Fabrizio Furano

(Activities)
What I do or I have recently done:

- Tech coordination of DPM: Disk Pool Manager
  - From legacy to the new stuff
  - The LFC
  - Our direction: DMLite
- Fostering an HTTP ecosystem for HEP
  - Making it possible to use it productively
  - The Dynamic Federations
  - The HTTP/DAV plugin for the XROOTD framework
- In the past I have done many contributions to HEP computing and data access
  - I have been directly working for ATLAS, ALICE, XROOTD, BaBar
  - Proactive since 2004 in the topics of WAN direct data access and storage federations
  - Before HEP (1998-2002/3): coordination of telephone calls and events related to communication
Main Goals of DPM

- Provide a scalable, open storage solution that integrates well with the Grid
- Give advanced features on top of the data access frontends
  - e.g. Strong auth, ACLs, draining, cataloguing, etc...
- Give quality tools to users and administrators
- Improve the feature set and performance
- Move towards using
  - standard protocols
  - standard building blocks
- Allow easy integration with old and new tools and systems
• Historically DPM was born to pool storage nodes that are (almost) independent
• The architecture can accommodate virtually any protocol. We already support many.
• Based on the idea of separation between metadata and data access
  – Direct data access to/from Disk Nodes
• Strong authentication / authorization, pooling, etc.

```
NFS  HTTP/DAV  XROOT  SRM
NS  DPM  GRIDFTP
NFS  HTTP/DAV  XROOT  GRIDFTP  RFIO
```
The head node contains the components that coordinate the activity of DPM
- Roughly: namespace management, pool management

The disk servers contain just storage media (disks) and data access frontends
- These frontends are in general instrumented in order to talk to the head node, e.g. to tell it that a client has finished uploading something
Architecturally it’s a loosely coupled system where the development tasks blur into the deployment and configuration. E.g.
- Maintaining build time dependencies among many components that give different parts of the functionalities
- Compiling many components that have to be gathered into a release
- Maintaining setup procedures that configure N kinds of components

The shape of an installation depends a lot on the choices and requirements of the sysadmin. It can vary a lot.

We maintain or coordinate all the items that allow this flexibility and broad adaptability.
Don’t forget LFC

- LCG File Catalogue
- A server that gives services:
  - Name space handling for huge namespaces
  - Replica location
- LFC is a slightly different usage of one of the core components of the DPM head node
Deployment & Usage

- DPM is the most widely deployed grid storage system
  - ~200 sites in 50 regions
  - Over 300 VOs
  - ~45 PB (10 sites with > 1PB)
- LFC enjoys wide deployment too
  - 58 instances at 48 sites
  - Over 300 VOs
Software Availability

- DPM and LFC are available via EMI and EPEL
- From 1.8.3 on we’re Fedora compliant for all components, available in multiple repositories
  - EMI1, EMI2 and UMD
  - Fedora / EPEL
- Latest production release is 1.8.6
- Some packages will never make it into Fedora
  - YAIM (We support Puppet)
  - Oracle backend
- The fresh DPM collaboration contributes/maintain some of the components
In the last year we put consistent effort into a modern, highly customizable plugin-based architecture called DMLite.

- It can be shaped in order to give the well known functionalities and much more.
- It can be plugged into many things, e.g. Apache.
- We want to use the DMLite approach more and more.
- The tech discussion is ongoing, this would mean at least:
  - Changing/redoing the instrumentations of the frontends in order to use it instead of the historical daemons
  - Make available renewed interfaces for users/sysadmins
• DMLITE is the result of a significant refactoring effort to make DPM modular
• Better separation between frontends and backend
• Proper plugins (shared libraries)
• Cleaner, more open, much improved performance
• Improved integration with standard building blocks
• Hadoop, Memcache, S3, Lustre, …
Direction for the releases

- We are stepping out from the ETICS builds
- Using Bamboo now
- ETICS had a lot of release administration features, bamboo has very few
  - Issue: using all those mandatory features to compose a release is very effort consuming. This (together with other items) slowed down considerably the cycle dev-debug/build/release
- We are evaluating a much simpler method to define releases
  - Headed to proactivity and ease of development
  - Each component defines its own ‘trunk’ and ‘release-candidate’ level, directly in SVN or GIT
  - We build them both continuously and all the packages are available separately at any time in a well defined place in AFS, always the same
  - The release is self-descriptive and needs “just” to be blessed to become an official one
  - “just” means that all the automated tests are green and that the collaborators said that everything is good
The EMI project provided the context for evolving towards HTTP/DAV
This has been a starting point to evolve the Grid components towards “standard protocols”, in particular HTTP
dCache, DPM/LFC did, STORM too

All these storage products now support HTTP/DAV as data access protocol
Our goal is to make HTTP/WebDAV data access and management possible in the context of HEP
It means to contribute in giving a “critical mass” of features so that the thing can be easily adopted
– This means filling the gap between simple-minded Web usage and prod-level HEP computing

We developed components to support the vision of using HTTP/DAV for HEP computing
– There are several components, some of them belong to joint effort projects
– There are many ways in which even simple components can be used in a design
– We want to make this possible
Why HTTP/DAV?

- It’s there, whatever platform we consider
  - HTTP is moving much more data than HEP worldwide
- We (humans) like browsers, they give an experience of simplicity
- Goes towards convergence
  - Users can use their preferred devices to access their data
  - Users can use their preferred applications to access their data
  - Sophisticated custom applications are allowed in the same way
- Interesting technical features
  - Multitalented, covers most existing use cases, while allowing new stuff
  - Applications just go straight to the data, wherever they are running
  - Staging (GET/PUT) or direct chunked access
  - Data pre-placement changes from a constraint (LAN-only access) to an optimization choice (as WAN access can be made to work well)
- Attractive for a professional to be formed in these systems
This is what we want to see as users

Sites remain independent and participate to a global view

All the metadata interactions are hidden and done on the fly

NO persistency needed here, just efficiency and parallelism

Aggregation

/dirl
/dirl/file1
/dirl/file2
/dirl/

Storage/MD endpoint 1

Storage/MD endpoint 2

With 2 replicas
Dyn Federations in a nutshell

• Goals:
  – Browse/access a huge repository made of many sites without requiring a static index
    • No "registration", no maintenance of catalogues
  – Redirect intelligently clients asking for replicas
  – Automatically detect and avoid sites that go offline
  – Accommodate both algorithmic name translations and catalog-based ones, at the same time
  – Correctly map on the fly existing SRM TURLS to HTTP URLs
  – Flexibility: accommodate almost any kind of endpoint
  – High robustness (=correctly treats failures)
  – High performance (=many requests per second per frontend)
  – Allow interactivity (achievable only with quick systems)
  – Scalability (=frontends can be replicated indefinitely)
  – Geographical scalability (=can have many frontends in different places)
How it works

The cache remembers what happened

The next metadata interactions will very likely be cached
• XROOTD is a high performance data access framework
• Supports clustering, many kinds of plugins, tape backends, WAN, has a clean, well documented protocol, can also be a DPM frontend.
• The XROOTD framework is multiprotocol
  – Can even accommodate multiple protocols on the same TCP port
• Idea: use our HTTP+WebDAV+XROOTD expertise to write an HTTP protocol plugin for the XROOTD framework
• The framework makes it quite easy and very performant
• Idea: allow the pure XROOTD sites to install it without changing their support for the XROOTD native protocol
  – A pure, clean, add-on with a trivial setup
• Many sites are using it, in HEP and others.
• A pure XROOTD site will be able to:
  – support XROOTD clients like before
  – join an XROOTD federation
  – support HTTP/DAV clients
  – join a Dynamic HTTP federation
  – hence, participate to the HTTP ecosystem
That’s it...

- This was supposed to introduce myself
- As coordinator of this I am acting as a sort of hub
- Foster the creation of a common direction
- AND give my own contribution as experienced developer
Thank you!

Questions?