OpenStack & GPFS: overview dell'infrastruttura di prova

Alessandro Brunengo

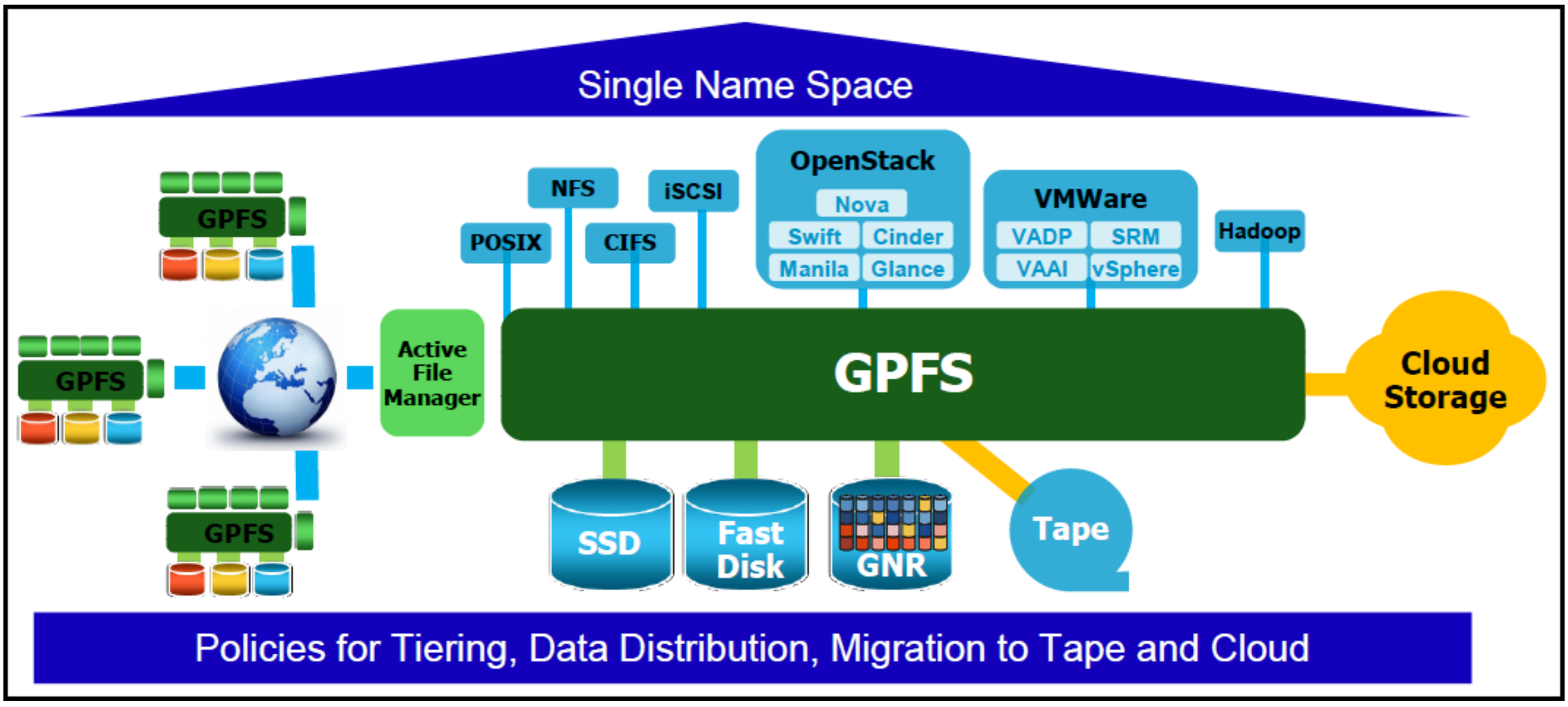
INFN Sezione di Genova

Reminder: GPFS per OpenStack

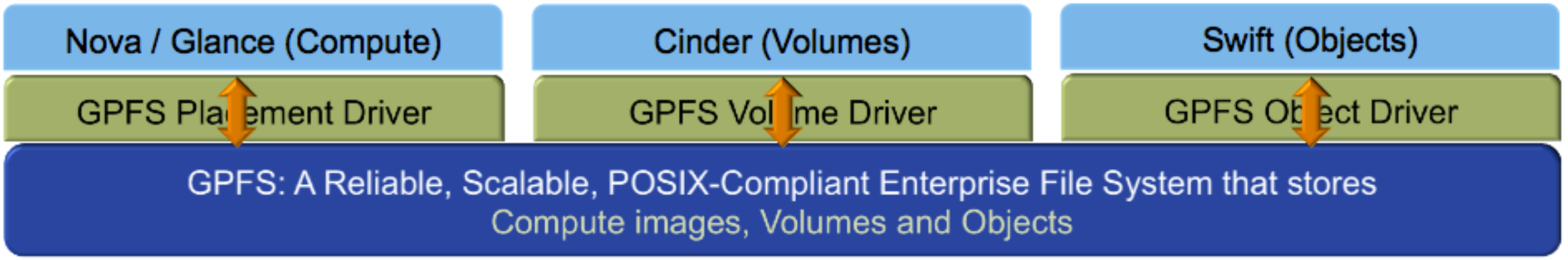
- GPFS come unico data layer del CED
- Unifica immagini delle VM, block device, oggetti e file
- Singolo namespace ovunque stiano i dati
- Parità de-clusterizzata - RAID GPFS nativo
  - questa e' una opzione, non configurata nella infrastruttura di test
- Gestione della localizzazione dei file (storage pool)
  - tiering per costi e performance
  - ottimizzazione per i tempi di accesso
- Tutto a livello software

Il punto di vista di GPFS

- È decisamente "GPFS-centrico"
  - e non solo per OpenStack



GPFS e OpenStack



- Un driver per ciascuno dei servizi che ha bisogno di un backend di storage
  - evita molte delle copie dei dati
  - garantisce accesso locale
  - usa le feature di GPFS (ad esempio copy-on-write)
  - ridondanza a livello di file system: non ha bisogno di repliche
- Esempio: creare un nuovo volume
  - copy-on write di una snapshot
  - nessuna formattazione
  - salta l'operazione più time consuming

Ambiente

- C'erano (un mese fa) due opzioni
  - RHEL6 + Icehouse
    - backporting del supporto ai network namespace sul kernel
      - inizialmente sul kernel RDO, ora sul kernel standard
    - release di OpenStack di primavera 2014
  - RHEL7 + Juno
    - network namespace nativo sul kernel
    - release di OpenStack recentissima
- Abbiamo scelto la prima opzione
  - anche se le abbiamo investigate entrambe…

Icehouse su Scientific Linux 6

- Abbiamo fatto una installazione del tutto manuale
  - repository di RDO-Icehouse
  - kernel specifico con backporting di features (ora incluse nel kernel della distribuzione standard)
    - "con questo kernel GPFS non compilerà mai" (cit. Brunengo)
  - installazione di GPFS 4.1 (e invece… GPFS compila)
  - installazione dei vari servizi sulle varie macchine
    - ! non sempre il manuale è affidabile

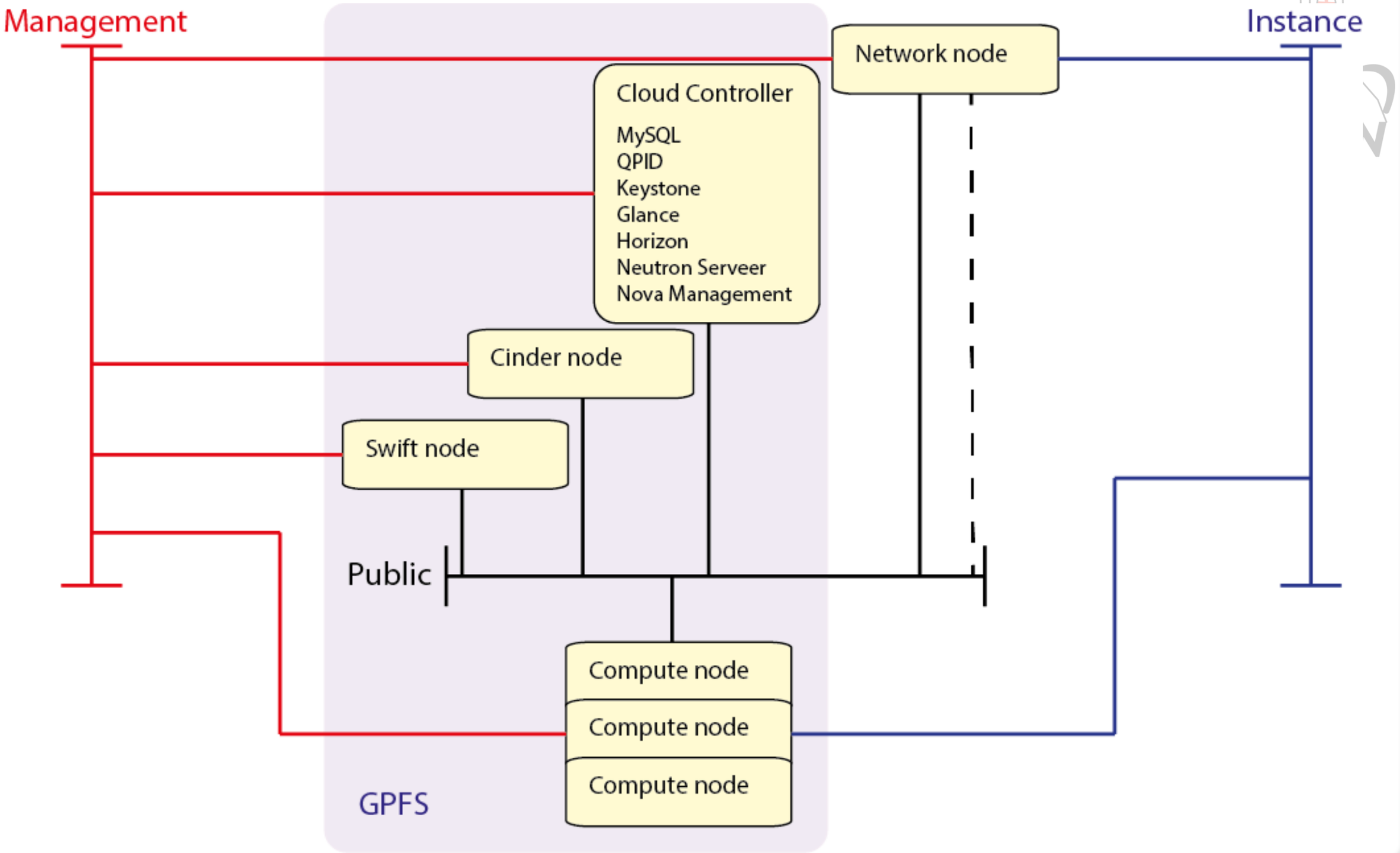
Note

Unlike other services, Networking typically does not require a separate step to populate the database because the `neutron-server` service populates it automatically. However, the packages for these distributions sometimes require running the **neutron-db-manage** command prior to starting the `neutron-server` service. We recommend attempting to start the service before manually populating the database. If the service returns database errors, perform the following operations:

- Configurazione dei pacchetti

- http://docs.openstack.org/icehouse/install-guide/install/yum/content/

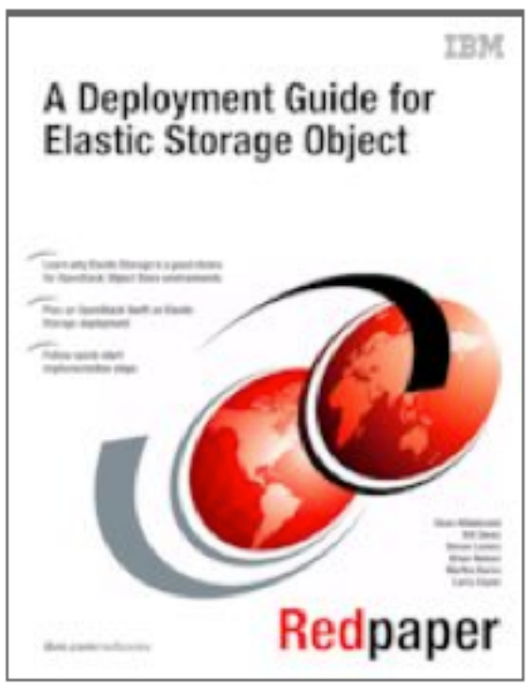
Overview dell'infrastruttura



IBM presenta lo stesso schema

A Deployment Guide for Elastic Storage Object

An IBM Redpaper publication



View online

Download PDF (0.9 MB)

Get Adobe® Reader®

Download EPUB (0.9 MB)

for e-book readers

Download on iBookstore (FREE)

Read in Google Books (FREE)

More options

Discuss this paper (0 comments)

→ Tips for viewing

→ Permanent link

Profile

Publish Date
26 September 2014

Last Update
28 September 2014

Rating: ★★★★★☆
(based on 1 review)

→ Rate this paper

Author(s)

- Pubblicato 26 settembre 2014
- http://www.redbooks.ibm.com/abstracts/redp5113.html?Open

Infrastruttura di test

- Interamente virtuale, ospitata su un server della installazione Ovirt di produzione.
  - Volumi presi da fette LVM di raidset acceduti via SAN
- Tre nodi con ruoli di servizio per il cluster GPFS, incluso ruolo di NSD server
  - un disco locale per nodo, piu' un disco shared
- Cloud nodes (visti prima)

Configurazione di GPFS

- Creazione del cluster, degli NSD, del file system
  - default replica per metadati:3
  - default replica per dati: 1
  - opzione –filesetdf (in caso di abilitazione di quota sul fileset, df su una dir del fileset fornisce i limiti di quota del fileset stesso e non i valori del file system)
- Creazione di un independent fileset
  - i-node space dedicato
  - data placement policy limitate al fileset
- Creazione nel fileset di directory dedicate a glance, cinder, nova. Un fileset dedicato per swift.

Glance

- La configurazione di GLANCE per l'utilizzo di GPFS consiste unicamente nella configurazione del placement:
  - # service openstack-glance-api stop
  - # service openstack-glance-registry stop
  - # openstack-config --set /etc/glance/glance-api.conf \
    DEFAULT filesystem_store_datadir \
    /gpfs/fs1/openstack/glance/images
  - # service openstack-glance-api start
  - # service openstack-glance-registry start

CINDER: GPFS driver

- GPFS implementa un driver per CINDER a partire dalla release 3.5, ufficiale a partire dalla 4.1
- La configurazione di CINDER richiede di specificare alcuni parametri sul nodo che esegue il *volume service*
  - volume_driver = cinder.volume.drivers.ibm.gpfs.GPFSDriver
  - gpfs_image_dir (no default): dove GLANCE mette le immagini
  - gpfs_image_share_mode (copy | copy_on_write): se si utilizza GPFS per glance
  - gpfs_mount_point_base (no default)
  - gpfs_storage_pool (system)
  - gpfs_sparse_volume (true)
  - gpfs_max_clone_depth (0)

Cinder (cont.)

- Possibilita' di definire parametri dei volumi al momento della creazione tramite specifica di "--metadata":
  - fstype, fslable: formatta il volume e lo etichetta
  - data_pool_name, replicas, dio: attributi del file sottostante
  - write_affinity, block_group_factor, write_affinity_failure_group: caratteristiche legate ad FPO
    - alcune (tutte?) non sono modificabili dopo la creazione

CINDER: multiple backend

- Cinder supporta diversi backend tramite la definizione di volume_type. Si possono definire diversi volume type con diverse caratteristiche.
- Definire i volume_type in cinder.conf:
  - enabled_backends = GPFS,GPFS-gold
  - [GPFS]
  - …
  - gpfs_storage_pool = system
  - gpfs_mount_point_base = /gpfs/…/cinder
  - volume_backend_name = GPFS
  - [GPFS-gold]
  - …
  - gpfs_storage_pool = pool1
  - gpfs_mount_point_base = /gpfs/…/cinder/gold
  - volume_backend_name = GPFS-gold

CINDER: volume type

- Definire il volume type name ed lo storage backend al volume type
  - # cinder type-create gpfs
  - # cinder type-key gpfs set volume_backend_name=GPFS
  - # cinder type-create gpfs-gold
  - # cinder type-key gpfs-gold set \
    volume_backend_name=GPFS-gold

- Creazione di volumi
  - # cinder create --volume-type gpfs-gold …

NOVA

- Anche per NOVA, come per GLANCE, la configurazione si limita alla regola di placement delle immagini effimere
  - La configurazione va eseguita sui compute node
  - Permette la migrazione on-line delle VM
  - I compute node devono essere client del cluster GPFS

  - # service openstack-nova-compute stop
  - # openstack-config --set /etc/nova/nova.conf \
    DEFAULT instances_path \
    /gpfs/fs1/openstack2/nova/instances
  - # openstack-config --set /etc/nova/nova.conf \
    libvirt disk_cachemodes \
    "file=writeback”

  - # service openstack-nova-compute start

SWIFT

- Ogni nodo e' client del cluster GPFS, quindi ogni nodo vede tutti i dati
- Non e' necessario:
  - separare le funzioni di proxy da quelle di storage node
  - configurare swift replication per oggetti: si demanda la replica a GPFS
  - eseguire ribilanciamento dei dati nel ring
- Swift rings non sono device, ma "virtual device" che corrispondono a directory dentro il fileset usato da swift
  - ogni virtual device dell'object ring e' aggiunto al nodo localhost: ogni nodo puo' accedere ad ogni volume
  - per evitare access contention e' opportuno definire piu' di un virtual device (suggerito: 10 per storage node)

SWIFT rings

- L'object ring non usa device, ma "virtual device" che corrispondono a directory dentro il fileset usato da swift
  - ogni virtual device dell'object ring e' aggiunto al nodo localhost: ogni nodo puo' accedere ad ogni volume
  - per evitare access contention e' opportuno definire piu' di un virtual device (suggerito: 10 per storage node)
- Container ed account ring sono inseriti nel modo canonico, assegnandoli ai nodi in modo bilanciato. Questo si realizza assegnandoli agli indirizzi IP espliciti dei nodi

SWIFT roles

- Data l'equivalenza dei nodi, i servizi di swift devono essere eseguiti su tutti i nodi, con l'eccezione di
  - openstack-swift-object-replicator: solo su un nodo
    - non fa replica (swift viene configurato con replica 1) ma fa cleanup di file corrispondenti ad oggetti rimossi ed erroneamente rimasti sul fs; serve solo su un nodo
  - openstack-swift-object-updater: solo su un nodo
    - fa update dei container listing di object che non sono stati caricati con successo, o cancellati senza update del listing: serve solo su un nodo
  - openstack-swift-object-auditor: disabilitato
    - verifica le checksum degli oggetti e delle repliche; conviene demandare la funzione a GPFS, tramite ad esempio funzione equivalente di GPFS Native RAID

SWIFT configration

- Una guida dettagliata e' presente nel Red Book IBM "A Deployment Guide for Elastic Storage Object"
- Il documento contiene una serie di script per automatizzare l'installazione
- Considerazione:
  - non abbiamo avuto il tempo di valutare l'impatto dei valori di parametri di configurazione suggeriti dal manuale
  - verificata solo la funzionalita'