Journal of Physics: Conference Series **219** (2010) 052023 doi:10.1088/1742-6596/219/5/052023

CASTORFS — A Filesystem To Access CASTOR

Alexander MAZUROV[1], **Niko NEUFELD**[1]

[1] LHCb Online Team, CERN, Geneve 23, Switzerland

E-mail: alexander.mazurov@cern.ch

Abstract. CASTOR provides a powerful and rich interface for managing files and pools of files backed by tape-storage. The API is modelled very closely on that of a POSIX filesystem, where part of the actual I/O part is handled by the rfio library. While the API is very close to POSIX it is still separated, which unfortunately makes it impossible to use standard tools and scripts straight away. This is particularly inconvenient when applications are written in languages other than C/C++ such as is frequently the case in web-apps. Here up to now the only the recourse was to use command-line utilities and parse their output, which is clearly a kludge. We have implemented a complete POSIX filesystem to access CASTOR using FUSE (Filesystem in Userspace) and have successfully tested and used this on SLC4 and SLC5 (both in 32 and 64 bit). We call it CastorFS. In this paper we will present its architecture and implementation, with emphasis on performance and caching aspects.

1. Introduction

In this article we describe the architecture, implementation and deployment of a filesystem to access CASTOR named **CASTORFS** [1].

2. CASTOR

CASTOR [2]

- stands for the CERN Advanced STORage manager;
- is a hierarchical storage management (HSM) system developed at CERN used to store physics production files and user files;
- manages disk cache(s) and the data on tertiary storage or tapes.

Files can be stored, listed, retrieved and accessed in CASTOR using command line tools or applications built on top of various data transfer protocols.

Example of using command line tools:

- *rfdir /castor/cern.ch*
  - List directory contents
- *rfcp /castor/path/to/file /tmp*
  - Copy files from CASTOR to local filesystem

3. CASTOR filesystem

CASTOR logically presents files in a UNIX (POSIX) like directory hierarchy of file names. This suggests to implement a new filesystem capable of operating on files stored on CASTOR using standard Unix operation system calls and commands like `open, read, cp, rm, mkdir, ls, cat` and `find`.

We have implemented a complete POSIX filesystem to access CASTOR using FUSE (Filesystem in Userspace) [3] and have successfully tested and used this on SLC4 and SLC5 (both in 32 and 64 bit). We call it **CASTORFS**.

4. Implementation

The FUSE library is the main component of CASTORFS. Using it with the CASTOR RFIO (Remote File I/O) and CASTOR NS (Name Server) libraries allows implementing a fully functional filesystem in a userspace program.

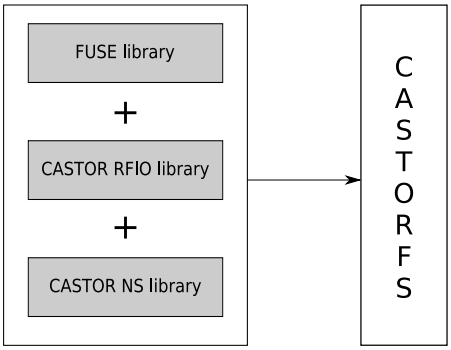

Figure 1. Implementation.

The CASTOR RFIO library provides the main input/output operations for files in CASTOR: `rfio_readdir, rfio_open, rfio_read, rfio_mkdir, rfio_stat,` etc — each operation has a corresponding POSIX operation: readdir, open, read, mkdir, stat, etc...

The CASTOR NS library provides information about files and directories in CASTOR. The library not only allows to retrieve standard meta-data like creation and modification time, owner and size of a file/directory, but also to get some additional attributes, such as the hecksum of a file and the status of a file with respect to migration from disk cache to tape.

5. Usage

An example of using CASTORFS, which mounts a new filesystem to a `/castorfs` local directory:

```
$> /opt/castorfs/bin/castorfs /castorfs  -o allow_other, castor_uid=10446,
   castor_gid=1470, castor_user=lbtbsupp, castor_stage_host=castorlhcb,
   castor_stage_svcclass=lhcbraw
```

From the command line one can configure:

- the connection parameters for CASTOR;
- information about the CASTOR user;
- the mode of the connection: read only or full access.

After mounting a CASTORFS filesystem one can run any of your favorite tools intended to work with files or directories. For example:

```
$> ls la /castorfs/path/to/file
$> mkdir p /castorfs/path/to/dir
$> cat /castorfs/path/to/file
$> rm /castorfs/path/to/file
$> find /castorfs | grep strangefilename.raw
```

6. Extended attributes

The CASTOR name server stores some interesting file meta-data, for example: a file checksum, a status of the migration to tapes (migrated or not).

In CASTORFS we implemented a mapping of additional file information to filesystem extended attributes. These can be accessed in the following way:

```
$> getfattr -d /castorfs/cern.ch/lhcb/online/data.tar

# file: castorfs/cern.ch/lhcb/online/np.tar
user.checksum="569145875"
user.checksum_name="adler32"
user.nbseg="1"
user.status="migrated"
```

7. Packaging

CASTORFS is available as a RPM package for Scientific Linux CERN (SLC4 and SLC5, both 32 and 64-bit architectures) and such will run unchanged on CentOS and RHEL distributions.

In the LHCb Online cluster (CERN) the CASTORFS RPM has been distributed to the compute farm using Quattor/LinuxFC [4] (a system administration toolkit for automated installation, configuration and management of Linux-clusters).

If you have the same user (UID, GID) on the client host as she is registered with CASTOR, you can use the FUSE and CASTOR packages provided by their vendors. If you want to map your local user to a different UID/GID on CASTOR, you need to apply our patches to the FUSE and CASTOR libraries. These patches will be obsolete once CASTOR can use certificates.

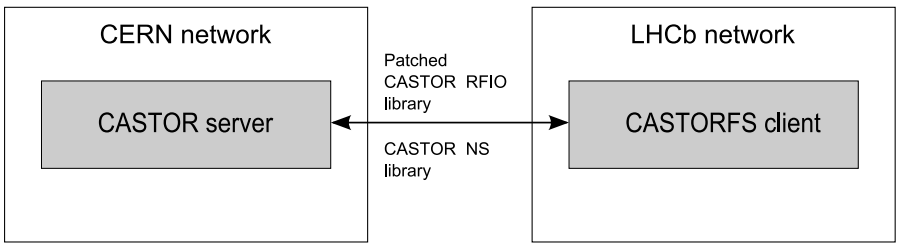

Figure 2. Using the CASTORFS client at the LHCb experiment at CERN.

8. Perfomance

Due to some limitation in the Linux kernels prior 2.6.27, we have a performance problem for writing and reading files to/from CASTOR compared to native RFIO about 14 times slower for writing and 3 times slower for reading[1](see Table **??**).

When testing CASTORFS with the latest kernels we observe significantly better performance.

[1] This is for the first read, subsequent reads are much faster, as they go through the buffer-cache

Table 1. Read and write perfomance.

Tool	Read (Mb/s)	Write (Mb/s)
POSIX *cp* command on CastorFS	31	5
rfcp command (based on CASTOR RFIO library)	100	70

9. Other implementations — XCFS

The Data Management group of the CERN IT department has implemented a CASTOR file system based on FUSE and the xROOT IO protocol — XCFS [5]. XCFS shows a better perfomance then CASTORFS, because it does not rely on rfio, but at the time of this writing it is still in a stage and requires some extra work to adopt it for using in a network different from the CERN network, such as the LHCb Online network.

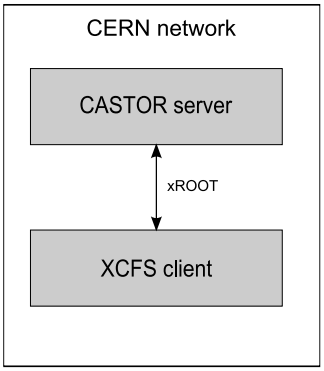

Figure 3. XCFS tested only at CERN network. It requires some extra work to adopt it for using in a network other than the CERN network (see Figure **??**).

10. Future plans

An improvement of read- and write performance is the main item in the future plan.

- Implementation of a caching mechanism for storing information about frequently requested CASTOR file meta-data.
- Creating a CASTORFS package for the newest Linux kernel.

References

[1] CASTORFS web page
— https://lbtwiki.cern.ch/bin/view/Online/CastorFS
[2] CASTOR web site
— http://cern.ch/castor
[3] FUSE web site
— http://fuse.sourceforge.net
[4] Quattor web site
— http://apps.sourceforge.net/mediawiki/quattor/index.php?title=Main_Page
[5] XCFS Presentation at ACAT 2008
— http://indico.cern.ch/materialDisplay.py?contribId=171&sessionId=29&materialId=slides&confId=34666