GPFS Advanced Features

Alessandro Brunengo
Mirko Corosu
INFN-Genova

Contenuto

- Storage pool
- Policy
- Tiering
- Object service: advanced features

Storage pool

Storage pool

- Gli storage pool sono collezioni di dischi di un file system
  - attraverso gli storage pools e' possibile partizionare i device utlilizzati dal file system in gruppi
  - unitamente alle policies, e' possibile realizzare operazioni di collocazione, movimentazione, rimozione tra storage pool diversi
- Esistono due tipi di storage pool
  - internal: pensati per lo storage on line; raggruppano i dischi veri e propri, volumi fisici gestiti direttamente da GPFS
  - external: pensati per il near line storage, o per l'archiviazione; questi sono oggetti gestiti da un applicativo esterno a GPFS (ad es. TSM)
    - GPFS mette a disposizione strumenti per definire una interfaccia tra GPFS e l'applicativo
    - GPFS non gestisce gli oggetti specifici che memorizzano i dati (dischi, tape), ma solo la movimentazione dei dati da e verso lo stprage pool esterno

Utiizzo degli storage pool

- L'appartenenza ad uno storage pool e' una caratteristica di ogni file
  - tutti i blocchi dei dati di un file risiedono nello stesso pool
  - e' possibile cambiare pool di appartenenza di un file tramite policy (che vedremo dopo)
- Usando gli storage pool ed opportune policy, e' possibile
  - configurare tiering spostando in modo automatico dati in pool piu' o meno performanti in base a vari attributi del file, tra cui:
    - data di ultimo accesso al file
    - frequenza di accesso ad un file
  - assegnare storage performante a particolari utenze
    - utilizzando policy su fileset, ad esempio
  - migliorare le prestazioni e l'affidabilita'
    - separando in pool diversi dischi lenti e veloci
    - contenendo la perdita dei dati in caso di failure
    - limitando l'impatto di un RAID rebuild sulle prestazioni del solo pool

Internal storage pool

- In un file system possono essere definiti fino a 8 storage pool differenti
  - il pool system, che deve esistere, e contiene i metadati e le strutture del file system
    - puo' anche contenere dati
  - il pool system.log, opzionale, che contiene i system recovery logs
  - gli user pool, che possono contenere solo dati
- L'appartenenza di un NSD ad un pool e' un parametro dello stanza file con cui si inserisce un disco nel file system (parametro "*pool*")
  - se il pool non esiste, viene creato
  - il pool viene rimosso quando l'ultimo disco di quel pool viene rimosso dal file system

Storage pool "system"

- Lo storage pool system contiene i metadati
- Opzionalmente puo' contenere anche user data
  - cosa un disco possa ospitare (dati/metadati/entrambi) e' un differente attributo del disco (parametro "usage" dello stanza file)
  - se un disco nello storage pool system puo' contenere dati, implicitamente lo storage pool system conterra' anche dati
  - se un disco puo' contenere metadati, deve appartenere allo storage pool system
- Il file system non puo' essere creato se non ha almeno un disco nel pool system
- La block size del pool system puo' essere diversa dalla block size del file system
  - solo se i dischi nel pool system hanno usage = metadataOnly

Storage pool "system.log"

- E' possibile creare un pool di nome system.log
  - questo pool contiene solo i system recovery logs (journal data)
- E' opzionale, se usato questo pool ha la stessa block size del pool system
- E' consigliato
  - disporre di un numero sufficiente di dischi da supportare il replica factor dei metadati
  - utilizzare dischi di performance almeno uguale a quelli del pool system
- E' pensato per alte prestazioni in configurazioni particolari, per tipologie di accesso particolari
  - high availability write cache

Pool per user data

- Gli user storage pool sono opzionali (max 7 per file system)
  - vengono automaticamente creati quando si inserisce nel file system un disco il cui attributo pool di appartenenza specifica uno storage pool inesistente
  - vengono rimossi automaticamente quando viene rimosso dal file system l'ultimo disco che vi appartiene
- Gli user storage pool possono contenere solo blocchi di user data
- Gli user pool possono avere block size diversa da quella del pool system
  - ma devono avere tutti la stessa block size (parametro –B della creazione del file system)

Visualizzazione degli storage pool

- Per visualizzare gli storage pool definiti in un file system, eseguire il comando:

  # mmlsfs fs1 -P

- Il comando mmdf mostra l'occupazione del fiel system per ciascun pool
- E' possibile visualizzare e modificare il pool di appartenenza di un file
  - mmlsattr, mmchattr

Policy

Policy e policy rules

- Una policy e' un insieme di regole (policy rules) che descrive il ciclo di vita dei dati in base ad attributi del file
- Una policy rule e' uno statement (a sintassi SQL-like) che dice a GPFS cosa fare dei dati di un file di uno specifico pool se sono soddisfatti determinati criteri
- Sono definite due tipologie di policy
  - placement policy: definisce il collocamento del file alla sua creazione
  - management policy: definiscono
    - spostamento (migration policy)
    - cancellazione (deletation policy)
    - compressione
    - encryption
- Vedremo solo esempi di placement policy e migration policy

Placement policy

- Definisce in quale pool mettere i dati di un file alla sua creazione
- Sintassi:

  RULE '*name*' SET POOL '*pool-name*'
  [FOR FILESET ('*fileset-name*')]
  [REPLICATE (*data-replicaton*)]
  [LIMIT (*percentage*)]
  [ACTION (SQL-expr)]
  [WHERE SQL-epr]

- se il file appartiene al fileset *fileset-name (opz.)*, e il limite di occupazione del pool e' inferiore a *percentage* (opz.), e l'espressione WHERE e' soddisfatta (opz.), il file:
  - viene creato nel pool *pool-name*
  - viene applicato override del default data replica factor (opz.)
  - viene eseguita l'ACTION (opz.)

Placement policy (cont.)

- WHERE e' seguita da una espressione SQL che, se falsa, la policy non viene applicata. Esempio:

  WHERE NAME LIKE '%.root'

  - WHERE in placement policy puo' usare solo NAME, USER_ID, GROUP_ID
- ACTION nella placement policy e' seguita solitamente da una funzione che definisce un extended attribute
- Esempio pratico e semplice di placement rule:

  RULE 'gold' FOR FILESET ('fast_fs') SET POOL 'gold'
  RULE 'default' SET POOL 'basepool'
- Attenzione: fare in modo che la placement policy abbia sempre una soluzione di default da applicare

Placement policy (cont.)

- Per applicare la placement policy si deve creare un file con le policy rules, ed eseguire il comando:

  # mmchpolicy <dev> <file-name>

- Per vedere la placement policy di un filesystem:

  # mmlspolicy <dev> -L

- La policy di default e'

  RULE 'DEFAULT' SET POOL 'system'

  e si restora con il comando

  # mmchpolicy <dev> DEFAULT

Migration policy

- Le migration policy dicono a GPFS cosa fare dei dati di un file se sono soddisfatti certi criteri
- Le migration policy possono definire criteri in base a tutti gli attributi di un file (perche' gia' esiste), tra cui
  - date (access, modification,…)
  - heat (frequenza di accesso)
  - presenza di attributi estesi
  - se il file e' un clone
  - …
- I criteri possono anche riguardare l'occupazione di un certo storage pool
  - migrazione automatica per fare spazio

Migration policy (cont.)

- La sintassi e' simile

```
RULE 'name' MIGRATE
  [FROM POOL 'src-pool-name] …
  TO POOL 'dest-pool-name'
  [FOR FILESET ('fileset-name')] …
  WHERE SQL-expr
```

- Se sono soddisfatti i criteri, i dati del file verranno migrati

Applicazione delle management policy

- Una management policy viene definita creando un file con le policy rules (che possono essere migrate, delete, o altro tipo di policy)
- Si deve quindi eseguire il comando:

  # mmapplypolicy <dev> -P <policy-file>

- L'operazione viene operata in tre fasi
  - selezione dei candidati in base alle regole
  - selezione tra i candidati in base a pesi o soglie
  - esecuzione della policy sui file selezionati
- La management policy non viene "installata", ma eseguita
  - se si vuole eseguire ciclicamente una policy la si deve mettere in cron, o legare agli script di callback che GPFS esegue in occasione di determinati eventi (non vedremo le callback)

Tiering

Tiering

- Utilizzando storage pool e policy e' possibile configurare un vero tiering automatico dei dati di un sistema di storage basato su Spectrum Scale
  - placement gold per applicazioni importanti
  - migrazione bronze per dati raramente acceduti o viceversa
- Le policy gestiscono anche la movimentazione da e verso pool external (tape), implementando un completo HSM

Supporto del tiering per OpenStack

- Il tiering automatizzato di GPFS puo' essere utilizzato anche sui volumi usati da Glance, Cinder, Nova e Swift
  - in modo trasparente ai quattro servizi
- Il driver di GPFS per Cinder supporta il parametro gpfs_storage_pool
  - il pool cosi' definito viene utilizzato per il placement dei volumi creati da Cinder su un determinato backend
- E' quindi possibile implementare una regola per il placement differenziato:
  - si definiscono due o piu' backend di tipo gpfs, sulla stessa directory (gpfs_mount_point_base), ma con storage pool diverso
    - ricordare di abilitarli in cinder.conf
  - si definiscono due (o piu') volume type in Cinder
  - ciascun volume type si aggancia al relativo backend tramite l'attributo volume_type_name
  - quando si crea un volume, a seconda del volume type scelto, il volume sara' creato su uno storage pool differente

Object service: advanced features

Object service capabilities

- L'object service di Spectrum Scale dispone delle seguenti capabilities:
  - file-access (unified file and object access)
  - multi-region (multiregion object deployment)
  - s3 (Amazon S3 API emulation)
- Queste capabilities devono essere abilitate e poi configurate
- I dettagli per la configurazione di questi servizi sono non sufficientemente banali e vengono lasciati alla consultazione dei manuali
  - cercheremo di vedere domani il multi-region

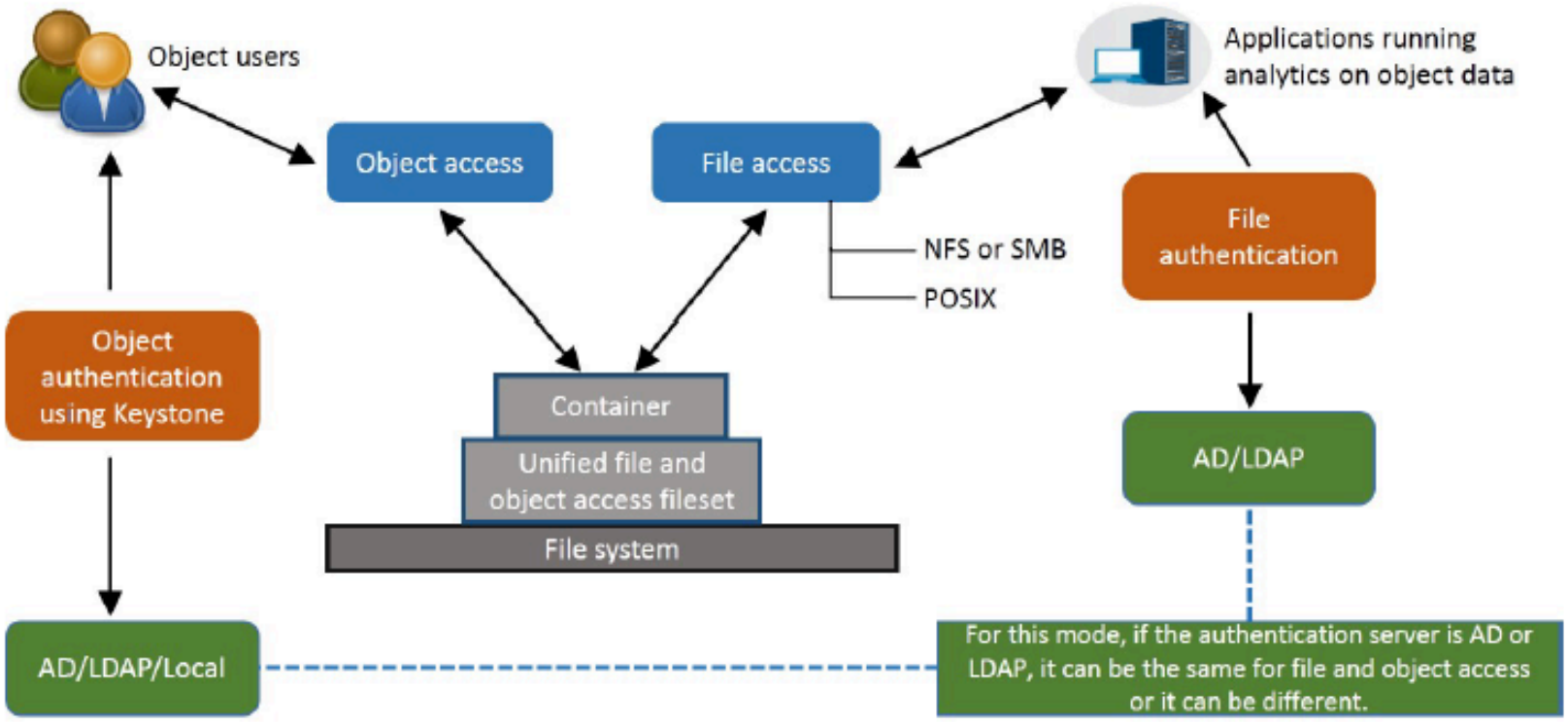
Unified file and object access

- Questa funzionalita' permette di
  - accedere a dati inseriti nell'object service tramite protocollo file
    - direttamente da un client del cluster GPFS, o tramite i protcolli NFS o SMB
    - definisce criteri di naming particolari (e ricostruibili) per il path dei file rappresentanti object
  - accedere a dati inseriti tramite interfaccia file (posix/NFS/SMB) attraverso il protocollo object
    - tramite un service (objectizer) che gira ciclicamente per creare oggetti da file

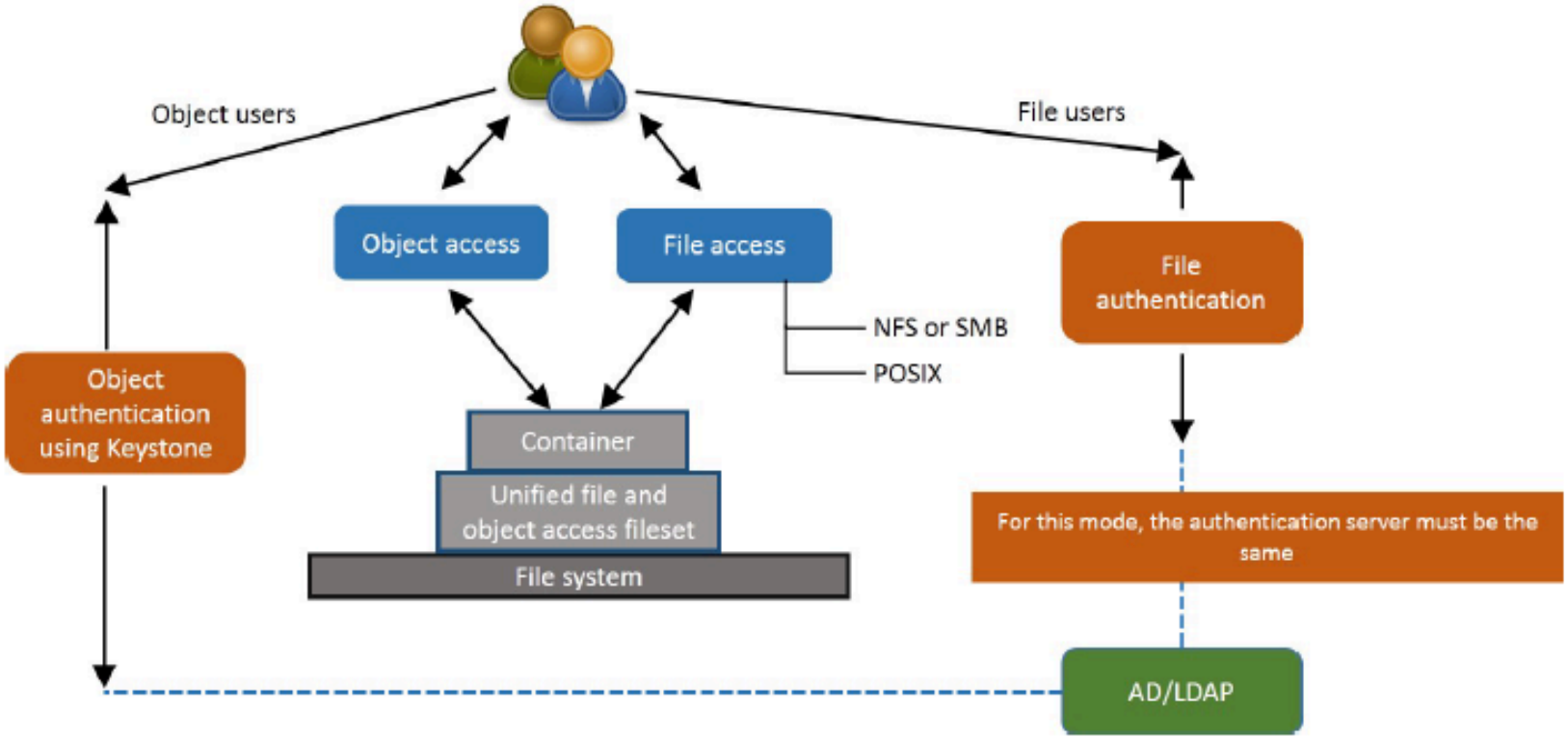
Identity management mode

- Nella configurazione del file-access si possono utilizzare due modelli di identity management:
  - local_mode: l'identita' di file e oggetti non e' unificata
    - i file corrispondenti agli oggetti sono salvati su disco con ownership dello user (unix) swift
    - gli oggetti creati tramite objectizer saranno disponibili via object interface secondo gli ACL (Swift) del container di appartenenza
  - unified_mode: l'identita' tra file e oggetti e' unificata
    - richiede che la userauth dei service file e object sia configurata in modo identico
    - in questo modo la ownership di file e oggetti viene mantenuta coerente
    - ricordarsi pero' che gli ACL file e gli ACL Swift sono diversi ed indipendenti
    - questa modalita' e' supportata solo per userauth via AD o LDAP

local_mode id management



unified_mode id management



Use case per il file-access

- analisi analitiche del contenuto dell'object storage accedendo agli oggetti tramite interfaccia posix
  - questo rende la procedura di analisi molto piu' efficiente, e non incide sull'object export
    - traffico dati solo via NSD server
- data ingestion via file
  - applicazioni di analisi che possono migrare a soluzioni basate su object access
    - mantengono l'accesso posix (legacy)
    - ottengono in modo efficiente l'accesso via object interface dei dati gia' presenti sul file system
- in caso di unified_access, il mantenimento di ownership permette di applicare GPFS policy UID/GID based

S3 API emulation

- L'object service di Spectrum Scale supporta l'accesso agli object tramite interfaccia S3
- L'interfaccia e' supportata solo se si utilizza l'autenticazione tramite internal keystone
  - ma se il keystone esterno la supporta, dovrebbe funzionare
- Per il suo utilizzo si deve
  - abilitare la capability s3
  - modificare attraverso il comando

    # mmobj config change –ccrfile …

    i file keystone-paste.ini e proxy-server.conf
  - configurare le credenziali EC2
    - e' necessario creare una coppia di access/secret per ogni coppia di user/project
    - si fa tramite comando openstack credential create –type ec2 ...

Storage policy per object storage

- Spectrum Scale object supporta l'esecuzione di policy per implementare funzionalita' (due al momento) sullo storage dedicato all'object service
  - file-access: gia' visto, viene implementato tramite policy
  - compression (vediamo questo esempio)
- Una object policy viene implementata associando un container ad una storage policy, la storage policy ad un fileset, e su questo fileset viene eseguita la funzionalita'
  - objectizer e gestione dei ring per file-access
  - compressione/decompressione per la compression policy

Implementazione della compression per l'object storage

- Lo scopo e' creare un meccanismo che permetta di applicare la compressione agli oggetti contenuti in certi container
- Si crea una object policy:

  # mmobj policy create <*policy-name*> --enable-compression --compression-schedule "MM:hh:dd:ww"

  - viene creato un fileset con nome *policy-name*, montato sul filesystem usato dall'object service
  - si deve configurare quando la compressione degli oggetti caricati sui container venga operata
  - la definizione riguarda giorno (di settimana e anno), ora e minuto. Ad ogni ricorrenza il processo di compressione viene applicato al fileset
- Ogni container associato a questa storage policy verra' salvato in questo fileset

Associare container a storage policy

- Per associare un container ad una data storage policy, e' necessario specificare l'attributo X-Storage-Policy col valore del nome della policy:

  # swift post test1 –H "X-Storage-Policy: *policy-name*"

- Questo parametro non puo' essere in seguito modificato
- E' possibile visualizzare lo stato di compressione di un file tramite il comando

  # mmlsattr –L <file>

- Ricordarsi che upload e download di file compressi comporta una perdita di prestazioni

Esercitazione

- https://wiki.ge.infn.it/calcolo/index.php/Corso_Cloud_Storage_Es6