**Object storage**

**Fragmentation Percentage**

Number of jumps / Max jumps   
= Number of jumps / (number of blocks - 1)

**Two blocks**

0 / 2-1 = 0

1 / 2-1 = 1

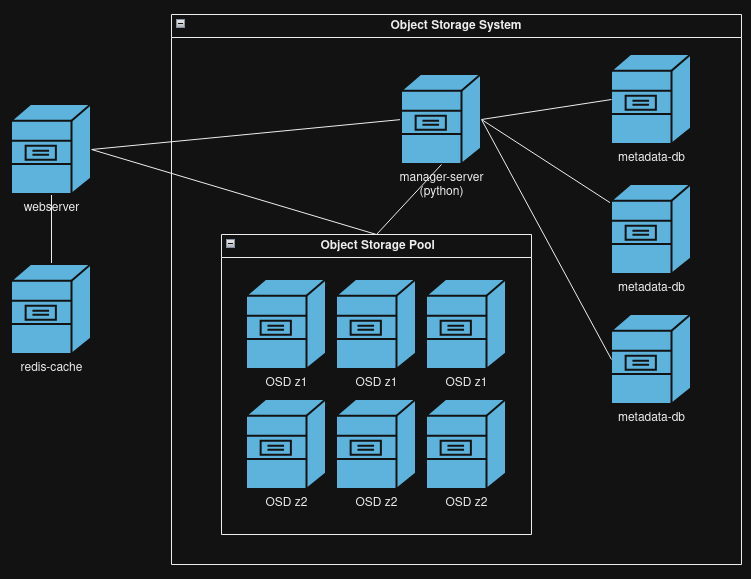
**Three blocks**

0 / 3-1 = 0

1 / 3-1 = 0.5

2 / 3-1 = 1

**Diagram**



**Database and storage architecture**

**Json**



**Storage and redundancy**

For every OSD there will be a redundant one that serves as a clone for high availability and fault tolerance. This can be visualized as two zones. Requests will be load balanced between the two zones. Arbitrarily, there will be a minimum of 3 OSDs in each zone at a time. If objects are greater than 512MiB they will be scattered across at minimum 3 OSDs in a zone to prevent one OSD’s drives from becoming unfairly overloaded. Contiguous block can at most be 512MiB for scalability reasons.

**Scaling**

When a new OSD is added, 1/number of nodes percent of the data from the other OSDs will be moved to that OSD to ensure an even distribution. In the case of the cumulative data being greater than the available storage of the new node, each node will transfer data based on the amount of storage that node has using weights.

**Data flow**

**GET**

1. Client makes request to management server to get data using the UUID
2. Management server returns the specific endpoints and blocks where the data is stored
3. Client makes requests to endpoints which return the raw data

**POST**

1. Client makes request to management server to upload a BLOB
2. The management server searches the free list table to find contiguous data blocks - it will prioritize finding blocks in the free list first, if no contiguous blocks can be found, it will place the blocks at the end pointer (found in a separate table).
3. returns the endpoints and blocks where the

**Free list**

A table in the database that holds block ids that used to hold data which has been previously deleted.

**OSD DNS**

Every object storage device process instance will have its own domain