CHINA
OpenStack Days

CHINA
OpenStack Days
IT大咖说
知识分享平台

CHINA RUNS ON OPENSTACK

CHINA
OpenStack Days

GlusterFS as a Backend For File Service With Manila in China Mobile

Liu Yuan, Senior Architect & StorageTeam lead
Jin Weiyi, Senior Software Engineer

Agenda

- Cloud File Service
- Manila
- EFS System in China Mobile

Shared file Services

of all storage sold is for file-based use cases

per IDC, 2012

The Short History of Cloud File Services

File Share Service Market Trends

- OpenStack Manila: June 2013
- Microsoft Azure Files: May 2014
- Amazon Web Services Elastic File Services: April 2015

Storage Serivces	AWS	Azure	Openstack
Object Storage	S3	Blob & Table Storage	Swift
Archival(cold) Storage	Glacier	Azure Backup	?
Block Storage	EBS	Block Blob Storage	Cinder
Share File Storage	EFS	Azure Files	Manila

Cloud File Services Comparison

	Amazon EFS	阿里云文件存储	BC-NFS + Manila
发布时间	2015年4月(Preview) 2016年6月(Pro. Ready)	2016年3月 2017年5月发布NAS Plus	2015年10月(Liberty) 持续开发中
数据可靠性	多个Zone(EBS单个Zone)	99.99999999%	依赖后端存储
服务稳定性	SLA	99.9%（SLA）	依赖后端存储
访问协议	NFS v4.1	NFS v3, NFS v4(不全面）	Block,Ganesha(NFS), GlusterFS, CephFS
弹性扩容支持	自动扩容、缩容（无限制）	扩容(最大1PB)	Manila支持，同时依赖后端存储
容量|性能线性扩展	是, 支持1-1000s 实例并发	是 (无具体并发数字)	依赖后端存储
网络访问模式	VPC	VPC, 共享网络	VPC, 共享网络
本地数据中心访问	AWS Direct Connect	不支持	不支持
访问安全	VPC, 安全组，IAM授权，IP	VPC, 安全组，RAM授权，IP	VPC，安全组，IP
定价(月)	0.3 $ / GB (3x-10x EBS)	2元/GB (2x-5x 块存储)	N/A

	Amazon EFS	阿里云文件存储	BC-NFS + Manila
实测性能	多个EC2高度并行数据访问保持高IO性能（110MB/s)	多个ECS并行访问数据IO性能差*(2MB/s)	多个EC2高度并行数据访问保持较高IO性能(100 MB/s)
性能模式	支持两种：一般用途和最大IO	不区分	不区分

一般用途（General Purpose）：
- 模式适用于大多数文件系统应用场景；
- 在低并发访问模式下，提供较高的IO性能；

最大IO（Max I/O）：
- 适用于高度并发访问需求的场景，如大数据分析等；
- 在高并发访问模式下，提供很高的IOPS和吞吐量；

*阿里云17年5月发布云NAS Plus，性能得到极大提升

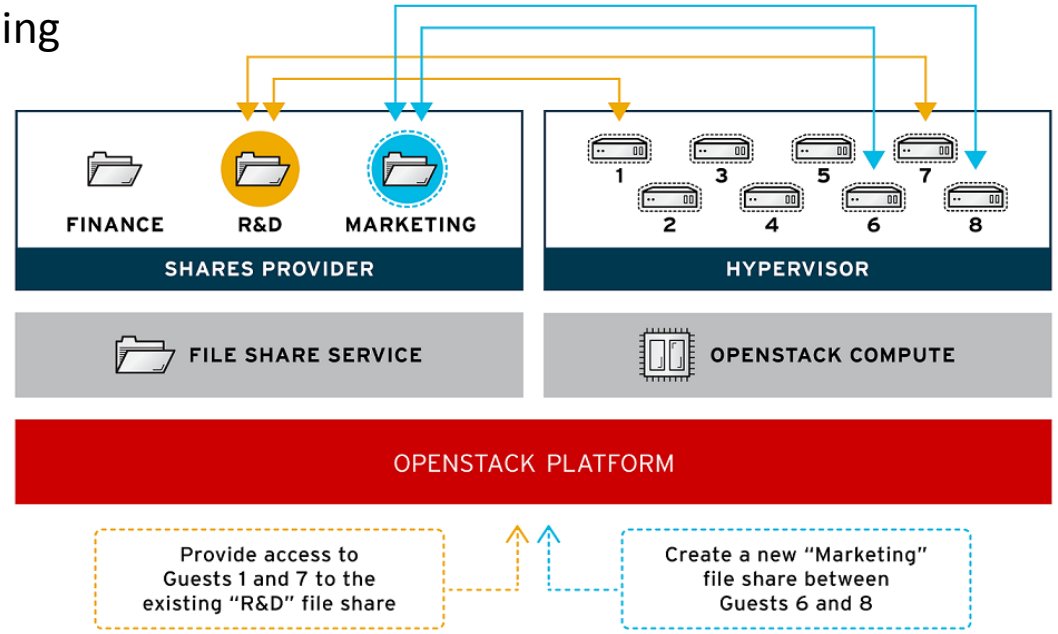
Why GlusterFS

方案	Generic Cinder Block	GlusterFS	CephFS
是否需要Service Server	是	否	否
后端存储	Ceph RBD	GlusterFS	CephFS
访问协议	NFS/CIFS	NFS/CIFS/POSIX	CephFS
优点	简单，成熟，虚机无需增加网卡链接存储私有网络	简单，较成熟有HA方案，支持IP认证	?
缺点	需Service VM, 无HA方案，性能差	需虚机增加网卡同存储私有网络链接	复杂，不成熟，Cephx认证

Manila: The OpenStack Shared File Service Program

Bringing self-service, shared file services to the cloud

- An Open, Standard API for self-service management & provisioning of shared file systems.
- Vendor neutral API for provisioning and attaching filesystem-based storage such as NFS, CIFS, CephFS, HDFS and other network filesystems.



Manila: Overview of Key Concepts

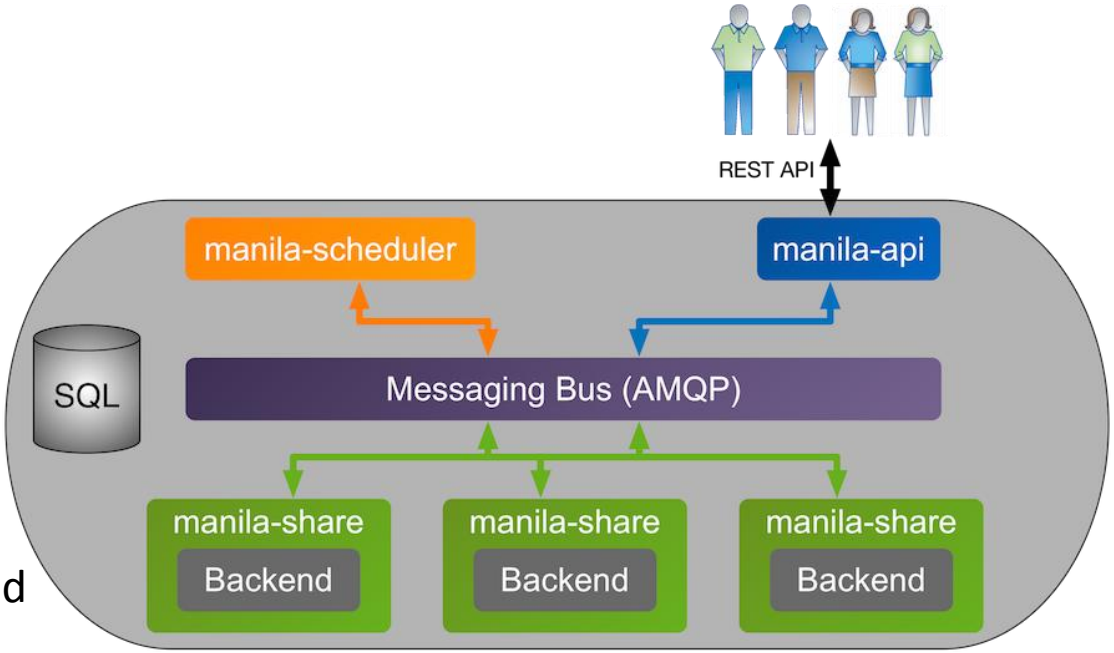
- *Share (an instance of a shared filesystem)*
  - User specifies size, access protocol, “share type”
  - Can be accessed concurrently by multiple instances
- *Share access rules (ACL)*
  - Defines which clients can access the share
  - Specified by IP in CIDR notation
- *Share network*
  - Defines the Neutron network & subnet through which instances access the share
  - A share can be associated with a single share network
- *Security service*
  - Finer-grained client access rules for Authn/z (e.g. LDAP, Active Directory, Kerberos)
  - Share can be associated to multiple security services

Manila: Overview of Key Concepts

- *Snapshots*
  - Read-only copy of share contents
  - New share can be created from a snapshot
- *Backend*
  - Logical storage pool and provider of shares
  - a share resides on a single backend
- *Driver*
  - Vendor or technology-specific implementation of backend API

Manila: Core Processes

- manila-api
  - Exposes REST API through WSGI
- Mania-scheduler
  - Makes provisioning decisions for share requests
- Manila-share
  - Manager share processes per backend
  - Responsible for communicating with storage subsystems



Manila: Generic Share Driver

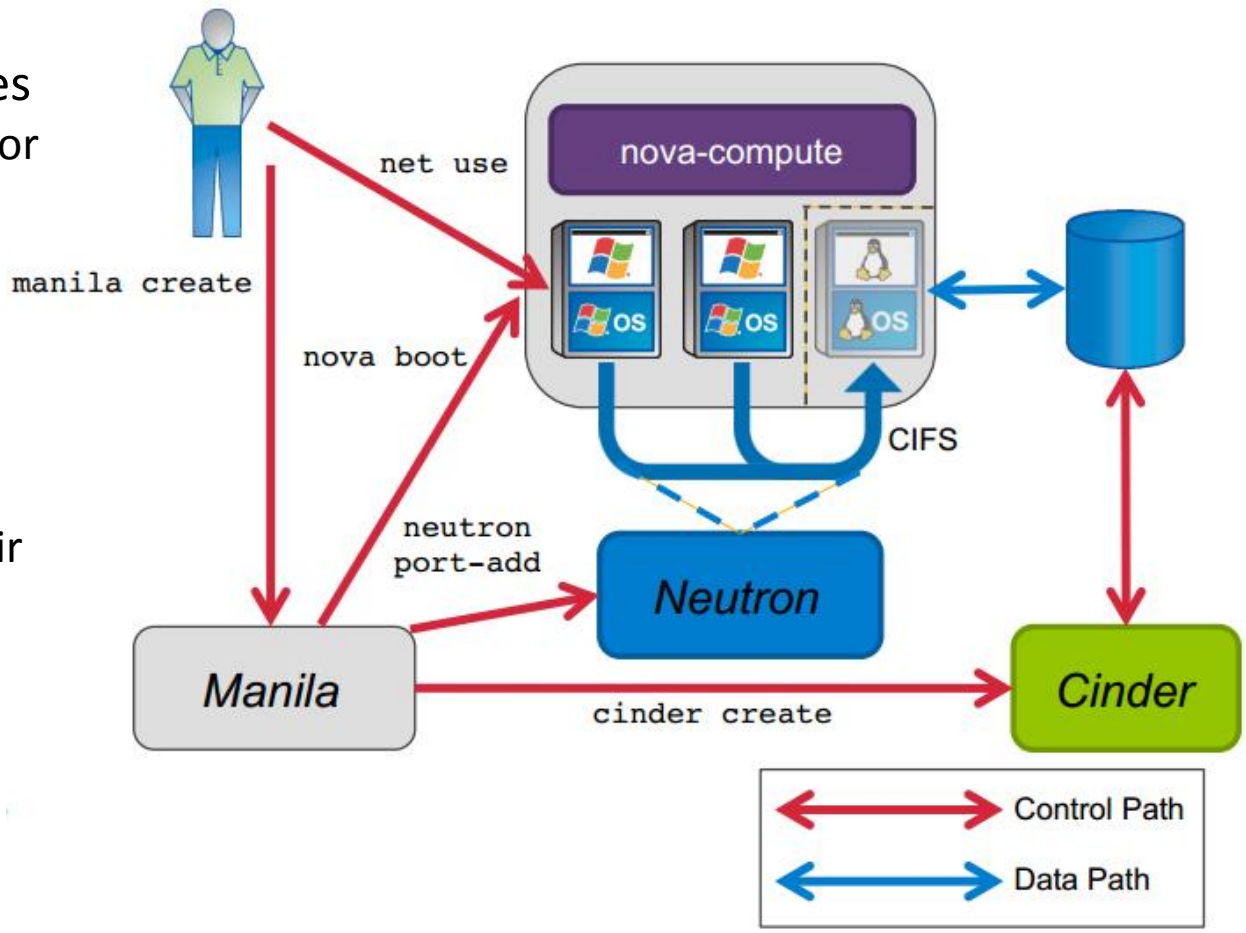
Creates a Nova instance to offer NFS/CIFS shares backed by Cinder volumes

– New instance is created for each ”share network”

– Connected into existing Neutron network & subnet

– Instance flavor, source Glance image, & SSH keypair are configurable in manila.conf

– Manila creates shares in instance using Linux commands over SSH



Manila: Generic Share Driver

Pros:

– Manage both control path and data path.

– Take advantage of openstack core modules: nova, neutron and cinder.

Cons:

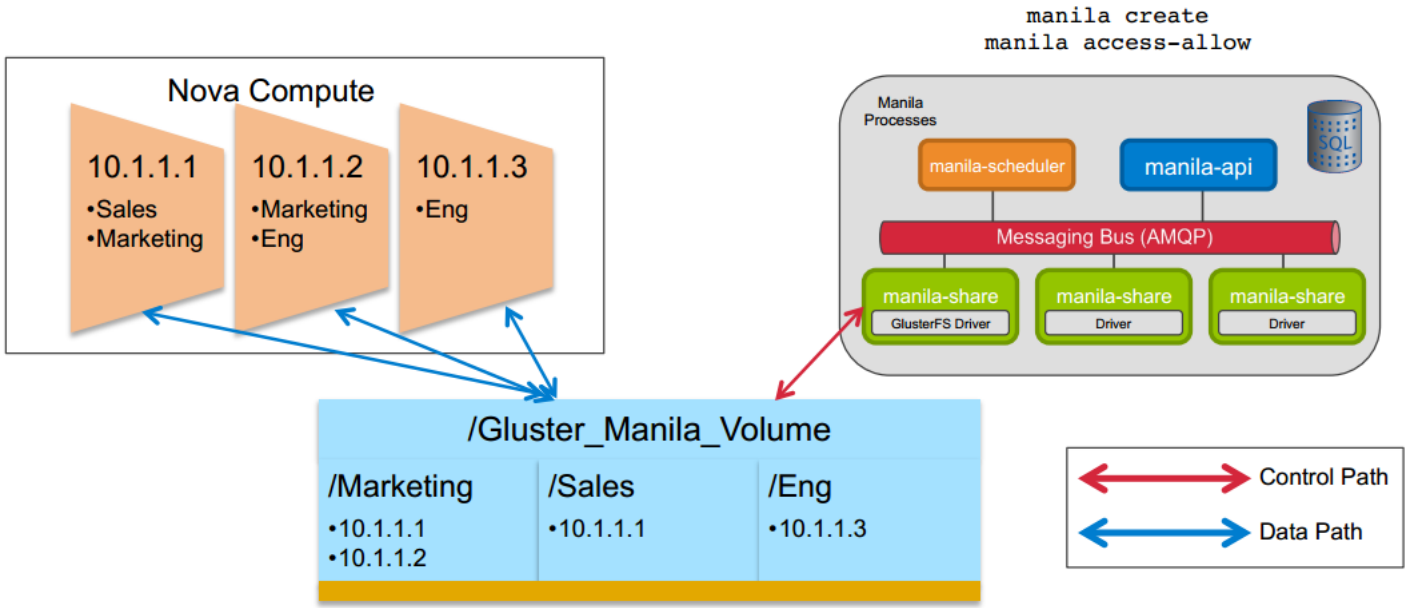
– It is unstable, share servers have SPOF(Single Points Of Failure) problems.

– Extra compute resources overhead.

– Compatibility issues with 3rd party neutron network plugin.

Generic share driver is a Reference Implementation driver, not applicable in production.

Manila: GlusterFS Share Driver



Manila: GlusterFS Share Driver

Using GlusterFS as the storage back end for serving file shares to the Shared File Systems clients.

Two driver types:

- GlusterFS Native driver
  - Share layout only support GlusterFS volume.
  - Instances use glusterfs protocol to access shares.
  - Instances directly talk with the GlusterFS back end storage pool.
  - Access to each share is allowed via TLS Certificates.

- GlusterFS driver
  - Two share layouts implemented: GlusterFS volume & top-level subdirectories.
  - Both of NFS ganesha & Gluster NFS supported.
  - Shares can be accessed by NFSv3 & v4 protocols.

Data path is not controlled by manila GlusterFS Share Driver

Manila: GlusterFS Share Driver

Why not GlusterFS Native driver?

- Only Glusterfs protocol access allowed

- Invasive operations to user client
  - Embedding TLS Certificates
  - Requirement of GlusterFS client application

- Out of band management of driver
  - GlusterFS volumes are not created on demand
  - Certificate setup (aka trust setup) between instance and storage backend

Manila: flaws with GlusterFS driver

GlusterFS driver is just a Demo Implementation!

- Uncompleted implementation
  - NFS Ganesha portion is semi-finished
  - Without HA NFS-Ganesha cluster features

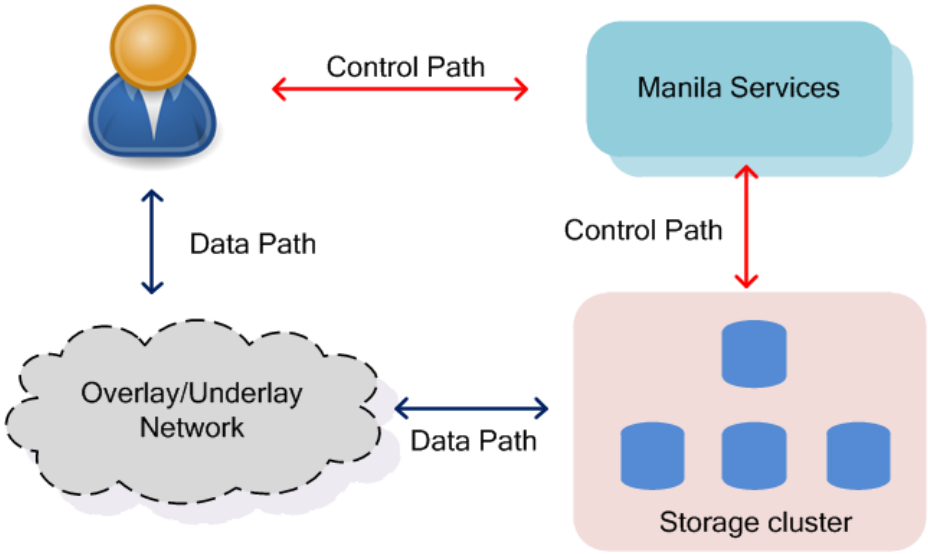
- SPOF problems
  - Control path between share driver and GlusterFS cluster may be disabled by the failure of some GlusterFS server
  - Share instance(s) may lose control by the failure of some backend

- Lack of consistency guarantees
  - Share status in DB and backend may be different

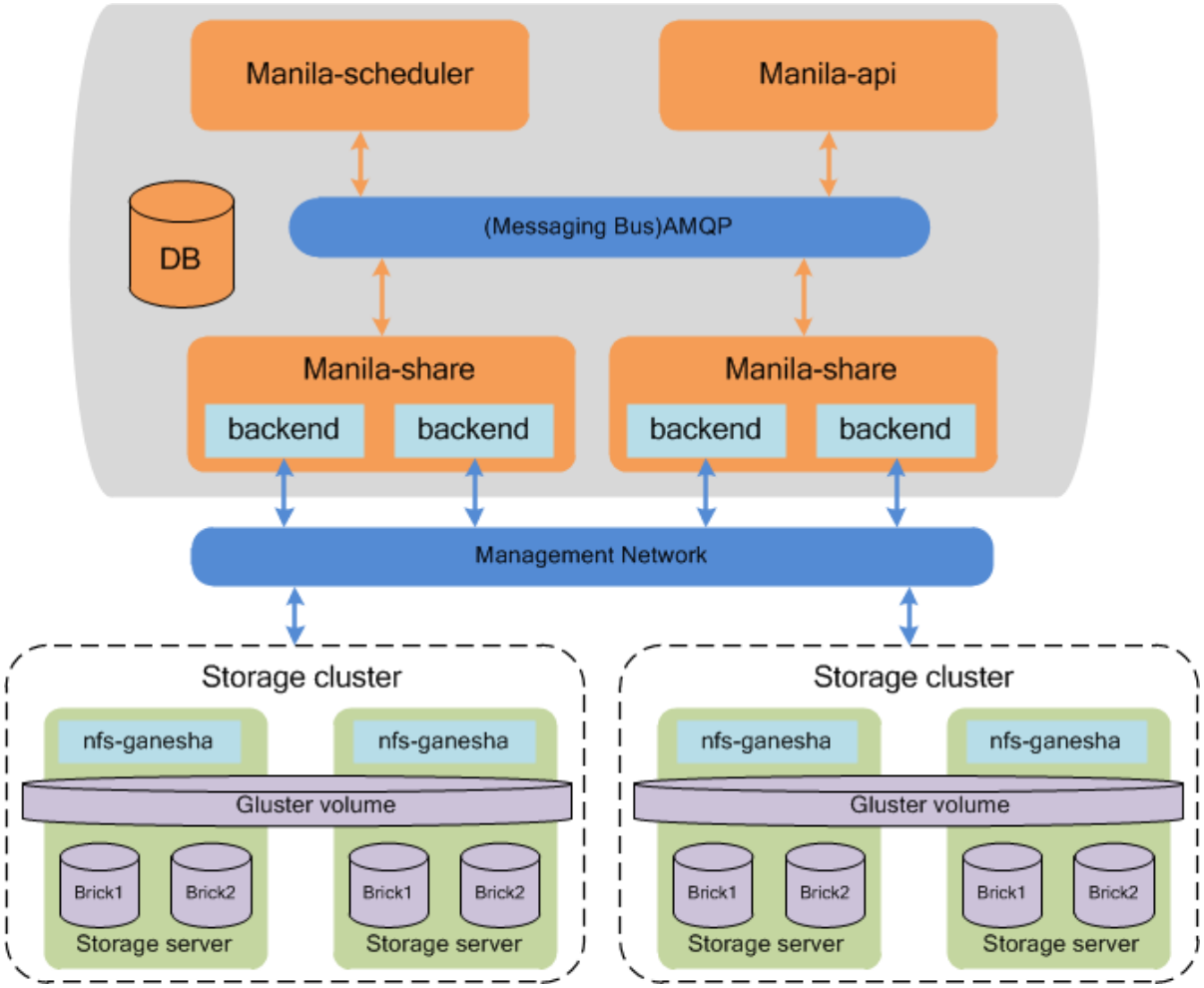
Overview of EFS system

Manila is in contorl plane.

- Control plane
  - Provider self-service shared file system service
  - Lifecycle of shared file system is controlled by its owner
- Data plane
  - Provider data path to access shared file system in storage pool



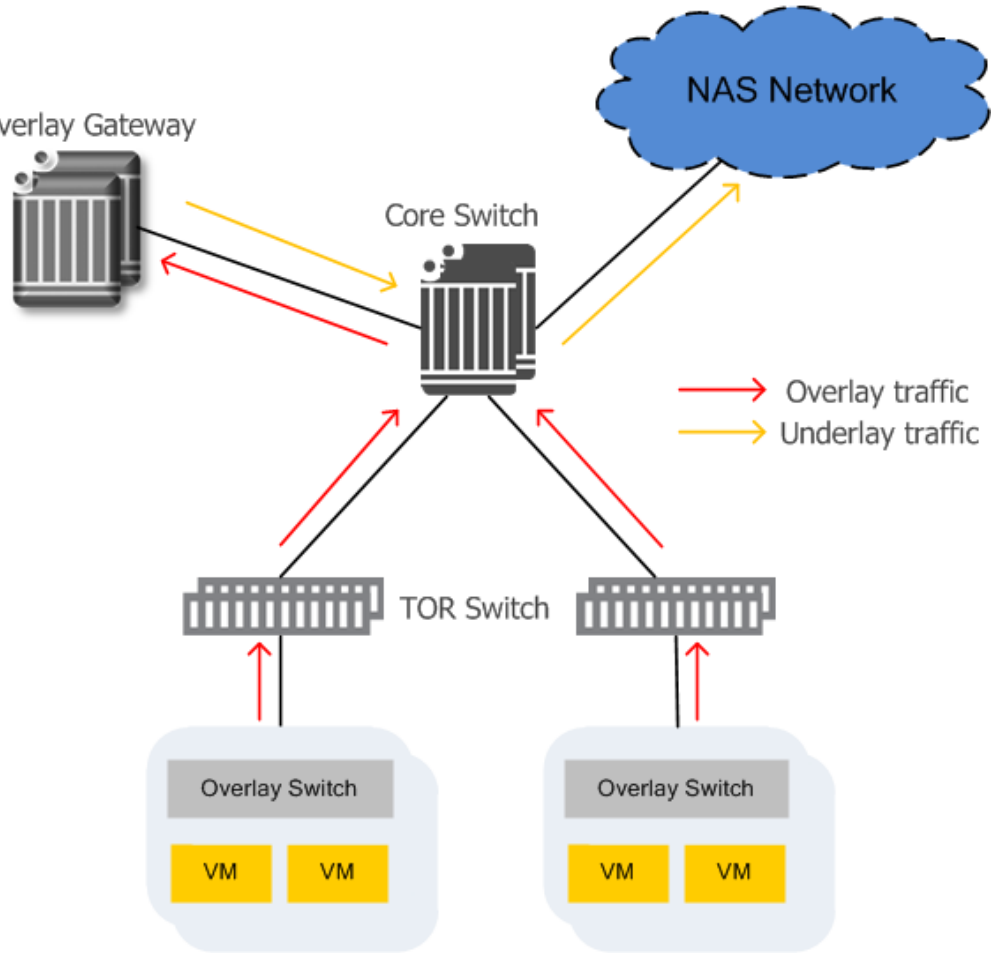
EFS system: control plane



EFS system: data plane

Basic requirements of data plane:

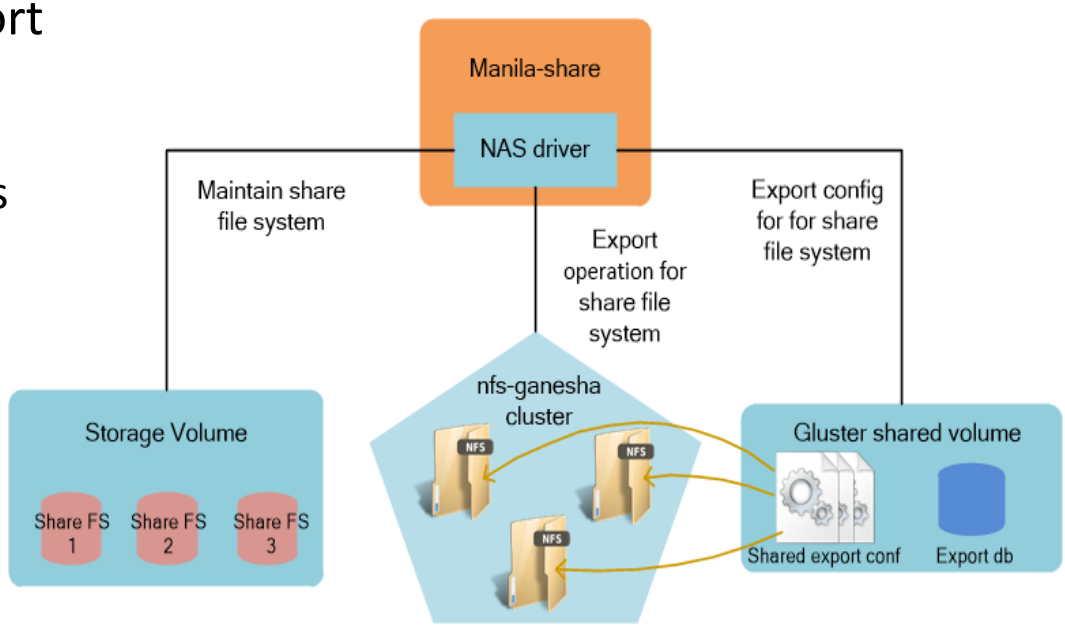
- ➢ Identification
  - NAS servers must get identities(ip addresses) from clients
- ➢ Network reachability
  - Make sure of network layer reachability between clients and NAS servers



EFS system: GlusterFS driver refactor

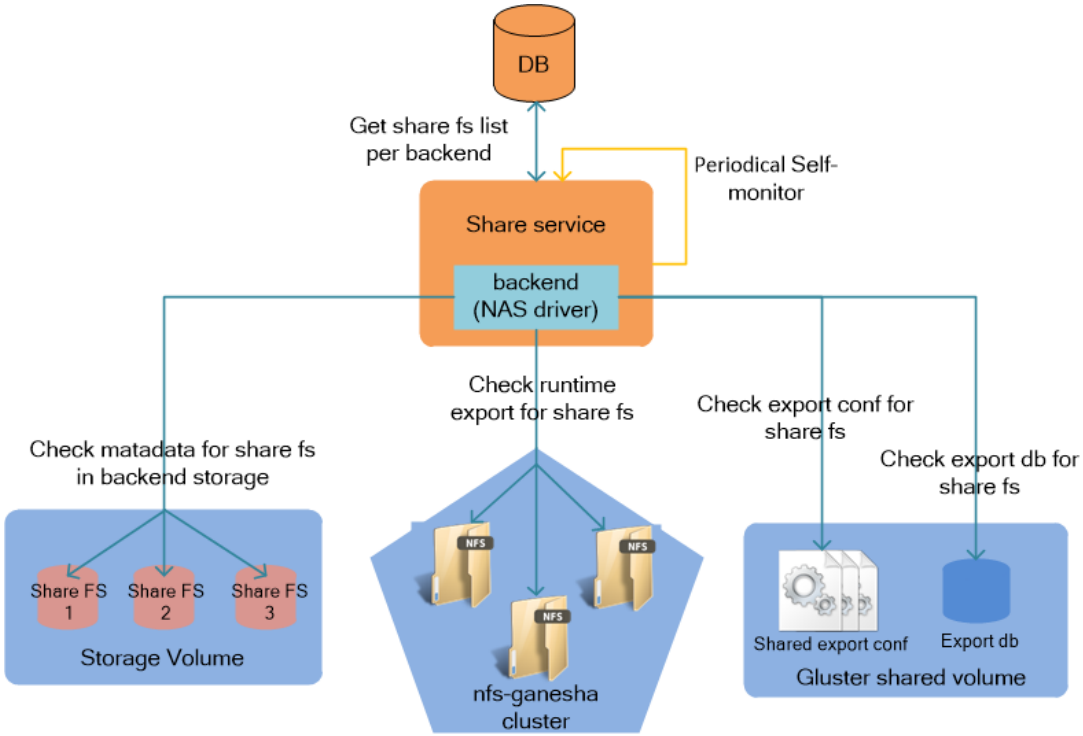
A production-ready driver:

- NFS ganesha cluster support
  - Guarantee export id uniqueness
  - Guarantee export status consistency
- No SPOF problems
  - Multiple control paths between diver and GlusterFS cluster



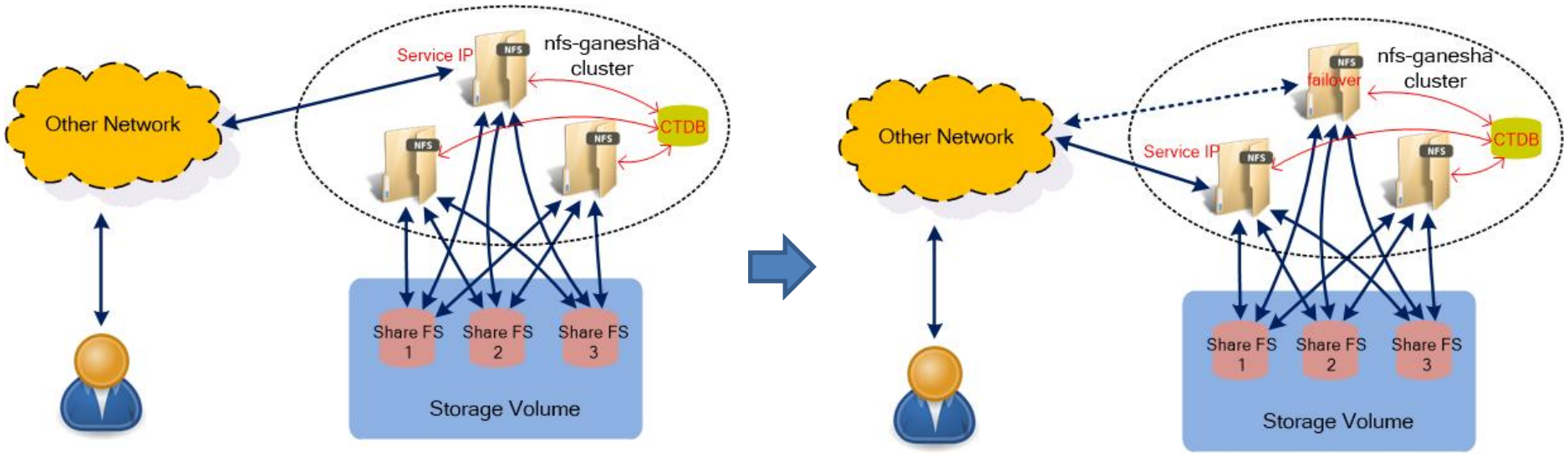
EFS system: GlusterFS driver refactor

Share instance self-monitor mechanism



EFS system: service continuity

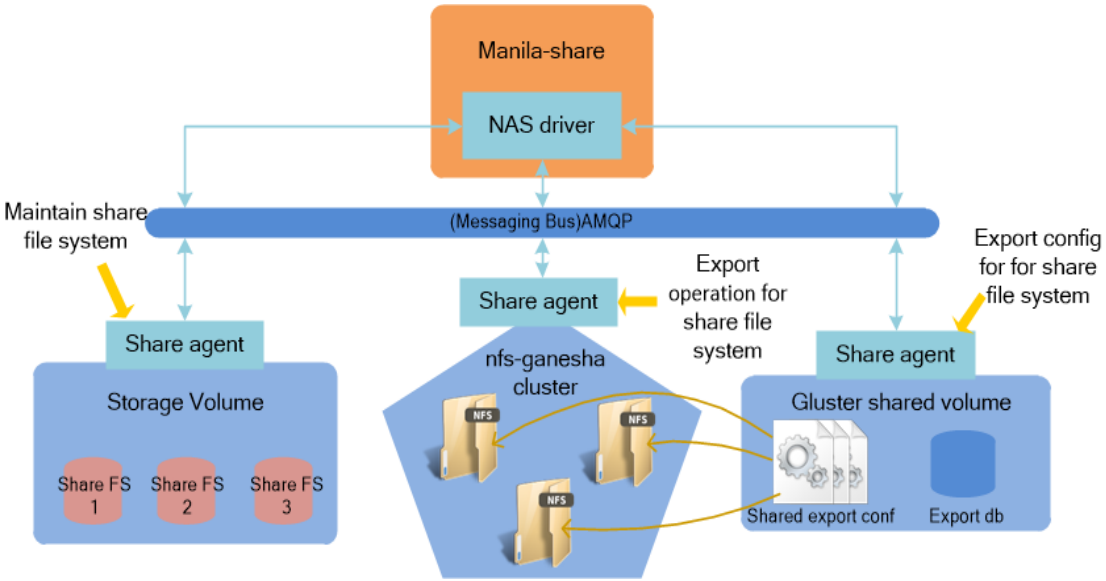
Prerequisites: export status consistency in the scope of ganesha cluster



EFS system: GlusterFS driver evolution

New component type named share agent:

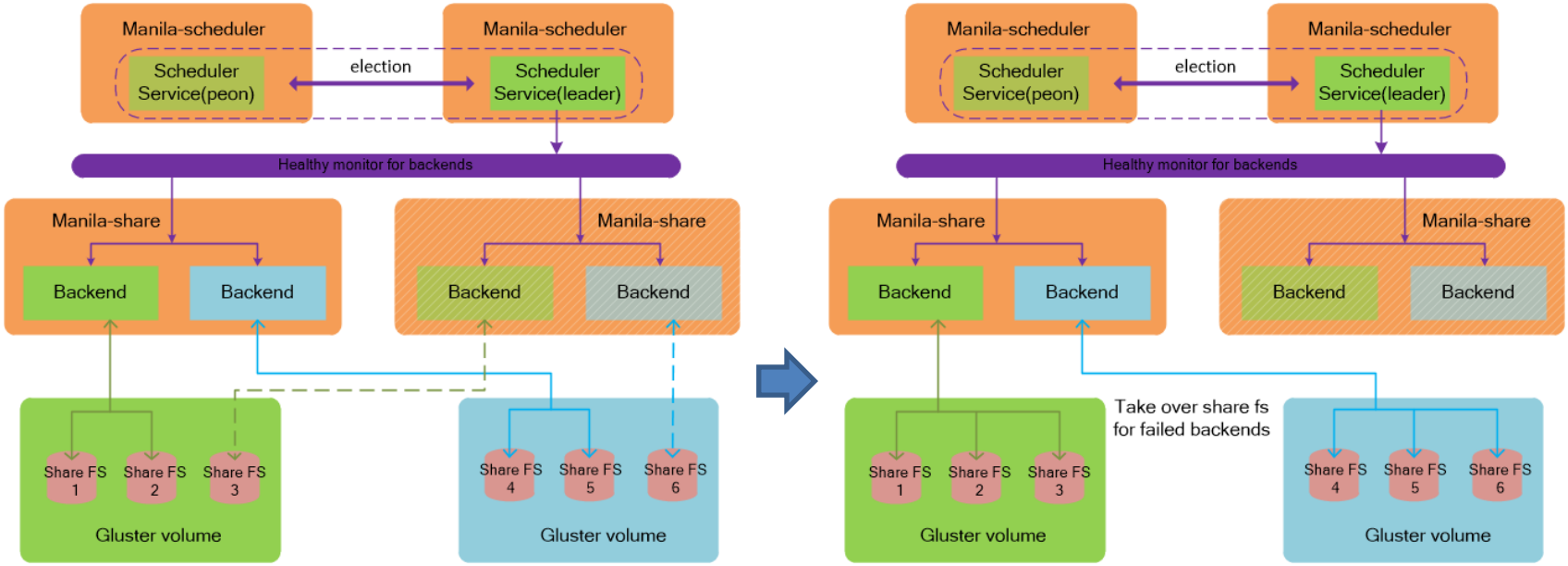
- Transfer implementation layer from share drive to share agent
- Integration with storage subsystem
- AMQP-based system instead of SSH-based control path



EFS system: backend HA

Make sure all the shares are always under control:

- Backends status monitored by leader scheduler service
- Shares control migration within backend if failure occurrence



THANK YOU