Table of Contents

AtlasXrootdSystems

Introduction

This page describes a project within ATLAS to create an XRootd-based infrastructure to improve access to ATLAS data.

The goals of this project are multiple: on one hand, to create better federation between Tier2 sites, while at the same time serving the needs of Tier3s.

This is work-in-progress, so all details on this page are subject to change. In particular, the installation process will be streamlined.

Global Namespace, LFC

One aspect of this work is to create an "ATLAS global namespace", that is, a uniform way of referring to files. We have a de-facto global namespace, which is file GUIDs, but this is not user-friendly.

The proposal is to have the global namespace based on LFC paths, with some small modifications (e.g. paths will start with `/atlas/` instead of `/grid/atlas` and some other cleanups are applied.).

For Tier3 sites, the "global namespace" name will be the same as the storage location (also referred to as PDP, "physical dataset path"). Tier1 and Tier2 sites are storing the files with possibly differing path conventions, so the LFC must be consulted to convert the global name into a physical name. This is handled by an Xroot plugin called "xrd-lfc" (short name) or "XrdOucName2NameLFC.so" (actual file name).

LFC lookup results are cached, for performance reasons and to reduce load on the LFC. This cache is controllable via the parameters `lfc_cache_ttl` (default: 2 hours) and `lfc_cache_maxsize` (default: 500000 entries).

dCache bypass

This plugin can also function as a "dcache bypass", that is, dCache sites can expose files sitting in dcache pools via the xrootd protocol (read-only). This of course is already possible in dcache, but the xrd-lfc implementation is different - rather than having the dCache software emulate xrootd, there is actually a (native) xrootd server running alongside dCache. This service can either read files from the dcache backend using the dcap libraries, or, in an enhanced-performance mode, can read data files directly from dcache pools. (This mode of deployment assumes that there would be one xrootd server on each dcache pool hosts).

Federation with other storage backends

Similarly, other non-xroot storage (Posix, Lustre, GPFS, Hadoop, etc) can be exposed in the same way via the xroot protocol, using different back-end I/O libraries.

Installation

Install xrootd packages

If xrootd/cmsd are not already installed, the easiest way to get them is from the OSG yum repository:

```
$ cat /etc/yum.repos.d/osg-xrootd.repo
[caltech]
name=OSG Xrootd Packages for Enterprise Linux 5 - $basearch
baseurl=http://newman.ultralight.org/repos/xrootd/$basearch/
enabled=1
gpgcheck=0
$  yum install xrootd
```

Correct group membership for xrootd user

Note that the installer will create an `xrootd` user if it does not already exist. This user will need to have group permissions to read files from site storage, e.g. the `usatlas` group for USA sites. You can either adjust the group membership for the `xrootd` user after the installer runs, or else create the `xrootd` user first and place them in the correct group, e.g.

```
xrootd:x:102:21005:ATLAS XRD:/etc/xrootd:/bin/bash
```

where 21005 is the numeric GID for the USATLAS group.

Modify startup scripts

The xrd-lfc plugin will be loaded dynamically by the `xrootd` and `cmsd` daemons. They will need access to a grid environment containing `liblfc.so`, for example the worker-node client package. They also need the `LFC_HOST` environment variable set. The easiest way to make sure the environment is set up correctly is to modify the startup scripts:

Modify the scripts `/usr/bin/xrootd.sh` and `/usr/bin/cmsd.sh` as follows:

```
$ cat /usr/bin/xrootd.sh
#!/bin/sh
. /share/wn-client/setup.sh
export LFC_HOST=YOUR_LFC_HOST
/usr/bin/xrootd ${1+"$@"}

$ cat /usr/bin/cmsd.sh

#!/bin/sh
. /share/wn-client/setup.sh
export LFC_HOST=YOUR_LFC_HOST
/usr/bin/cmsd ${1+"$@"}
```

(Here, `/share/wn-client/setup.sh` path is an example, it needs to be set appropriately for your site).

Provide a grid proxy

Since xrd-lfc is an LFC client, it needs an x509 proxy with appropriate VOMS attributes to access the LFC. Currently this is done manually via `voms-proxy-init` and the resulting `x509up_uxxxx` file installed with 'xxxx' replaced by the UID of the `xrootd` user. Using the flags

```
-vomslife=96:0 -hours 96
```

is helpful so that the proxy does not need to be renewed daily. For production, this will need to be automated or replaced with a service proxy, or some other means of authenticating with LFC, see e.g. https://twiki.cern.ch/twiki/bin/view/LCG/LfcAdminGuide#Trusted_Hosts (However, the `shift.conf` only works for clients running as root, and xrd-lfc is running as

Install the xrd-lfc plugin

Source code is currently available at http://repo.mwt2.org/viewvc/xrd-lfc/ and a prebuilt (64-bit) binary is available at http://www.mwt2.org/~cgw/xrd-lfc/

Download (or compile) XrdOucName2NameLFC.so and place it in `/etc/xrootd`

Configure xrootd and xrd-lfc

`/etc/xrootd/Authfile` should contain the single line:

```
u * /atlas lr
```

The main xrootd config file is `/etc/xrootd/xrootd.cfg`. This also contains the configuration for the xrd-lfc plugin.

Here's a minimal example of a config file:

```
## Port specifications; only the redirector needs to use a well-known port
## "any" will cause rooted to bind to any available port.  Change as needed for firewalls.
xrd.port 1094

## The roles this server will play.
all.role server

## The known managers
all.manager atl-grdr.slac.stanford.edu:1213

## Since global LFN starts with /atlas
all.export /atlas r/o

oss.namelib /opt/xrootd/XrdOucName2NameLFC.so root=/pnfs match=uchicago.edu

## For reading dCache files in non-bypass mode:
#ofs.osslib /usr/lib64/libXrdDcap.so

all.adminpath /var/run/xrootd
all.pidpath /var/run/xrootd

## Logging verbosity
xrootd.trace emsg login stall redirect
ofs.trace none
xrd.trace conn
cms.trace all
```

Some comments: = atl-grdr.slac.stanford.edu= is the "Atlas Global Redirector" hosted at SLAC.

The configuration for xrd-lfc is contained on the `oss.namelib` line.

Configuration parameters for xrd-lfc:

- ~~lfc_host~~: Do not use, set via LFC_HOST env. var.
- lfc_cache_ttl: (Optional)cache time to live, in seconds (default: 7200, 2 hours)
- lfc_cache_maxsize: (Optional) maximum number of entries in LFC cache (default: 500000)
- root start of physical filesystem path in an SFN, e.g. /pnfs/domain.edu This **must** be set if LFC lookups do not return a "bare" filesystem path (prefixes like srm://server:port/manager must be removed and this parameter is used to identify the beginning of the non-prefix component)
- match (Optional) string or comma-separated list of strings. If set, LFC replies will only be considered if they contain at least one of the match strings. This is used to handle shared LFC between sites, or to restrict to particular space tokens.
- nomatch (Optional) As above, but LFC replies will be rejected if any of the strings matches as a substring. This is used, e.g. to avoid accessing tape files.
- dcache_pool[s] (dCache bypass-mode only). Path or paths to dCache physical file pools. If this is set, dCache bypass mode will be used, see section "dCache bypass mode" below.
- force_direct (dCache bypass-mode only). See "dCache bypass mode" below

Xrd/Posix/Lustre backend

If the storage backend is xrootd, or Posix-compatible (Lustre, GPFS, etc), then there should be no ofs.osslib line in the config file, since files can be read directly by the xrootd daemon using standard Posix calls.

dCache backend without bypass mode

To read files from a dCache system, then ofs.osslib must be set to a path to libXrdDcap.so, e.g.

```
ofs.osslib /usr/lib64/libXrdDcap.so
```

libXrdDcap is available from the OSG yum repository (package xrd-dcap) but as of this writing that version has several bugs which make it non-usable. Instead, get a patched version from here: http://www.mwt2.org/~cgw/xrd-lfc/

dCache bypass mode

In addition to reading dCache files using dcap protocol, a configuration is supported where xrootd runs alongside the dCache pool software, and reads directly from the dCache pools. (This requires a pathname->pnfsid lookup, which is cached alongside the LFC lookup result).

To enable this feature, the dcache_pool or dcache_pools directive is used. This pattern is glob-expanded, so for instance a pattern like /dcache/pool*/data is handled correctly. It may be a comma-separated list of such paths.

If a file is found directly in the pool, it will be read from there. If ofs.osslib is set to libXrdDcap.so and the file is not found on the pool, it can still be read using dcap protocol.

However if force_direct is specified, then files will **only** be served from the paths specified in dcache_pool[s]. The idea is that xrootd is running on all dcache pools, and only the pool which hosts the file will respond to a location query, so files will always be served directly from the pools, bypassing dcap mode entirely. This offers higher performance and incurs less traffic and on the local network.

Here's an example of such a configuration:

```
oss.namelib /etc/xrootd/XrdOucName2NameLFC.so root=/pnfs match=uchicago.edu lfc_host=uct2-grid5.u
```

If there are more than a few dCache pools, rather than having them pointing to the ATLAS global redirector, a local redirector should be used. In this case the `all.manager` directive should read

```
all.manager LOCAL_REDIRECTOR_HOST:1213
```

This machine will not be involved in data transfers, only control messages, so it will not not require significant resources.

Here's a sample `xrootd.cfg` for a local redirector (in this example, `uct2-grid5.uchicago.edu`)

```
# grdr is the global redirector, xrdr is the local redirector
#
set grdr = atl-grdr.slac.stanford.edu
set xrdr = uct2-grid5.uchicago.edu
set exportpath = /atlas
all.adminpath /var/run/xrootd
all.pidpath   /var/run/xrootd
all.manager $(xrdr):1213
xrootd.async off
xrd.port 1094
all.export $(exportpath) r/o
```

Note the the local redirector does not need an `oss.namelib` directive.

Starting the services

Both the `xrootd` and `cmsd` daemons must be running on all participating hosts. They are controlled via standard init scripts, e.g.

```
/etc/init.d/xrootd start
/etc/init.d/cmsd start
```

You may use `chkconfig` to arrange to start these automatically at boot.

```
# /sbin/chkconfig --levels=345 xrootd on
# /sbin/chkconfig --levels=345 cmsd on
```

Log files go to `/var/log/xrootd` and are automatically rotated.

Testing

First, try to access a file from the local xroot service. You will need the global namespace path to a file which is present at your site. This is basically the LFC path, with the leading `/grid/atlas/dq2/` replaced with `/atlas/`

For example, using MWT2 (with LFC host uct2-grid5.uchicago.edu), the LFC path to `DBRelease-12.9.3.tar.gz` is

```
$   LFC_HOST=uct2-grid5.uchicago.edu lfc-ls -l /grid/atlas/dq2/ddo/DBRelease/v120903/ddo.000001.A

-rwxrwxr-x   1 102      102              462113519 Nov 01 10:20 /grid/atlas/dq2/ddo/DBRelease/v1
```

so the global namespace path is `/atlas/dq2/ddo/DBRelease/v120903/ddo.000001.Atlas.Ideal.DBRelease.v120903/DBRel`

Copy the file from the local redirector:

```
$ rm /tmp/DBRelease-12.9.3.tar.gz ; xrdcp  xroot://uct2-grid5.uchicago.edu//atlas/ddo/DBRelease/v
[xrootd] Total 440.71 MB        |====================| 100.00 % [93.5 MB/s]
```

If that succeeds, try again using the global redirector (atl-grdr.slac.stanford.edu)

```
$ rm /tmp/DBRelease-12.9.3.tar.gz ; xrdcp  xroot://atl-grdr.slac.stanford.edu//atlas/ddo/DBReleas
[xrootd] Total 440.71 MB        |====================| 100.00 % [106.1 MB/s]
```

Finally, try "extreme copy" mode (-x) which uses multiple sources in parallel:

```
$ rm /tmp/DBRelease-12.9.3.tar.gz ; xrdcp  -x xroot://atl-grdr.slac.stanford.edu//atlas/ddo/DBRel
Extreme Copy enabled.
Source #1 root://192.41.230.187:1094//atlas/ddo/DBRelease/v120903/ddo.000001.Atlas.Ideal.DBReleas
Source #2 root://129.107.255.18:1094//atlas/ddo/DBRelease/v120903/ddo.000001.Atlas.Ideal.DBReleas
Source #3 root://131.225.233.52:1094//atlas/ddo/DBRelease/v120903/ddo.000001.Atlas.Ideal.DBReleas
Source #4 root://128.135.158.185:1094//atlas/ddo/DBRelease/v120903/ddo.000001.Atlas.Ideal.DBRelea
Source #5 root://128.135.158.188:1094//atlas/ddo/DBRelease/v120903/ddo.000001.Atlas.Ideal.DBRelea
Source #6 root://134.79.201.11:1094//atlas/ddo/DBRelease/v120903/ddo.000001.Atlas.Ideal.DBRelease
[xrootd] Total 440.71 MB        |====================| 103.63 % [70.8 MB/s]
```

This shows the file being copied from AGLT2, SWT2, FNAL, MWT2 (2 different servers), and SLAC.

Troubleshooting

If any of these tests fail, re-run with '-d 3' for extra debug output. Also, examine the logs for xrootd, cmsd, and the LFC server log as well.

The LFC server log should show the xrd-lfc connecting, e.g.:

```
12/14 16:52:27  3622,0 Cns_srv_startsess: NS098 - startsess (XRD-LFC@iut2-s1.iu.edu)
```

Contact

For any issues, email: cgw@hepSPAMNOT.uchicago.edu

References

- Presentations from Atlas Software Week Dec 2010:
    - "XRootd Demonstrator description, schedule and results": http://indico.cern.ch/contributionDisplay.py?contribId=46&confId=76896 (Doug Benjamin)
    - "Discussion of Federated ATLAS XRootd": http://indico.cern.ch/contributionDisplay.py?contribId=93&confId=76896 (Charles Waldman)
    - "XRootd Name to Name plugin status and testing": http://indico.cern.ch/contributionDisplay.py?contribId=47&confId=76896 (Charles Waldman)
    - "DQ2 clients news" (discusses Global Physical Dataset Path): http://indico.cern.ch/contributionDisplay.py?contribId=88&confId=76896 (Angelos Molfetas)

Major updates:
-- CharlesWaldman - 17-Dec-2010

Responsible: CharlesWaldman
Last reviewed by: **Never reviewed**

This topic: Atlas > AtlasXrootdSystems
Topic revision: r4 - 11-Jan-2011 - 21:20:10 - CharlesWaldman