Object Storage Quality of Service

Julian Satran - IBM
Mallikarjun Chadalpaka - HP
David Nagle – Panasas
What is Quality of Service?

A guarantee of:

- **Performance** (I/O operations/second, rate, latency and jitter)
  - Usually associated with a specific usage instance (short term, tactical)

- **Availability** (time to first-access)
  - Usually associated with usage history (long term, strategy)

- **Reliability** (resiliency to faults, recovery time)
  - Associated to content (static)

We will talk about performance
History

◉ The CMU papers about Object Storage suggest a session based QoS model
  ◇ Open and Close operations delineate a session
  ◇ A session has a QoS associated with it (presumably at open)

◉ An initial draft of an OSD standard has similar constructs (Open and Close) with the intent of accommodating QoS later

◉ The session model introduces state

◉ With Object Stores having very large numbers of objects accessed by many clients the amount of session state to be maintained was deemed excessive (unlikely to be needed for QoS)

◉ Session where dropped

◉ We decided to follow the networking model and attempt to handle QoS by “marking” access requests as belonging to an access category and optionally adding QoS hints to commands
Object Store – System Architecture

All operations are secured by credentials
Security achieved by cooperation of:
- Security Function – authenticates requestor and capability
- X Manager - Authorizes and generates capability
- ObS - validates credentials that a host presents.

Credential is cryptographically hardened
- ObS and the security function share a secret

Allow non-trusted clients to sit in the SAN

Allow shared access to storage without giving clients access to all data on a volume
Accessing an Object with QoS

1. Request Object access credential
2. Return Object access credential
3. Request QoS credential (token) – can be used for any object on an ObS
4. Return QoS credential
5. I/O request with Access + QoS credential
6. I/O data
Object Store added to a SAN FS

added:
- SAN security
- QoS
- Scalability
- Manageability
QoS token/capability

- Encodes enough information to define a workflow (an aggregate)
  - Limits the amount of state an ObS engine has to maintain
- Has to be associated with a partition or root to enable invalidation
- Association with a Service Level Agreement can be
  - Implicit (encoded in the token)
  - Explicit (communicated separately to the ObS)
  - Mixed (implicit for some classes, explicit for others)
- Service level can be expressed as:
  - Quantitative performance requirements (e.g., I/O rate or bandwidth)
  - Relative performance (differentiated service, part of total resources)
- Marking and traffic shaping has to be performed only on the endpoints
Some assumptions on QoS token standardization

- Request/Response shaping is done through queuing
- Request rejection can be done only in the extreme to protect against DOS attacks
- QoS association with ObS access may introduce reordering as a side effect. The effects of reordering have to be carefully examined. A flow encoding in the QoS token may be part of the solution
- QoS token maps into a response behavior that has to be standardized
- A QoS standardization should specify ObS and network behavior not implementation
- Mechanisms to surface QoS to applications
- What relief (e.g., buffering/caching) to surface in token (e.g., non-cached flow)
- Backward compatibility
QoS token – straw man structure

✧ A string with the following fields:
  ◇ Class type
    ◇ Part-of-total
    ◇ Rate
      ◇ Assured (defined rate is associated with flow delivery at deadline with high priority)
      ◇ Minimum (minimum rate assured defined for flow)
      ◇ Minimum and Maximum (minimum and maximum rate defined for flow)
  ◇ Part of total guaranteed number (n means 1/n)
    ◇ 0 – best effort etc.
  ◇ Relief flags – e.g., cache guarantee
  ◇ Flow ID.
QoS/SLA engineering

- An application/OS implements its own QoS goal by:
  - Allocating and managing local resources to tasks
    - Buffer Space/Cache
    - Compute Power
  - Contracting QoS with “external” providers:
    - Network
    - Storage
- Local and external resource costs are specific to an environment
- In many cases the use of local resources may reduce QoS requirements versus external providers
- Judicious use of centralized resources can be achieved by pricing
Challenges

- Request scheduling and interaction with disk scheduling
- Provide enough direct and implied information for efficient access and placement
- Interaction with interconnect network scheduling and QoS