LCG PEB Meeting , 17 May 2005

“ARDA xrootd testing”

Derek Feichtinger

http://cern.ch/arda

Enabling Grids for E-science in Europe

www.eu-egee.org

cern.ch/lcg

EGEE is a project funded by the European Union under contract IST-2003-508833

Contents

1. Introduction
2. Single Server xrootd
3. Load balancing xrootd systems
4. Further developments
   - Security, Proxy, cell self organization
5. Possible Integration/usage with
   - SRM
   - Disk pool managers
6. Conclusion

Introduction

xrootd:

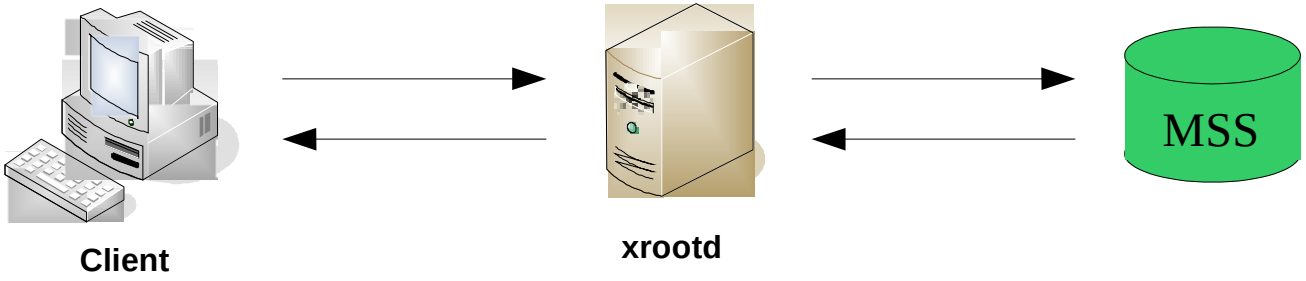
xrootd (eXtended Root Daemon) was written at SLAC and INFN Padova as part of the work to migrate the BaBar event store from Objectivity to Root I/O.

Full fledged file transfer server offering also a POSIX I/O client library.

ROOT (beginning with V4.01-02) contains the new TXNetClient class, but xrootd is a project on its own and can be used without ROOT.

➡ App. Area offers ROOT V4.04-02

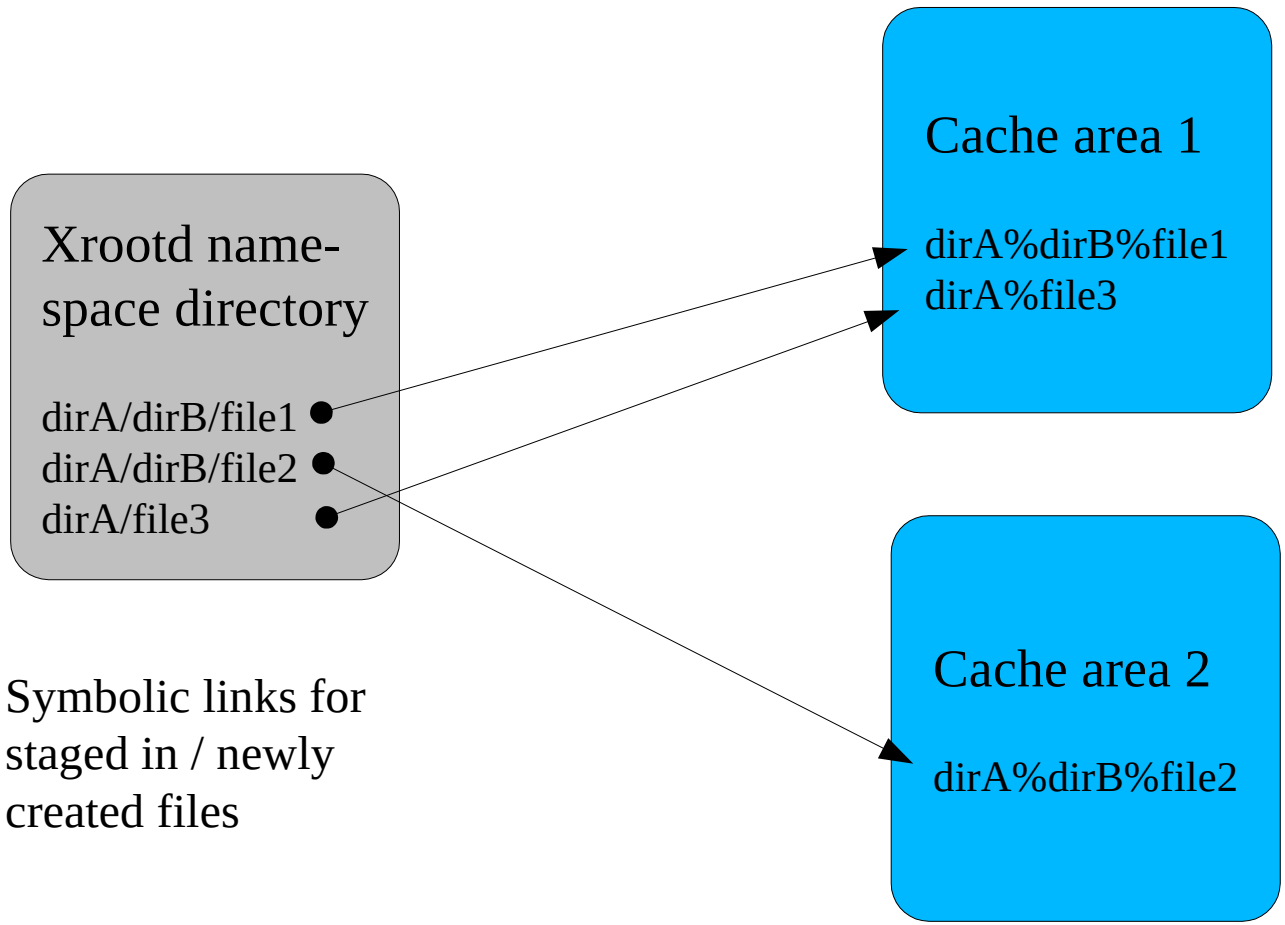
Single file server



- Connection multiplexing
- Economical usage of resources
- Heavily multithreaded
- Can make use of mmaped files and can use asynchronous I/O for multiple parallel requests by the same client
- *Opaque* information for handing over meta instructions
- Rootd compatibility mode for older ROOT versions

- Interfacing to MSS via hooks that call external scripts
- Protocol offers prestaging requests

Staging / write cache



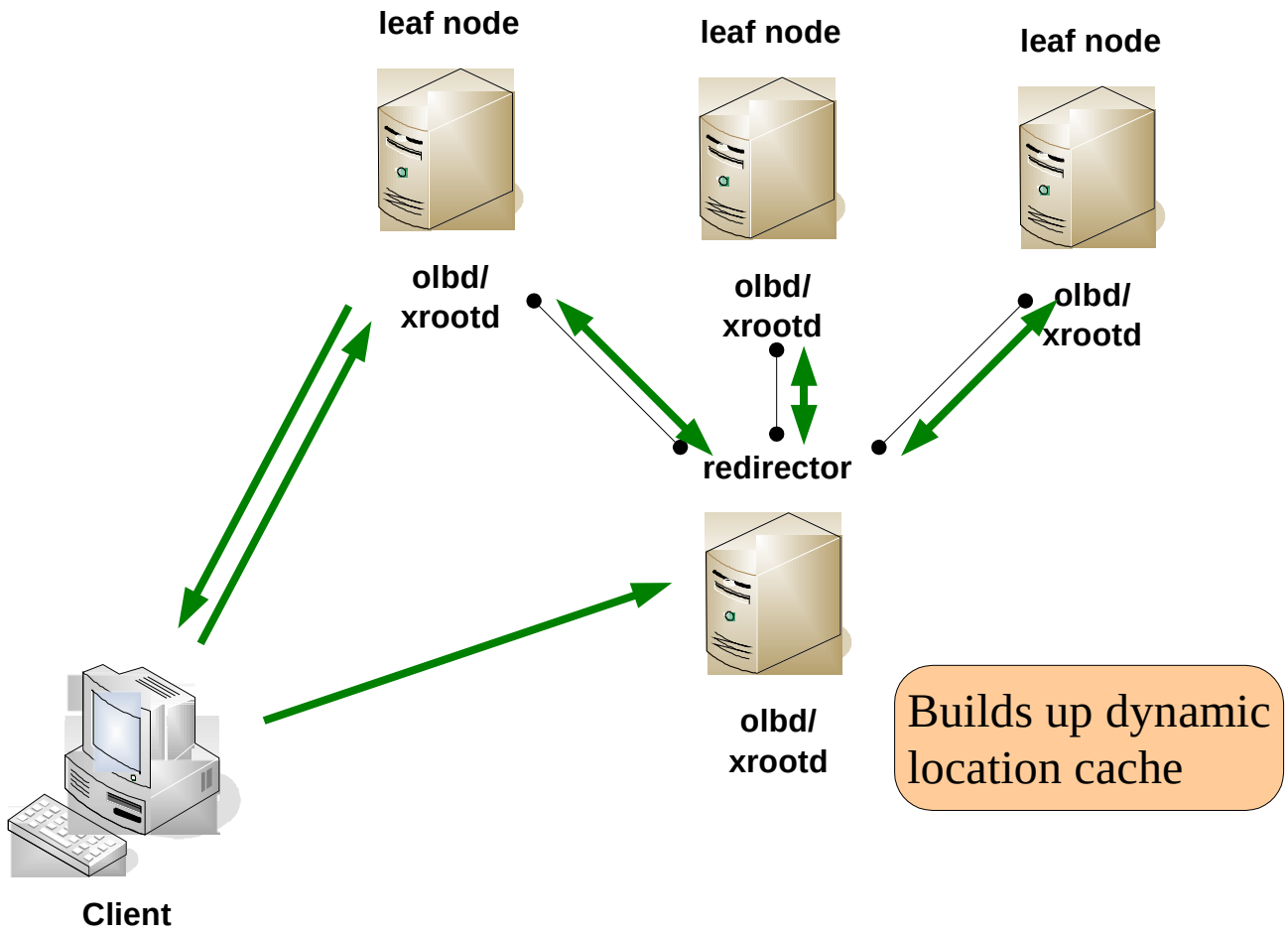
Symbolic links for staged in / newly created files

Opaque information can be used to choose cache area.

Load Balancing

- Multiple xrootd servers can be turned into a load balanced system by running a **olbd** process with every xrootd.
- An xrootd/olbd pair can run in redirector (master) mode or in leaf node (slave) mode.
- A redirector keeps connections to all its subordinate leaf nodes.
- Clients contact redirectors to get access to the total namespace of the system.
- Redirectors dynamically build up a cache of file locations.

Redirection example



xrootd/olbd general features

- Redundancy on the redirector level is gained by using multiple DNS aliased redirectors (leafs can sign up to more than one redirector).
- System can accommodate dynamic joining and leaving of nodes (leafs and redirectors). If a file is hosted on 2 leafs, clients can survive the disconnecting of the leaf they are connected to.
- Unsolicited response mechanism (in asynchronous I/O). E.g. a server can at any point decide to redirect a client or request deferral.

Read only system

- Offers access to single namespace spread over several data servers.
- Redirector reassigns slaves with minimal work involved to leaf nodes, who do the bulk of the work. Scales very well.
- Load balances over servers that contain the same requested file.
- Can cope with changes due to externally managed disk space (e.g. migrating of files between nodes). Client knows that it has been reconnected.

Configuration effort: trivial

Read only system + MSS

- Improved load balancing: If a server hosting a specific file is overloaded, the redirector assigns the request to another server that will stage in another replicate of the file.
- Load information is retrieved by the olbd via a script supplied by the administrator and additional olbd directives can be used to define statistical weights for the different load factors (Network I/O, CPU load, ...)

Configuration effort:

- Interfacing via simple scripts to MSS
- Configuration of cache area
- External purging system necessary for cache management

Read write system + MSS

- A newly created file triggers creation of a 0 length file in the MSS and the associated xrootd pool file is marked by a *.lock file.
- Migration to MSS must be done by an external process (e.g. the MPS system available with xrootd). Consistent versions in xrootd pool and MSS cannot be guaranteed, so a usage policy must be put in place.

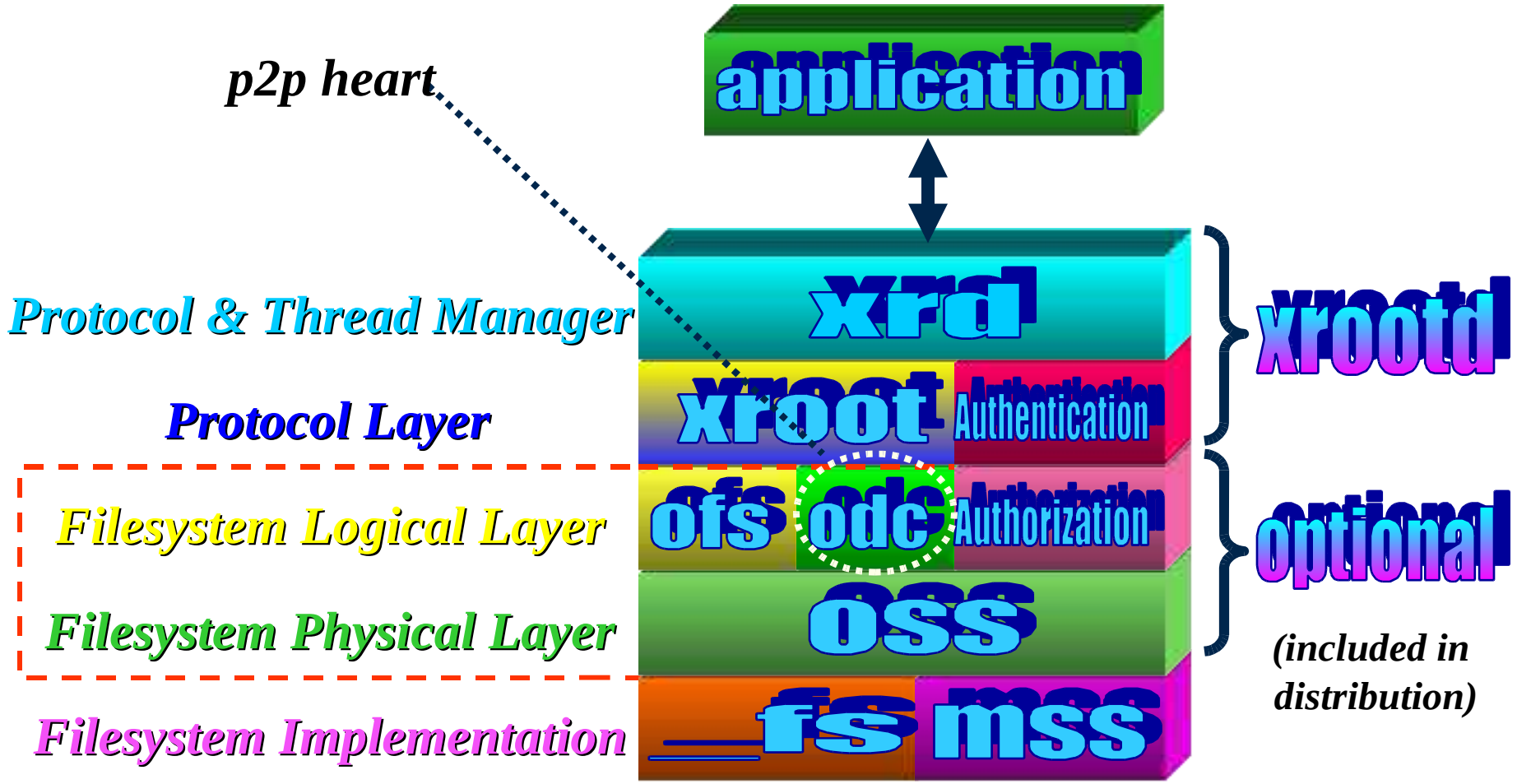
  ➡ ARDA/ALICE extension: write trigger
- No replication is allowed, since xrootd offers no mechanism to invalidate stale replicates.
- Every creation of a new file incurs a (typically) 5 s penalty.

Configuration effort:
- moderately difficult for xrootd/olbd, but rather complicated for the MPS system
- Requires a usage policy to avoid incosistencies

Security

- Xrootd needs not to be run as root.
- Authentication
  - Currently supports Kerberos authentication.
  - GSI and passwd based authentication will soon be ready (by G. Ganis, CERN).
- Authorization
  - uid/gid based with policies specified in an ACL data base (flat file)
  - ARDA/ALICE extension for generic catalogue ACL based authorization (using catalogue signed/target-encrypted tokens passed as xrootd opaque information. Paper being finished)

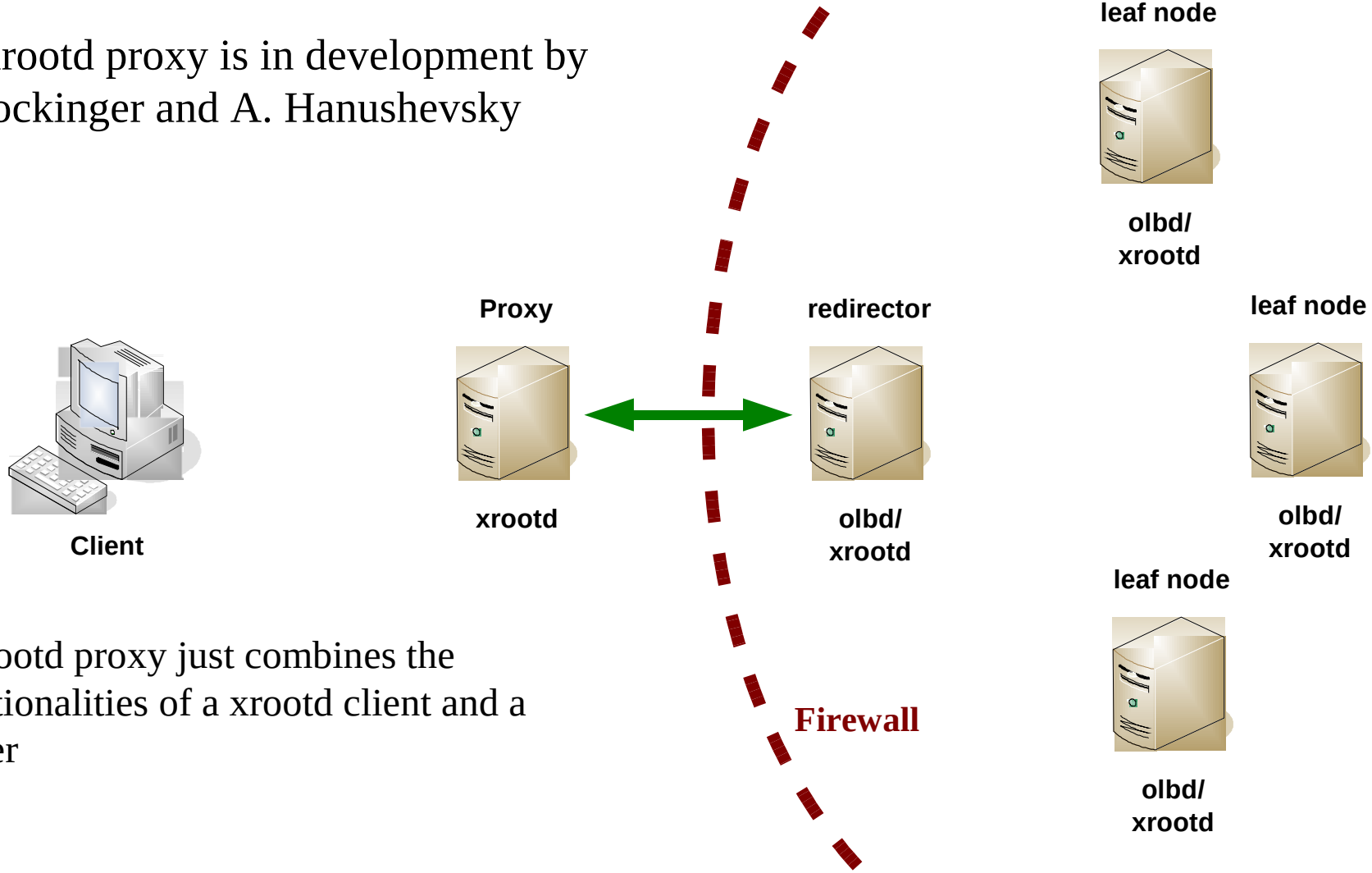
Modular Architecture



Graphic taken from a talk by A. Hanushevsky

WAN operation: xrootd proxy

The xrootd proxy is in development by H. Stockinger and A. Hanushevsky



A xrootd proxy just combines the functionalities of a xrootd client and a server

Configuration via self organization

- Self organization of a huge xrootd system into cells
- Three hierarchy system:
  - Manager
  - Cell supervisor
  - Leaf node (cells of 64 leaf nodes)
- Test by A. Hanushevsky: 900 nodes reached a stable configuration after 56 seconds.

Possible Integration with SRM

- Trivial approach
  - Treat SRM system like any other MSS backend (i.e. use interfacing scripts to stage in files). Interfacing at leaf node level.
  - Needs special solutions for transfering additional information (e.g. For pinning. Could be implemented via opaque info).
  - Needs solution for transferring authentication/authorization
  - Retains performance of xrootd but sacrifices some functionality of SRM.
- Interface olbd layer with SRM
  - Full functionality of SRM possible (e.g. reservation of space).
  - Scalability problem. Redirector has to carry out more work. Loses advantages of an xrootd load balanced system.

Use with another disk pool manager

- If very efficient file read access is required, it might make sense to add a read only xrootd pool to an existing disk pool manager like DPM.

  Deploy on separate pool (use xrootd MSS hooks to stage in files from the other pool)

  Deploy on top of other pool. Again, files should only be staged in via interfacing to the other pool. xrootd is resilient enough to deal with the externally managed pool. Care must be taken that the load balancing algorithm of one pool does not interfere with the other, if both ways should be open to the user.

Monitoring

- Xrootd provides monitoring on the leaf node level based on UDP packets (user, open files, bulk informtation on transfers). Jacek Becla of SLAC working on this
- ARDA/ALICE is implementing client level monitoring via MonALISA. This provides tracing on the job level. (Catalin Cirstoiu working on this).

Further application possibility

- Use xrootd/olbd on worker nodes

  Use olbd to find out best placement of jobs on nodes:

  - Use a olbd bulk location request (resolves by default in 5 sec) to find location of files on worker nodes
  - Use location information to split job (e.g. For a PROOF type application) across the nodes. Nodes can still reach the files on the other nodes, but network traffic is minimized.

Conclusion

- xrootd provides a full fledged file transfer system.
- Highly optimized for making economic use of resources.
- Read only load balanced systems (+MSS) are ideal for setups where clients need to open and read many files in non sequential, but rather sparse fashion.
- Read/write load balanced systems (+MSS) are more difficult to manage and require giving up some of the elegance, versatility and performance of the simple read only systems.
- Modular architecture guarantees easy extensibility by third parties.

Acknowledgements

- Xrootd team for discussions and bug fixes
  - A. Hanushevsky, F. Furano, P. Elmer, J. Becla
- ARDA
  - Implementation of authorization prototype: A. Peters
  - Monitoring integration with MonALISA: C. Cirstoiu
- CERN IT
  - J.-P. Baud and B. Panzer for discussions