
  - distributed cluster computing?
  - large-scale parallel processing?

- Three-point checklist
  - Resources are not managed centrally
  - Open standards
  - Quality of service

- Data Grid
  - controlled sharing and management of large amounts of distributed data
  - How much is large amounts?
    - Moore’s law (computing: exponential growth)
      vs. Kryder’s law (storage: doubling every year)
      vs. Nielsen’s law (network: 0.5 per year)
      vs. Wirth’s law (software: is getting more slower than hardware gets faster)
    - “LHC era” computing: $10^5$ CPUs, 10s Petabytes storage
The ATLAS Computing Model

- Decentralised structure
  - make use of existing Grid technology
- Sites are organised in Tiers
  - hierarchical
  - each Tier has a specific role
    - Tier-0
      - record RAW detector data
      - distributed data to Tier-1s
      - calibration and first-pass reconstruction
    - Tier-1s
      - permanent storage
      - capacity for reprocessing and bulk analysis
    - Tier-2s
      - Monte-Carlo simulation
      - user analysis
The ATLAS Computing Model

- Sites are also organised in clouds
  - not the “computer science” definition of clouds, though!
- Every cloud has a major Tier-1 and associated Tier-2s
- Mostly geographical and/or political
  - support
  - deployment
  - funding
DQ2 (Don Quijote 2)

- Dataset
  - set of files
- DQ2 enforces dataset
  - placement
  - replication
  - deletion
  - access
  - consistency
  - monitoring
  - accounting

DQ2 Clients & API

Common Modular Framework

Centrals Catalogs
- Repository, Content
- Location, Accounting, Subscription
- Tracer

Site Services
- Transfer
- Consistency
- Deletion

Production
Analysis
Data Export
Interactive
Physics Metadata

WLCG
LHC COMPUTING GRID
OPEN SCIENCE GRID
NORDUGRID
Datasets

- DQ2 stores system metadata for files and datasets
  - owner, filesizes, checksums, …
  - datasets are versioned
- DQ2 does not store physics metadata though
  - we do not know about events, luminosity, …
  - separate metadata catalogue project that interfaces with DQ2 (called AMI)
- Datasets have 3 different states
  - Open: dataset version is mutable and files can be added and removed
  - Close: dataset version is immutable. a new open version can be made though
  - Frozen: dataset is immutable (subject to hardware reliability :-)
- Dataset hierarchy
  - flat namespace
  - datasets can be aggregated into containers (still look like datasets to users)
  - derived/overlapping datasets with the same (logical) files
Central catalogues

- **Dataset Repository Catalogue**: Holds all dataset names, their unique IDs, and system metadata.

- **Dataset Content Catalogue**: Maps each dataset to its constituent files. Files are identified by a GUID (Grid Unique IDentifier) and a LFN (Logical File Name).

- **Dataset Location Catalogue**: Stores locations of each dataset.

- **Container Catalogue**: Maintains versioning information and information on containers.

- **Local File Catalogue**: One catalog per cloud (in the US per site), providing logical to physical file name mapping.
SRM and Space Tokens

Storage systems implement a common interface

- **Storage Resource Manager (SRM)**
  - gridftp as common transfer protocol
  - storage specific access protocols

- **Space Tokens**
  - partitioning of storage resources according to activities

- Each ATLAS site is identified by a site name and according space token
  - DESY-ZN_PRODDISK

- 'srm': 'token:ATLASPRODDISK:srm://lcg-se0.ifh.de:8443/srm/managerv2?SFN=/pnfs/ifh.de/data/atlas/atlasproddisk/'
Accounting

- Space Tokens provide easy accounting

Graphs showing used disk space for DESY-ZN_DATADISK and CERN-PROD_DATADISK.
Data movement

- Datasets are subscribed from a site to another site
  - dataset placement request
  - wished for as automatic updates/synchronisation in the future

- Transfer Agents (Site Services) enforce the request for a given site
  1. Resolve the dataset content
     - via central catalog
  2. Look for missing files at destination site
     - via destination site LFC
  3. Finds existing location of missing files
     - ask location catalog and source site LFC
  3b. Optionally trigger stage recall
     - if data is on tape storage, then initiate a stage request from tape to disk buffer in advance
  4. Trigger data movement
     - via File Transfer Service (FTS)
  5. Register destination file in destination LFC
File Transfer Service (FTS)

- FTS is a third party point-to-point file transfer service
  - one server per cloud
- Channels are usually privileged, pledged network links
  - optical private networks
  - high-speed links
  - no multi-hop
  - every other transfer is going through the internet
- The FTS channel at T1 of cloud X defines channels for
  - T1(X)-T2(X) and T2(X)-T1(X)
  - T1s-T1(X)
  - *-T1(X) and *-T2s(X)
  - CERN-T1s are served from CERN FTS
Users and clients

- Command line clients and Python APIs exist for all possible DQ2 operations
  - creating datasets, registering files, requesting subscriptions, ...

- High-level tools to support user workflows
  - dq2-get ... download data from the grid
  - dq2-put ... upload data into the grid
  - dq2-ls ... query the data on the grid

- Both ATLAS analysis tools (pAthena and Ganga) are integrated with DQ2 and DDM
  - user define input datasets
  - jobs go to the data (located via DDM automatically)
  - output is organised in datasets again

- Writing output datasets is tricky
  - where do we put the data?
  - directly from the worker node to the user’s site?
  - at the site where the job ran? then subscribe the dataset to the user’s site?
  - lots of issues here... user’s want to have their data “near” them, but that doesn’t really make sense in a distributed system
Central deletion service

- **Generally, we do not allow users to delete data**
- Users can mark data as obsolete and the central deletion service will resolve dependencies and schedule deletion.
  - Overlapping/derived datasets share files.

**DDM Deletion Activity Overview**

<table>
<thead>
<tr>
<th>Clouds</th>
<th>Submitted datasets</th>
<th>Waiting datasets</th>
<th>Deleted files for last hour</th>
<th>Errors for last hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>CANADASITES</td>
<td>255</td>
<td>368</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CERN</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FRANCESTES</td>
<td>0</td>
<td>17610</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FZKSTES</td>
<td>19</td>
<td>677</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ITALYSTES</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NDGF</td>
<td>0</td>
<td>56</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NLSITES</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SPAINSTES</td>
<td>0</td>
<td>2232</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TAIWANSTES</td>
<td>2590</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>UKSTES</td>
<td>295</td>
<td>1381</td>
<td>8560</td>
<td>0</td>
</tr>
<tr>
<td>USASITES</td>
<td>34025</td>
<td>2958</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**IN2P3-CC_ATLASDATADISK**

Free space - last 7 days
Common Computing Readiness Challenge

- Test the computing, data export and consolidation with all LHC experiments at the same time
- ATLAS Full Data Flow
Common Computing Readiness Challenge

12h backlog fully Recovered in 30 minutes

Subscriptions injected every 4 hours and immediately honored

All Experiments

Averaged Throughput during the last 24 hrs (21/05 - 22/05) VO-wise Data Transfer From All Sites To All Sites

TO-T1 throughput
Hmmmmm...

- this looks pretty solid, where’s the catch?
Hmmm... 

- this looks pretty solid, where’s the catch?

- in reality, keeping all services consistent is a nightmare (and that’s an understatement)
Hmmmmm...

- this looks pretty solid, where’s the catch?

- in reality, keeping all services consistent is a nightmare (and that’s an understatement)
  - software breaks
  - hardware breaks
  - user errors or mistakes
  - uncontrollable third party influences
Consistency service

- every time something breaks, we need manual operator intervention
- the consistency service is now helping to make this automated
- schedules a file for checking on every modification
  - checks availability and correctness of file
    - in central catalogues
    - in local catalogues
    - in storage namespace
    - in storage
- tedious and time-consuming process
  - must take care not to overload system with consistency checks
- it’s a design flaw/feature of the system
  - multiple heterogeneous systems working together
- but we have it reasonably under control now
What we cannot control though

- is our users
  - and in a sense that’s both good and bad

- remember the Computing Model
  - data is moved centrally
  - user submits analysis job
  - job runs at Tier-2
First Beam Day

- Users going to storage directly (instead of DQ2)
  - killed the data export

- One user overloading a site leads to a large number of errors
  - kill site performance
  - and it wasn’t even beam data...
How will we tackle the use(r) case?

- DQ2 Clients trace user access non-intrusively
  - who, what, when, how
Conclusions

- DDM works well for data export, consolidation and simulated production
  - many parts of the system are in stable use since years
  - we are confident that it can take “LHC era” load

- The real challenge is now to support users
  - educate them (how not to abuse the system)
    - sadly, many users are very opportunistic
    - and putting arbitrary restrictions is never a good idea
      (and usually leads to angry emails or clever ideas how to circumvent them)
  - tracer information provides necessary insights
  - we have simulation projects and studies ongoing
  - overall goal is to achieve restriction-free and policy-free access to data

- DDM Operations is now focusing on day-to-day activities
- DQ2 people now focusing on tackling the use(r) case technology-wise
  - and I might even get a PhD out of it 😊
Distributed Data Management
on the Grid

Mario Lassnig