CMS evolution on data access: xrootd remote access and data federation

Giacinto DONVITO
INFN-BARI
Outlook

- CMS computing model: first assumptions and results
- Technology trends and impact on HEP Computing models
- Xrootd concepts and features
- CMS framework enhancements
- Xrootd Redirector use cases and basic ideas
- Xrootd Redirectors actual deployment
- Xrootd first test and monitoring
Outlook

- CMS computing model: first assumptions and results
- Technology trends and impact on HEP Computing models
- Xrootd concepts and features
- CMS framework enhancements
- Xrootd Redirector use cases and basic ideas
- Xrootd Redirectors actual deployment
- Xrootd first test and monitoring
CMS computing model: first assumptions

- The Computing Models of the LHC experiments were designed using Monarc (Models of Networked Analysis at Regional Centers for LHC Experiments) Model assumption:

  - Resource are located in distributed centers with a hierarchical structure
    - Tier 0 Master computing center, full resource set (disks and tapes)
    - Tier 1 National Centre, resource subset (disks and tapes)
    - Tier 2 Major Centre (only disks)
    - Tier 3 University Department
  - The network will be bottleneck
  - We will need a hierarchical mass storage because we cannot afford sufficient local disk space
  - The file catalogue will not scale
  - We will overload the source site if we ask for transfers to a large number of sites
  - We need to run job “close” to the data to achieve efficient CPU utilization
  - We need a a priori structured and predictable data utilization
CMS computing model: first assumptions

Tier 1
- France Regional Centre
- Germany Regional Centre
- Italy Regional Centre

Tier 0
- CERN Computer Centre
- Offline Processor Farm (~20 TIPS)
- FermiLab (~4 TIPS)

Tier 2
- Caltech (~1 TIPS)
- Tier2 Centre (~1 TIPS)
- Centre (~1 TIPS)

Tier 4
- Physics data cache
- Institute (~0.25 TIPS)

1 TIPS is approximately 25,000 SpecInt95 equivalents

There is a “bunch crossing” every 25 nsecs. There are 100 “triggers” per second. Each triggered event is ~1 MByte in size.

Physicists work on analysis “channels”. Each institute will have ~10 physicists working on one or more channels; data for these channels should be cached by the institute server.

[Image courtesy of Harvey Newman, Caltech]
CMS computing model: first assumptions

Take CMS data flow as example

CMS T1→T2 transfers likely to be bursty and driven by analysis demands

~10/100 MB/s for worst/best connected T2's (2006)

CMS T2→T1 transfers almost entirely fairly continuous simulation transfers

~ 10MB/s (NOTE: aggregate input rate into T1's comparable to rate from T0!)

NB: averaged sustained throughputs.

Tier-0
4.6 M-SI2K
0.4 PB disk
4.9 PB tape
5 Gbps WAN

225 MB/s
(RAW)

Tier-1
2.5 MSI2K
1.0 PB disk
2.4 PB tape
~10 Gbps WAN

40 MB/s
(RAW, RECO, AOD)

Tier-1s

Tier-2
0.9 M-SI2K
0.2 PB disk
1 Gbps WAN

Up to 1 GB/s
(AOD analysis, calibration)

Tier-2s

Tier-0

WNs

280 MB/s
(RAW, RECO, AOD)

240 MB/s
(skimmed AOD,
Some RAW+RECO)

280 MB/s
(RAW, RECO, AOD)

225 MB/s
(RAW)

WNs

Tier-1

WNs

48 MB/s
(MC)

Tier-1s

60 MB/s
(skimmed AOD,
Some RAW+RECO)

12 MB/s
(MC)

Tier-2s

Tier-1s

Tier-1

Tier-2

Tape

Courtesy: J.Hernandez (CMS)
CMS computing model: first results

- Data transfers among sites is much more reliable than expected
  - WAN network performances are growing fast and the network infrastructure is reliable
- Retrieving files from a remote sites could be easier than retrieving it from a local hierarchical mass storage system
  - It looks like we need a clearer partition between disk and tape
    - Using tape only like an archive
- Geographically distributed Job submission and matchmaking are working well
  - The tool developed to help the final users are easy and make the usage of the grid transparent to the user
CMS computing model: first results

It is “easy” to get out of a single T0/T1 up to 300-400MB/s sustained of WAN transfer to all others CMS sites.
A very small number of datasets are used very heavily.
Outlook

- CMS computing model: first assumptions and results
- Technology trends and impact on HEP Computing models
- Xrootd concepts and features
- CMS framework enhancements
- Xrootd Redirector use cases and basic ideas
- Xrootd Redirectors actual deployment
- Xrootd first test and monitoring
Technology trends and impact on HEP Computing models
Technology trends and impact on HEP Computing models

- The WAN available bandwidth is comparable with the backbone available at LAN level
- Network flows are already larger than foreseen at this point in the LHC program, even with lower luminosity
- Some T2’s are very large
  - All US ATLAS and US CMS T2’s have 10G capability.
- Some T1-T2 flows are quite large (several to 10Gbps)
- T2-T2 data flows are also starting to become significant.
- The concept of transferring file regionally is substantially broken!
Technology trends and impact on HEP Computing models

- The vision progressively moves away from all hierarchical models to peer-peer
  - True for both CMS and ATLAS
  - For reasons of reduced latency, increased working efficiency
- The challenge in the next future will be the available IOPS on storage systems
  - It is clear we need to optimize the IO at the application level as the disk will not increase the performance too much in the future
- The hierarchical mass storage system could not cope with dynamic data stage-in request
  - The cost per TB of the disk allow us to build huge disk-only facilities
Outlook

- CMS computing model: first assumptions and results
- Technology trends and impact on HEP Computing models
- Xrootd concepts and features
- CMS framework enhancements
- Xrootd Redirector use cases and basic ideas
- Xrootd Redirectors actual deployment
- Xrootd first test and monitoring
Xrootd concepts and features

What Is xrootd?

- A file access and data transfer protocol
  - Defines POSIX-style byte-level random access for
    - Arbitrary data organized as files of any type
    - Identified by a hierarchical directory-like name
  
- A reference software implementation
  - Embodied as the xrootd and cmsd daemons
    - xrootd daemon provides access to data
    - cmsd daemon clusters xrootd daemons together
Xrootd concepts and features

What Isn’t xrootd?

- It is not a POSIX file system
  - There is a FUSE implementation called xrootdFS
    - An xrootd client simulating a mountable file system
    - It does not provide full POSIX file system semantics
- It is not an Storage Resource Manager (SRM)
  - Provides SRM functionality via BeStMan
- It is not aware of any file internals (e.g., root files)
  - But is distributed with root and proof frameworks
    - As it provides unique & efficient file access primitives
Xrootd concepts and features

Primary xrootd Access Modes

✦ The root framework
  ✦ Used by most HEP and many Astro experiments (MacOS, Unix and Windows)

✦ POSIX preload library
  ✦ Any POSIX compliant application (Unix only, no recompilation needed)

✦ File system in User Space
  ✦ A mounted xrootd data access system via FUSE (Linux and MacOS only)

✦ SRM, globus-url-copy, gridFTP, etc
  ✦ General grid access (Unix only)

✦ xrdcp
  ✦ The parallel stream, multi-source copy command (MacOS, Unix and Windows)

✦ xrd
  ✦ The command line interface for meta-data operations (MacOS, Unix and Windows)
What Makes **xrootd** Unusual?

- A comprehensive plug-in architecture
  - Security, storage back-ends (e.g., tape), proxies, etc
- Clusters widely disparate file systems
  - Practically any existing file system
    - Distributed *(shared-everything)* to JBODS *(shared-nothing)*
  - Unified view at local, regional, and global levels
- Very low support requirements
  - Hardware and human administration
Xrootd concepts and features

Protocol Driver
(XRD)

Authentication
(gsi, krb5, etc)

Authoritication
(dbms, voms, etc)

Protocol (1 of n)
(xroot, proof, etc)

Logical File System
(ofs, sfs, alice, etc)

Physical Storage System
(ufs, hdfs, hpss, etc)

Let’s take a closer look at xrootd-style clustering

Replaceable plug-ins to accommodate any environment
Xrootd concepts and features

A Simple xrootd Cluster
Xrootd concepts and features

Exploiting Stackability

Data is uniformly available
By federating three distinct sites

An exponentially parallel search!
(i.e. $O(2^n)$)

Federated Distributed Clusters
Federated Distributed Clusters

- Unites multiple site-specific data repositories
  - Each site enforces its own access rules
    - Usable even in the presence of firewalls
  - Scalability increases as more sites join
    - Essentially a real-time bit-torrent social model
      - Federations are fluid and changeable in real time
        - Provide multiple data sources to achieve high transfer rates
  - Increased opportunities for data analysis
    - Based on what is actually available
Xrootd concepts and features

Copy Data Access Architecture

 понравилось The built-in **File Residency Manager** drives

- Copy On Fault
- Demand driven (fetch to restore missing file)
- Copy On Request
- Pre-driven (fetch files to be used for analysis)

```bash
open(“/my/file”) xrdcp -x xroot://mm.org//my/file /my:
```

**Client**

**Manager** (a.k.a. Local Redirector)

**Manager** (a.k.a. Local Redirector)

**Manager** (a.k.a. Local Redirector)

**Meta-Manager** (a.k.a. Global Redirector)

xrdcp copies data using two sources.
Xrootd concepts and features

Direct Data Access Architecture

- Use servers as if all of them were local
- Normal and easiest way of doing this
- Latency may be an issue (depends on algorithms & CPU-I/O ratio)
- Requires Cost-Benefit analysis to see if acceptable
Xrootd concepts and features

Cached Data Access Architecture

- Front servers with a caching proxy server
- Client access proxy server for all data
- Server can be central or local to client (i.e. laptop)
- Data comes from proxy’s cache or other servers
Outlook

- CMS computing model: first assumptions and results
- Technology trends and impact on HEP Computing models
- Xrootd concepts and features
- CMS framework enhancements
- Xrootd Redirector use cases and basic ideas
- Xrootd Redirectors actual deployment
- Xrootd first test and monitoring
CMS framework enhancements

- The CMSSW 4_X contains ROOT 5.27/06b, which has significant I/O improvements over 3_x
- CMSSW 4_x also contains modest I/O improvements on top of out-of-the-box ROOT I/O
- New ROOT added:
  - Auto-flushing: All buffers are flushed to disk periodically, guaranteeing some level of locality.
  - Buffer-resizing: Buffers are resized so approximately the same number of events are in each buffer.
  - Read coalescing: “Nearby” reads (but non-consecutive) are combined into one.
CMS framework enhancements

- Incremental improvements through 3_x and 4_x: Only one event tree, improved event read ordering, TTreeCache became functional, caching non-event trees.

- Improved startup: While TTreeCache is being trained, we now read all baskets for the first 10 events at once. So, startup is typically one large read instead of many small ones.

![Diagram showing improved startup latencies](image)

We read all branches for the first 20 events; compared to the entire file size, the bandwidth cost is minimal. On high latency links, this can cause a 5 minute reduction in time to read the first 20 events.
CMS framework enhancements

Upcoming Enhancements

✦ “Real” Asynchronous prefetch (using threads and double-buffering). Brian’s opinion: the current patch set is problematic and not usable by CMS. At least a year out.


✦ ROOT implementation of things in CMSSW: multiple TTreeCaches per file. Improved startup latencies.
Outlook

- CMS computing model: first assumptions and results
- Technology trends and impact on HEP Computing models
- Xrootd concepts and features
- CMS framework enhancements
- Xrootd Redirector use cases and basic ideas
- Xrootd Redirectors actual deployment
- Xrootd first test and monitoring
Xrootd Redirector use cases and basic ideas

- The goal of this activity is to decrease the effort required for the end-user to access CMS data.

- We have built a global prototype, based on Xrootd, which allows a user to open (almost) any file in CMS, regardless of their location or the file’s source.

- Goals:
  - **Reliability**: Seamless failover, even between sites.
  - **Transparency**: Open the same filename regardless of source site.
  - **Usability**: Must be native to ROOT.
  - **Global**: Cover as many CMS files as possible.
The target of this project:

- **End-users**: event viewers, running on a few CMS files, sharing files with a group.
- **T3s**: Running medium-scale ntuple analysis

None of these users are well-represented by CMS tools right now.

So, a prototype is better than nothing…
Xrootd Redirector use cases and basic ideas

**Xrootd Prototype**

- Have a global redirector users can contact for all files.
  - Can use any ROOT-based app to access the prototype infrastructure! Each file only has one possible URL
- Each participating site deploys at least 1 `xrootd` server that acts like a proxy/door to the external world.
Xrootd Redirector use cases and basic ideas

Global Xrootd Federation

Another Xrootd Cloud
CMS/ATLAS Site

CMS Site

Global Redirector

2. Queries all sites for file (if not in cache)
1. Open file
3. Redirects user to site
4. Serves data

User

Q: Open /store/foo
A: Check Site A
Q: Open /store/foo
A: Success!

Global Xrootd Redirector

Site A
Lustre Storage

Site B
Hadoop Storage

Site C
dCache Storage
Xrootd Redirector use cases and basic ideas

1. Attempt login
4. Login successful!
5. Open /store/foon
4. Open successful!

2. Request mapping for DN / VOMS
3. Result: User "brian"

The authentication is always delegated to the site hosting the file.
Xrootd Redirector use cases and basic ideas

“Fallback” Case

- CMSSW_3_9_x includes ability to open a file remotely if the local file is missing.

Example: T3 at Omaha

- We don’t have the effort to efficiently maintain CMS PhEDEx at Omaha.
- This T3 only reads from the global xrootd system. Good continuous test.
- 6,000 wall hours in the last day.
Xrootd Redirector use cases and basic ideas

CMSSW Improvements

- In order to improve WAN streaming performance, we worked hard with the CMSSW team to optimize the I/O code.

- A sample, I/O-intensive analysis of 60k evts reading data from FNAL dCache/Xrootd:

<table>
<thead>
<tr>
<th>Site</th>
<th>Ping time</th>
<th>Wall time</th>
</tr>
</thead>
<tbody>
<tr>
<td>FNAL</td>
<td>.1ms</td>
<td>80s</td>
</tr>
<tr>
<td>Nebraska</td>
<td>17ms</td>
<td>80s</td>
</tr>
<tr>
<td>CERN</td>
<td>128ms</td>
<td>161s</td>
</tr>
</tbody>
</table>

T3 Benefits

- A T3 no longer needs to learn CMS data movement tools to access data.

- If the T3 is xrootd-based, we can use caches to improve data locality.

- If the T3 is not xrootd-based, they can just “fall back” to the global T3 cluster if the file is not local.
Xrootd Redirector use cases and basic ideas

Caching Case

Notice xrootd can download from multiple sites at once! This helps one avoid overloaded sites; bittorrent-like.
Outlook

- CMS computing model: first assumptions and results
- Technology trends and impact on HEP Computing models
- Xrootd concepts and features
- CMS framework enhancements
- Xrootd Redirector use cases and basic ideas
- Xrootd Redirectors actual deployment
- Xrootd first test and monitoring
We have yet to run into a storage system that cannot be integrated with Xrootd. Techniques:

- C API: Requires a new XrdOss plugin. Example: HDFS, dcap, REDDNet (?)
- Native implementation: Latest dCache has built-in Xrootd support; combine with xrootd.org cmsd to join federation.
- Direct access: Let xrootd read the other system’s files directly from disk (dCache @ FNAL)
In this case: we need a native xrootd door reading file using hdfs standard C library

This is quite similar to the installation of a gridftp door
In this case: we need a native xrootd door reading file using xroot library

- This machine is configurated in "proxy-mode"
- This needs that the dCache instance already provide a dCache xrootd endpoint
In this case: we need one or more native xrootd doors that are clients of the local parallel file-system (GPFS/Lustre)
Note the use cases all involve wide-area access to CMS data, whether interactively or through the batch system.

WAN performance for the software is key to this project’s success.

Current monitoring shows 99% of the bytes CMS jobs requests are via vectored reads - many jobs have about 300 reads per file total. That’s at least 30s of I/O that a locally running job wouldn’t have.

We plan to invest our time in I/O to make sure these numbers improve, or at least don’t regress.
Xrootd Redirectors actual deployment

**Xrootd deployment**

- **US Xrootd Redirector:**
  - @Nebraska
  - Collecting data from US Sites:
    - Nebraska, Purdue, Caltech, UCSD, FNAL

- **EU Xrootd Redirector:**
  - @INFN_Bari
  - Collecting data from EU Sites:
    - INFN-Bari, INFN-Pisa, DESY, Imperial College (UK)
Xrootd Redirectors actual deployment

Xrootd deployment

- Xrootd provides per-file and per-connection monitoring.
  - I.e., a UDP packet per login, open, close, disconnect, and summarizing every 5s of activity.
  - This is forwarded to a central server, where it is summarized and forwarded to several consumers.

- Also provides a per-server summary (connections, files open) that is fed directly to MonaLisa.
Xrootd Redirectors actual deployment

Xrootd distributed performance

- Using CMSSW 4_1_3 and a I/O bound analysis
- Job are running on T2_IT_LNL and data are hosted in T2_IT_Bari
  - ping time: 12ms
- CPU Efficiency drop from 68% to 50%
  - ~30% of performance drop
- This is about to be one of the worst case
Xrootd Redirectors actual deployment

Xrootd distributed performance

- Best US CMS T2 efficiency: about 80%
- CPU efficiency: about 60% in Omaha

Graph showing traffic and bit rate per second from 2010/10/05 to 2010/10/12.
Conclusion

✧ Xrootd redirector infrastructure is fulfilling few use cases that are not covered by official CMS tools

✧ It is already been used:

✧ from both final users
  ✧ Debugging code, visualization etc.

✧ and computing center
  ✧ That do not have enough man power to fully support CMS locally

✧ Tuning CMSSW is very important in order to provide good performance when accessing data remotely