CMS Use of a Data Federation

Peter Elmer, Princeton University
(for the CMS Computing group)
CMS at the Large Hadron Collider (LHC)

- General purpose detector for both proton-proton and Heavy Ion collisions at the LHC
- A collaboration of ~3800 physicists and engineers from ~180 institutions in 39 countries
- Computing resources are provided to the experiment by about 60 computing centers
The baseline CMS Computing Model was developed in 2005: an era of unreliable sites, unreliable storage systems, an unreliable grid and poor monitoring if one wanted to understand what was happening.

We opted for a very simple model:

- Datasets are statically placed at sites, either by the central computing operations or by request of analysis groups.
- Jobs go where the data is located and read data locally.
- No data is moved on response to job submissions.
- Uses more storage, a tradeoff between efficiency and system complexity.
**PhEDEx - Physics Experiment Data Export**

Generic data transfer and placement system
Design focused on scalability and reliability from the beginning
A transfer management database plus a set of loosely coupled, stateless agents

**Volume:**
Up to 250TB/day transferred (500k transfer successes/week)

**Files:**
11M (24M incl. replicas)

**Throughput:**
Up to 2-2.5GB/s aggregate in peak weeks

**CMS average data transfer volume (2004-2012)**
Computing tool with the greatest longevity in CMS
Storage Heterogeneity

Breakdown of storage solutions at sites that are nodes in the CMS PhEDEx topology.
Data Federations

• The maturity of the computing system leads to an emphasis on new goals: efficiency of storage use (eventually fewer replicas?), lower latency for end users, greater transparency relative to storage system heterogeneity, improved usability for analysis.

• Much of the traffic is to T2 sites, where analysis is happening.
Data Federations

• Exploit WAN data access as a complement to local access: allow jobs running at one site to transparently access files located at other sites.

• In short, create a federated data storage system which provides access to files residing in a heterogeneous mix of site storage systems and independent administrative domains.

• Requires a common global namespace. For efficiency applications also need to handle gracefully higher latency file reads and bandwidth limitations.

• A data federation has however several interesting use cases.
Use cases

**Interactive use-case**
- debugging single events / files
  - hosted remotely, event viewer
Use cases

**Interactive use-case**
- debugging single events/files
  - hosted remotely, event viewer

**Fallback use-case**
- if a grid jobs fails to open a file, no application failure: have it try again remotely
  - loss of efficiency, but no crash
Use cases

**Interactive use-case**
- debugging single events/files
  - hosted remotely, event viewer

**Fallback use-case**
- if a grid job fails to open a file, no application failure: have it try again remotely
  - loss of efficiency, but no crash

**Overflow use-case**
- purposely allow a job to go to sites not hosting the full input dataset
  - if one data source is in the federation, a forced fallback will “backfill” otherwise-idle analysis CPUs and work around non-optimal data distribution
The CMS data federation is implemented using xrootd. Single entry point for file access is the "redirector". It maintains no permanent file location information, but queries registered xrootd data servers as to whether they have the file. The client is then redirected to one which can provide access.

GSI-based authentication is used, via xrootd plugin to map GSI to username.
Global file namespace

• Since 2005 CMS has imposed a single logical file namespace globally. File access (the PFN) at a given site is determined by a site-specific set of transformation rules, i.e. the so-called "trivial file catalog".

• The use of these rules is setup such that a "fallback" access can be specified. If the job attempts to read the file via the standard (first match) method and it fails, the job will then attempt to open the file via the fallback method. This is configured as the entry point into the global data federation.

• This bit of application-level logic permits the implementation of the use cases mentioned earlier.
Optimizing high-latency file access

- o(50ms) latencies within regions and o(100ms) latency between regions, much greater than local access. Manage reads carefully to avoid small reads.

Use TTreeCache to group many reads into a single, large, vector read. Limit I/O round-trips to one per 512kB read.

Top 10 data products are colored, the rest black.
Deployment

• Funded project "Any Data, Anywhere, Anytime" (AAA) in the US. The first deployment began with sites there as a regional federation.

• Now being expanded to include all CMS sites in a global federation

• Access to data in the federation can of course come from anywhere

• The realization of such a system is critically dependent on proper monitoring
Global Data Federation
xrootd server side monitoring (UDP)
xrootd monitoring

- UDP packets describing individual user sessions
- XRootd server
- UDP-TCP replicator @ UCSD: allows several analysers
- TTree: full UDP packets
- XrdMon collector / analyser @ UCSD
- TTree: File Access Reports
- Active MQ
- CMS Dashboard @ CERN
- Embedded HTTPD
- Table of current transfers
- OSG Gratia @ UNL
- Produce daily usage reports
Growth of remote access data access

30 sites provide access to their data, 37 use as fallback
Realtime monitor

Allows operators/sysadmins to spot jobs causing problems, for example the origin of excessive load.
Data outgoing via xrootd by site - Past Month

Dominated by a few sites at the moment
Standard PhEDEx Data Transfers - Past Month

CMS PhEDEx - Transfer Volume
30 Days from 2013-04-15 to 2013-05-15

Data Transferred [TB]

Time

- Maximum: 218.87 TB
- Minimum: 1.53 TB
- Average: 90.65 TB
- Current: 1.53 TB

... plus 14 more
CMS Jobs by Access Type

Jobs per access type

Local

Remote

0 20000 40000 60000 80000 100000 120000 140000 160000 180000 200000 220000 240000 260000 280000

Jobs number

app-succeeded app-failed site-failed aborted cancelled app-unknown pending running
Daily Summary Report for Operators

<table>
<thead>
<tr>
<th>Source Site</th>
<th>Volume GB</th>
<th># of Transfers</th>
<th>Yesterday Diff</th>
<th>One Week Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMS Xrootd Site Unknown</td>
<td>966</td>
<td>2,987</td>
<td>29%</td>
<td>2165%</td>
</tr>
<tr>
<td>GLOW</td>
<td>2,164</td>
<td>2,920</td>
<td>-35%</td>
<td>-79%</td>
</tr>
<tr>
<td>GLOW_Internal</td>
<td>145,069</td>
<td>93,343</td>
<td>22%</td>
<td>79%</td>
</tr>
<tr>
<td>MIT</td>
<td>378</td>
<td>6,793</td>
<td>14%</td>
<td>-90%</td>
</tr>
<tr>
<td>Nebraska</td>
<td>2,300</td>
<td>11,940</td>
<td>-42%</td>
<td>-90%</td>
</tr>
<tr>
<td>Purdue</td>
<td>327</td>
<td>2,670</td>
<td>-77%</td>
<td>-3%</td>
</tr>
<tr>
<td>T1_FR_IN2P3</td>
<td>19,227</td>
<td>37,739</td>
<td>16%</td>
<td>237%</td>
</tr>
<tr>
<td>T1_IT_CNAF</td>
<td>350</td>
<td>1,585</td>
<td>15%</td>
<td>-74%</td>
</tr>
<tr>
<td>T1_UK_RAL</td>
<td>58</td>
<td>332</td>
<td>-40%</td>
<td>-85%</td>
</tr>
<tr>
<td>T2_IT_Bari</td>
<td>3,325</td>
<td>10,504</td>
<td>310%</td>
<td>-61%</td>
</tr>
<tr>
<td>T2_IT_Pisa</td>
<td>369</td>
<td>793</td>
<td>105%</td>
<td>-2%</td>
</tr>
<tr>
<td>UCSD</td>
<td>1,872</td>
<td>5,956</td>
<td>27%</td>
<td>1917%</td>
</tr>
<tr>
<td>UFL</td>
<td>98</td>
<td>2,018</td>
<td>-10%</td>
<td>-78%</td>
</tr>
<tr>
<td>USCMS-FNAL-WC1</td>
<td>13,219</td>
<td>8,542</td>
<td>-66%</td>
<td>-5%</td>
</tr>
<tr>
<td>Vanderbilt</td>
<td>160</td>
<td>3,370</td>
<td>-13%</td>
<td>361%</td>
</tr>
</tbody>
</table>

Quick ascii-art/table at-a-glance morning summary of system statistics from the previous day.
Summary

• The CMS Computing Model relies on static data placement, job movement to the data and local data access. This simple model has served us well during commissioning and the first LHC run.

• Over the past 1.5 years we have been deploying and commissioning a system, based on xrootd, allowing for WAN access to data by jobs and building a global data federation from heterogeneous resources.

• The system has been growing steadily and we expect that it will be a critical piece of our computing system for the next LHC run.
Proper Credits

- UCSD: Frank Wuerthwein, Matevz Tadel, Igor Sfiligoi
- UW: Dan Bradley, Sridhara Dasu
- UNL: Brian Bockelman, Ken Bloom
- The xrootd team: Andy Hanushevsky, Lukasz Janyst, Andreas Peters, Gerri Ganis, et. al.
- dCache implementation: Gerd Behrmann
- Many T1/T2 sysadmins who deployed the system