Utilizing CRUSH Algorithm on Ceph to Build a Cluster of Reliable Data Storage

Imam Nur Bani Yusuf
School of Electrical Engineering and Informatics
Bandung Institute of Technology
inby04@gmail.com

Eueung Mulyana
School of Electrical Engineering and Informatics
Bandung Institute of Technology
eueung@gmail.com

Hendrawan
School of Electrical Engineering and Informatics
Bandung Institute of Technology
hendrawan@stei.itb.ac.id

Adrie Taniwidjaja
School of Electrical Engineering and Informatics
Bandung Institute of Technology
acarnila@gmail.com

***Abstract*— Ceph uses CRUSH algorithm (Controlled Replication Under Scalable Hashing) to achieve data redundancy in storage cluster. The use of CRUSH algorithm can replace the role of controller to achieve data redundancy, so that dependency on vendors can be eliminated. CRUSH algorithm allows one to build a data storage cluster that has high scalability and flexibility. Unfortunately, no study has discussed comprehensively how Ceph uses CRUSH algorithm to create redundant data and spread it to storage clusters. This paper aims to examine the mechanism for placing Ceph data using CRUSH algorithm comprehensively. The research method used is by conducting literature studies and experiments. The experiment was carried out by directly trying out the configuration applied to the data storage cluster. By understanding the workings of CRUSH algorithm, it is expected that people can customize the data placement carried out by Ceph according to the needs of services that will be run on the data storage cluster.**

Keywords*— *Ceph, CRUSH, SDS

I. INTRODUCTION

The reliability of the data storage cluster can be achieved by implementing a data redundancy mechanism. The data redundancy mechanism that is widely used today is RAID (Redundant Array of Independent Disks). RAID technology uses special controllers to create redundant data. The use of a controller causes the storage cluster to have poor scalability because to expand the data storage cluster, a controller of the same type is needed with the installed controller. In addition, if there is damage to the controller, the controller must be replaced with the same type as well.

Ceph is one of the most popular open source Software Defined Storage implemented. Ceph data storage clusters offer better flexibility and scalability with CRUSH algorithm. Ceph uses CRUSH algorithm (Controlled Replication Under Scalable Hashing) to achieve data redundancy in data storage clusters. The choice of devices used to build the cluster can be adjusted to the technological development of the hardware and cluster expansion is not limited to the type of hardware that has been used in the data storage cluster.

The amount of redundant data that is created by CRUSH algorithm can be configured according to the service requirements that will be run in the cluster. How data and replicas are distributed in a data storage cluster can also be configured according to the specified domain of failure. CRUSH algorithm makes it possible to build data storage clusters without a single point of failure.

Ceph design has been explained in [4] and benchmarking of CRUSH algorithm has been done in [9]. But, in both papers it has not been explained in detail the parameters that affect CRUSH algorithm manifests redundancy on Ceph and how to customize the distribution of data using CRUSH algorithm according to the needs of certain design specifications.

This paper will explain how CRUSH algorithm and Ceph place data and replicas on Ceph data storage clusters and what parameters affect them. A case study will also be presented to customize the CRUSH algorithm according to the needs of certain design specifications.

II. RELATED WORKS

Some research on optimizing the mechanism of data redundancy in cloud storage has been carried out in [2] and [11]. [2] discusses how to place data redundancy by considering the Quality of Service factor. [11] optimizing capacity usage caused by redundancy mechanisms in cloud storage with a method called deduplication. Optimization of Ceph data placement to increase write throughput has also been done on [10].

Building a storage cluster of Ceph data is discussed in [1] and [3]. [1] describes an example of a Ceph implementation for a data storage cluster application. The data storage cluster designed consists of 40 virtual servers on 4 different Proxmox servers. [3] discussed the implementation of Ceph to handle many heterogeneous services by allocating one pool specifically for one service.

III. CEPH COMPONENT

Ceph consists of 3 main daemons: Monitor, Manager, and OSD. The Monitor has the main function to monitor the condition of all components that make up the Ceph data storage cluster. The monitor summarizes the condition of all

components in the cluster map. The cluster map consists of: OSD map, CRUSH map, and placement group map.

In order for the monitor functionality to run fully, the manager must be installed on the same node as the monitor. The manager provides additional monitoring plugins such as: RESTful plugins, ceph-dashboard plugins, and plugins for integration with monitoring services such as Influx, Zabbix, Telemetry, Telegraf, Iostat, and Hello.

OSD is the daemon that is responsible for storing data on Ceph data storage clusters. The OSD daemon can be initiated on a node that has a data storage device, such as a flash disk, *hard disk drive*, and *solid-state drive*. There are two possible OSD configurations, that are: making only one partition on the data storage device and using all of the partitions on an OSD or dividing the data storage device into several partitions then each partition is used by a different OSD daemon

IV. CEPH DATA PLACEMENT

Ceph converts files that will be saved to the data storage cluster into objects of a certain size. If the file size is larger than the maximum size of the object, the file is divided into several objects. This mechanism is called striping.

The striping mechanism has several advantages including:

1. Can speed up the process of writing data to a data storage cluster because the writing process can be done in parallel
2. If there is a failure on the storage device, the data recovery process is faster

Data that has been converted into an object is ready to be stored in the Ceph data storage cluster. The object will be saved to a placement group contained in a pool. Pool is a logical container that is used to hold data, while placement groups are a subset of pools. Pool is an abstraction of a storage unit allocated by a Ceph administrator for one / particular group of users.

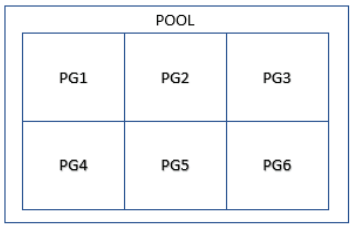

Figure 1 Relationship Between Pool dan Placement Group

Ceph client will access the node running the monitor daemon (called the monitor node) to get a cluster map. After getting a cluster map, Ceph client initiates a connection to the pool. If the connection to the pool is successful, Ceph client gets four parameters, that is:

1. Pool ID

   Pool IDs are integer integers. Pool ID is a unique ID that distinguishes one pool from another. Pool ID is assigned to a pool automatically by Ceph when the pool creation process is successful.

2. CRUSH rule ID that is applied to the pool

   CRUSH rule parameter is defined when the process of creating a pool. This parameter determines how objects and replicas are stored in clusters. CRUSH rule ID is an integer.

3. PG number

   This parameter provides information on the number of placement groups in a pool. The PG number is defined as the process of creating a pool. If not defined, by default the value matches the `default pg num osd pool parameter` defined in the cluster configuration file stored in the `/etc/ceph` directory.

4. Pool size

   This parameter determines how much total (data + replica) is made if using the replication redundancy mechanism or the total number (k + m) if using erasure code. By default, the value matches the `default size osd pool` parameter defined in the cluster configuration file stored in the `/etc/ceph` directory.

Objects are stored at a specific placement group in a pool. Each placement group has an ID called PG ID. PG ID is unique and has the following writing format:

$$\text{PGID} = \text{X.Y} \ldots (\text{III.1})$$

X is an integer that states the ID pool and Y are hexadecimal numbers obtained from the following equation [6]:

$$\text{Y} = \text{hash}(\text{ObjectID}) \% \text{pgnumber} \ldots (\text{III.2})$$

When data is successfully converted into an object, the object automatically gets an object ID.

Ceph needs a cluster map to determine cluster configurations and conditions. The cluster map consists of: OSD map, CRUSH map, and placement group map. CRUSH map consists of: CRUSH hierarchy and CRUSH rules.

After getting a specific PG ID for an object, then Ceph maps the PG ID using CRUSH rule to a number of OSD, where the value of n is equal to the number (data + replica) or (k + m) obtained from the pool size parameter. To place objects and replicas, CRUSH rule requires information about cluster hierarchy and which OSD are functioning (can be used to store objects and replicas). Both of this information are obtained from CRUSH hierarchy and the OSD map.

All mapping of PG ID to a number of OSDs is recorded in placement group map. The function of this recording is so that the Ceph can easily find out the location of objects and replicas.

V. CRUSH PROCEDURE

CRUSH procedure is part of CRUSH rule defined on CRUSH map. Below is an example of a default crush rule.

```
rule replicated {

    ruleset 0
    type replicated
    min_size 1
    max_size 10
    step take default
    step choose firstn 0 type host
    step choose firstn 0 type osd
    step emit
}
```

CRUSH procedure starts with the `step` function. The structure of CRUSH procedure is as follows:

1. `step take <bucket-name>`

   defines the starting point of CRUSH procedure and determines the execution domain for the procedure defined after the `step take <bucket-name>`. The `<bucket-name>` parameter value is derived from bucket names defined in CRUSH hierarchy. For example, there is a hierarchy like this:

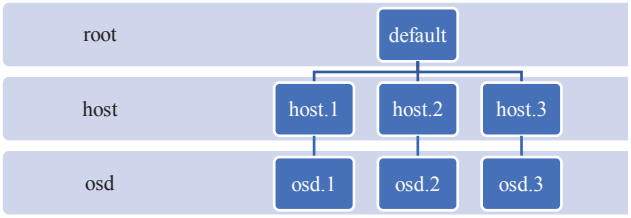

Figure 2 Hierarchy Example 1

   In the hierarchy above, there is one bucket of the root type named default, 3 buckets of type host named host.1, host.2, and host.3 which are located below the default bucket, and on each host, there is a bucket of type osd with the name osd.1, osd.2, and osd.3.

   If the `step take default` procedure is applied at the hierarchy above, CRUSH procedure starts with a bucket named default and domain execution of CRUSH procedure after the `step take default` is a bucket that is below the default bucket, that are host.1, host.2, and host. 3. The <bucket-name> parameter selection does not have to start on the root type bucket.

2. `step [choose|chooseleaf] [firstn|indep] n type <bucket-type>`

   the command has a function to select a number of n buckets defined in CRUSH hierarchy at a certain level of all available buckets. The bucket selection must reach the OSD type bucket. If not, CRUSH rule is invalid (error).

choose vs chooseleaf

The choose method works to select a number of n buckets randomly in the specified `<bucket-type>` domain from all available buckets.

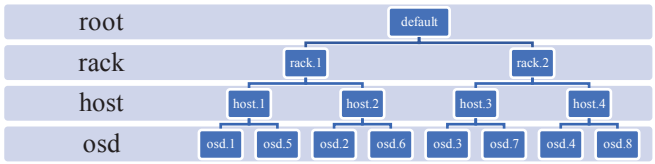

Figure 3 Hierarchy Example 2

If `step choose firstn n type rack` is applied at the hierarchy above, meaning that bucket in the rack domain, namely rack.1 and rack.2, a number of n buckets of all available buckets are selected. The bucket selection stops at this stage. The procedure for selecting buckets after selecting buckets on the rack domain must be defined again using `step [choose | chooseleaf] [firstn | indep] n type <bucket-type>` until the selected bucket type is osd.

The chooseleaf method works to select a number of n buckets randomly in the specified `<bucket-type>` domain from all available buckets and in the bucket domain under the specified `<bucket-type>`, 1 bucket randomly selected from all available buckets. The selection of 1 bucket stops until the selected bucket type is osd.

firstn dan indep

firstn is used for replication redundancy mechanism while indep is used for erasure code redundancy mechanism.

variation of n value

The value of n is influenced by the pool size parameter of a pool. The pool size parameter (p) by default the value is the same as the `default size osd pool parameter` defined in the cluster configuration file stored in the `/etc/ceph` directory. This parameter is obtained when the Ceph client is successfully connected to a pool. The hierarchy used for the explanation of variations in the value of n is same as Figure 2.

For example, the value of p = 3 and CRUSH procedure used is `step choose firstn n type host`. There are 3 possible values of n:

- n = 0, means choosing a number of p buckets from all available buckets

  If n = 0 and p = 3, Ceph will randomly select 3 hosts from all available hosts. In this example only 3

buckets are available, the bucket host.1, host.2, and host.3 are definitely selected.

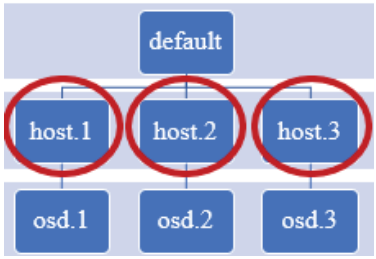

Figure 4 Bucket Selection for n = 0 and p = 3

- n > 0 and n < p, means choosing a number of n buckets from all available buckets

  If the value of n = 2 and p = 3, the bucket chosen randomly is 2 out of 3 buckets available. In this example, the selected bucket is host.1 and host.3.

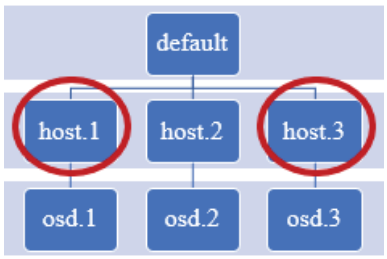

Figure 5 Bucket Selection for n = 2 and p = 3

- n < 0, means choosing a number of (p-n) buckets from all available buckets

  If n = -1 and p = 3, the number of buckets chosen randomly are 2 because 3 - 1 = 2. In this example, the randomly selected bucket is host.1 and host.2.

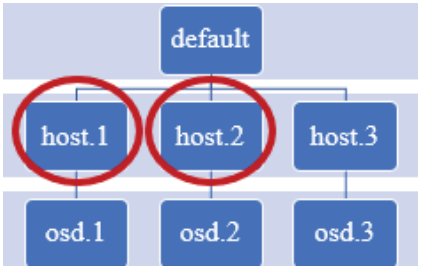

Figure 6 Bucket Selection for n = -1 and p = 3

3. `step emit`

Every CRUSH procedure always ends with `step emit`.

VI. CASE STUDY

The hierarchy used for the case study is as follows.

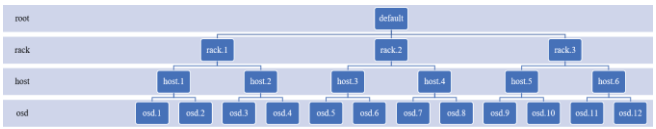

Figure 7 Case Study Hierarchy

Data and replicas want to be stored in a host located on a different rack because each rack has a different power source. The total amount of data and replicas is three.

The first step is to determine the starting point. Starting points are chosen at the level above the domain where the data is distributed. In this example, data is spread on domain rack, so the starting point chosen is a bucket located above the rack, which is root. The root bucket name defined in the CRUSH hierarchy is default.

```
step take <bucket-name>
step take default
```

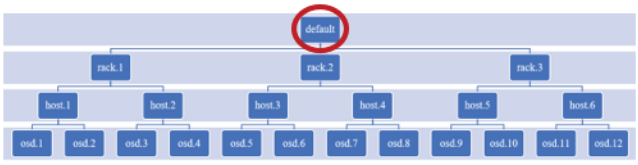

Figure 8 First Step

The next step is to choose a rack based on the parameters n and p. firstn is used because the redundancy mechanism used is replication. When using erasure code, firstn is replaced with indep.

p = 3 (obtained from the total amount of data and replicas), then the value of n must be n = 0 so the rack type bucket selected randomly is equal to the value of p = 3 from all available buckets. There are only three rack type buckets available in the Figure 7, namely rack.1 and rack.2, and rack.3, then the whole bucket is definitely selected.

```
step choose firstn 0 type rack
```

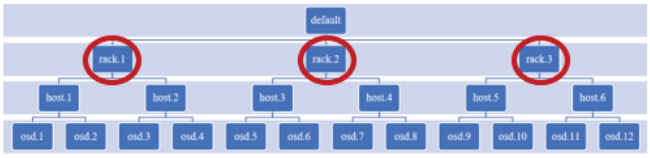

Figure 9 Second Step

The next step is to choose the host type bucket. On each rack, 1 host will be selected from all available bucket hosts. With a value of p = 3 and a value of n = 1, the host type bucket that are randomly selected are one of all available bucket hosts. In rack.1, host.1 and host.2 are available, in rack.2, host.3 and host.4 are available, and in rack.3, host.5 and host.6 are available. On each rack, 1 bucket host will be chosen randomly.

```
step choose firstn 1 type host
```

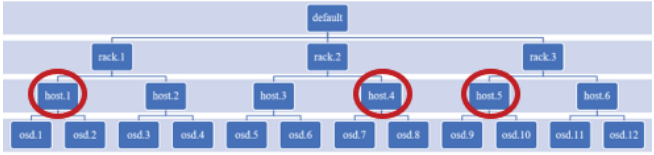

Figure 10 Third Step

The next step is to choose OSD. On each host selected in the previous step, 1 osd will be selected. With the value p = 3 and the value n = 1, the osd type bucket chosen randomly is 1 of all the osd buckets available in the host selected in the previous step.

```
step choose firstn 1 type osd
```

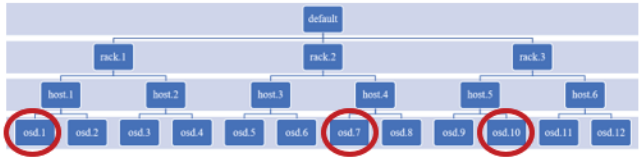

Figure 11 Last Step

The bucket selection has arrived at the osd type bucket. The next step is to end the CRUSH procedure.

```
step emit
```

The CRUSH procedure for placing data and replicas on 3 different racks is as follows

```
step take default
step choose firstn 0 type rack
step choose firstn 1 type host
step choose firstn 1 type osd
step emit
```

VII. Conclusion

Ceph places data by labeling objects with PG IDs that are calculated using equations I and II. The PG ID is then mapped to a number of OSDs. All mapping of PG ID to a number of OSDs is recorded in placement group map. The mechanism of mapping PG ID to OSD is controlled by CRUSH rules defined on CRUSH map. CRUSH rule requires information about cluster hierarchy and which OSD are functioning (can be used to store objects and replicas). Both of this information are obtained from CRUSH hierarchy and the OSD map. All mapping of PG ID to a number of OSDs is recorded to placement group map.

References

[1] Azginoglu, N., Eren, M., Celik, M. and Aydin, Z. (2018). Ceph-based storage server application. *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*.

[2] Matt, J., Waibel, P. and Schulte, S. (2017). Cost- and Latency-Efficient Redundant Data Storage in the Cloud. *2017 IEEE 10th Conference on Service-Oriented Computing and Applications (SOCA)*.

[3] Meyer, S. and Morrison, J. (2015). Supporting Heterogeneous Pools in a Single Ceph Storage Cluster. *2015 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC*

[4] Weil, S., Brandt, S., Miller, E. and Maltzahn, C. (2006). *Ceph: A Scalable, High-Performance Distributed File System*. *Proceedings*. [United States]: [Association for Computing Machinery], pp.307-320.

[5] Fisk, N. (2017). *Mastering Ceph*. Birmingham, UK: Packt Publishing.

[6] Singh, A. (2017). *Learning Ceph - Second Edition*. S.l.: Packt Publishing.

[7] Ceph. (n.d.). *how data is stored in ceph cluster Archives - Ceph*. [online] Available at: https://ceph.com/category/how-data-is-stored-in-ceph-cluster/ [Accessed 26 Jun. 2019].

[8] Docs.ceph.com. (n.d.). *Welcome to Ceph — Ceph Documentation*. [online] Available at: http://docs.ceph.com/docs/ [Accessed 26 Jun. 2019].

[9] Weil, S., Brandt, S., Miller, E. and Maltzahn, C. (2006). CRUSH: Controlled, Scalable, Decentralized Placement of Replicated Data. *ACM/IEEE SC 2006 Conference (SC'06)*.

[10] Wu, C., Hsu, T., Yang, H. and Chung, Y. (2017). File placement mechanisms for improving write throughputs of cloud storage services based on Ceph and HDFS. *2017 International Conference on Applied System Innovation (ICASI)*.

[11] Xu, X. and Tu, Q. (2015). Data Deduplication Mechanism for Cloud Storage Systems. *2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*.