LAD'21 – Septembre 28-30, 2021

Lustre-OpenStack integration

Thomas Leibovici, CEA
Jean-Sébastien Bevilacqua, LINAGORA

Interests

- Access data that resides in your Lustre filesystem securely from the Internet

  ➔ **Connect your HPC storage infrastructure to the rest of the world**

- Use Lustre as a common storage backend for various purposes

  ➔ **Save money & reduce maintenance effort**

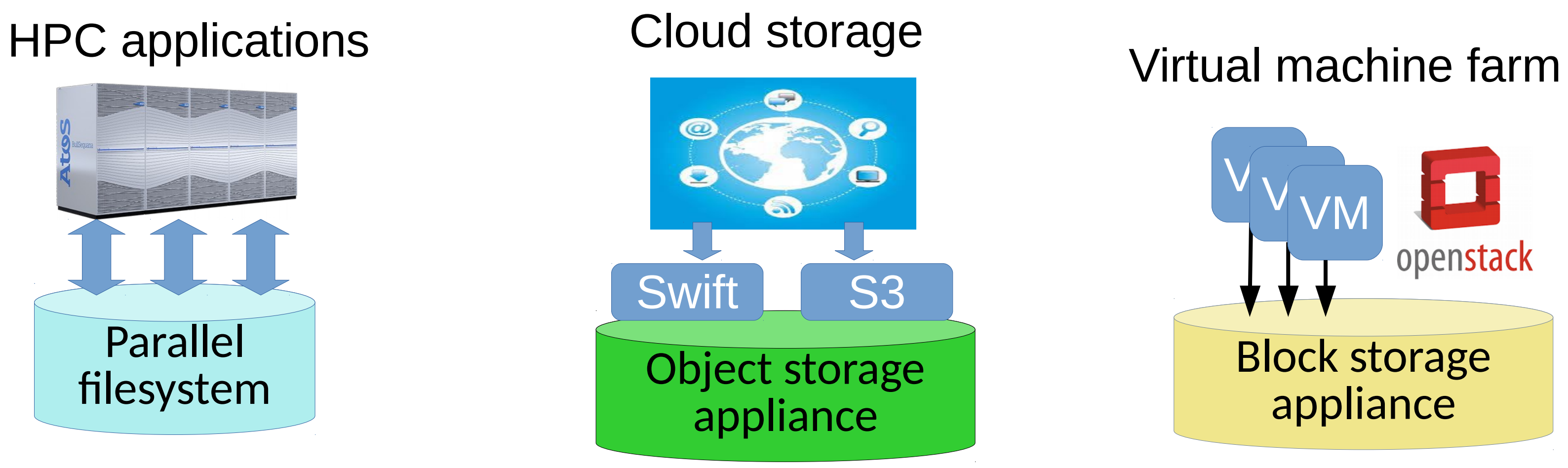
Accessing Lustre over the Internet: security concerns

Issues with public sharing of Lustre over the internet

- Clients must run Lustre client (compatible) software
- Clients needs to be trusted
- Fine control of access rights of any client
- Authentication mechanism needs to be configured on client and server side
- Clients can make the filesystem unstable (wrong behavior, taking ldlm lock and disappear, exploiting known bugs…)
- Can Lustre support millions of clients?

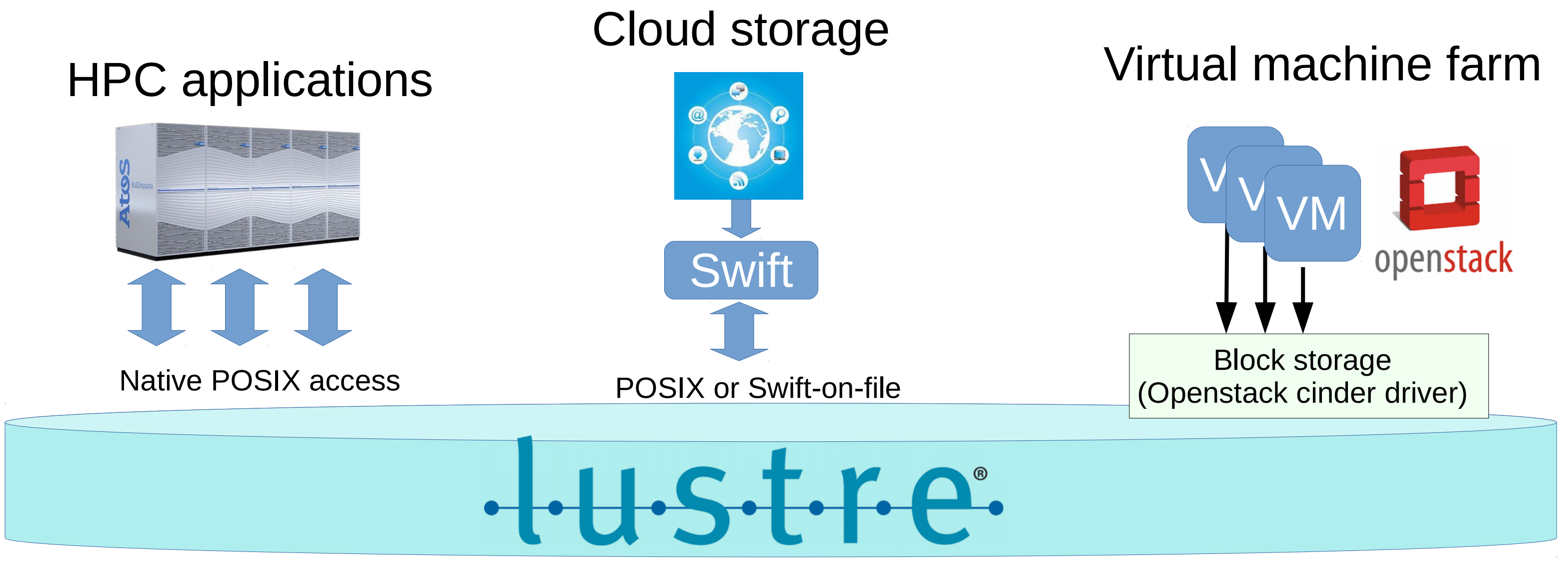
➔ **Obviously not a good idea!**

Fragmentation of resources in compute centers and data centers

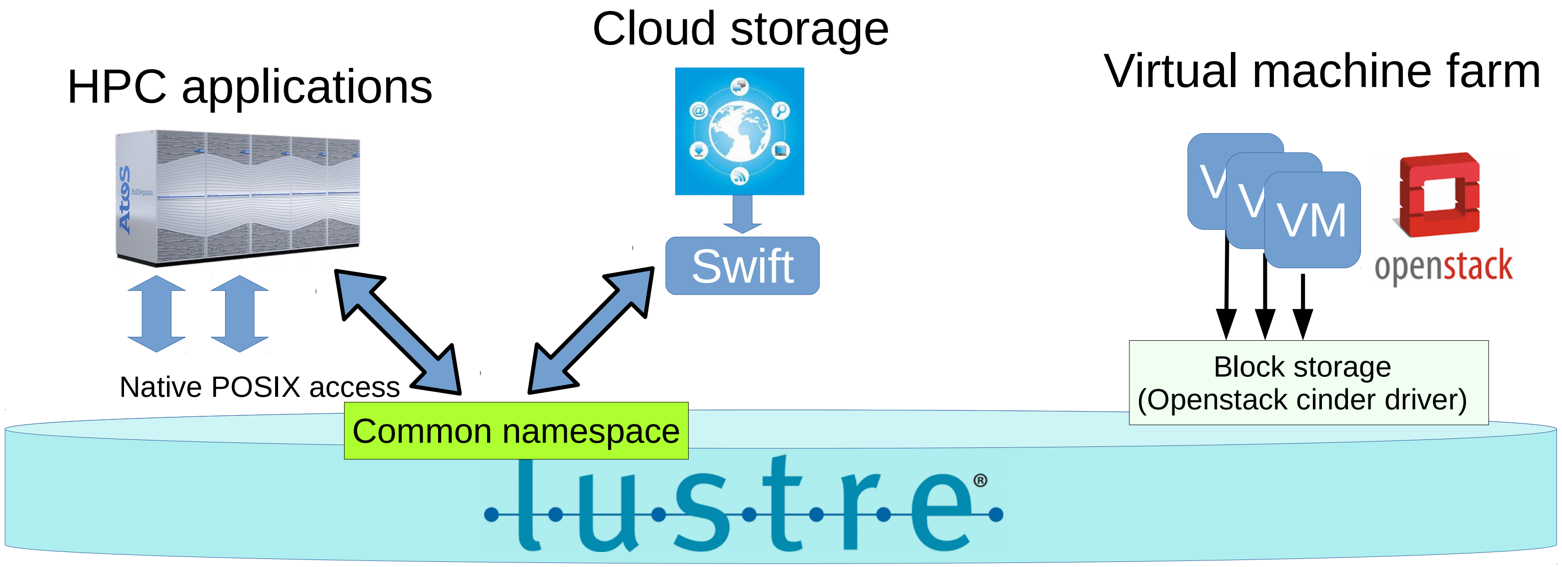


- ➔ Not flexible: fragmented resources can't be allocated as needed
- ➔ No data sharing between the systems
- ➔ Requires many fields of expertise: Lustre, CEPH, SAN, NAS, object store…

Using Lustre as a single storage backend for all purposes



Icing on the cake: unified view between Swift and POSIX



Bi-directional namespace synchronization:
Data pushed through Swift can be accessed by HPC application and vice versa

A little background

- The **ICEI** project (European H2020 project) aims to implement a federated infrastructure across European HPC sites, called **FENIX**
- It includes funding for the implementation of new services required in this project
  - “Swift-over-Lustre” is one of the identified required features
- Due to its long experience with Lustre, CEA was designated to lead the RFP to develop it
- The company selected at the end of the tender procedure is LINAGORA

LINAGORA

LUSTRE AND OPENSTACK INTEGRATION

30 September 2021

SOMMAIRE

0. INTRODUCTION
1. LUSTRE AND CINDER INTEGRATION
2. LUSTRE AND SWIFT BASIC INTEGRATION
3. LUSTRE AND SWIFT DEEP INTEGRATION
4. CONCLUSION

Introduction / Presenter

Jean-Sébastien BEVILACQUA

Technical Manager

Work with CEA in order to integrate OpenStack and Lustre

lustre®

openstack®

LINAGORA

INTRODUCTION / PURPOSE OF THIS INTEGRATION

Lustre and Cinder integration

- Store Cinder volumes directly in a Lustre FS
- Attach these volumes to a VM with Nova

Lustre and Swift integration (basic)

- Write a guide in order to deploy Swift on a Lustre FS

Lustre and Swift integration (deep)

- Integrate deeper with SwiftOnFile

PART 1

LUSTRE AND CINDER INTEGRATION

Automatic mounting by Cinder of the remote Lustre FS

- At launch, the Cinder service has to take care of the availability of the Lustre FS (Cinder manages the mount)

Volume management in Lustre FS

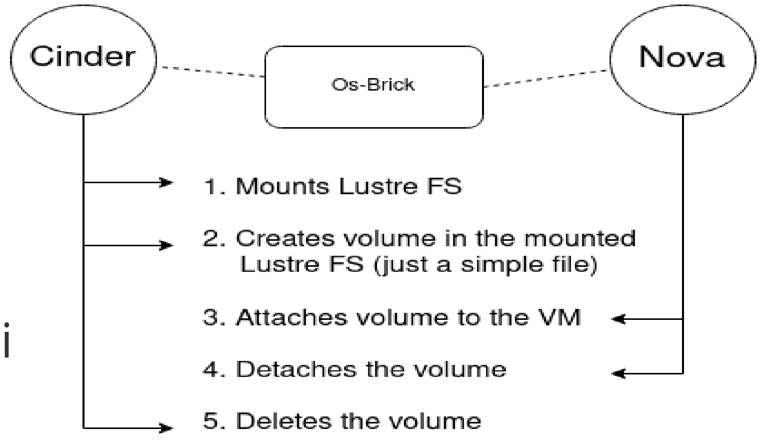
- Cinder has to be able to create, delete and manage disks in the Lustre FS

Attaching volumes to a VM via Nova

- Nova has to be able to attach disks stored in Lustre FS

Development should be done in multiple modules

- Cinder: disk management
- Nova: attachment and detachment of di
- os-brick: underlying library launching mount operations

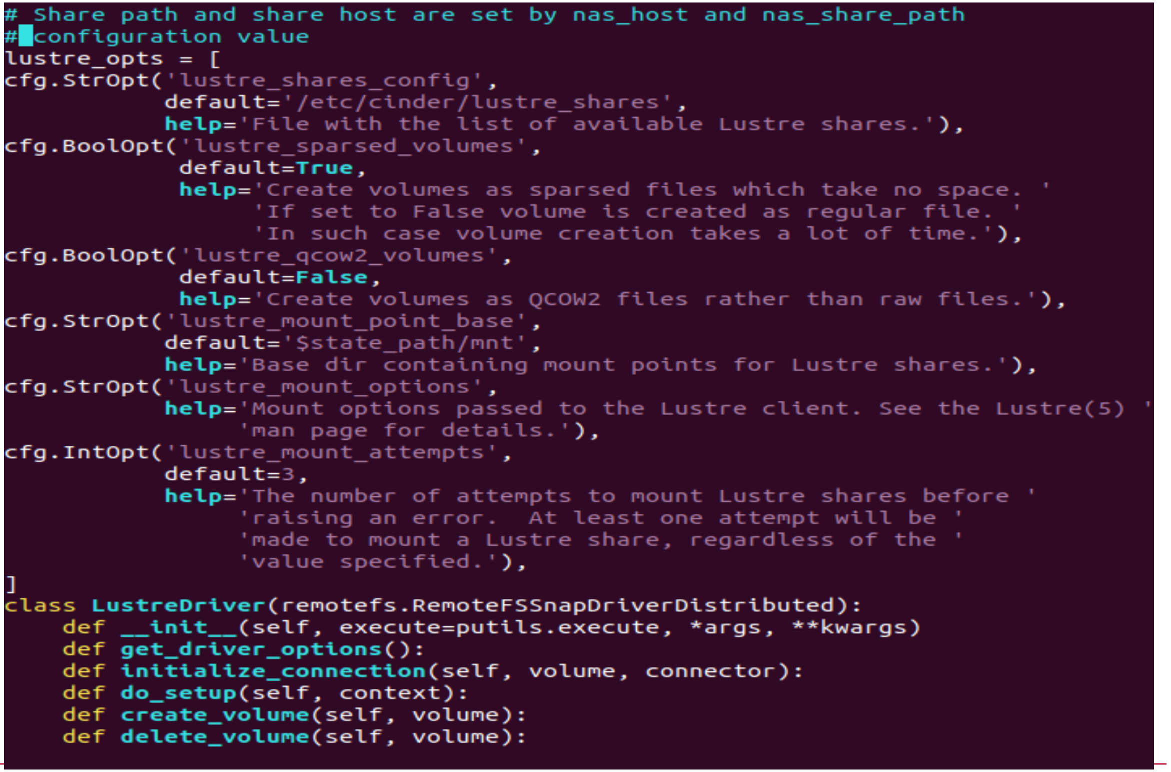

Cinder
Os-Brick
Nova
1. Mounts Lustre FS
2. Creates volume in the mounted Lustre FS (just a simple file)
3. Attaches volume to the VM
4. Detaches the volume
5. Deletes the volume

Operations performed by Cinder:

- When starting the service, mounting the Lustre FS configured in /etc/cinder/cinder.conf (nas_host, nas_share_path)
- Mounting of the FS in a directory directly managed by Cinder
- Volume creation and deletion = Creation and deletion of a file

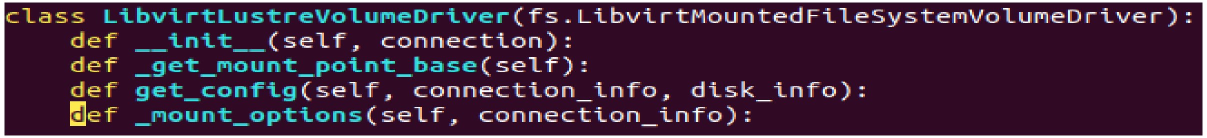
Note : A volume corresponds to a file in the FS

Development of a new driver: cinder/volume/drivers/lustre.py



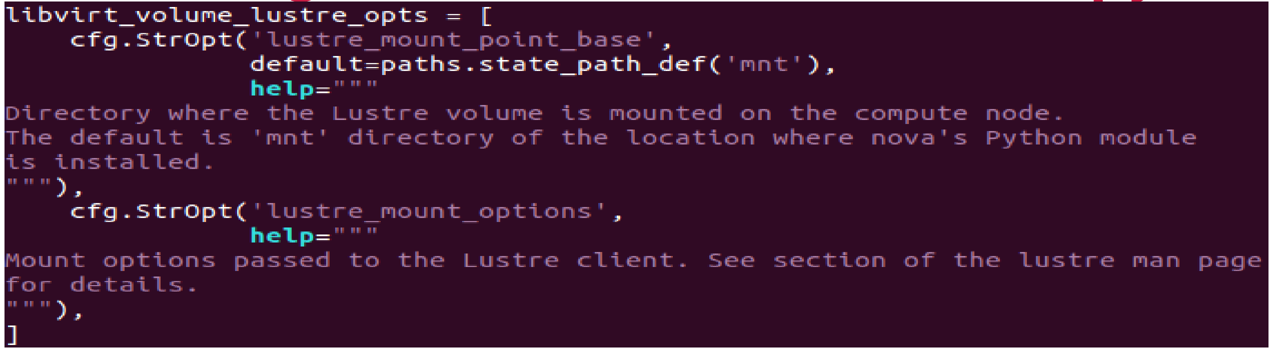
```
# Share path and share host are set by nas_host and nas_share_path
# configuration value
lustre_opts = [
cfg.StrOpt('lustre_shares_config',
           default='/etc/cinder/lustre_shares',
           help='File with the list of available Lustre shares.'),
cfg.BoolOpt('lustre_sparsed_volumes',
            default=True,
            help='Create volumes as sparsed files which take no space. '
                 'If set to False volume is created as regular file. '
                 'In such case volume creation takes a lot of time.'),
cfg.BoolOpt('lustre_qcow2_volumes',
            default=False,
            help='Create volumes as QCOW2 files rather than raw files.'),
cfg.StrOpt('lustre_mount_point_base',
           default='$state_path/mnt',
           help='Base dir containing mount points for Lustre shares.'),
cfg.StrOpt('lustre_mount_options',
           help='Mount options passed to the Lustre client. See the Lustre(5) '
                'man page for details.'),
cfg.IntOpt('lustre_mount_attempts',
           default=3,
           help='The number of attempts to mount Lustre shares before '
                'raising an error.  At least one attempt will be '
                'made to mount a Lustre share, regardless of the '
                'value specified.'),
]
class LustreDriver(remotefs.RemoteFSSnapDriverDistributed):
    def __init__(self, execute=putils.execute, *args, **kwargs)
    def get_driver_options():
    def initialize_connection(self, volume, connector):
    def do_setup(self, context):
    def create_volume(self, volume):
    def delete_volume(self, volume):
```

Development of a new driver : nova/virt/libvirt/volume/lustre.py



```
class LibvirtLustreVolumeDriver(fs.LibvirtMountedFileSystemVolumeDriver):
    def __init__(self, connection):
    def _get_mount_point_base(self):
    def get_config(self, connection_info, disk_info):
    def _mount_options(self, connection_info):
```

With custom configuration : nova/conf/libvirt.py



```
libvirt_volume_lustre_opts = [
    cfg.StrOpt('lustre_mount_point_base',
               default=paths.state_path_def('mnt'),
               help="""
Directory where the Lustre volume is mounted on the compute node.
The default is 'mnt' directory of the location where nova's Python module
is installed.
"""),
    cfg.StrOpt('lustre_mount_options',
               help="""
Mount options passed to the Lustre client. See section of the lustre man page
for details.
"""),
]
```

PART 2

LUSTRE AND SWIFT BASIC INTEGRATION

PART 2 / LUSTRE AND SWIFT BASIC INTEGRATION

Simple installation of Swift with slight modifications

- Only one device, the Lustre FS (normally you need several devices to manage the distribution of data)
- Lustre takes care of replication
- Very small Swift Ring because only one device
- All Swift features are supported → Utilization of Lustre is transparent
- Lustre striping optimisation may be required

Note : The Ring is a file allowing to all service to know where others service are located.

PART 3

LUSTRE AND SWIFT DEEP INTEGRATION

PART 3 / LUSTRE AND SWIFT DEEP INTEGRATION / GOALS

To be able to access files stored in Swift directly through the FS

- Swift uses its own unreadable storage architecture

Swift: `/mnt/sdb1/2/node/sdb2/objects/981/f79/f566bd022b9285b05e665fd7b843bf79/1401254393.89313.data`

To be able to recover a file via Swift which has been set up in the FS

- Swift has to know the new added file

Partial solution : SwiftOnFile

x / swiftonfile

PART 3 / SWIFTONFILE / OVERVIEW

Pros of SwiftOnFile

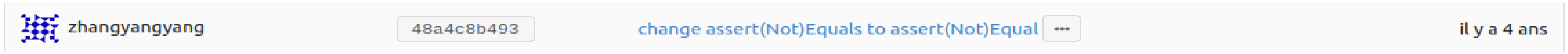
- Implemented as a storage policy → No need to modify the Swift code
- SwiftOnFile allows files to be stored on the FS in a readable way

Swift: `/mnt/sdb1/2/node/sdb2/objects/981/f79/f566bd022b9285b05e665fd7b843bf79/1401254393.89313.data`
SoF: `/mnt/swiftonfile/acct/cont/obj`

Cons of SwiftOnFile

- Not compatible with Python 3
- Not compatible with the last Swift API
- Can't synchronize file from Lustre to Swift

And the most important : Sof is no longer maintained


zhangyangyang 48a4c8b493 change assert(Not)Equals to assert(Not)Equal ··· il y a 4 ans

PART 3 / SWIFTONFILE / CON 1 : PYTHON 3 SUPPORT

Python 3 support is mandatory

- Swift no longer supports Python 2 but only Python 3
- OS Victoria version (last stable) targets Python 3

Difficult migration

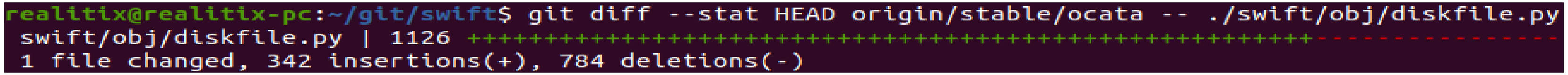
- SwiftOnFile communicates a lot with the outside world (IO)
- Major breaking change with string IO when passing from Python 2 to Python 3

PART 3 / SWIFTONFILE / CON 2 : NEW SWIFT API

SwiftOnFile is based on the Swift API to implement the storage policy

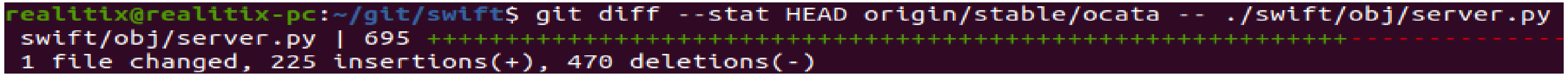
- Lots of API changes in the two code areas used by SwiftOnFile

swift/obj/diskfile.py



```
realitix@realitix-pc:~/git/swift$ git diff --stat HEAD origin/stable/ocata -- ./swift/obj/diskfile.py
 swift/obj/diskfile.py | 1126 +++++++++++++++++++++++++++++++++++++++++++++++++++++++----------------
 1 file changed, 342 insertions(+), 784 deletions(-)
```

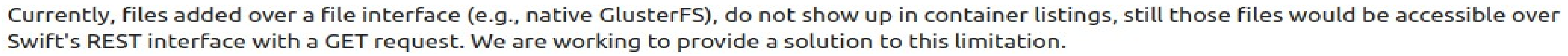
swift/obj/server.py



```
realitix@realitix-pc:~/git/swift$ git diff --stat HEAD origin/stable/ocata -- ./swift/obj/server.py
 swift/obj/server.py | 695 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++--------------
 1 file changed, 225 insertions(+), 470 deletions(-)
```

PART 3 / SWIFTONFILE / CON 3 : TWO-WAY SYNCHRONIZATION

SwiftOnFile does not support synchronization from FS to Swift


Currently, files added over a file interface (e.g., native GlusterFS), do not show up in container listings, still those files would be accessible over Swift's REST interface with a GET request. We are working to provide a solution to this limitation.

New service to be implemented

- Must run alone and listen to the fs → daemon
- Must perform well with Lustre → use of changelog
- Should also offer a generic version → Ideal to be contributed: open-source software!

PART 3 / SWIFTONFILE / TEST COVERAGE

SwiftOnFile's current test coverage : 60 %

- Increase test coverage to its maximum
  - → 100% test coverage on new developments
  - → Increase the test coverage to 100% on the existing code too

Goal :

PART 4

CONCLUSION

PART 4 / CONCLUSION

A lot of work to do

- Part 1 : Cinder part currently under development
- Part 2 : Swift and Lustre quickstart guide already completed
- Part 3 : SwiftOnFile under study

Work in progress

- Provisional schedule for part 1 : Start of the contribution Q1 2022
- Provisional schedule for part 3 : Start of the contribution Q2 2022
- It may take several months to be integrated into master and available for everyone

THANK YOU FOR YOUR ATTENTION