On Asymmetrical Distributed Storage Implementation

Presented by:
Josh Salomon, Senior Principal Software Engineer, Red Hat jsalomon@redhat.com
Orit Wasserman, Senior Principal Software Engineer, Red Hat owasserm@redhat.com
Agenda

- Why symmetry is important in Software Defined Storage?
- Asymmetrical Software Defined Storage Architecture
- Ceph asymmetrical architecture
Software Defined Storage and Ceph
What is Software Defined Storage (SDS)

- Software only
- Run on commodity hardware:
  - Standard servers
  - Standard disks
  - Standard networking
- Distributed and Highly Available
- Elastic
  - Easy to expand
- Scalable
Distributed Software Defined Storage (SDS) Architecture

- Client access the storage remotely
- Data is distributed randomly across the devices
- Replicated data reads are performed from a single device and writes involve multiple devices
Ceph

https://ceph.com/en/
https://github.com/ceph/ceph
Ceph

- Open source project
- Software Defined Storage
- Runs on commodity hardware
- Commodity servers
- IP networks
- HDDs, SSDs, NVMe, NV-DIMMs, …
- A single cluster can serve object, block, and file
- Highly Available and resiliency with self healing
- Scalable
Ceph Architecture

OBJECT

RGW
S3 compatible Object storage gateway

BLOCK

RBD
Virtual Block device

FILE

CEPHFS
Distributed network file system

LIBRADOS
Low-level storage API

RADOS
Reliable, elastic, distributed storage layer with replication and erasure coding
Ceph Rados Components

**Monitor**
- Central authority for authentication, data placement, policy
- Coordination point for all other cluster components
- Protect critical cluster state with Paxos
- 3, 5 or 7 per cluster

**OSD (Object Storage Daemon)**
- Stores data on an HDD or SSD
- Services client IO requests
- Cooperatively peers, replicates, rebalances data
- 10s-1000s per cluster
CRUSH: Calculated Placement

- Get map of cluster layout (num OSDs etc) on startup
- Calculate correct object location based on its name
- Read from or write to appropriate OSD
Why symmetry is important in Software Defined Storage?

Some math and physics ideas, and how they apply to Software Defined Storage
Symmetry in Software Defined Storage (SDS) Architecture

▪ Why limit to SDS?
  ▪ Technically asymmetry is not limited to SDS
  ▪ SDS have a variety of possible configuration, hardware and components they support
  ▪ In the long run after several expansions they become asymmetric
  ▪ The use cases we are focusing on are mainly applicable for SDS systems
    ▪ Closed storage systems and appliances use similar ideas but refer to them differently
Device Symmetry

- Performance
  - IOPS/Throughput and latency
- Technology
  - Media (HDD vs SSD)
  - Protocol (SATA vs NVMe)
- Load
- Size
- Fill rate
  - The devices should be full in the same percentage (all devices should be X% full at any given time), when a single device is 100% full the system becomes 100% full
Weakest Link in the Chain Effect

- Take a system with N devices, each with M IOPS
  - Total max performance: NxM IOPS
  - A system with 100 devices with 200 IOPs will produce 20,000 IOPS
- Under some circumstances
  - Many simultaneous clients (more than devices)
  - Each work in sync mode (request, wait for response, request, etc)
  - As a result: system is fully loaded
- If one device has less performance (O < M IOPS)
  - The system performance after some time will reduce to NxO IOPS
    - As if all the devices had O IOPs
    - A system with 99 devices of 200 IOPS and one device of 100 IOPS will produce 10,000 IOPS
Advantages of Symmetry

- Easier to manage capacity
- Easier to manage performance
- Simpler to configure
- Easier to validate
Symmetrical Software Defined Storage

- So we want symmetrical system!
- Not always…

- Can we break the assumptions in such a way that we can gain from breaking the symmetry?

- Yes!!!
Asymmetrical Software Defined Storage Systems
So Why Consider Asymmetry?

- Follow the money...
  - Cloud economics
  - When life gives you lemons, make lemonade...
- Reality
  - Gradual upgrade/migration
Breaking the Symmetry

- Is it worth it?
  - Reducing cost
  - Increase reliability
- Functional break
  - Get the most out of it
Cost Analysis (AWS - Aug 2021)

- Comparing m5 and i3en instances - not apples to apples!

<table>
<thead>
<tr>
<th>Type</th>
<th>vCPU</th>
<th>Memory</th>
<th>Network BW</th>
<th>Storage</th>
<th>15 TB $/month</th>
<th>IOPs / max BW</th>
<th>Price / month</th>
</tr>
</thead>
<tbody>
<tr>
<td>m5.8xlarge</td>
<td>32</td>
<td>128 GiB</td>
<td>10 Gb/s</td>
<td>EBS only</td>
<td>$1200 - $1875+</td>
<td>45 K / 1.5 GBs</td>
<td>$1,106</td>
</tr>
<tr>
<td>i3en.6xlarge</td>
<td>24</td>
<td>192 GiB</td>
<td>25 Gb/s</td>
<td>2x7500 NVMe SSD</td>
<td>0 (included)</td>
<td>500 K / 4 GBs</td>
<td>$1,953</td>
</tr>
</tbody>
</table>

[1] - Note the network bandwidth may be the bottleneck here

- Cost comparison
  - For similar bandwidth we need 2 m5.8xl instances for every i3en.6xl instance
    - $3,412 / month (2 instances + 15TB EBS gp2) relative to $1,953 / month
  - For a similar price EBS based system has much lower bandwidth.
- Can we combine options and gain price and performance together?
Performance Analysis (AWS - Aug 2021)

- **Fact summary**
  - Local storage has 10x 4KB IOPs and ~3x the bandwidth
  - Local storage is cheaper (for consistent high load system)
  - Local storage reliability may not be in our control
  - EBS storage is much more reliable than a local storage.

- **Replicated storage facts**
  - Read is served by a single device
  - Write is served by multiple devices
Performance Analysis (AWS - Aug 2021)

- Fact summary
  - Local storage has 10x 4KB IOPs and ~3x the bandwidth
  - Local storage is cheaper (for consistent high load system)
  - Local storage reliability may not be in our control
  - EBS storage is much more reliable than a local storage.

- Replicated storage facts
  - Read is served by a single device
  - Write is served by multiple devices
  - Let the cheaper, faster but less reliable devices serve all the reads
What is the Gain

- Reads are faster
  - Faster latency and bandwidth for most workloads
- Write bandwidth is higher
  - Bandwidth of the slower storage devices is not used for read I/Os
  - No change to the latency
- Points to watch
  - Network bandwidth
  - Workload Read/Write distribution (%)
More Options

▪ Replica count
  ▪ Very reliable slower device (99.999%)  
    ▪ We may use 2 replications, one on the local storage and one on cloud storage  
  ▪ Protection from AZ failure  
    ▪ We may replicate to 2 instances of cloud storage in 2 AZs + one local storage replication on a third zone  
  ▪ Consider the data importance  
    ▪ Not all data is equal - in some cases we may compromise durability over cost  
     ▪ Especially if we can reconstruct the data easily  
▪ We have many options to balance cost, performance and reliability
Implementing Asymmetry in Ceph

▪ Primary affinity - not good enough
  ▪ The primary will always be a local device, but not guarantee that the others will never be local devices
▪ Custom CRUSH rule (based on device class)
  ▪ A bit more complex, but effective
Sample Asymmetric CRUSH Rule

```plaintext
rule local_with_EBS {
    id 3
    type replicated
    min_size 1
    max_size 10
    step take default class local
    step chooseleaf firstn 1 type host
    step emit
    step take default class EBS
    step chooseleaf firstn 0 type host
    step emit
}
```
A Different View

- So what do we have here
  - For every piece of data we have
  - One copy on a fast storage, used by all read IOs
    - One or more copies on a slower storage used mainly for write IOs
  - So it looks just like ...
A Different View

▪ So what do we have here
  ▪ For every piece of data we have
  ▪ One copy on a fast storage, used by all read IOs
    ▪ One or more copies on a slower storage used mainly for write IOs
  ▪ So it looks just like ...

▪ Full size read NVMe flash cache for the entire storage pool
Other use cases

- Phased upgrade of the system
  - Assuming you have a large system and you want to add capacity
    - What if you can't get the same devices as in your current system?
    - What if you want to upgrade technology (from older SSD to NVMe-SSD) but not at once?
Conclusions

- In some cases symmetrical storage systems are optimal
- In some cases requiring full symmetry is not feasible
- When symmetry can't be forced, we can still make the best of the configuration we have
  - Split the system into symmetrical pools
  - Combine different devices into the same pool, but route more load (reads/writes) to the faster devices and less load (writes only) to the slower devices
  - Some middle solution that plays with the load between the devices but is not all or nothing for reads
Please take a moment to rate this session.

Your feedback is important to us.