1 LTFS Archiver Overview

1.1 Summary

This software component tool is an advanced prototype developed by RAI, integrated within P4 during the PrestoPRIME evaluation test-beds of November 2011. This tool has been later improved during the fourth year of the project and showed with the new features during the evaluation test-beds 2012.
The new release 1.0 includes the source code as Linux shell scripting, the present document as an overall description and the technical documentation necessary for the installation and use.
LTFS Archiver has been successfully deployed and used during the 2012 test-bed, it has been integrated under the P4 platform, in order to support a second safety purpose copy of the multimedia files on LTO5 tapes, by using LTFS file system.

1.2 LTFS Archiver

The description of the software is organised as follow.
Chapter 1.2.1 briefly explains the new functionalities introduced in the last version 1.0 with respect of the preceding one version 0.9
Chapters 1.2.2 and 1.2.3 respectively give a short high level description of the system architecture and of the logic used by the software.

1.2.1 What's new in this version

From the preceding version 0.9 to the current one 1.0, a set of important changes has been made as listed in the following:

- Introduction of Tape Pool concept
- Introduction of MakeAvailable service that allows to have read access on an LTO tape directly through a share avoiding local copies
- The new modality “Tape on Shelf” that allows to manage non automated LTO drives (e.g. desktop drives)
- Introduction of a web interface for managing manual load/unload operations and basic monitoring
- Possibility to ask for an entire directory archiving
- Possibility to ask for a md5 checksum value for each file archived
- Better management of the operation queue, in particular more writings targeted on the same tape are made together saving LTO access time

1.2.2 Description

LTFS Archiver is a software module capable to deal with LTO5 libraries (HP model supported so far) and simple desktop LTO5 drives (HP model tested so far) also in a mixed configuration. The goal of the software is to effectively manage the storage of generic files, even if it is optimised for working with multimedia (big, order of several GBytes) files.

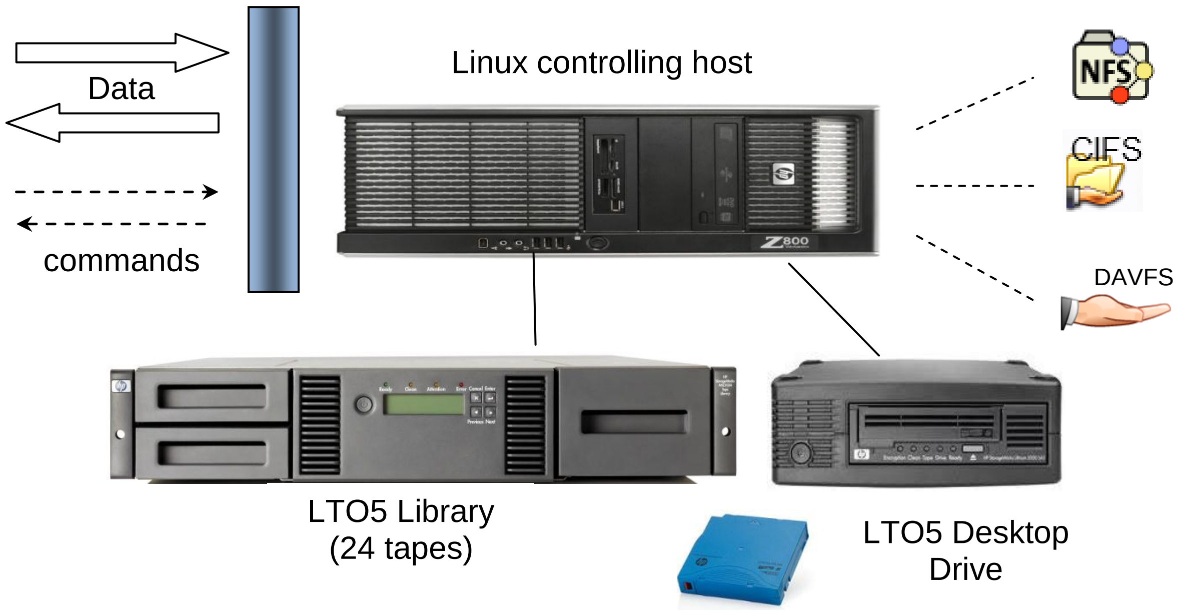

Figure 1 - System configuration example

The software has to be installed on a Linux based host that is connected to the LTO library and/or drive (see Figure 1), indeed the software requires mtx[1] and mt[2] system tool commands to pilot the library and the drives. Figure 2 is a picture taken during the 2012 test-bed held in Paris where LTFSArchiver has been configured with both an HP LTO5 Library and an HP LTO5 desktop drive, all integrated with the P4 platform.
Incoming and outgoing data areas (Figure 1 on the right) have to be previously mounted on the local file system, tests have been successfully carried out with NTFS, CIFS and DAVFS. In Figure 1 the vertical bar in the upper left part represent the software that manages the requests (thin arrows) issued by the users and the data flows (thick arrows) of the files that go from and to the LTO tapes.

---

[1] Media Changer Devices, http://source- forge.net/projects/mtx/
[2] Control Magnetic Tape Drive Operation

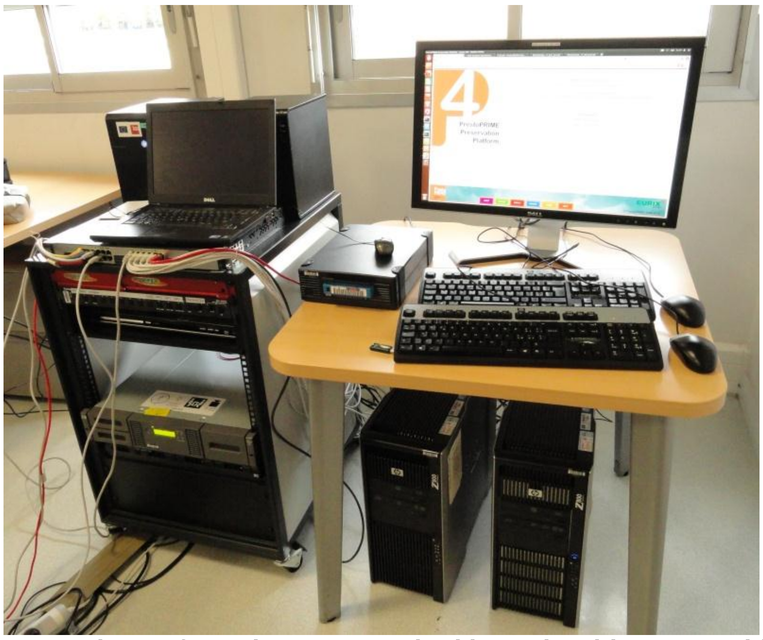
Figure 2 - Picture from the 2012 test-bed in Paris with LTFSArchiver configured with an LTO5 library and an LTO5 desktop drive

In this system the file localisation is dealt with special strings called *flocat* (file locations) where all the information necessary to access the file is contained, a flocat has the following form:

[<tapeid>:]/<path>/<filename>

Where tapeid is is used only when the file is located on a tape and has a common prefix *lto-ltfs* followed by a identification string like *000001L5* corresponding to the barcode of the data tape itself.
An example of a flocat for a file stored on a tape is:
lto-ltfs:000001L5:/02b34872-3512-4b61-8f6f-e897ea99150b/PPRIME_Example1.mxf
while an example for a file on a share is:
/mnt/d104a/p4test/PPRIME_Example2.mxf
In case of tape, files are always stored under a folder (e.g. *02b34872-3512-4b61-8f6f-e897ea99150b)* whose name corresponds to the UUID of the submit operation (request, see later).

1.2.3 Implementation

The software is a stable prototype written mostly with shell scripting, it proved to work rather reliably during the 2011 and 2012 project test-beds.
It supports multi-drive LTO library with parallel read/write in case that more than a drive is available, also a multi-library configuration is managed even if not well tested till now.
The core of the software is a daemon process that works using a booking/request database and gives commands to the library/ies and to the drives accordingly.
The database is minimal and includes two principal tables, one for the requests of get/store of the files and one for keeping track of the LTO tapes available in the library and their remaining free space.
One of the important concept used is the Tape Pool that is a group of tapes identified by a “PoolName”. The user can control which files will be recorded on tapes

belonging to a specified pool, permitting a consistent management of tapes and files also in complex contexts such an Archive serving several collections, departments or customers. If the pool is not specified, a default pool is used spanning all the tapes that has been loaded without specifying a pool.

The logic followed by the serving daemon is quite easy as it basically scans the database looking for requests to be satisfied (Figure 3). In case of a write request it looks for a free drive belonging to the specified pool and a tape with sufficient space to store in the submitted file. The writing operation is not done immediately but only booked in order to wait for other writings fittable on the same tape, this allows to optimize the writing execution. If after a configurable timeout no other write request arrive the writing is done anyway in order to give an upper delay limit.
The write logic favours the tape having the minimum free space yet sufficient to store in the file, this means that the system works trying to fill up partially used tapes before to use a new one. If the system is configured with a *mixed* configuration with both library and non automated drive (e.g. desktop model), tapes inside the library are preferred.
When both a tape with sufficient space and a free drive are found, the software either sends the command to load the tape into the drive in case of library or asks the user through a user web interface to manually load the tape in case of desktop drive. After that the tape is mounted for writing and a separate process is started for doing the real writing.

In case of a reading operation the client has to pass the *flocat* of the resource he wants to retrieve hence including the tape involved. The daemon checks that a drive is free and mounts the indicated tape supposed to contain the desired file, otherwise if the tape is on the shelf, the user is asked through the web user interface to manually load it in a specified desktop drive.
After this the software checks than the file is really on the tape and that there is enough space in the destination area, finally it launches a separate process for the extraction of the file on the indicated destination (local area or a remote share).

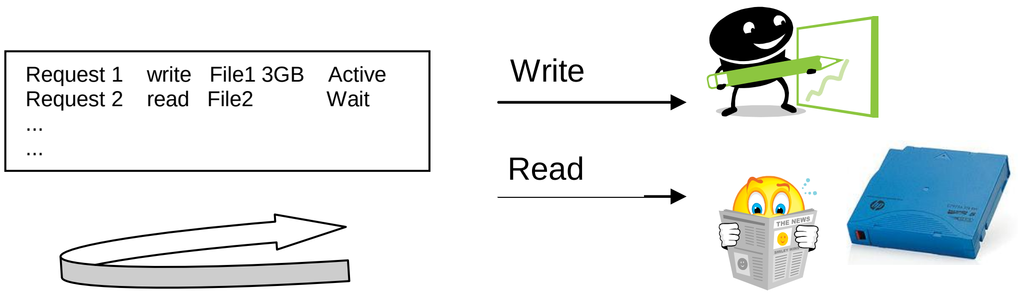

Figure 3 - LTF Serving daemon logic

The processes taking care of reading/writing operations report on the database the result of every operations, either success or error code and error description in case of problems. Higher level software can inspect the database to report with respect of these workflow exceptions, an example is the Web interface used for controlling the non automated LTO drives.