CRUSH in Ceph

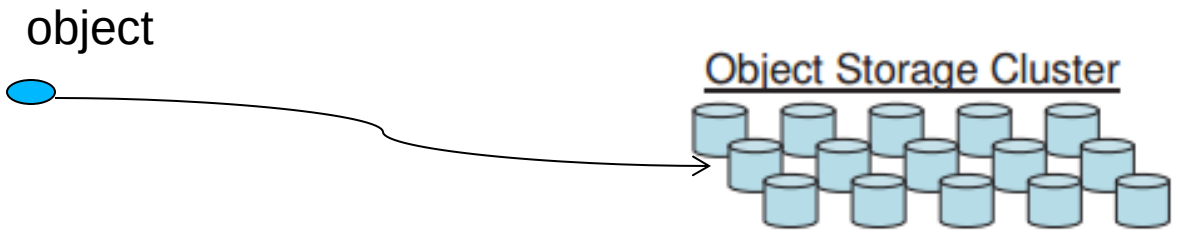
Shang Ding

dingshang@dislab.nju.edu.cn

Contents

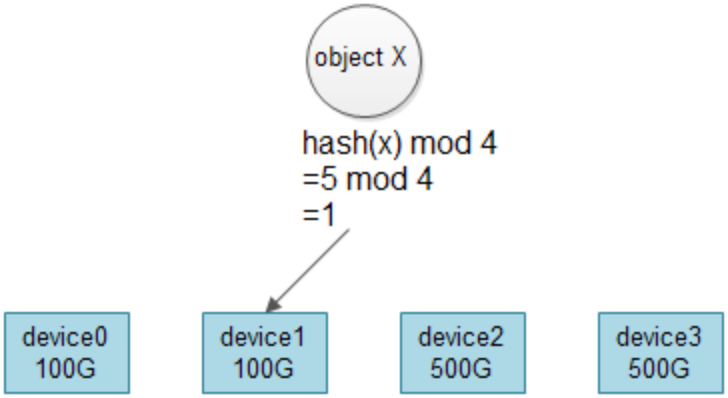
- Distributed Object Storage
- Simple Hash
- CRUSH
  - Hierarchical cluster map
  - Placement rules
  - Bucket types
  - Review

- Distributed Object Storage
  - File into objects
  - Object replica or erasure code



  - CRUSH
    - map object to object storage devices(OSDs)
    - computing instead of storing
    - a pseudo-random data distribution function

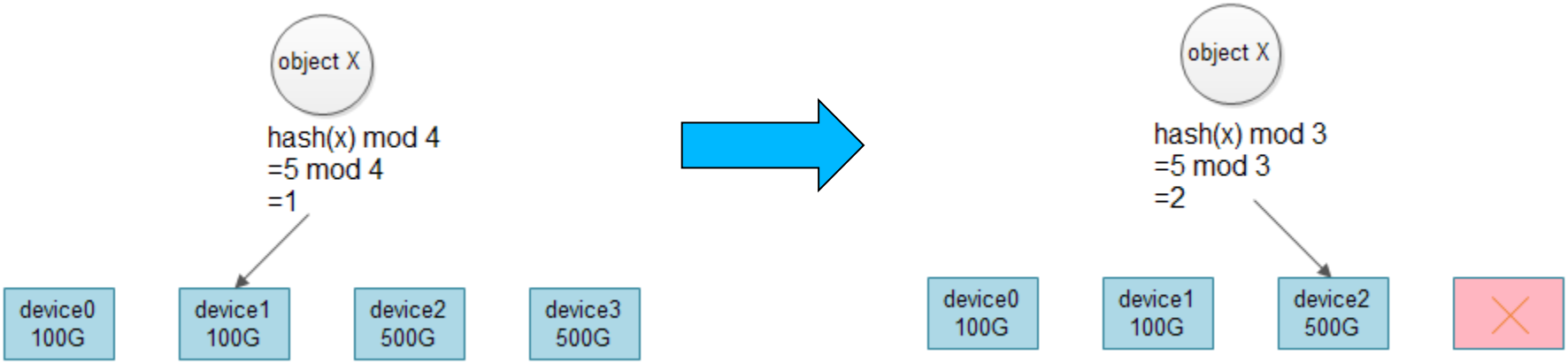
How Mapping: Simple Hash



- Problem
  - Not completely uniformly distributed
    - weight

How Mapping: Simple Hash

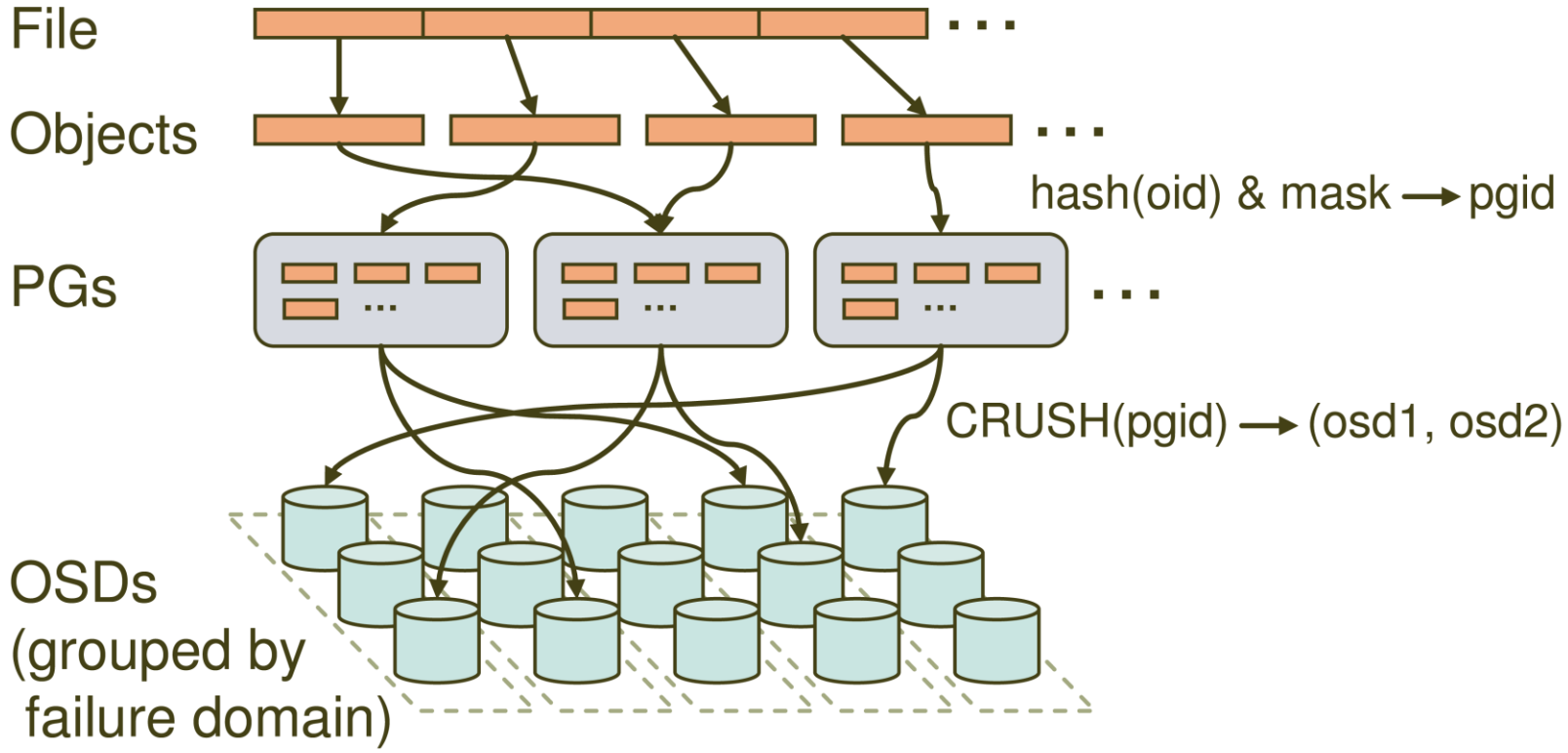
- Devices add/remove



- Problem
  - Data movement a lot when OSDs add/remove
    - structure devices

Mapping in Ceph

- 2 levels mapping
- PGs=placement groups

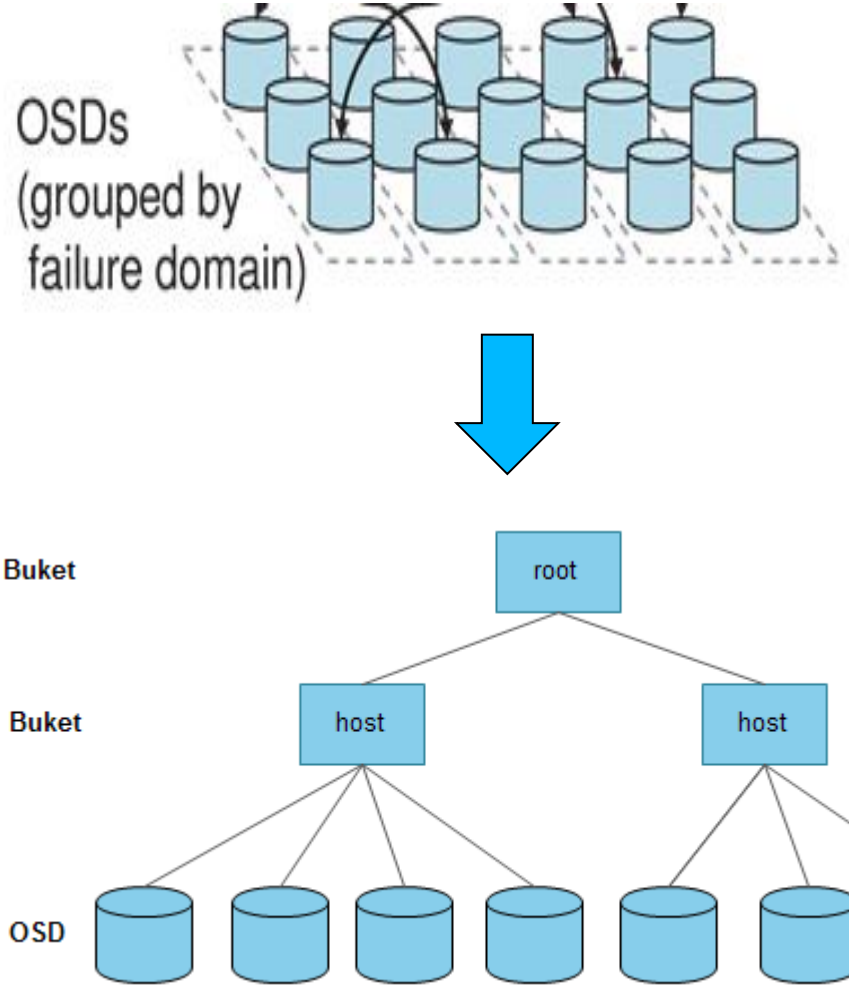


CRUSH(x) → (osd_1, osd_2)

- Short for "Controlled Replication Under Scalable Hashing"
- CRUSH(x) → (osd_1, osd_2)
  - Inputs
    - x is the placement group
    - Hierarchical cluster map
    - Placement rules
  - Outputs
    - a list of OSDs

Hierarchical Cluster Map

- Item
  - leaf node
    - OSD
  - no-leaf node
    - Bucket
- Bucket
  - Contains items
  - Weight
  - Select item using hash



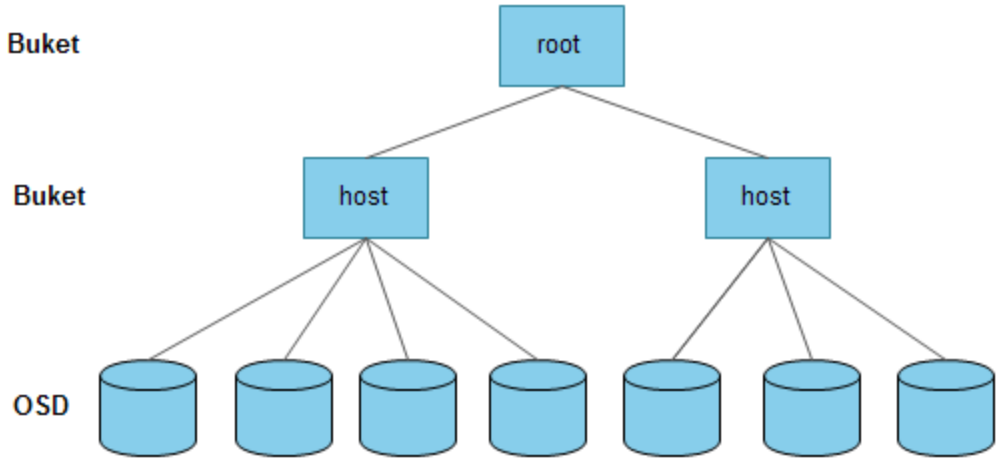
Collisions, Failure, and Overload

- Exception
  - Collisions
    - If an device has already been selected in the current set
  - Failure
    - If a device is *failed*
  - Overload
    - If a device is *overloaded*
- Re-choose
  - Hash(r,x) → Hash(r+1,x)

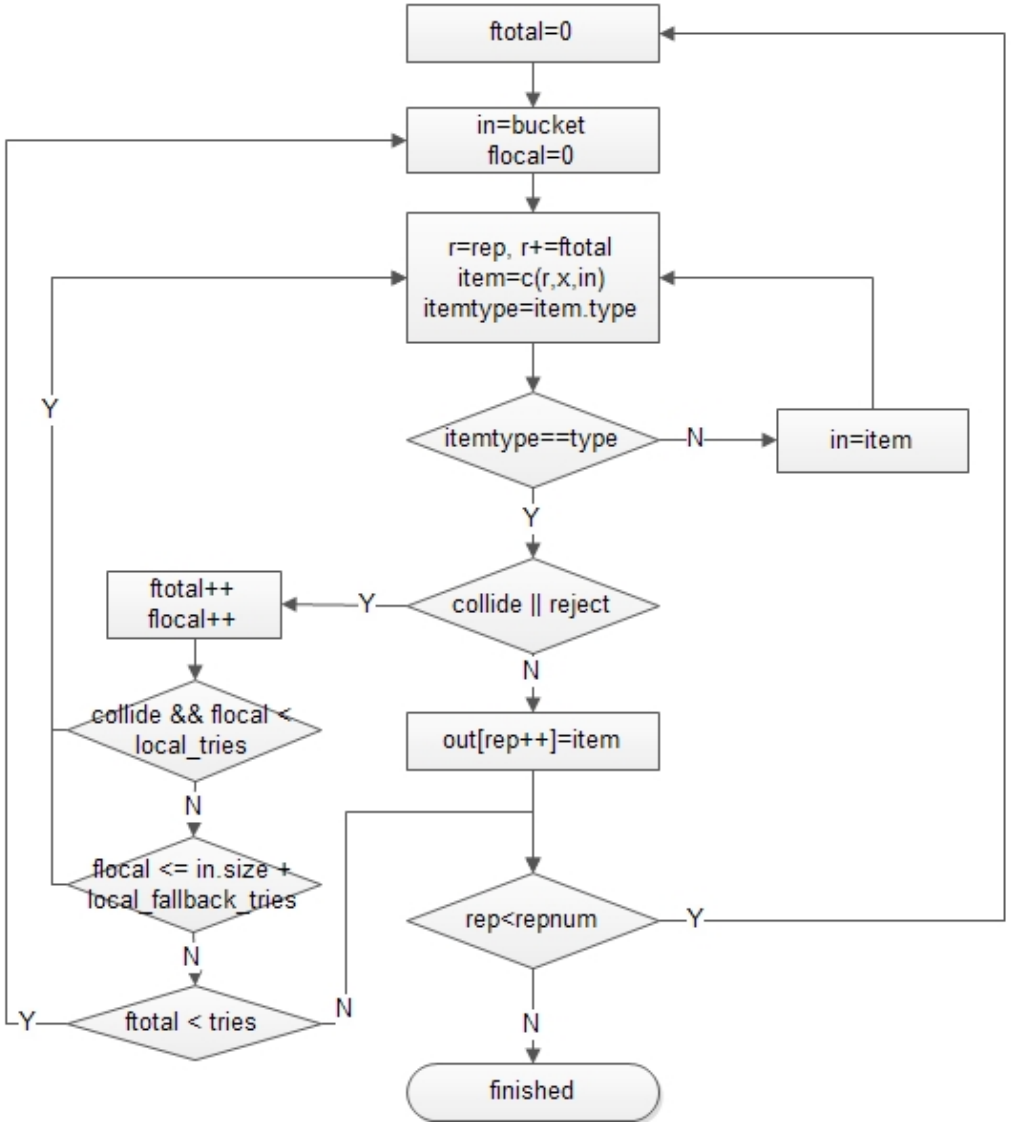
Placement rules

- Rule basic step
  - Step take (bucket-name)
  - Step select(*n*, *type*)
  - Step emit
- Rule
  - Step 1
  - …
  - Step n

Step1 take (host1)
Step2 select(1, OSD)
Step3 take (host2)
Step4 select(1, OSD)
Step5 emit

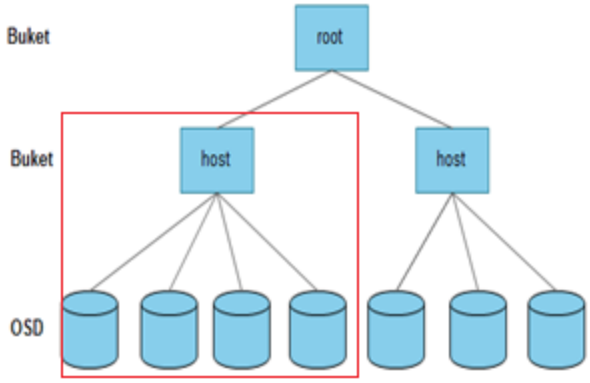


- Select(*n*, *type*)
  - c(r,x) choose an item
  - Many times call c(r,x)



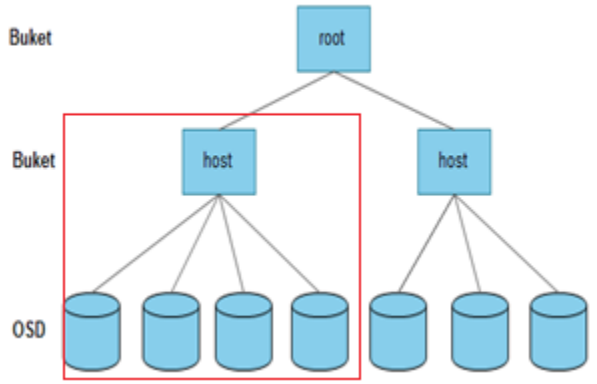
Bucket types

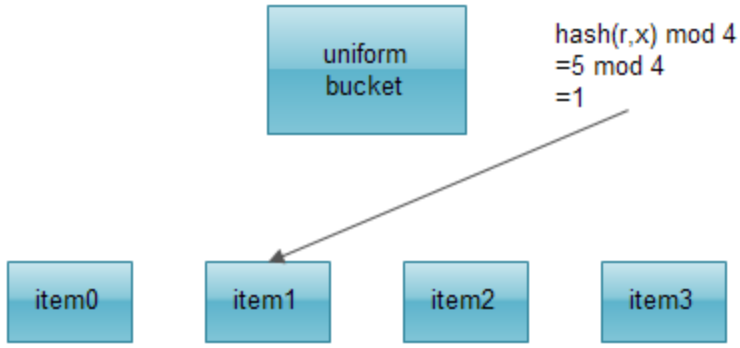
- Based on a different internal data structure and utilize a different function $c(r, x)$ for pseudo-randomly choosing nested items
- Types
  - *Uniform bucket*
  - *List bucket*
  - *Straw bucket*
  - *Tree bucket*



Uniform Bucket

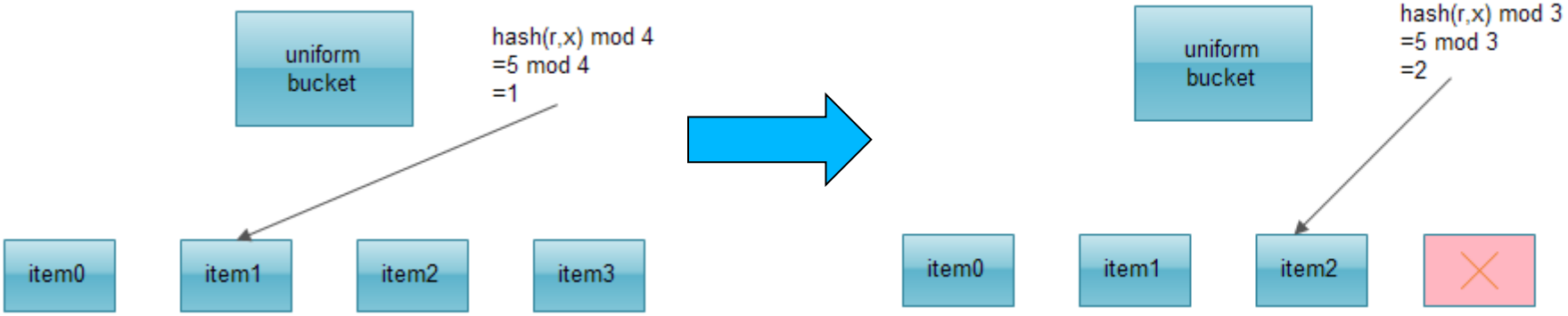
- Nested item weight
  - With same weight
- Algorithm
  - $c(r, x) = \text{hash}(r, x) \bmod n$ (n is size of bucket)
- Complexity
  - $O(1)$
- Example





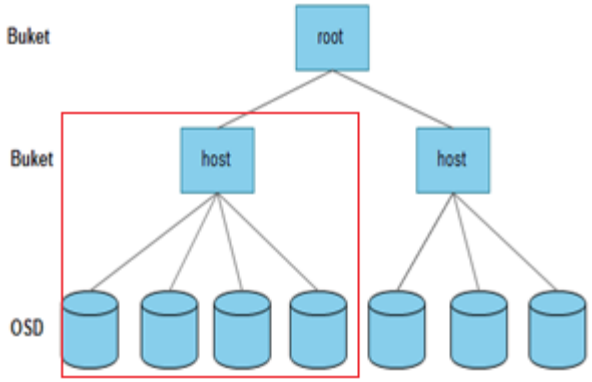
Uniform Bucket

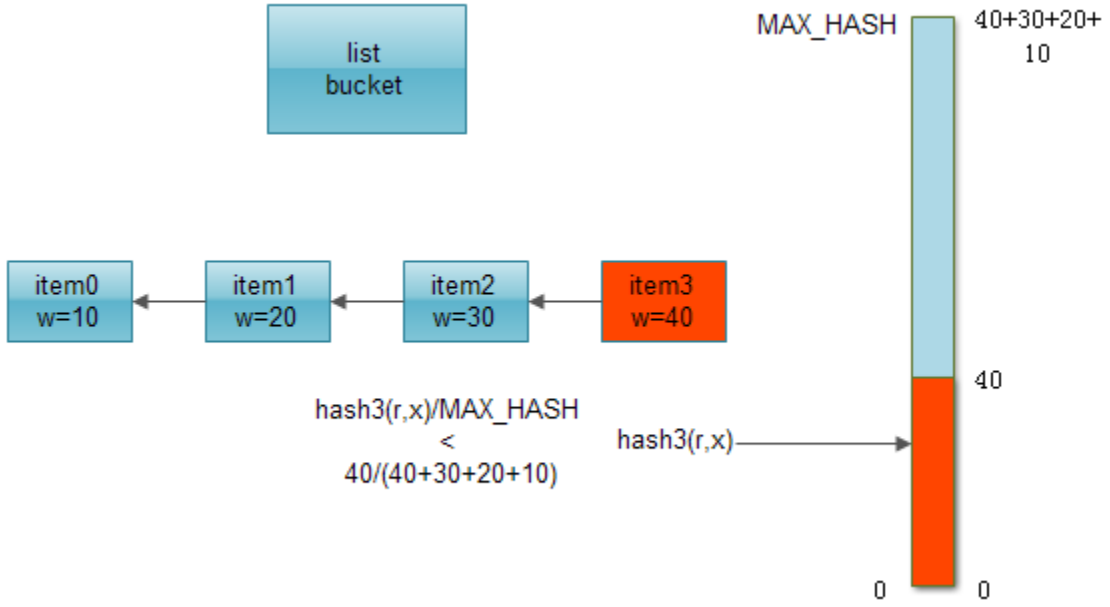
- Data movement when item adds/removes
  - Both poor
- Example
  - Remove



List Bucket

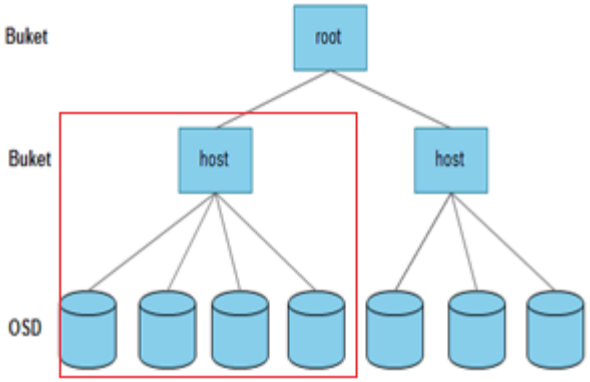
- Nested item weight
  - Items with arbitrary weights
- Algorithm
  - Structure their contents as a linked list
- Complexity
  - $O(n)$
- Example

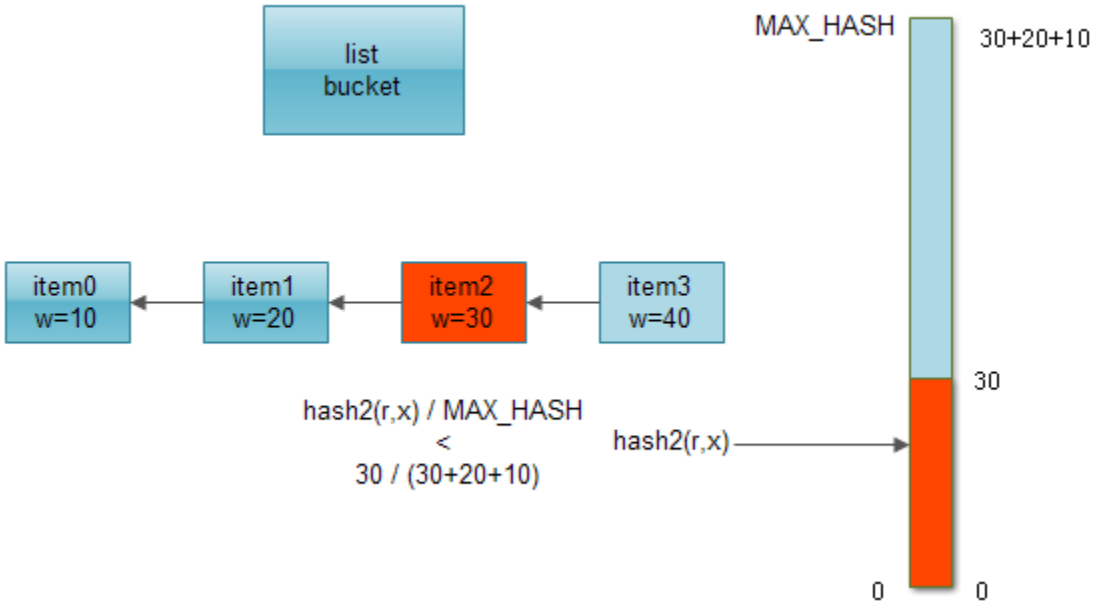




List Bucket

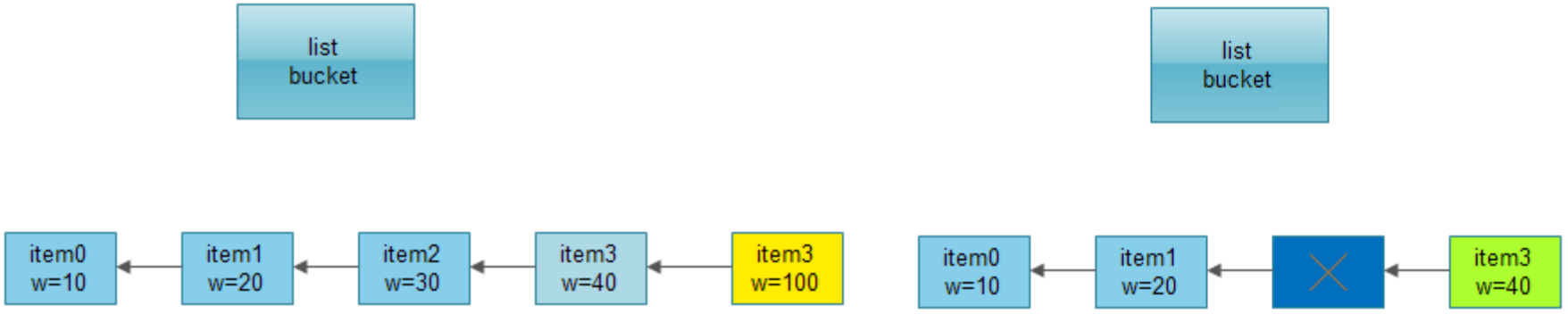
- Nested item weight
  - Items with arbitrary weights
- Algorithm
  - Structure their contents as a linked list
- Complexity
  - $O(n)$
- Example





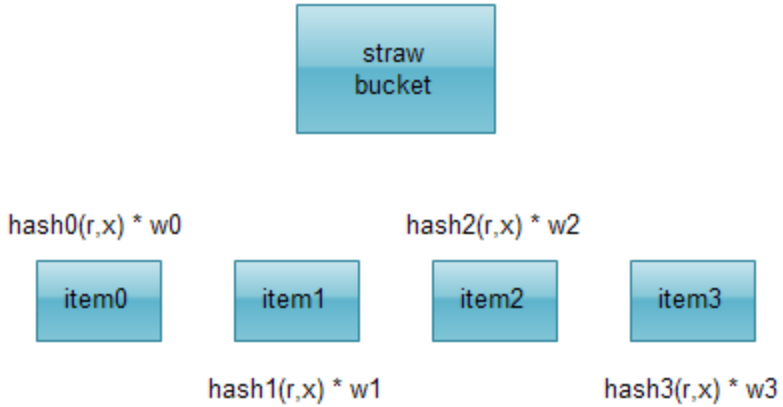
List Bucket

- Data movement when item adds/removes
  - Addition optimal, removal poor
- Example



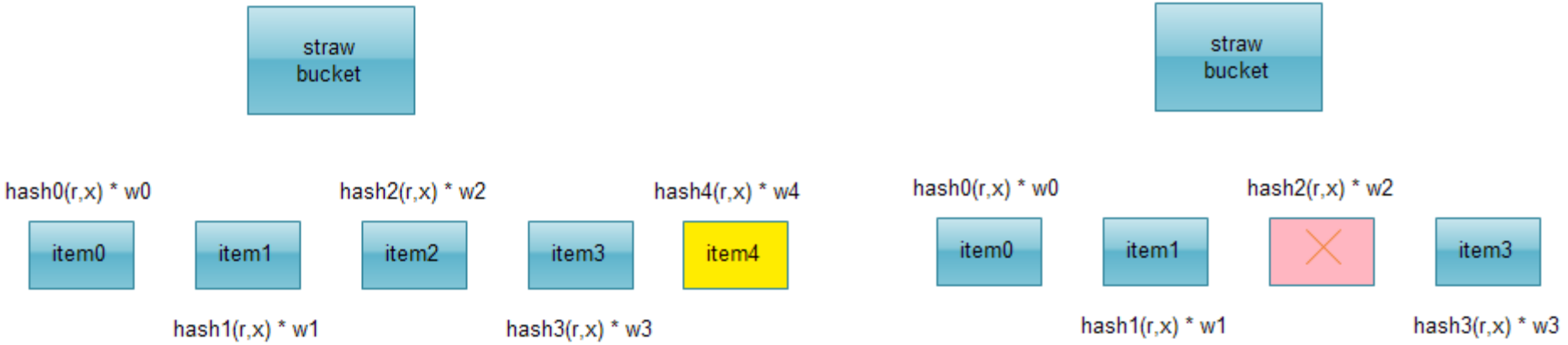
Straw Bucket

- Nested item weight
  - Items with arbitrary weights
- Algorithm
  - Compare hash value * weight
- Complexity
  - $O(n)$
- Example



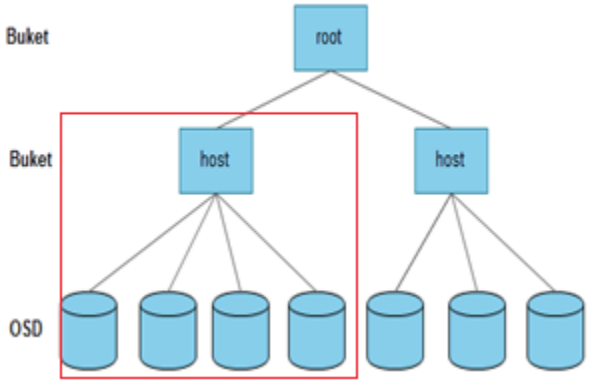
Straw Bucket

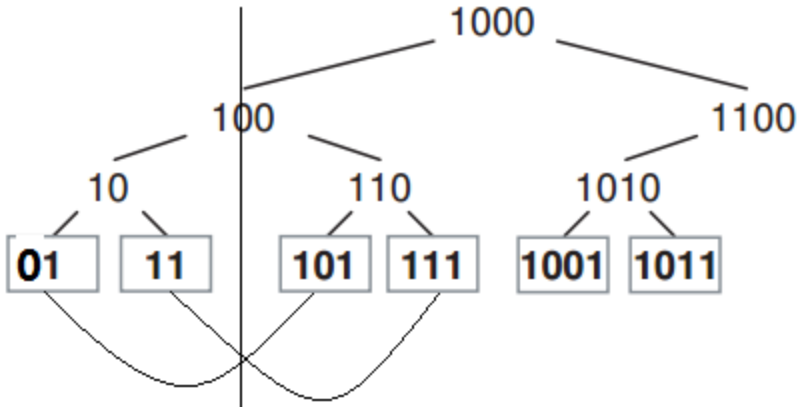
- Data movement when item adds/removes
  - Both optimal
- Example



Tree Bucket

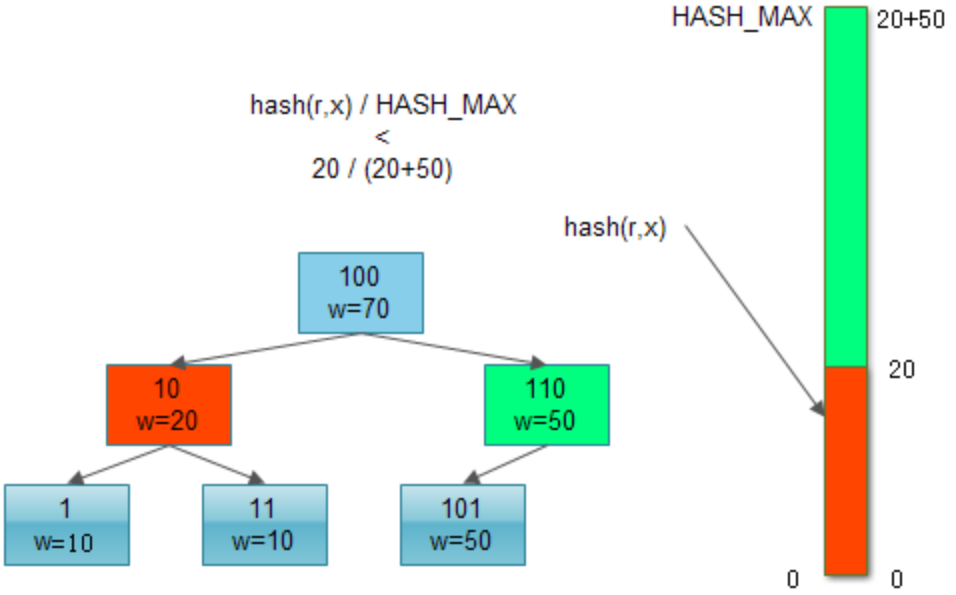
- Nested item weight
  - Items with arbitrary weights
- Algorithm
  - Structure their contents as a binary tree
- Complexity
  - *O*(*log n*)
- Example





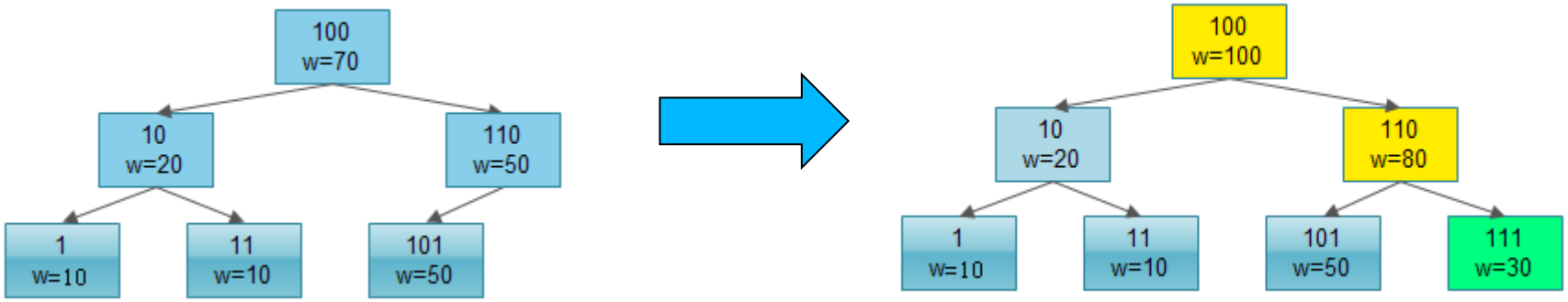
Tree Bucket

- Nested item weight
  - Items with arbitrary weights
- Algorithm
  - Structure their contents as a binary tree
- Complexity
  - *O*(*log n*)
- Example



Tree Bucket

- Data movement when item adds/removes
  - Both good but not optimal
- Example
  - Add item



Bucket Choice

Action	Uniform	List	Tree	Straw
Speed	O(1)	O(n)	O(log n)	O(n)
Additions	poor	optimal	good	optimal
Removals	poor	poor	good	optimal

- Uniform bucket with the fixed circumstances, *e.g.* a shelf of identical disks
- List bucket suit in circumstances only expected to expand
- Straw bucket suits in where addition and removal are critical
- Tree buckets are an all around compromise, providing excellent performance and decent reorganization efficiency

CRUSH Review

- Main component
  - Cluster map
  - Placement rules
  - Bucket types