Scalla Features

xrootd /olbd

Andrew Hanushevsky
Stanford Linear Accelerator Center
Stanford University
7-March-07

http://xrootd.slac.stanford.edu
Outline

- Introduction
- Data serving
  - xrootd
- Clustering
  - olbd
- Conclusion
What is **Scalla**?

- **Structured Cluster Architecture for Low Latency Access**
  - Low Latency Access to data via **xrootd** servers
    - POSIX-style byte-level random access
      - By default, arbitrary data organized as files
      - Hierarchical directory-like name space
    - Protocol includes high performance features
  - Structured Clustering provided by **olbd** servers
    - Exponentially scalable and self organizing
Scalla Design Points

- High speed access to experimental data
  - Write once read many times processing mode
  - Small block sparse random access (e.g., root files)
  - High transaction rate with rapid request dispersement (fast opens)

- Wide usability
  - Generic Mass Storage System Interface (HPSS, RALMSS, Castor, etc)
  - Full POSIX access
  - Server clustering for scalability

- Low setup cost
  - High efficiency data server (low CPU/byte overhead, small memory footprint)
  - Very simple configuration requirements
  - No 3rd party software needed (avoids messy dependencies)

- Low administration cost
  - Non-Assisted fault-tolerance
  - Self-organizing servers remove need for configuration changes
  - No database requirements (no backup/recovery issues)
xrootd Plugin Architecture

- Protocol Driver (XRD)
- Protocol (1 of n) (xrootd)
- File System (ofs, sfs, alice, etc)
- Storage System (oss, drm/srm, etc)
- Clustering (olbd)
- authentication (gsi, krb5, etc)
- lfn2PFN prefix encoding
- prefix encoding
- authorization (name based)
What is Actually Provided

Protocol (1 of n) (xrootd)

Protocol Driver (XRD)
xrootd

libXrdSec.so +
libXrdSecgsi.so
libXrdSeckrb4.so
libXrdSeckrb5.so
libXrdSecpwd.so
+ libXrdCrypto.so
libXrdSut.so
xrdgsiproxy
xrdpwdadmin

Clustering
(olbd)

includes simple entity authorization model,
partition aggregation
and generic MSS interface plus drivers:
mps_prep, mps_MigrPurg, mps_PreStage,
mps_Stage, mps_Xeq

File System
(ofs, sfs, alice, etc)

Storage System
(oss, drm/srm, etc)

No 3rd Party Software Required
(name based)

libXrdOfs.so

olbd

7-March-07 6: http://xrootd.slac.stanford.edu
Basic xrootd Startup

- Can be run without clustering
  - Simple or no configuration needed
    - `xrootd -l logfn path [ path [ . . . ] ]`
  - Startup/Shutdown scripts provided
    - StartXRD and StopXRD plus StartXRD.cf
  - Root privileges really not needed
    - xrootd refuses to run as user root
  - It’s really that simple
Simple Authorization

- Standard feature in libXrdOfs.so
  - Based on authenticated user and hostname
    - Name and group membership
    - Can use hostname if authentication not configured
  - Capability list oriented
    - Allows templates and fungible authorization
    - Privileges setup like Windows™ XP
Partition Aggregation

- Standard feature in libXrdOfs.so
  - N partitions can be aggregated
    - Uniform name space
    - Partitions are space load balanced
  - Reduces admin overhead
    - No special volume manager privileges needed
    - Reduces the granularity of failure
Parallel Stream Support

- Client can transfer data via || streams
  - Feature used by xrdcp for WAN transfers
- Can improve xfr rate on high latency links
  - Geneva -> Prague (RTT 17 ms)
    - 15: 22.2 MB/s 541%
    - 10: 18.5 MB/s 451%
    - 05: 12.7 MB/s 308%
    - 02: 6.7 MB/s 163%
    - 00: 4.1 MB/s 100%

7-March-07 10: http://xrootd.slac.stanford.edu
Proxy Support

- Fast way to deal with firewalls
  - Integrated with xrootd
  - Can setup load balanced proxy clusters
  - Read/Only Access

- SOCKS4 protocol supported
  - For Read/Write access
  - Can use ssh tunnels
Full Posix Access

- **libXrdPosixPreload.so** (dynamic)
  - Vectors C-library calls to xrootd when needed
    - `setenv LD_LIBRARY_PATH /path/libXrdPosix.so`
    - `setenv LD_PRELOAD /path/libXrdPosixPreload.so`
    - `setenv XROOTD_VMP server:port:/lpath`
  - Can use favorite Unix commands (e.g., vi)
    - `cp /lpath/file /tmp/file or`
    - `cp xroot://server:port//lpath/file /tmp/file`

- **libXrdPosix.so** (non-dynamic)
  - Safer way to use POSIX interface
Grid Access via FTP

- gsiftp (a.k.a. GridFTP)
  - Done using POSIX preload library
    - libXrdPosixPreload.so
  - Need to run a gsiftp node
MSS Interface

- Standard feature in libXrdOfs.so
  - Generic call-outs to MSS
    - Stage files to disk
    - Modify meta-data in MSS
    - Delete files from disk and MSS
  - Accommodates almost any MSS
    - Simple model effective for smaller sites
    - Greatly reduced administrative overhead
MSS/DRM/SRM Integration

- **CERN Castor2 and LCG DPM (GLite)**
  - Allows xrootd to share MSS disk pools
  - Currently being tested by Alice

- **Berkeley DRM**
  - Allows SRM access to single disk server
  - Being tested and enhanced by BNL

- **INFN to do StoRM integration**
  - Full SRM access to an xroot cluster
Clustering

- xrootd servers can be clustered
  - Increase access points and available data
  - Allows for automatic failover
- Structured point-to-point connections
  - Cluster overhead (human & non-human) scales linearly
  - Cluster size is not limited
  - I/O performance is not affected
- Always pairs xrootd & olbd servers
  - Symmetric cookie-cutter arrangement
olbd Roles

Manager Role
- Keeps track of path to the file
  - In-memory real-time database
  - Fully recoverable without any intervention
- Guides clients to the actual file
- Decides what server is to be used for a request

Server Role
- Keeps track of xrootd utilization and health
- Reports statistics to the manager
Basic cluster Startup

- One* manager node and up to 64 data servers
  - Configuration file needed
    - Is the same one for each node
  - Must start xrootd and olbd on each node
    - xrootd –l logfn –c configfn
    - olbd –l logfn –c configfn
  - Startup/Shutdown scripts provided
    - StartXRD and StopXRD plus StartXRD.cf
    - StartOLB and StopOLB plus StartOLB.cf
  - Root privileges really not needed
    - xrootd and olbd refuse to run as user root
  - While simple it requires some thought

*Can have any number.
The SLAC \(\frac{1}{4}\)PB “kan” Cluster

Clients

Managers

Data Servers

(> 250 TB)

kan001  kan002  kan003  kan004  bbr-rdr03  bbr-rdr04

bbr-rdr-a

kan059

kan001  kan002  kan003  kan004  (> 250 TB)
all.manager bbr-rdr-a+ 3121
olb.port 3121
olb.allow host kan*.slac.stanford.edu
olb.allow host bbr-rdr03.slac.stanford.edu
olb.allow host bbr-rdr04.slac.stanford.edu
olb.path rs /store

all.role manager if bbr-rdr-a+
all.role server if kan*.slac.stanford.edu

oss.cache public /kanga/cache*
oss.path /store readonly nocheck stage nodread
oss.mssgwcmd /usr/etc/ooss/pudc
oss.stagecmd |/usr/etc/ooss/mps_prep

oss.localroot /kanga
oss.remoteroot /kanga

xrootd.fslib /opt/xrootd/lib/libXrdOfs.so
xrootd.export /store

lfn to pfns mapping
/store/…
/kanga/store/…
Conclusion

- High performance data access systems achievable
  - The devil is in the details
- High performance and clustering are synergetic
  - Allows unique performance, usability, scalability, and recoverability characteristics
- Challenge is to do it with low human overhead
  - In Scala simplicity trumps features
    - Need fewer people and with lower skill sets (e.g., students)
Acknowledgements

Software Collaborators
- INFN/Padova: Fabrizio Furano (client-side), Alvise Dorigo
- Root: Fons Rademakers, Gerri Ganis (security), Beterand Bellenet (windows)
- Alice: Derek Feichtinger, Guenter Kickinger
- STAR/BNL: Pavel Jackl
- Cornell: Gregory Sharp
- SLAC: Jacek Becla, Tofigh Azemoon, Wilko Kroeger, Bill Weeks
- BaBar: Pete Elmer (packaging)

Operational collaborators
- BNL, CNAF, FZK, INFN, IN2P3, RAL, SLAC

Funding
- US Department of Energy
  - Contract DE-AC02-76SF00515 with Stanford University