BCNET

Shared IT Services for Higher Education & Research

Conference 2018

Ceph: All-in-One Network Data Storage

What is Ceph and how we use it to backend the Arbutus cloud

A little about me, Mike Cave:

- Systems administrator for Research Computing Services at the University of Victoria.
- Systems administrator for the past 12 years
- Started supporting research computing in April of 2017
- Past experience includes:
  - Identity management
  - Monitoring
  - Systems automation
  - Enterprise systems deployment
  - Network storage management

My introduction to Ceph

It was my first day…

My introduction to Ceph

It was my first day…

Outgoing co-worker: “You’ll be taking over the Ceph cluster.”

Me: “What is Ceph?”

- Today's focus:
  - Ceph: what is it?
  - Ceph Basics: what makes it go?
  - Ceph at the University of Victoria: storage for a cloud deployment

So, what is Ceph?

What is Ceph

- Resilient, redundant, and performant object storage
- Object, block, and filesystem storage options
- Scales to the exabyte range

What is Ceph

- No single point of failure
- Works on almost any hardware
- Open source (LGPL) and community supported

Ceph Basics

Ceph Basics

- Ceph is built around, what they call, RADOS
  - R: reliable
  - A: autonomic
  - D: distributed
  - O: object
  - S: storage
- RADOS allows access to the storage cluster to thousands of clients, applications and virtual machines
- All clients connect via the same cluster address, which minimizes configuration and availability constraints

Ceph Basics

Storage Options

1. Object storage

- RESTful interface to objects
- Compatible with:
  - Swift
  - S3
  - NFS (v3/v4)
- Allows snapshots
- Atomic transactions
- Object level key-value mapping
- Basis for Cephs advanced feature set

Ceph Basics

Storage Options

1. Object storage
2. Block storage

- Expose block devices through RBD interface
- Block device images stored as objects
- Block device resizing
- Offers read-only snapshots
- Thin provisioned, by default
- Block device more flexible than object storage

Ceph Basics

Storage Options

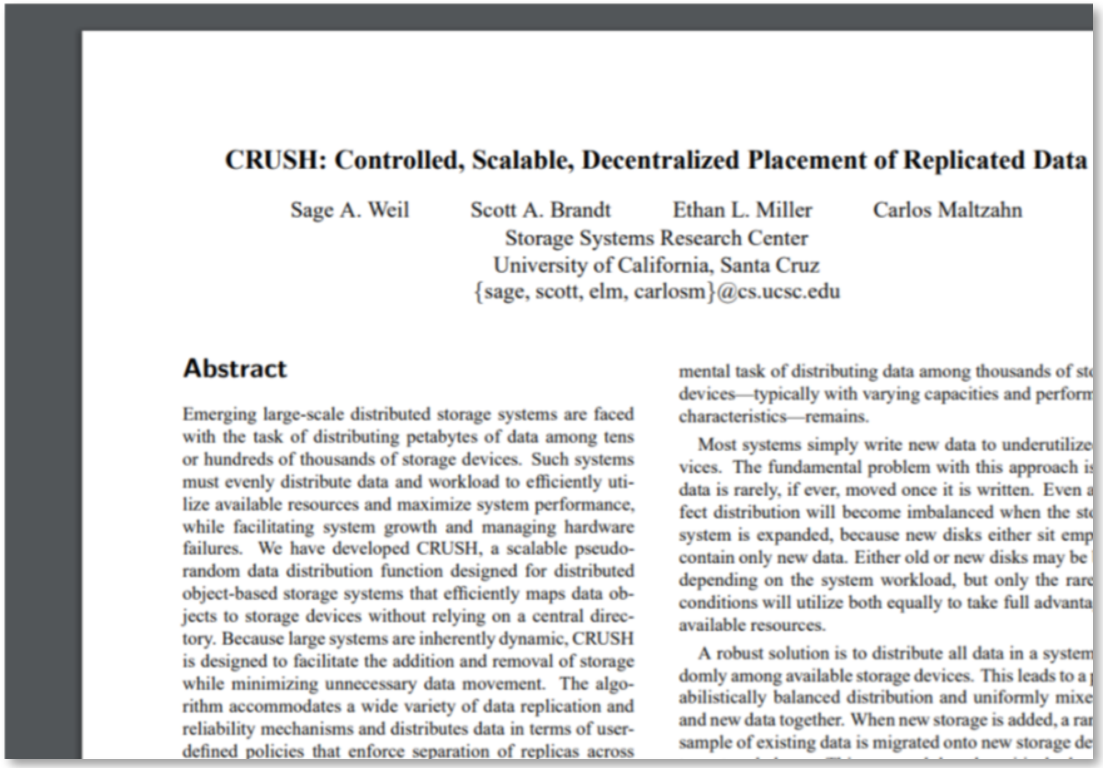
1. Object storage
2. Block storage
3. CephFS

- Supports applications that do not support object storage
- Can be mounted to multiple hosts through Ceph client
- Conforms to the POSIX standard
- High performance under heavy workloads

Ceph Basics

What is CRUSH

- As I mentioned earlier, the entire system is based on an algorithm called CRUSH



CRUSH: Controlled, Scalable, Decentralized Placement of Replicated Data

Sage A. Weil Scott A. Brandt Ethan L. Miller Carlos Maltzahn
Storage Systems Research Center
University of California, Santa Cruz
{sage, scott, elm, carlosm}@cs.ucsc.edu

Abstract

Emerging large-scale distributed storage systems are faced with the task of distributing petabytes of data among tens or hundreds of thousands of storage devices. Such systems must evenly distribute data and workload to efficiently utilize available resources and maximize system performance, while facilitating system growth and managing hardware failures. We have developed CRUSH, a scalable pseudo-random data distribution function designed for distributed object-based storage systems that efficiently maps data objects to storage devices without relying on a central directory. Because large systems are inherently dynamic, CRUSH is designed to facilitate the addition and removal of storage while minimizing unnecessary data movement. The algorithm accommodates a wide variety of data replication and reliability mechanisms and distributes data in terms of user-defined policies that enforce separation of replicas across

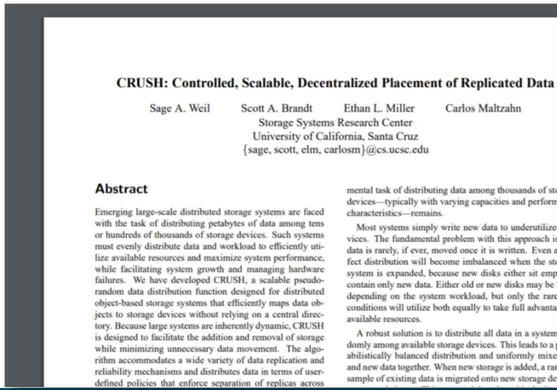
mental task of distributing data among thousands of st
devices—typically with varying capacities and perform
characteristics—remains.

Most systems simply write new data to underutilize
vices. The fundamental problem with this approach i
data is rarely, if ever, moved once it is written. Even a
fect distribution will become imbalanced when the st
system is expanded, because new disks either sit emp
contain only new data. Either old or new disks may be
depending on the system workload, but only the rar
conditions will utilize both equally to take full advanta
available resources.

A robust solution is to distribute all data in a system
domly among available storage devices. This leads to a
abilistically balanced distribution and uniformly mixe
and new data together. When new storage is added, a ra
sample of existing data is migrated onto new storage de

Ceph Basics
What is CRUSH

- As I mentioned earlier, the entire system is based on an algorithm called CRUSH
- The algorithm allows Ceph to calculate data placement on the fly at the client level, rather than using a centralized data table to reference data placement



CRUSH: Controlled, Scalable, Decentralized Placement of Replicated Data

Sage A. Weil   Scott A. Brandt   Ethan L. Miller   Carlos Maltzahn
Storage Systems Research Center
University of California, Santa Cruz
{sage, scott, elm, carlosm}@cs.ucsc.edu

Abstract

Emerging large-scale distributed storage systems are faced with the task of distributing petabytes of data among tens or hundreds of thousands of storage devices. Such systems must evenly distribute data and workload to efficiently utilize available resources and maximize system performance, while facilitating system growth and managing hardware failures. We have developed CRUSH, a scalable pseudo-random data distribution function designed for distributed object-based storage systems that efficiently maps data objects to storage devices without relying on a central directory. Because large systems are inherently dynamic, CRUSH is designed to facilitate the addition and removal of storage while minimizing unnecessary data movement. The algorithm accommodates a wide variety of data replication and reliability mechanisms and distributes data in terms of user-defined policies that enforce separation of replicas across

mental task of distributing data among thousands of st
devices—typically with varying capacities and perform
characteristics—remains.

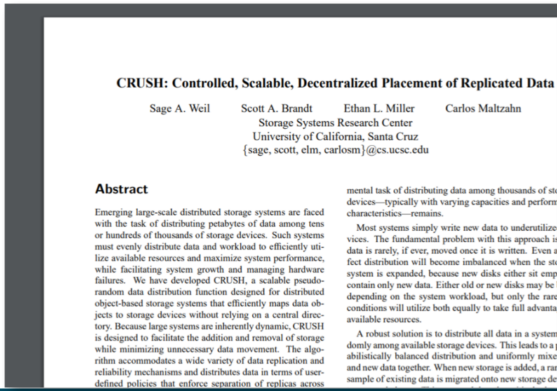
Most systems simply write new data to underutilize
vices. The fundamental problem with this approach i
data is rarely, if ever, moved once it is written. Even a
fect distribution will become imbalanced when the st
system is expanded, because new disks either sit emp
contain only new data. Either old or new disks may be
depending on the system workload, but only the rar
conditions will utilize both equally to take full advanta
available resources.

A robust solution is to distribute all data in a system
domly among available storage devices. This leads to a
abilistically balanced distribution and uniformly mix
and new data together. When new storage is added, a ra
sample of existing data is migrated onto new storage de

Ceph Basics

What is CRUSH

- As I mentioned earlier, the entire system is based on an algorithm called CRUSH
- The algorithm allows Ceph to calculate data placement on the fly at the client level, rather than using a centralized data table to reference data placement
- You do not have to worry about managing the CRUSH algorithm directly.
  - Instead you configure the CRUSH map and let the algorithm do the work for you.



CRUSH: Controlled, Scalable, Decentralized Placement of Replicated Data

Sage A. Weil Scott A. Brandt Ethan L. Miller Carlos Maltzahn
Storage Systems Research Center
University of California, Santa Cruz
{sage, scott, elm, carlosm}@cs.ucsc.edu

Abstract

Emerging large-scale distributed storage systems are faced with the task of distributing petabytes of data among tens or hundreds of thousands of storage devices. Such systems must evenly distribute data and workload to efficiently utilize available resources and maximize system performance, while facilitating system growth and managing hardware failures. We have developed CRUSH, a scalable pseudo-random data distribution function designed for distributed object-based storage systems that efficiently maps data objects to storage devices without relying on a central directory. Because large systems are inherently dynamic, CRUSH is designed to facilitate the addition and removal of storage while minimizing unnecessary data movement. The algorithm accommodates a wide variety of data replication and reliability mechanisms and distributes data in terms of user-defined policies that enforce separation of replicas across

mental task of distributing data among thousands of sto
devices—typically with varying capacities and perform
characteristics—remains.

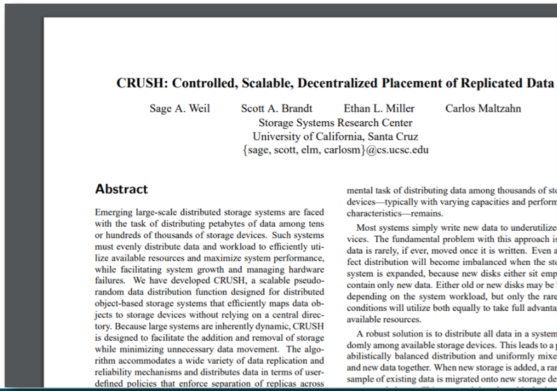
Most systems simply write new data to underutilize
vices. The fundamental problem with this approach is
data is rarely, if ever, moved once it is written. Even a
fect distribution will become imbalanced when the sto
system is expanded, because new disks either sit emp
contain only new data. Either old or new disks may be
depending on the system workload, but only the rare
conditions will utilize both equally to take full advanta
available resources.

A robust solution is to distribute all data in a system
domly among available storage devices. This leads to a
abilistically balanced distribution and uniformly mixe
and new data together. When new storage is added, a ra
sample of existing data is migrated onto new storage de

Ceph Basics

What is CRUSH

- As I mentioned earlier, the entire system is based on an algorithm called CRUSH
- The algorithm allows Ceph to calculate data placement on the fly at the client level, rather than using a centralized data table to reference data placement
- You do not have to worry about managing the CRUSH algorithm directly.
  - Instead you configure the CRUSH map and let the algorithm do the work for you.
- The CRUSH map lets you lay out the data in the cluster to specifications based on your needs
  - The map contains parameters for the algorithm to operate on
  - These parameters include
    - Where your data is going to live
    - And how your data is distributed into failure domains



CRUSH: Controlled, Scalable, Decentralized Placement of Replicated Data

Sage A. Weil Scott A. Brandt Ethan L. Miller Carlos Maltzahn
Storage Systems Research Center
University of California, Santa Cruz
{sage, scott, elm, carlosm}@cs.ucsc.edu

Abstract

Emerging large-scale distributed storage systems are faced with the task of distributing petabytes of data among tens or hundreds of thousands of storage devices. Such systems must evenly distribute data and workload to efficiently utilize available resources and maximize system performance, while facilitating system growth and managing hardware failures. We have developed CRUSH, a scalable pseudo-random data distribution function designed for distributed object-based storage systems that efficiently maps data objects to storage devices without relying on a central directory. Because large systems are inherently dynamic, CRUSH is designed to facilitate the addition and removal of storage while minimizing unnecessary data movement. The algorithm accommodates a wide variety of data replication and reliability mechanisms and distributes data in terms of user-defined policies that enforce separation of replicas across

mental task of distributing data among thousands of st
devices—typically with varying capacities and perform
characteristics—remains.

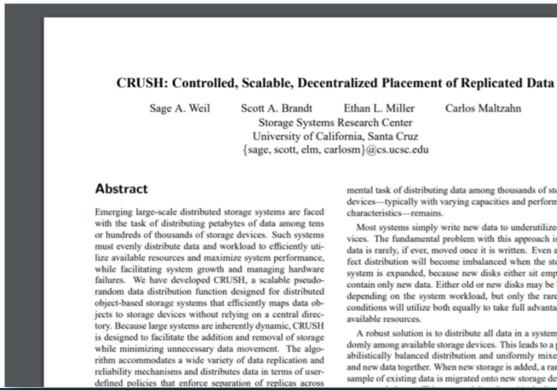
Most systems simply write new data to underutilize
vices. The fundamental problem with this approach i
data is rarely, if ever, moved once it is written. Even a
fect distribution will become imbalanced when the st
system is expanded, because new disks either sit emp
contain only new data. Either old or new disks may be
depending on the system workload, but only the rar
conditions will utilize both equally to take full advanta
available resources.

A robust solution is to distribute all data in a system
domly among available storage devices. This leads to a
abilistically balanced distribution and uniformly mixe
and new data together. When new storage is added, a ra
sample of existing data is migrated onto new storage de

Ceph Basics

What is CRUSH

- As I mentioned earlier, the entire system is based on an algorithm called CRUSH
- The algorithm allows Ceph to calculate data placement on the fly at the client level, rather than using a centralized data table to reference data placement
- You do not have to worry about managing the CRUSH algorithm directly.
  - Instead you configure the CRUSH map and let the algorithm do the work for you.
- The CRUSH map lets you lay out the data in the cluster to specifications based on your needs
  - The map contains parameters for the algorithm to operate on
  - These parameters include
    - Where your data is going to live
    - And how your data is distributed into failure domains
- Essentially, the CRUSH map is the logical grouping of the available devices you have available in the cluster



CRUSH: Controlled, Scalable, Decentralized Placement of Replicated Data

Sage A. Weil Scott A. Brandt Ethan L. Miller Carlos Maltzahn
Storage Systems Research Center
University of California, Santa Cruz
{sage, scott, elm, carlosm}@cs.ucsc.edu

Abstract

Emerging large-scale distributed storage systems are faced with the task of distributing petabytes of data among tens or hundreds of thousands of storage devices. Such systems must evenly distribute data and workload to efficiently utilize available resources and maximize system performance, while facilitating system growth and managing hardware failures. We have developed CRUSH, a scalable pseudo-random data distribution function designed for distributed object-based storage systems that efficiently maps data objects to storage devices without relying on a central directory. Because large systems are inherently dynamic, CRUSH is designed to facilitate the addition and removal of storage while minimizing unnecessary data movement. The algorithm accommodates a wide variety of data replication and reliability mechanisms and distributes data in terms of user-defined policies that enforce separation of replicas across

mental task of distributing data among thousands of sto
devices—typically with varying capacities and perform
characteristics—remains.

Most systems simply write new data to underutilize
vices. The fundamental problem with this approach is
data is rarely, if ever, moved once it is written. Even a
fect distribution will become imbalanced when the sto
system is expanded, because new disks either sit emp
contain only new data. Either old or new disks may be
depending on the system workload, but only the rare
conditions will utilize both equally to take full advanta
available resources.

A robust solution is to distribute all data in a system
domly among available storage devices. This leads to a
abilistically balanced distribution and uniformly mixe
and new data together. When new storage is added, a ran
sample of existing data is migrated onto new storage de

CRUSH

A Basic Example

A Basic CRUSH Example

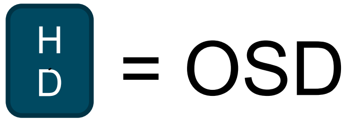
The Hardware

- Lets build a quick cluster…
- The basic unit of our cluster is the hard drive

A Basic CRUSH Example

The Hardware

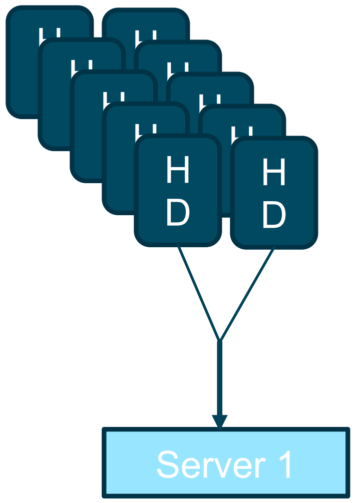
- Lets build a quick cluster…
- The basic unit of our cluster is the hard drive



A Basic CRUSH Example

The Hardware

- Lets build a quick cluster…
- The basic unit of our cluster is the hard drive
- We will have 10 OSDs in each of our servers



A Basic CRUSH Example

The Hardware

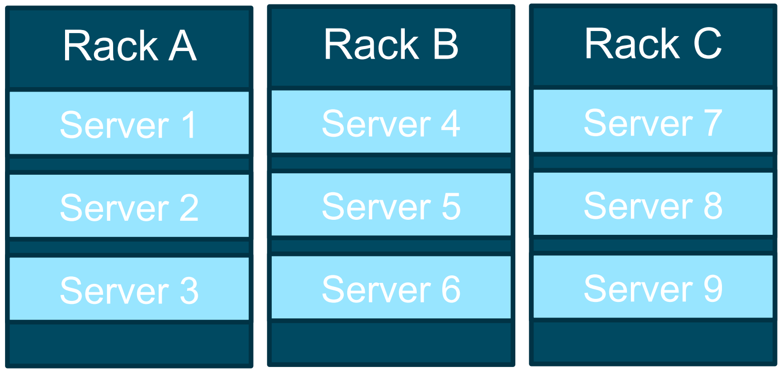
- Lets build a quick cluster…
- The basic unit of our cluster is the hard drive
- We will have 10 OSDs in each of our servers
- Add 9 servers

Server 1
Server 2
Server 3
Server 4
Server 5
Server 6
Server 7
Server 8
Server 9

A Basic CRUSH Example

The Hardware

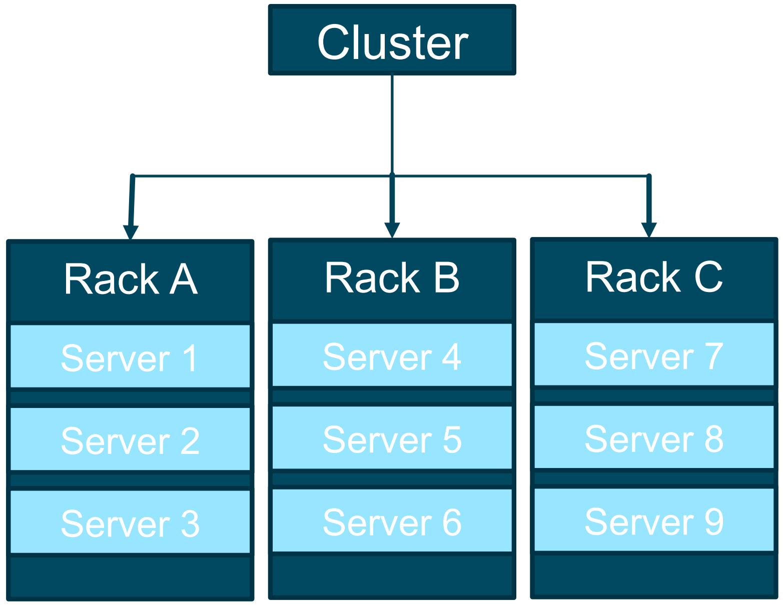
- Lets build a quick cluster…
- The basic unit of our cluster is the hard drive
- We will have 10 OSDs in each of our servers
- Add 9 servers
- Then we'll put them into three racks



A Basic CRUSH Example

The Hardware

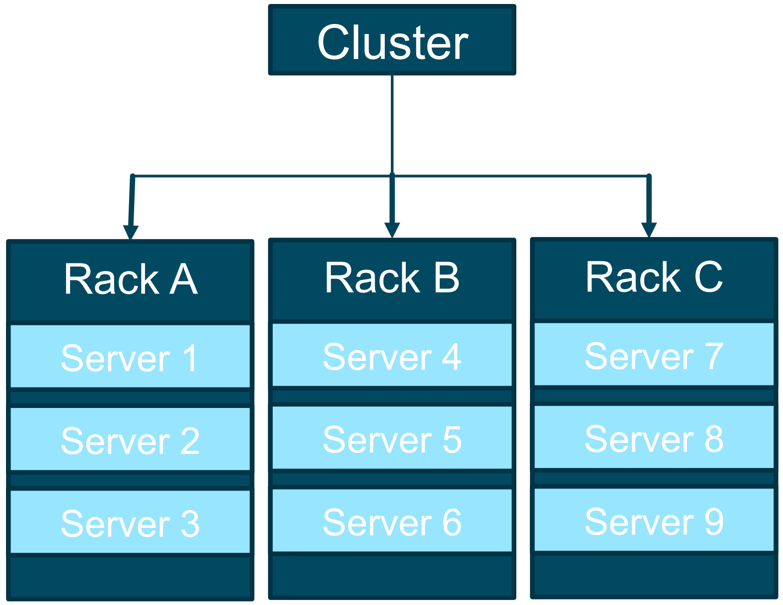
- Lets build a quick cluster…
- The basic unit of our cluster is the hard drive
- We will have 10 OSDs in each of our servers
- Add 9 servers
- Then we'll put them into three racks
- And now we have a basic cluster of equipment
- Now we can take a look at how we'll overlay CRUSH map



A Basic CRUSH Example

CRUSH Rules: Buckets

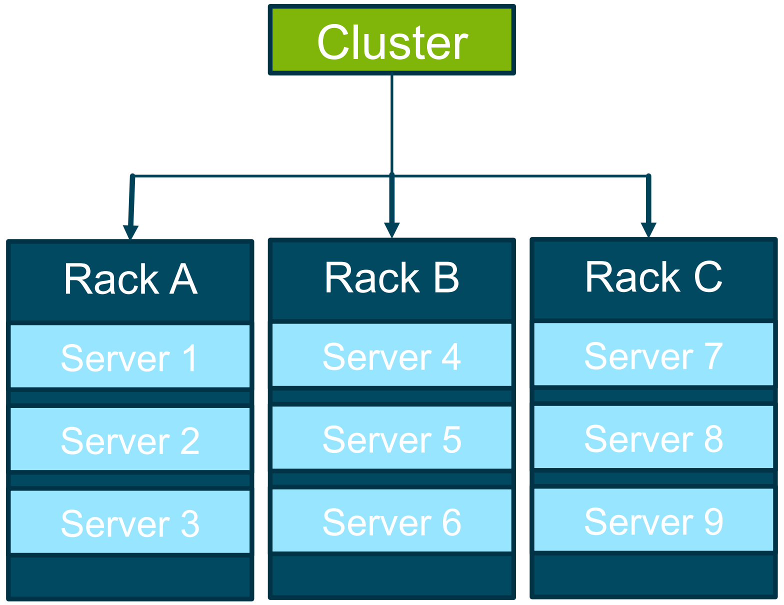
- Now we have the cluster built we need to define the logical groupings of our hardware devices into ‘buckets’ which will house our data
- We will define the following buckets:



A Basic CRUSH Example

CRUSH Rules: Buckets

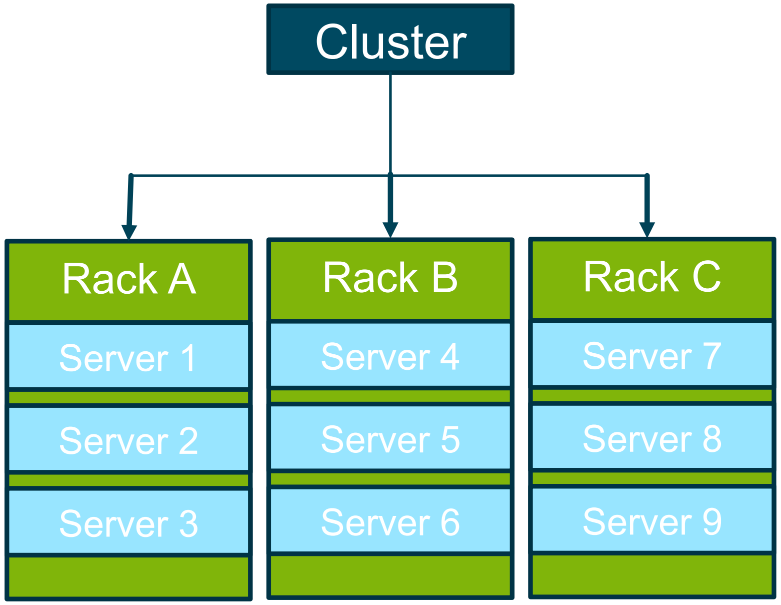
- Now we have the cluster built we need to define the logical groupings of our hardware devices into ‘buckets’ which will house our data
- We will define the following buckets:
  - Cluster - called the ‘root’ bucket



A Basic CRUSH Example

CRUSH Rules: Buckets

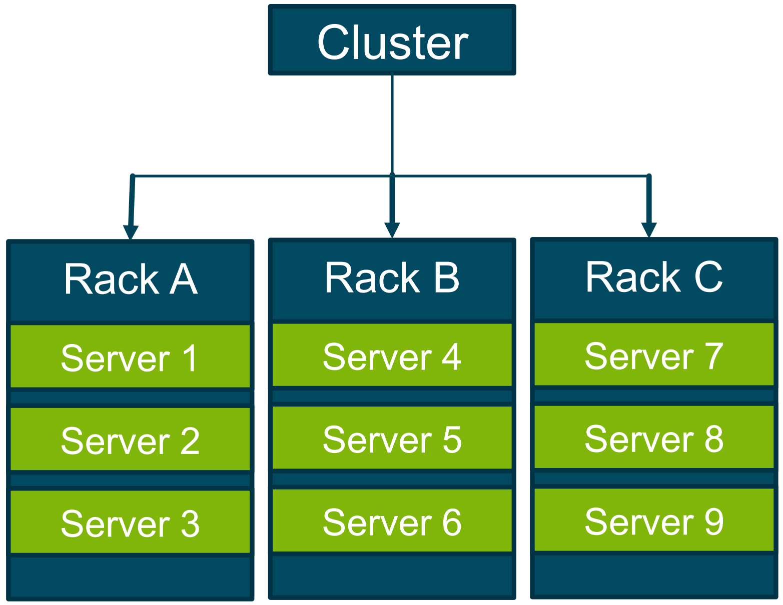
- Now we have the cluster built we need to define the logical groupings of our hardware devices into ‘buckets’ which will house our data
- We will define the following buckets:
  - Cluster - called the ‘root’ bucket
  - Rack – collection of servers



A Basic CRUSH Example

CRUSH Rules: Buckets

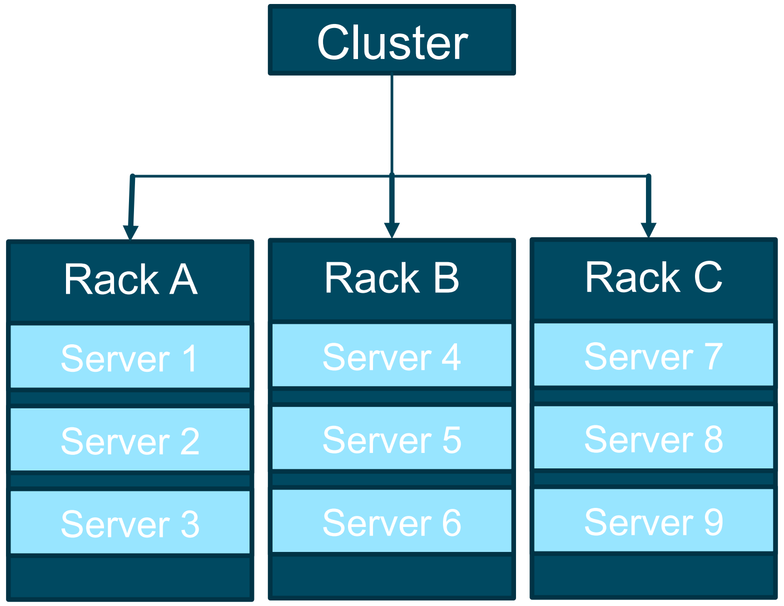
- Now we have the cluster built we need to define the logical groupings of our hardware devices into ‘buckets’ which will house our data
- We will define the following buckets:
  - Cluster - called the ‘root’ bucket
  - Rack – collection of servers
  - Server - collection of OSDs (HDs)



A Basic CRUSH Example

CRUSH Rules: Rule Options

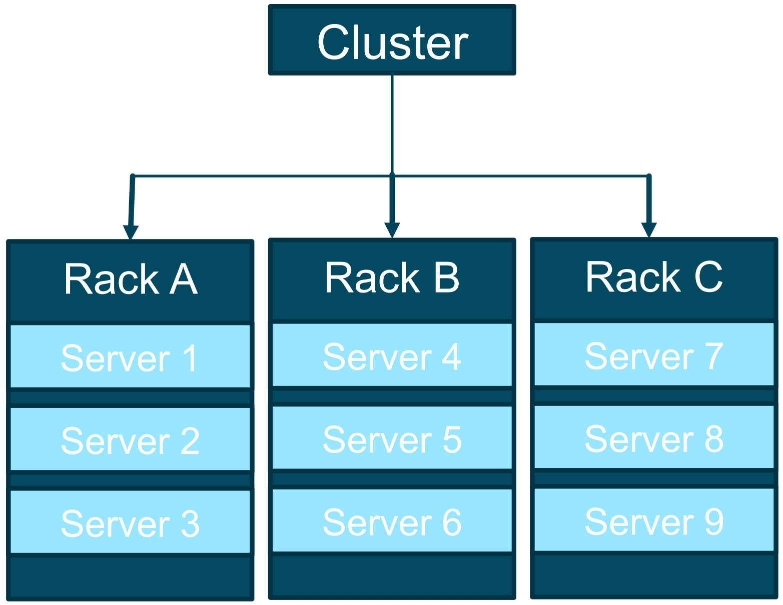
- CRUSH rules tell the cluster how to organize the data across the devices defined in the map



A Basic CRUSH Example

CRUSH Rules: Rule Options

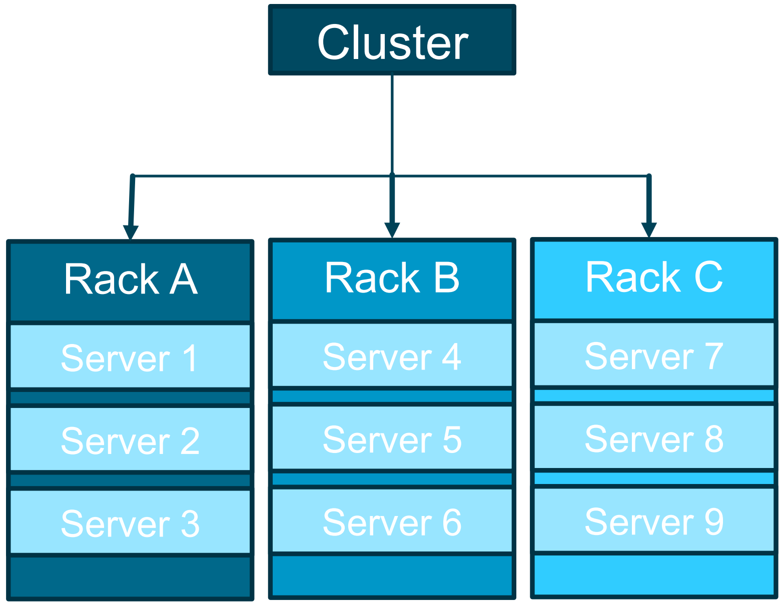
- CRUSH rules tell the cluster how to organize the data across the devices defined in the map
- In our simple case we'll define a rule called "replicated_ruleset" with the following parameters:
  - Location – root



A Basic CRUSH Example

CRUSH Rules: Rule Options

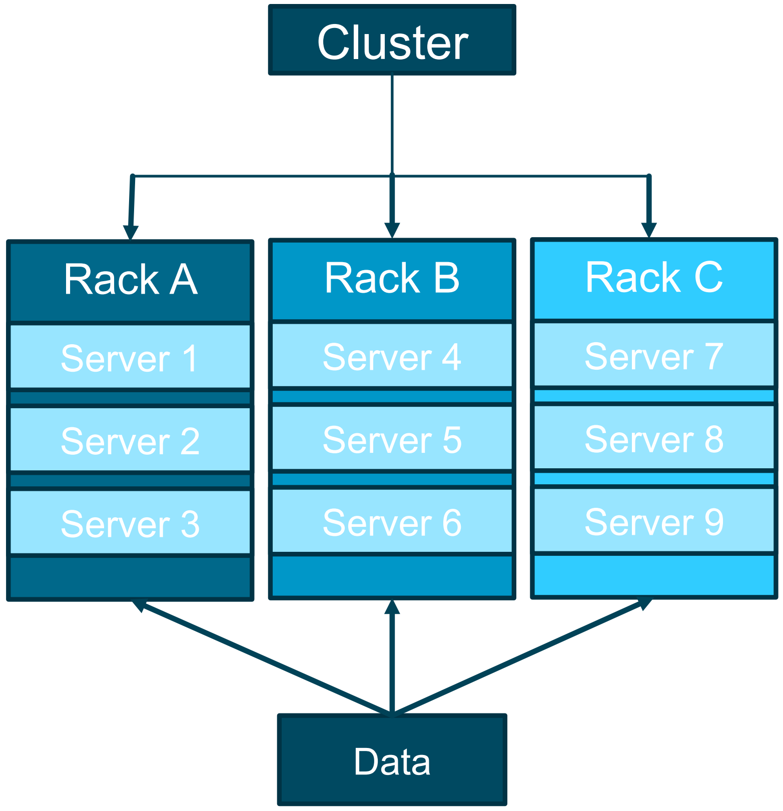
- CRUSH rules tell the cluster how to organize the data across the devices defined in the map
- In our simple case we'll define a rule called "replicated_ruleset" with the following parameters:
    - Location – root
    - Failure domain – Rack



A Basic CRUSH Example

CRUSH Rules: Rule Options

- CRUSH rules tell the cluster how to organize the data across the devices defined in the map
- In our simple case we'll define a rule called "replicated_ruleset" with the following parameters:
  - Location – root
  - Failure domain – Rack
  - Type – Replicated



A Basic CRUSH Example
Pools

- Data inside of Ceph is stored in ‘pools’
- The pool allows for specific bounds around how data is stored and who can access it

A Basic CRUSH Example

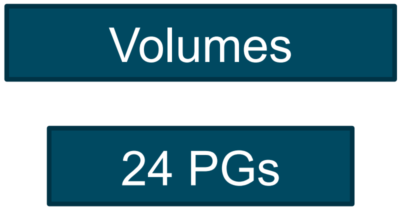
Pools

- Data inside of Ceph is stored in ‘pools’
- The pool allows for specific bounds around how data is stored and who can access it
- Some basic required:
  - Name of pool

Volumes

A Basic CRUSH Example
Pools

- Data inside of Ceph is stored in ‘pools’
- The pool allows for specific bounds around how data is stored and who can access it
- Some basic required:
  - Name of pool
  - Number of ‘placement groups’



A Basic CRUSH Example

Pools

- Data inside of Ceph is stored in ‘pools’
- The pool allows for specific bounds around how data is stored and who can access it
- Some basic required:
  - Name of pool
  - Number of ‘placement groups’
  - Storage rule
    - Minimum size - triple

Volumes

24 PGs

Replicated

A Basic CRUSH Example

Pools

- Data inside of Ceph is stored in 'pools'
- The pool allows for specific bounds around how data is stored and who can access it
- Some basic required:
  - Name of pool
  - Number of 'placement groups'
  - Storage rule
    - Minimum size - triple
  - Pool application association - rgw/rbd/cephfs

Volumes

24 PGs

Replicated

RBD

A Basic CRUSH Example

Pools

- Data inside of Ceph is stored in ‘pools’
- The pool allows for specific bounds around how data is stored and who can access it
- Some basic required:
  - Name of pool
  - Number of ‘placement groups’
  - Storage rule
    - Minimum size - triple
  - Pool application association - rgw/rbd/cephfs
  - Many pool options (class, size, cleaning, etc.)

Volumes

24 PGs

Replicated

RBD

A Basic CRUSH Example

Pools: Users

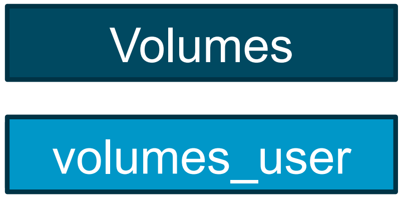
- Pool access is based on users and keys

Volumes

A Basic CRUSH Example

Pools: Users

- Pool access is based on users and keys
- You first create a user for your pool



A Basic CRUSH Example

Pools: Users

- Pool access is based on users and keys
- You first create a user for your pool
- Then assign standard POSIX permissions

Volumes

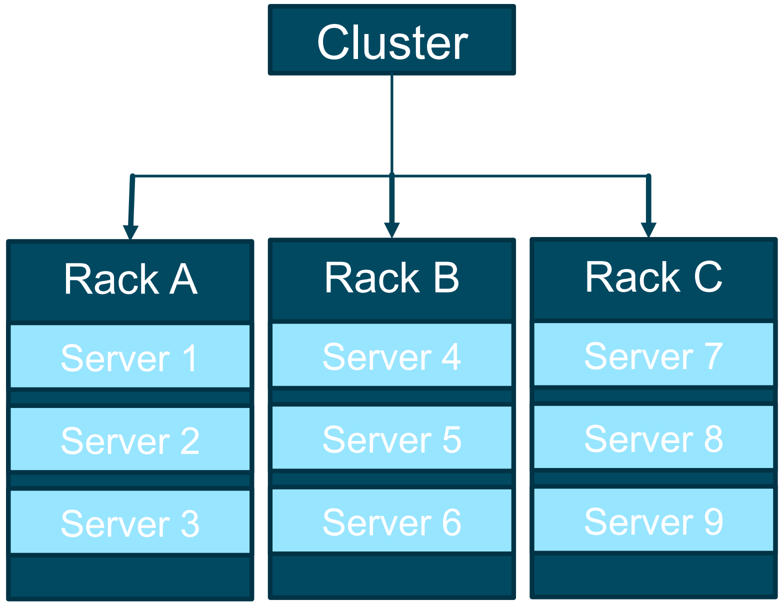
volumes_user

rwx

A Basic CRUSH Example

CRUSH Recap

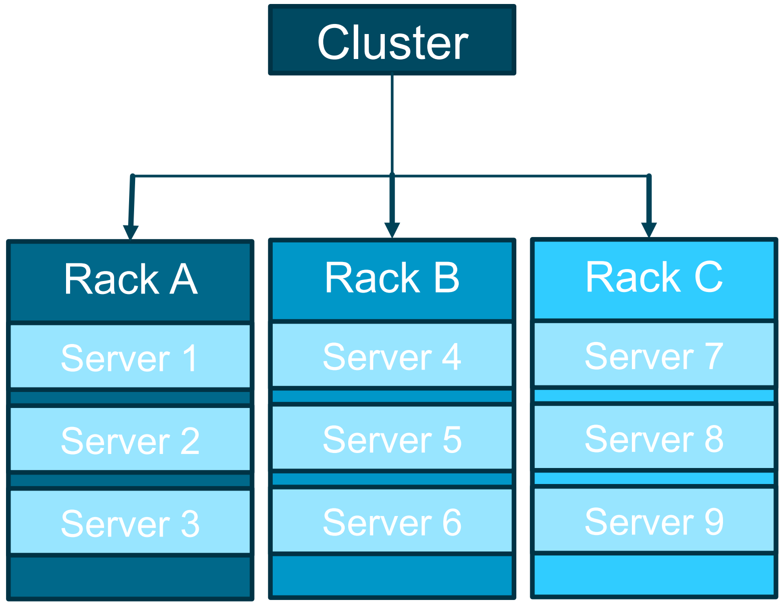
- Create the CRUSH map
  - Organize physical resources into 'Buckets'



A Basic CRUSH Example

CRUSH Recap

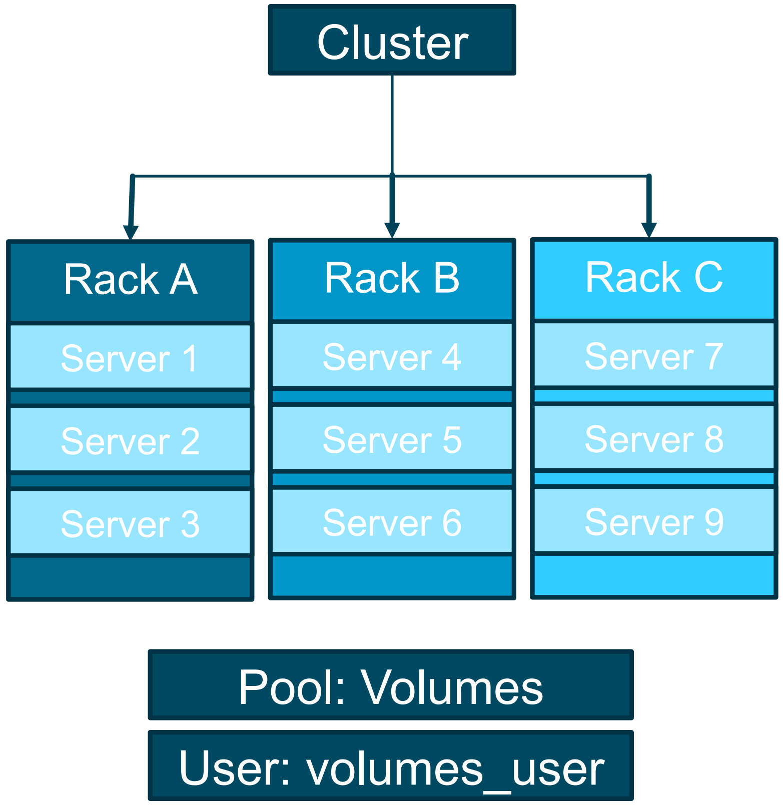
- Create the CRUSH map
  - Organize physical resources into ‘Buckets’
- Create your CRUSH rule
  - Data distribution into ‘buckets’



A Basic CRUSH Example

CRUSH Recap

- Create the CRUSH map
  - Organize physical resources into ‘Buckets’
- Create your CRUSH rule
  - Data distribution into ‘buckets’
- Create a data pool
  - Defines data management using CRUSH rule
    - Access
    - Distribution – PGs



Ceph Resiliancy

How does Ceph make sure the data is safe

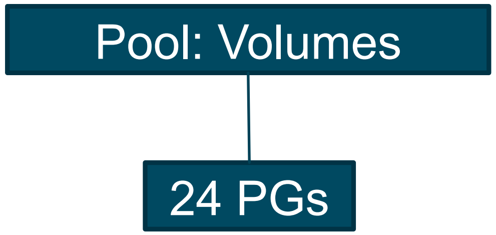
Resiliency

- Lets look at why by using the the Volumes pool as an example:

Pool: Volumes

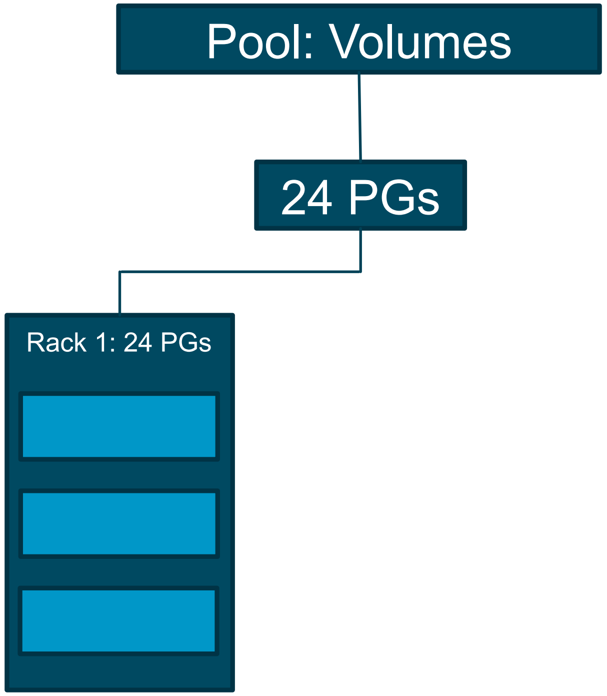
Resiliency

- Lets look at why by using the the Volumes pool as an example:
  - Defined 24 placement groups (PGs)
  - Using the “replicated_ruleset”



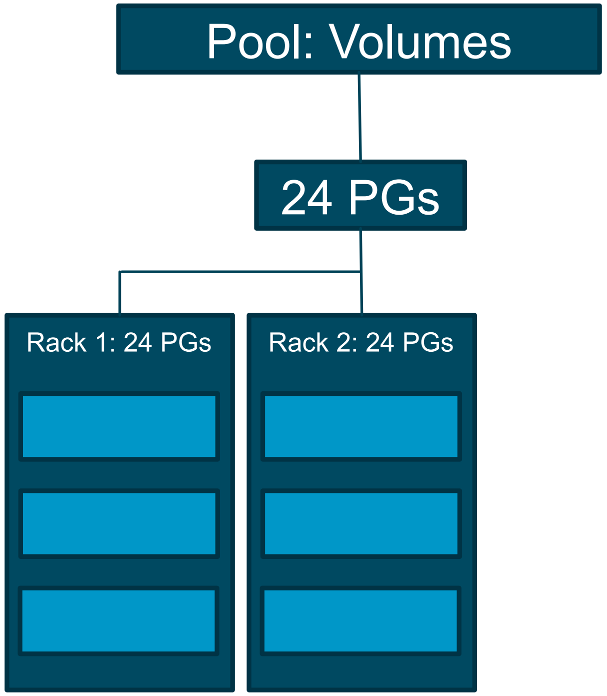
Resiliency

- Lets look at why by using the the Volumes pool as an example:
  - Defined 24 placement groups (PGs)
  - Using the “replicated_ruleset”
  - So it breaks down:
    - Each rack gets 24 PGs



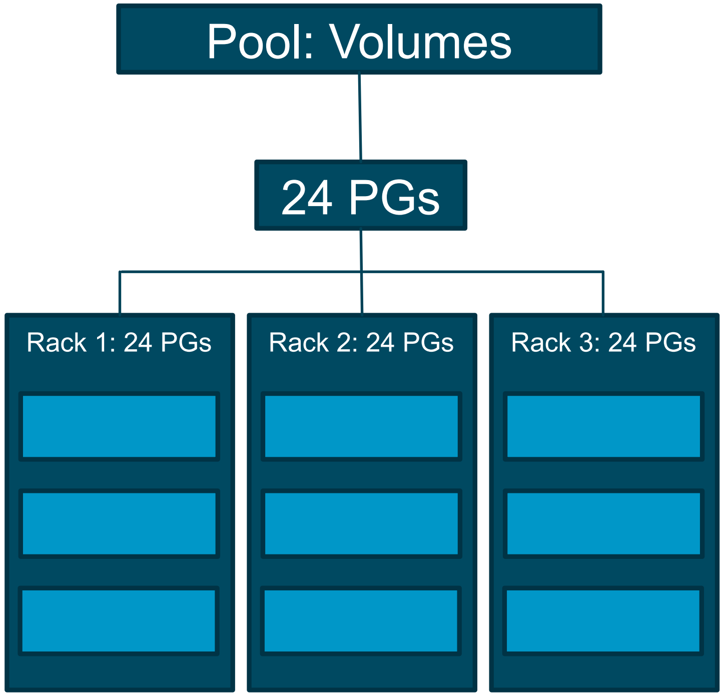
Resiliency

- Lets look at why by using the the Volumes pool as an example:
  - Defined 24 placement groups (PGs)
  - Using the “replicated_ruleset”
  - So it breaks down:
    - Each rack gets 24 PGs



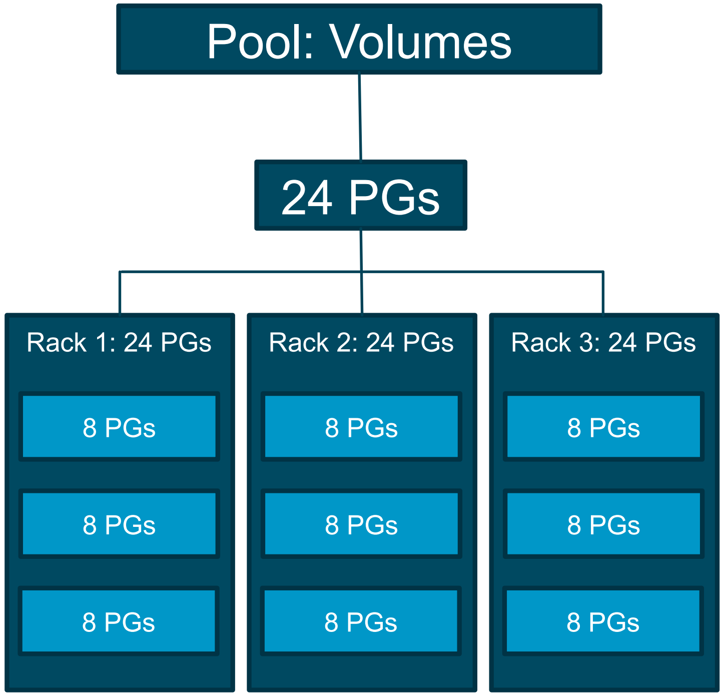
Resiliency

- Lets look at why by using the the Volumes pool as an example:
  - Defined 24 placement groups (PGs)
  - Using the “replicated_ruleset”
  - So it breaks down:
    - Each rack gets 24 PGs
      - All three racks have a copy of the data



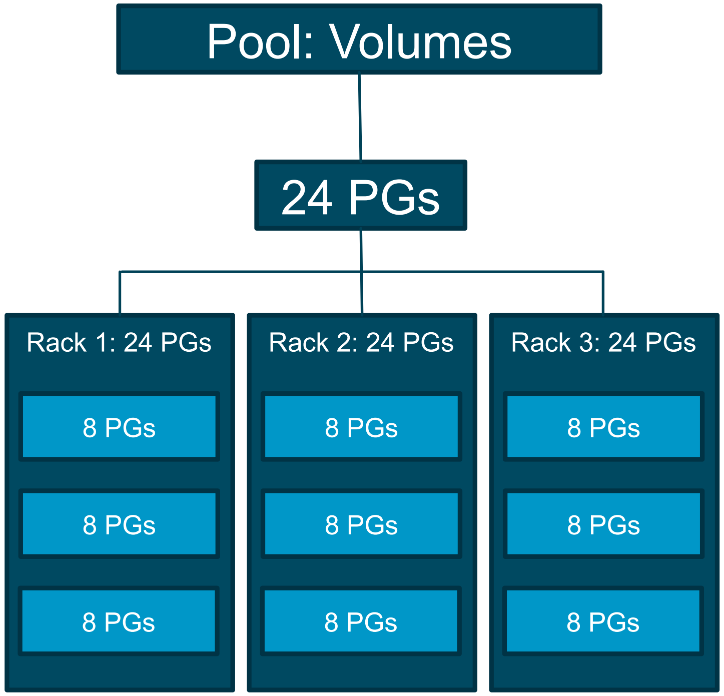
Resiliency

- Lets look at why by using the the Volumes pool as an example:
  - Defined 24 placement groups (PGs)
  - Using the “replicated_ruleset”
  - So it breaks down:
    - Each rack gets 24 PGs
      - All three racks have a copy of the data
    - Each server gets 8 PGs



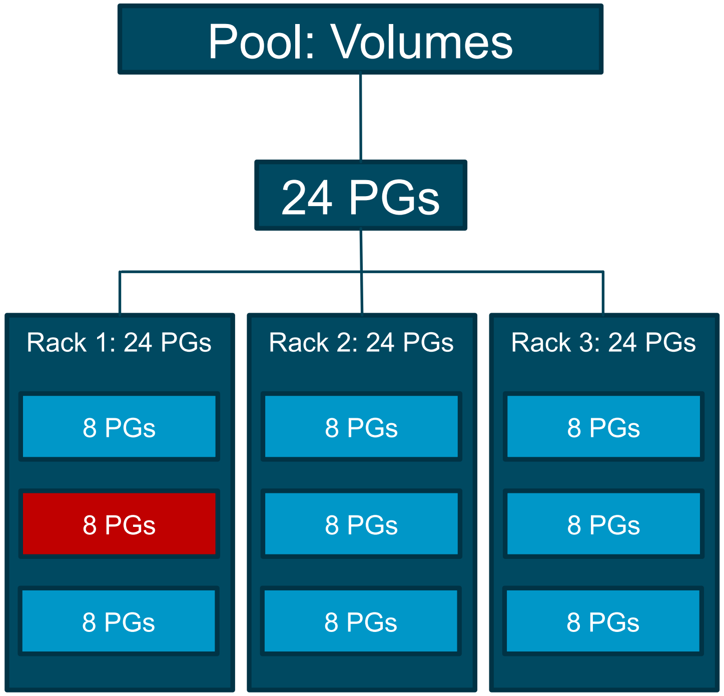
Resiliency

- Lets look at why by using the the Volumes pool as an example:
  - Defined 24 placement groups (PGs)
  - Using the “replicated_ruleset”
  - So it breaks down:
    - Each rack gets 24 PGs
      - All three racks have a copy of the data
    - Each server gets 8 PGs
- This means that if you lose an OSD, the data can be pulled from another OSD elsewhere in the cluster
- Even if you lose a rack you maintain data access



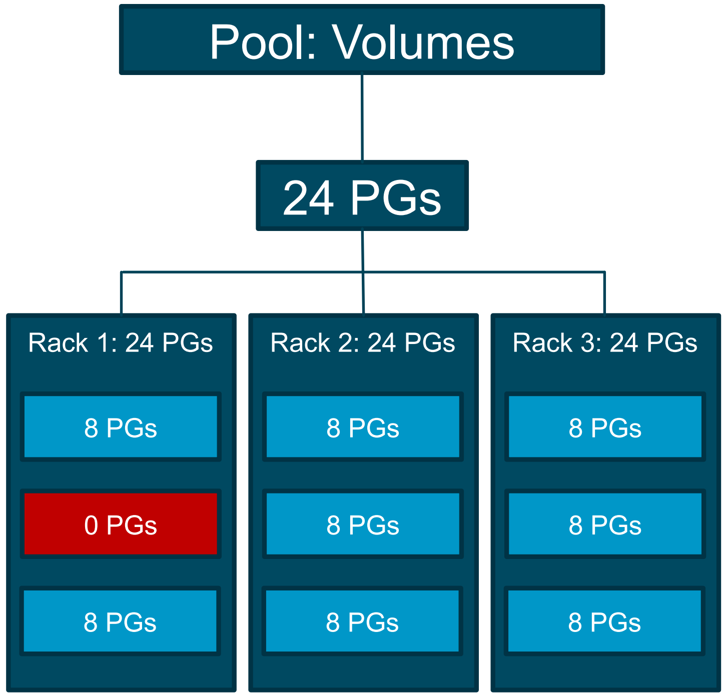
Resiliency

- What happens when you do lose a device, lets say an entire server?



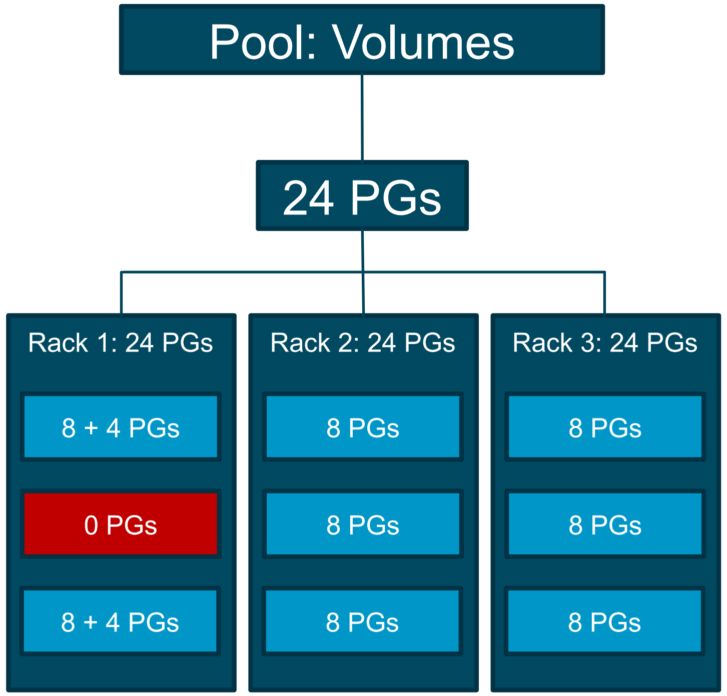
Resiliency

- What happens when you do lose a device, lets say an entire server?
- Well the system looks at that and says, okay no problem.
- First it drops that set of OSDs from the cluster



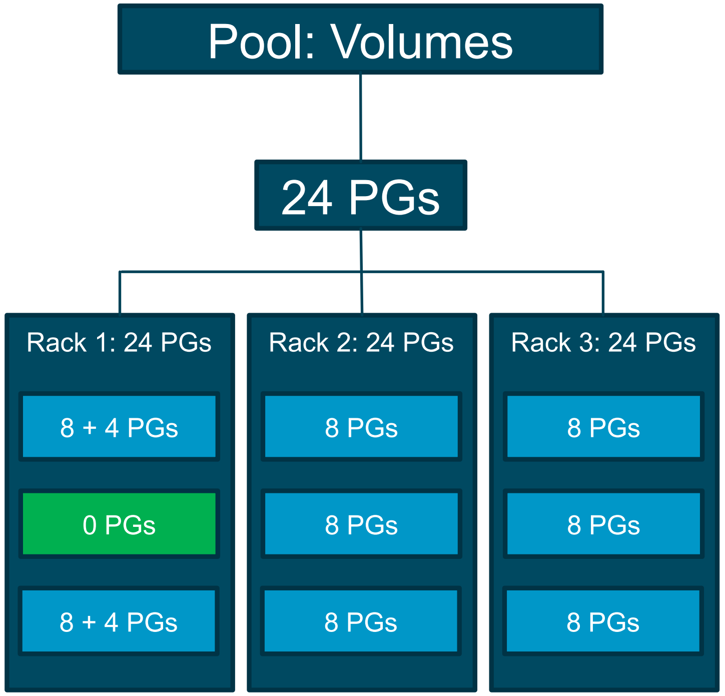
Resiliency

- What happens when you do lose a device, lets say an entire server?
- Well the system looks at that and says, okay no problem.
- First it drops that set of OSDs from the cluster
- Then is replicates the PGs from the other members of the cluster on to neighboring OSDs
  - While the server is out of the cluster you lose that capacity but once the PGs are replicated the cluster is healthy again.



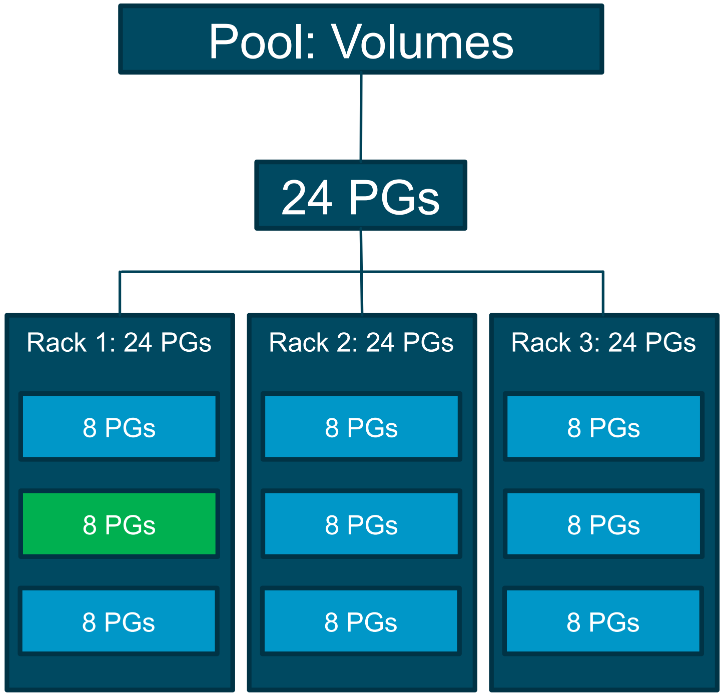
Resiliency

- Once the server is brought back online the cluster checks its health



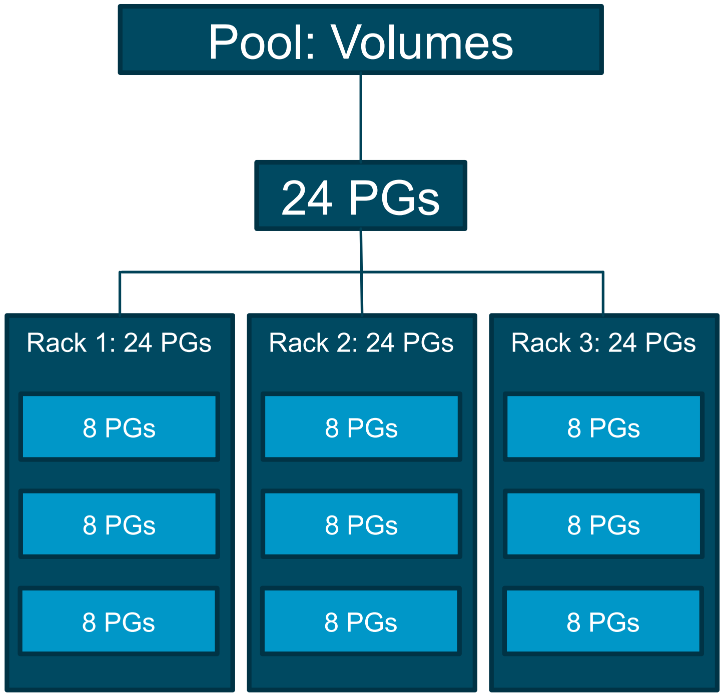
Resiliency

- Once the server is brought back online the cluster checks its health
- Then the PGs that are in the temporary locations are migrated back to the replaced server



Resiliency

- Once the server is brought back online the cluster checks its health
- Then the PGs that are in the temporary locations are migrated back to the replaced server
- The lost capacity is recovered and all operations continue normally



Ceph Management

How Ceph manages the cluster and client access

Ceph Management

- Ceph has two types of nodes:
  1. Data nodes – OSD servers

Ceph Management

Monitor nodes

- Ceph has two types of nodes:
  1. Data nodes – OSD servers
  2. Monitor nodes – cluster managers

Data

Monitor

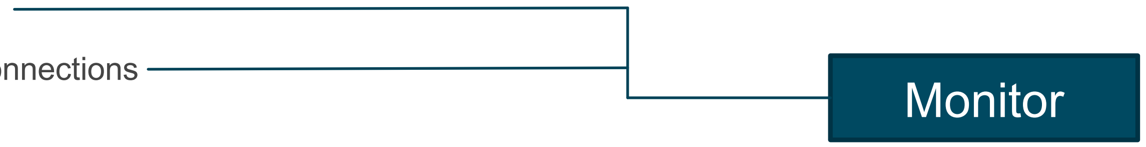
Ceph Management

Monitor nodes

- Tasks include:
  - Cluster health

Ceph Management
Monitor nodes

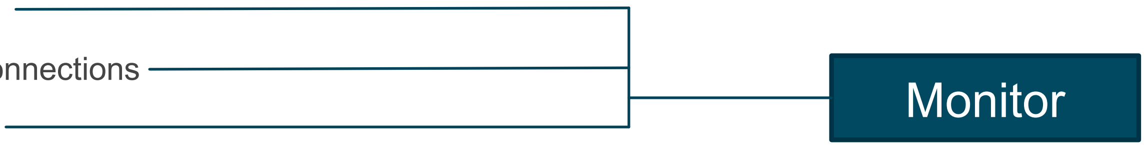
- Tasks include:
  - Cluster health
  - Initial client connections



Ceph Management

Monitor nodes

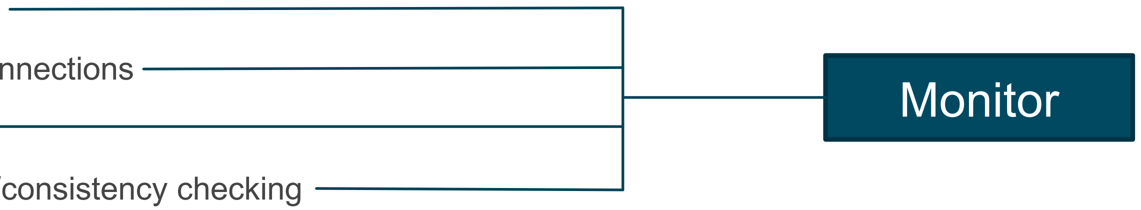
- Tasks include:
  - Cluster health
  - Initial client connections
  - Manager API



Ceph Management

Monitor nodes

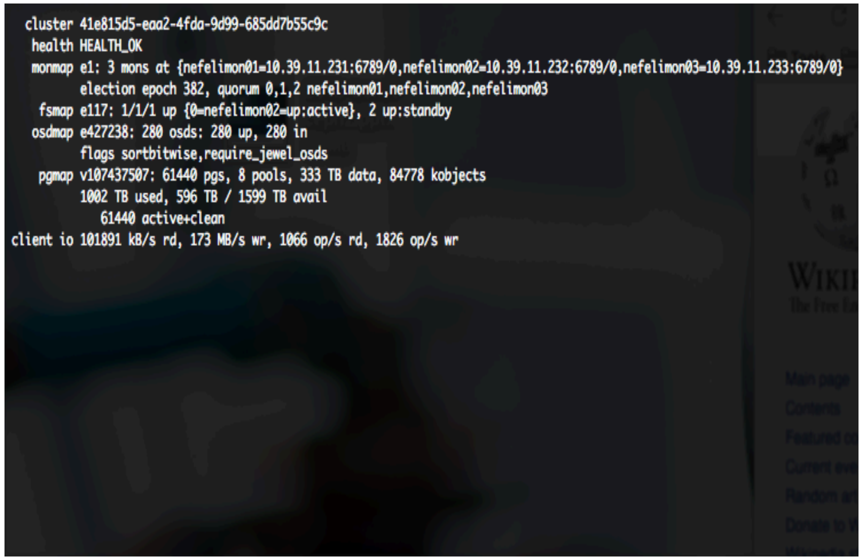
- Tasks include:
  - Cluster health
  - Initial client connections
  - Manager API
  - Data cleaning/consistency checking



Ceph Management

Monitor nodes: Monitoring

- The primary function is monitoring the cluster's performance and health
- These nodes watch
  - Data throughput of the cluster
  - Health of the OSDs
  - Heath of the PGs
  - Basic details at a glance
  - In-depth analysis for all aspects of the cluster performance

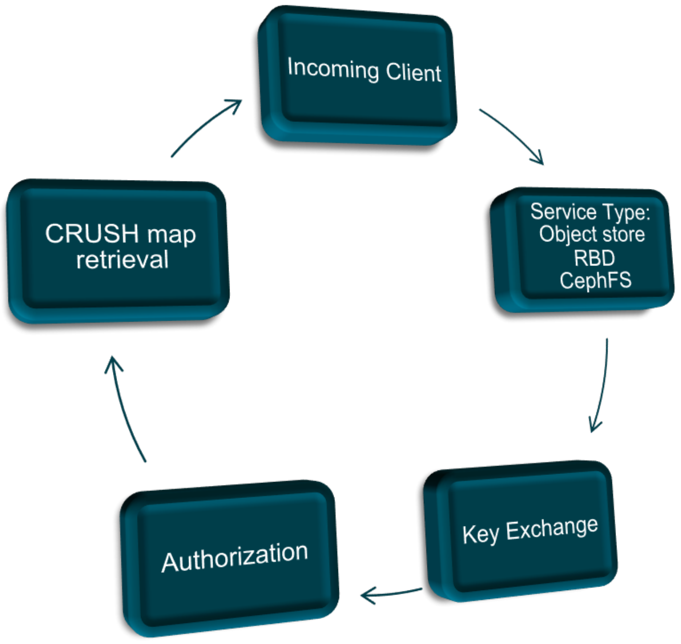


```
  cluster 41e815d5-eaa2-4fda-9d99-685dd7b55c9c
   health HEALTH_OK
   monmap e1: 3 mons at {nefelimon01=10.39.11.231:6789/0,nefelimon02=10.39.11.232:6789/0,nefelimon03=10.39.11.233:6789/0}
          election epoch 382, quorum 0,1,2 nefelimon01,nefelimon02,nefelimon03
    fsmap e117: 1/1/1 up {0=nefelimon02=up:active}, 2 up:standby
   osdmap e427238: 280 osds: 280 up, 280 in
          flags sortbitwise,require_jewel_osds
    pgmap v107437507: 61440 pgs, 8 pools, 333 TB data, 84778 kobjects
          1002 TB used, 596 TB / 1599 TB avail
             61440 active+clean
client io 101891 kB/s rd, 173 MB/s wr, 1066 op/s rd, 1826 op/s wr
```

Ceph Management

Monitor nodes: Initial Client Connection

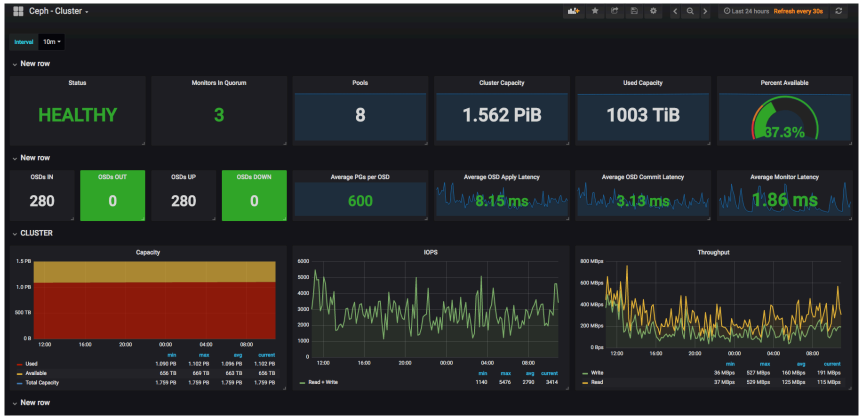
- Initial client connections involve a couple of things:
  - When the client connects it announces what type of connection its making (Object, RBD, or CephFS)
  - Exchanges keys for authentication/authorization
  - Gets a copy of the CRUSH map
- From there the client has all the information needed to read/write data in the cluster
- The monitors do not process data in the cluster for the clients – the clients speak directly with the OSDs that host the data



Ceph Management

Monitor nodes: Manager API

- The manager API brokers a couple of important functions:
  - Issuing commands to the cluster
  - Allows connections to third party applications
    - Graphana/Prometheus – visualization of cluster statistics



Status	Monitors In Quorum	Pools	Cluster Capacity	Used Capacity	Percent Available
HEALTHY	3	8	1.562 PiB	1003 TiB	37.3%

OSDs IN	OSDs OUT	OSDs UP	OSDs DOWN	Average PGs per OSD	Average OSD Apply Latency	Average OSD Commit Latency	Average Monitor Latency
280	0	280	0	600	8.15 ms	3.13 ms	1.86 ms

CLUSTER

Capacity

	min	max	avg	current
Used	1.090 PB	1.102 PB	1.096 PB	1.102 PB
Available	656 TB	669 TB	663 TB	656 TB
Total Capacity	1.759 PB	1.759 PB	1.759 PB	1.759 PB

IOPS

	min	max	avg	current
Read + Write	1140	5476	2790	3414

Throughput

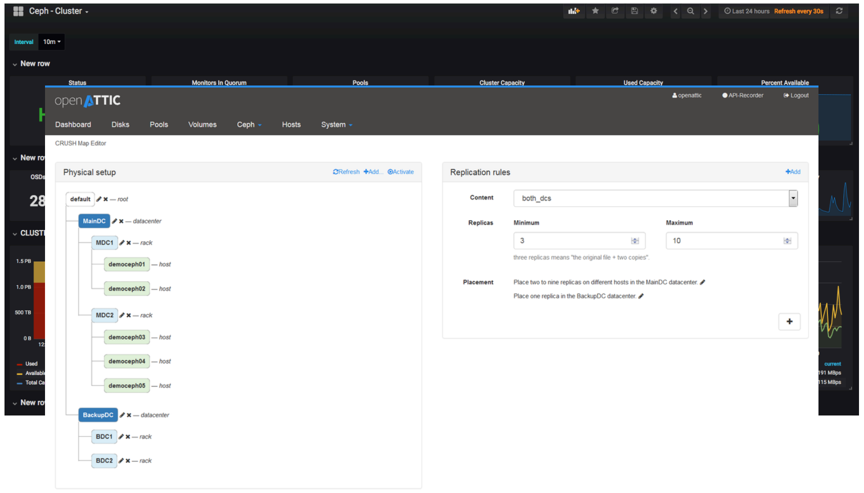
	min	max	avg	current
Write	36 MBps	527 MBps	160 MBps	191 MBps
Read	37 MBps	529 MBps	125 MBps	115 MBps

New row

Ceph Management

Monitor nodes: Manager API

- The manager API brokers a couple of important functions:
  - Issuing commands to the cluster
  - Allows connections to third party applications
    - Graphana/Prometheus – visualization of cluster statistics
    - openAttic – cluster management through a web GUI



Content: both_dcs

Replicas: Minimum 3, Maximum 10

three replicas means "the original file + two copies".

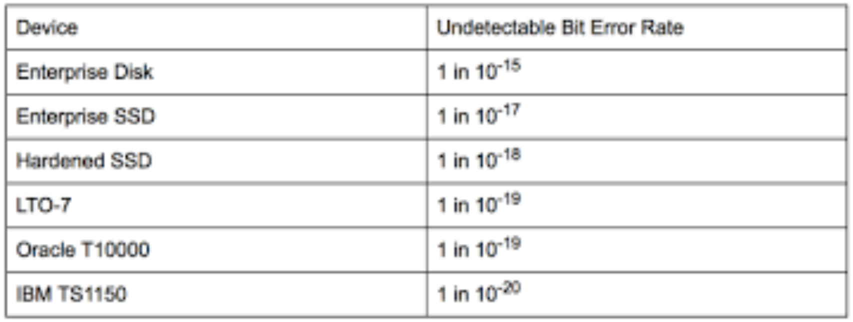
Placement: Place two to nine replicas on different hosts in the MainDC datacenter.

Place one replica in the BackupDC datacenter.

Ceph Management

Monitor nodes: data consistency/cleaning

- The monitor nodes are responsible for ensuring the consistency of the data in the PGs and guard against ‘bit rot’

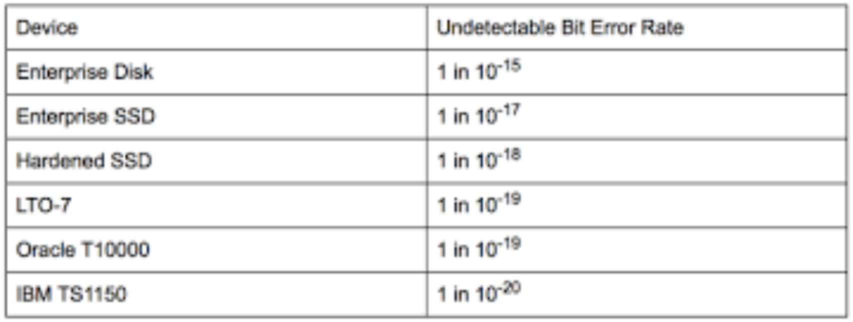


Device	Undetectable Bit Error Rate
Enterprise Disk	1 in 10^{-15}
Enterprise SSD	1 in 10^{-17}
Hardened SSD	1 in 10^{-18}
LTO-7	1 in 10^{-19}
Oracle T10000	1 in 10^{-19}
IBM TS1150	1 in 10^{-20}

Ceph Management

Monitor nodes: data consistency/cleaning

- The monitor nodes are responsible for ensuring the consistency of the data in the PGs and guard against ‘bit rot’
- The process is called ‘scrubbing’

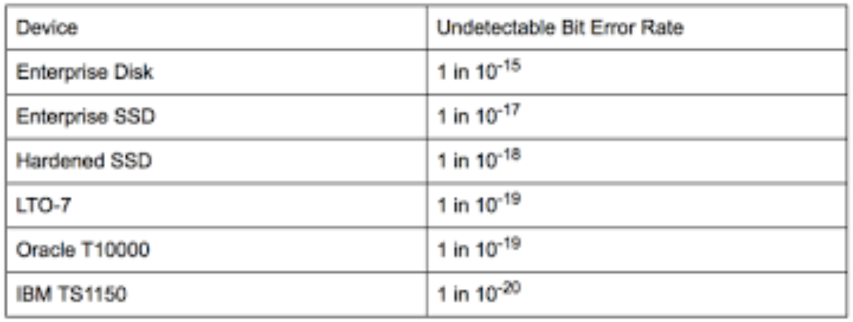


Device	Undetectable Bit Error Rate
Enterprise Disk	1 in 10^{-15}
Enterprise SSD	1 in 10^{-17}
Hardened SSD	1 in 10^{-18}
LTO-7	1 in 10^{-19}
Oracle T10000	1 in 10^{-19}
IBM TS1150	1 in 10^{-20}

Ceph Management

Monitor nodes: data consistency/cleaning

- The monitor nodes are responsible for ensuring the consistency of the data in the PGs and guard against ‘bit rot’
- The process is called ‘scrubbing’
- Scrubbing can cause some performance hits



Device	Undetectable Bit Error Rate
Enterprise Disk	1 in 10^{-15}
Enterprise SSD	1 in 10^{-17}
Hardened SSD	1 in 10^{-18}
LTO-7	1 in 10^{-19}
Oracle T10000	1 in 10^{-19}
IBM TS1150	1 in 10^{-20}

Ceph Management

Monitor nodes: data consistency/cleaning

- The monitor nodes are responsible for ensuring the consistency of the data in the PGs and guard against 'bit rot'
- The process is called 'scrubbing'
- Scrubbing can cause some performance hits
  - Best to schedule
  - Ensure entire cluster is checked weekly

Ceph at the University of Victoria

Backing a cloud deployment

Ceph at UVic
Current State

- Current cluster is:
  - 3 Monitor nodes
  - 18 Data nodes
    - 10 - 10x4TB
    - 8 – 20x8TB
  - 1.6 PB
    - 500T Usable
  - Redundant 10G client/replication network
  - Single 1G for management

Ceph at UVic

Future State

- New cluster:
  - 3 Monitor nodes
  - 42 Data nodes
    - 10 - 10x4TB
    - 8 – 20x8TB
    - 24 – 20x10TB
  - 6.4 PB Raw
    - ~4 PB Usable
    - Employing mixture of Erasure Coding and Replication
  - Redundant 10G client/replication network
  - Single 1G for management
  - Possible expansion for special projects

Ceph at UVic
Arbutus Cloud

- One of the largest non-commercial clouds in Canada
- Phase 2 is underway
- Hosting for researcher platforms and portals
- HPC in the cloud

openstack®

ceph

Questions

Please feel free to reach out to me via email:

mcave@uvic.ca

Resources

- Ceph:
  - http://ceph.com
  - http://docs.ceph.com/docs/master/
- CRUSH
  - https://ceph.com/wp-content/uploads/2016/08/weil-crush-sc06.pdf
- Compute Canada
  - https://www.computecanada.ca/
- OpenStack
  - https://www.openstack.org/