MAPX: Controlled Data Migration in the Expansion of Decentralized Object-Based Storage Systems

Li Wang
laurence.liwang@gmail.com
Didi Chuxing

Yiming Zhang
sdiris@gmail.com
(Corresponding)
NiceX Lab, NUDT

Jiawei Xu
titan_xjw@cs.sjtu.edu.cn
SJTU

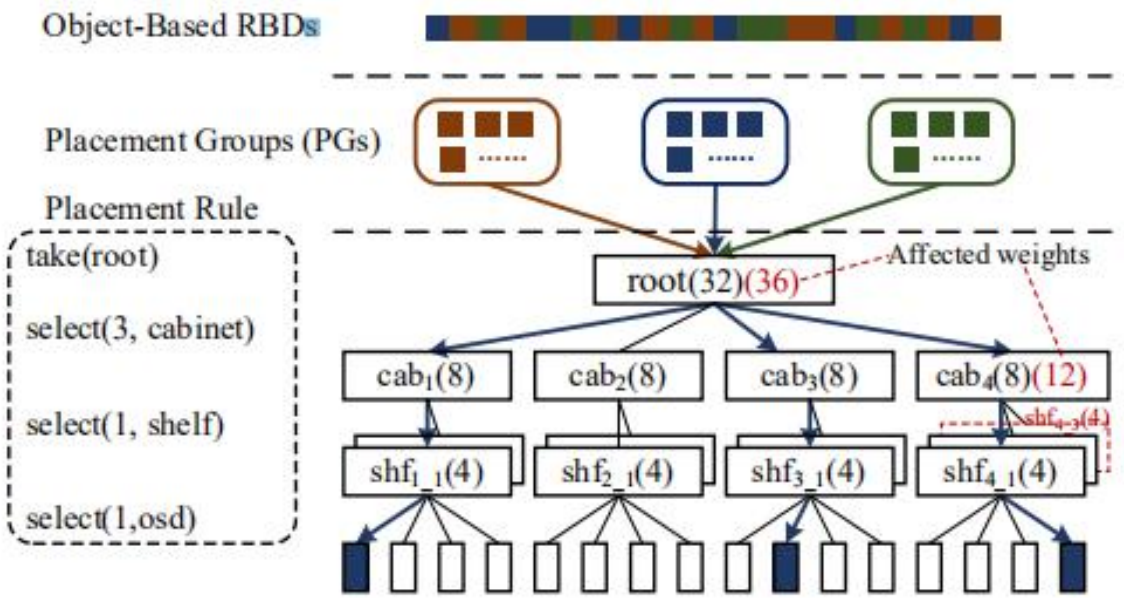
Guangtao Xue
xue-gt@cs.sjtu.edu.cn
SJTU

FAST 2020

Background

- **Object-based storage systems**
  - view various datas as different object
  - data placement scheme: Centralized VS Decentralized

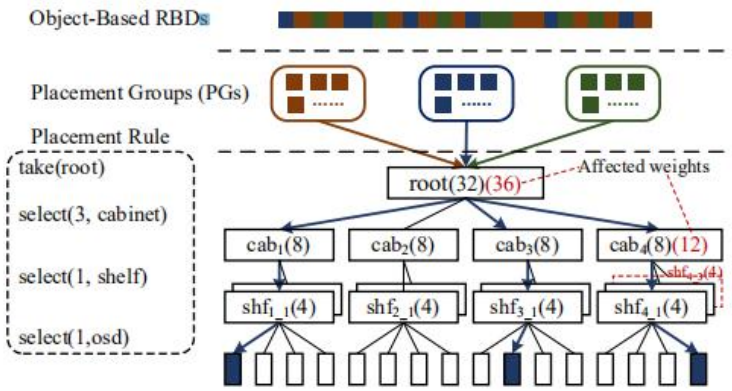
- **CRUSH**

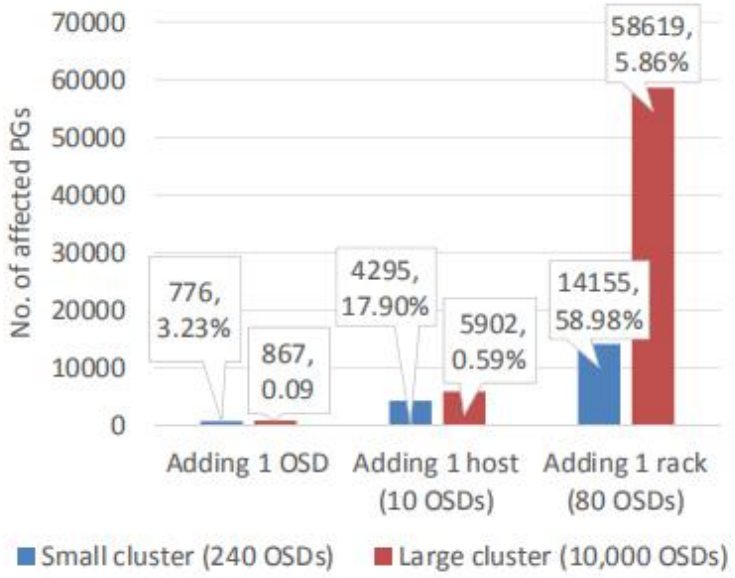


Background

➢**CRUSH**

- high scalability, robustness, and performance
- uncontrolled data migration after expanding the clusters causes significant performance degradation





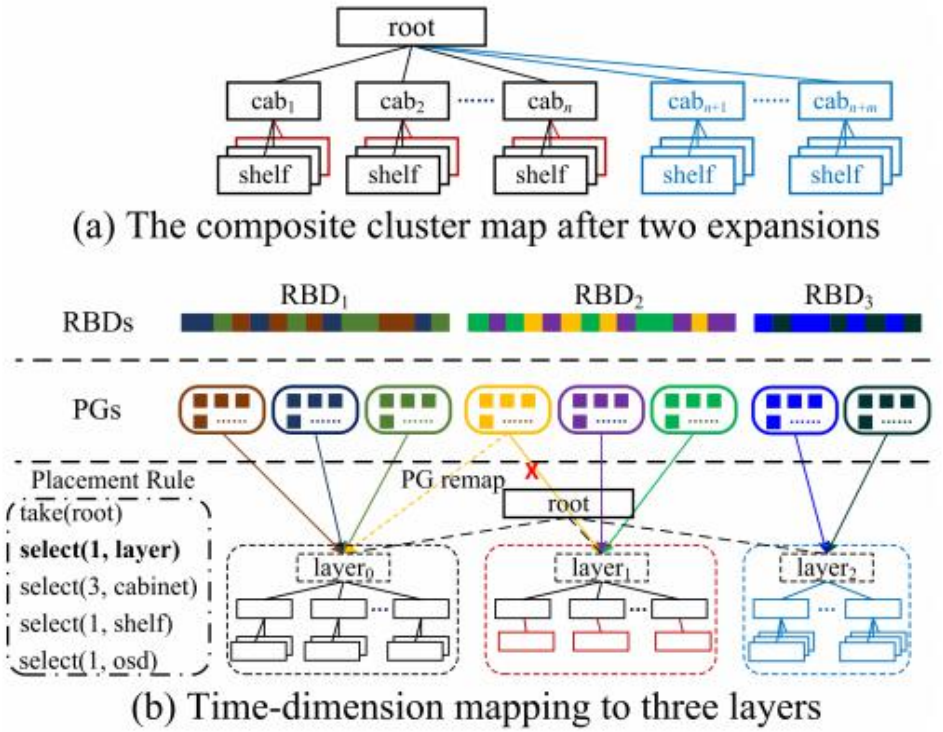
Motivation

- **Improving performance while enjoying CRUSH's advantage**
  - centralized placement methods

  — to keep existing objects unaffected during expansions and place only new objects onto the newly-added OSDs
  - Based on CRUSH

Main Idea

- **Layer**
  - Each expansion is viewed as a new layer of the CRUSH map represented by a virtual node beneath the CRUSH root.

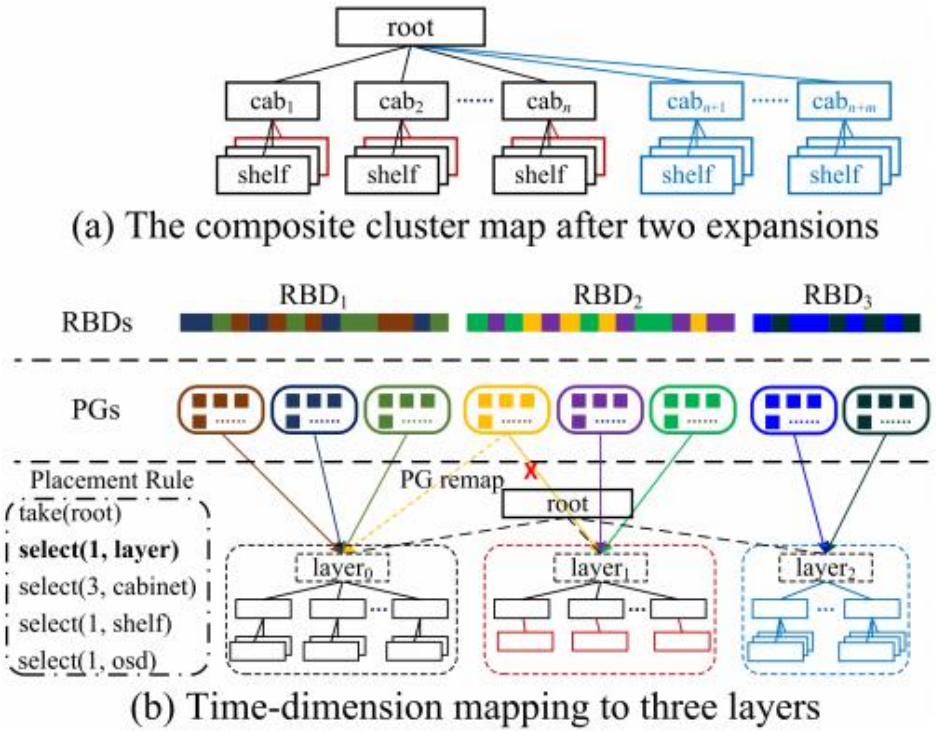


Main Idea

➢Timestamp

- uses an extra dimensional mapping (from object creation times to cluster expansion times) for controllable data migration in the expansion
- Limition:

The time-dimension mapping cannot support general object storage,but object-based storage scenarios (such as block storage and file storage)



Macro design

- **Migration-Free Expansion**
  - mainly application scenario,provides load balancing within each layer
- **Migration Control**
  - removals of objects, failures of OSDs, or workload changes……

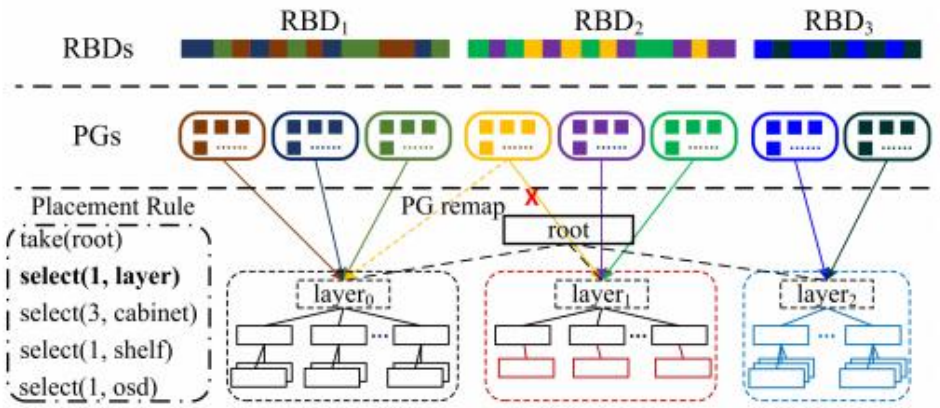
Migration-Free Expansion

➢ Mapping objects to PGs

- The new layer is assigned with a certain number of newly-created PGs
- Each Pg has a timestamp (tpgs) equal to the layer's expansion time (tl)
- Write/Read :Compute object's pgid

$$pgid = Hash(name) \bmod \text{INIT\_PG\_NUM}[j] + \sum_{i=0}^{j-1} \text{INIT\_PG\_NUM}[i],$$

Then mapping, tl ≤ to , tpgs ≤ to



(b) Time-dimension mapping to three layers

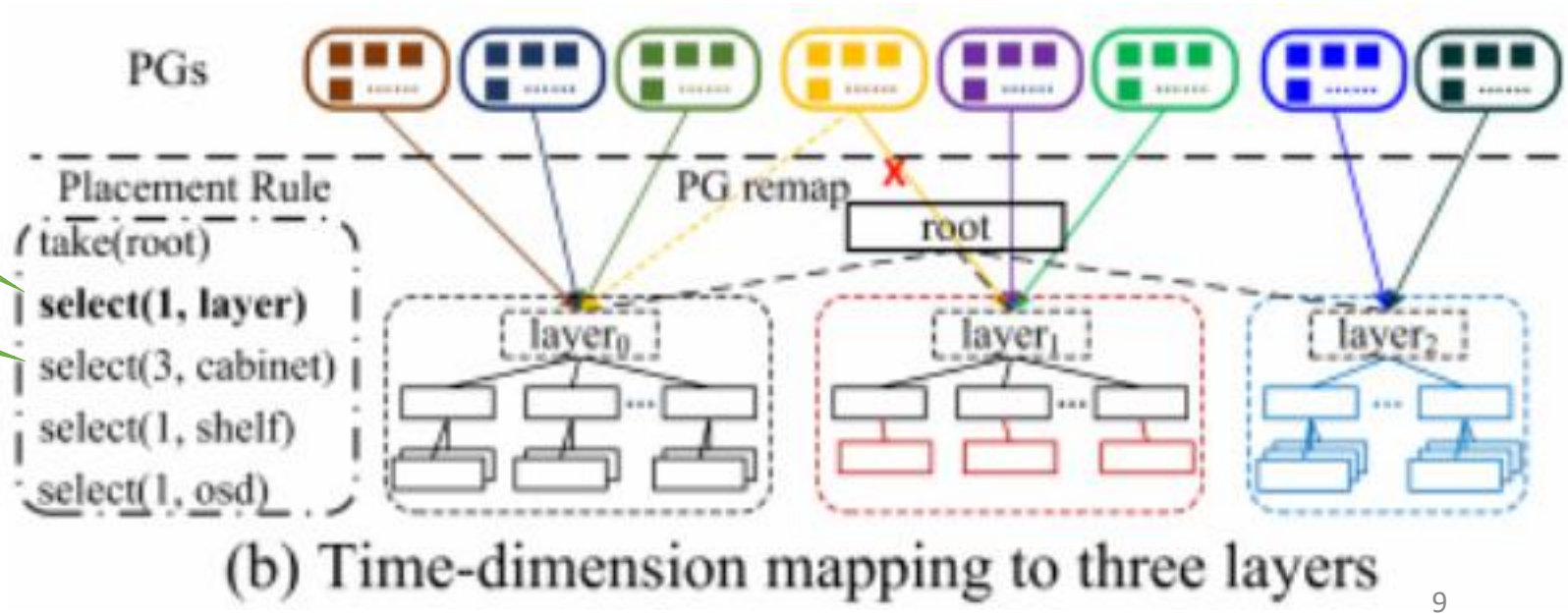
Migration-Free Expansion

➢Mapping PGs to OSDs

- Similar to CRUSH, MAPX maps aPG onto a list of OSDs following a sequence of operations in a user-defined placement rule
- Note:

Algorithm 1

if it has two cabs in the new layer



(b) Time-dimension mapping to three layers

Migration-Free Expansion

- **Algorithm 1**

Algorithm 1 Extended `select` Procedure of MAPX

```
1: procedure SELECT(number, type)
2:     if type ≠ "layer" then
3:         return CRUSH_SELECT(number, type)
4:     end if
5:     layers ← layers beneath currently-processing bucket
   ▷ each layer represents an expansion
6:     num_layers ← number of layers in layers
7:     pg ← current Placement Group
8:     o⃗ ← Φ                                   ▷ output list
9:     for (i = num_layers − 1; i ≥ 0; i − −) do
10:        layer ← layers[i]
11:        if layer.timestamp ≤ pg.timestamp then
12:            if layer was chosen by previous select then
13:                continue
14:            end if
15:            o⃗ ← o⃗ + {layer}
16:            number ← number − 1
17:            if number == 0 then
18:                break
19:            end if
20:        end if
21:    end for
22:    return o⃗
23: end procedure
```

Migration Control

➢PG remapping

- two timestamps :

  a static timestamp (tpgs) 、 a dynamic timestamp (tpgd)

- tpgd that could be set to any layer’s expansion time

➢Cluster shrinking

- view removing the layer’s devices as an inverse operation of expansions
- PG remapping

➢Layer merging

- just change layer's expansion time.

Implement

➢ Applicable scene

- not suitable for general object stores,but a large variety of object based storage systems

➢ Ceph-RBD

- the metadata-based timestamp retrieval mechanism： add timestamp in the rbd_header structure

➢ CephFS

- through it's inode

Evaluation

➢Major verification issues

- performance compared with CRUSH

➢Test tools

- fio benchmark
- four machines, three machines run the Ceph OSD storage servers and one runs the client. Each machine has dual 20- core Xeon E5-2630 2.20GHz CPU, 128GB RAM, and one 10GbE NIC, running CentOS 7.0. Each storage machine, installs four 5.5TB HDDs, and runs Ceph 12.2(Luminous) with the BlueStore backend.

➢Other

- OSD_max_backfills:characterize migration priority

Evaluation

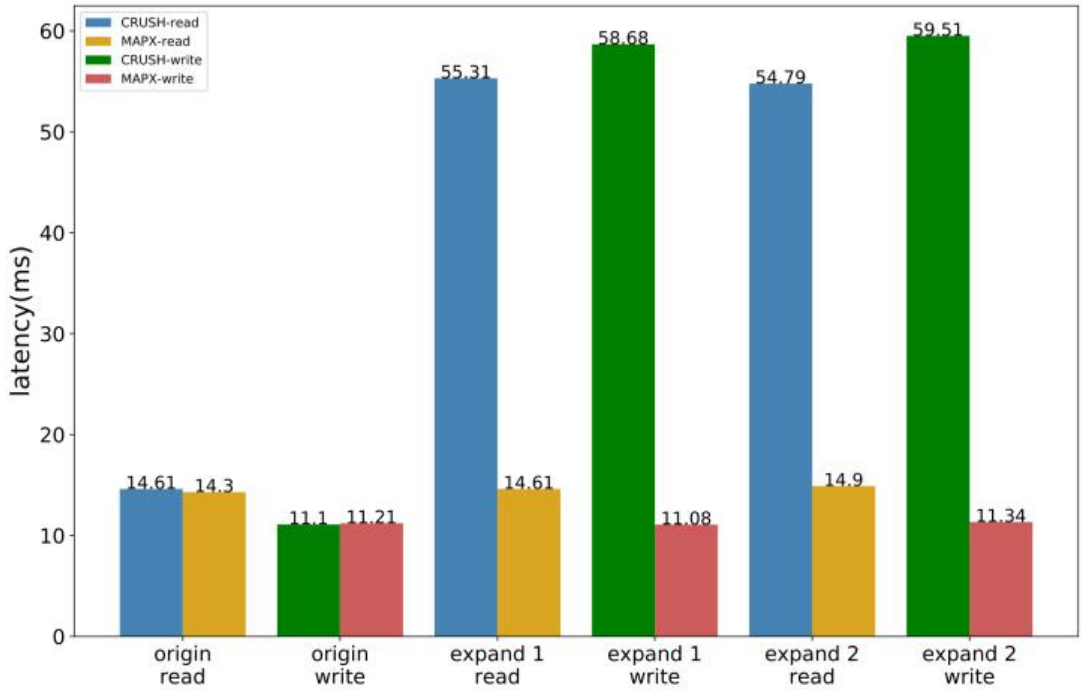


Figure 4: 99th percentile I/O latency of MAPX and CRUSH (during cluster expansions).

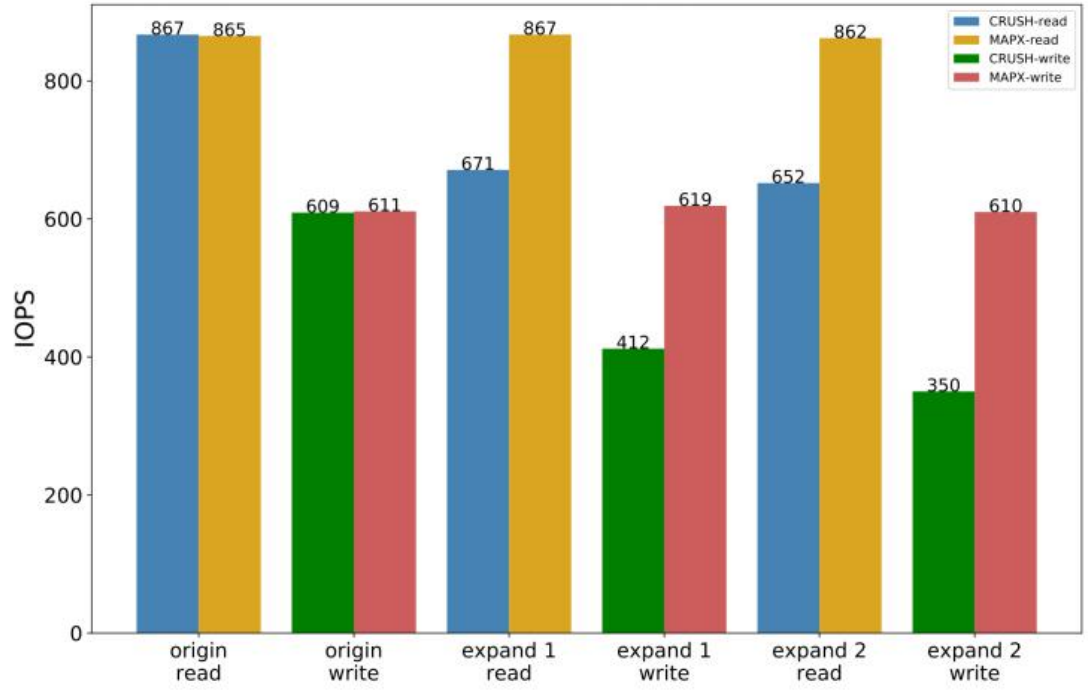


Figure 5: IOPS of MAPX and CRUSH (during cluster expansions).

Evaluation

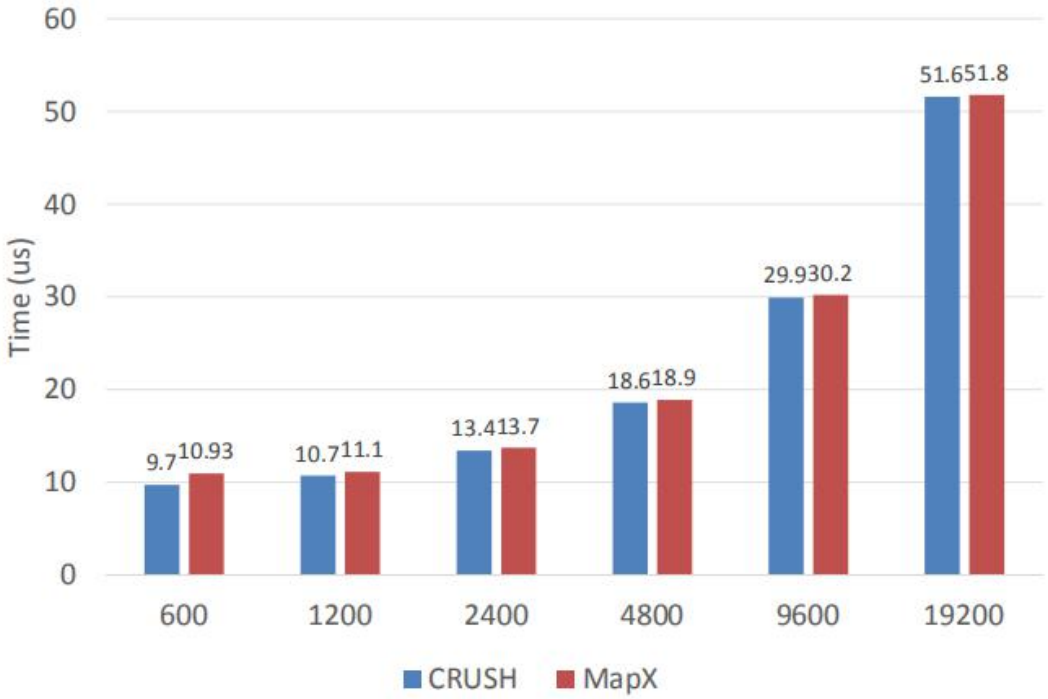


Figure 6: Computation overhead of MAPX and CRUSH.

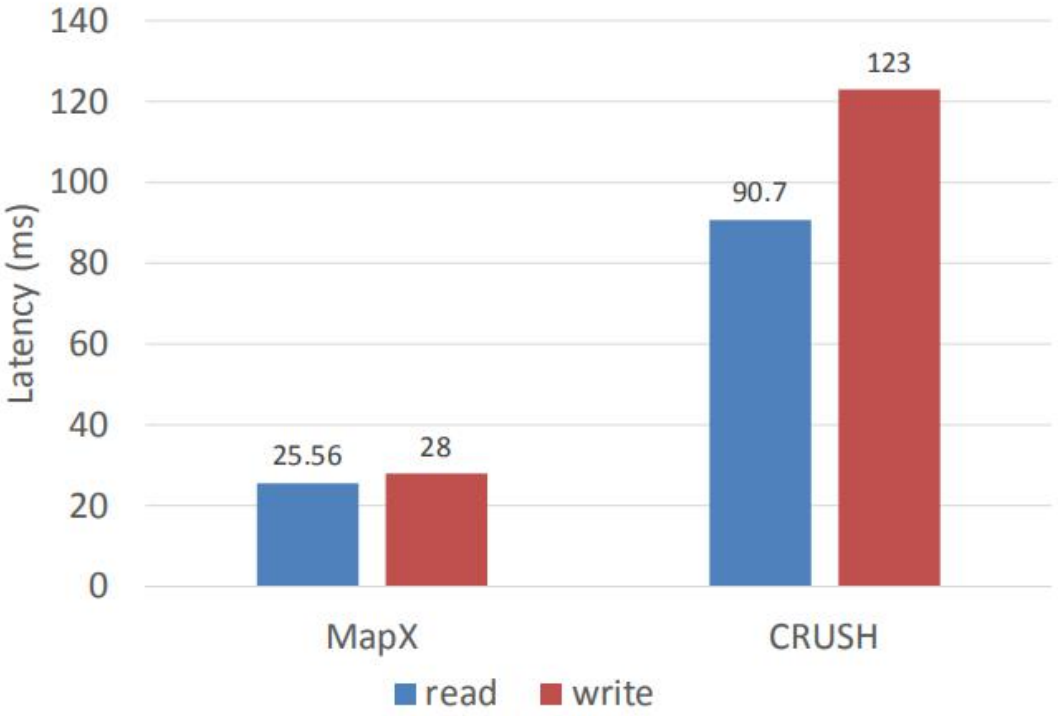


Figure 7: 99th percentile I/O latency of MAPX and CRUSH (during cluster shrinking).

➢**the higher come from the computation of the time-dimension mapping beneath the root**

Evaluation

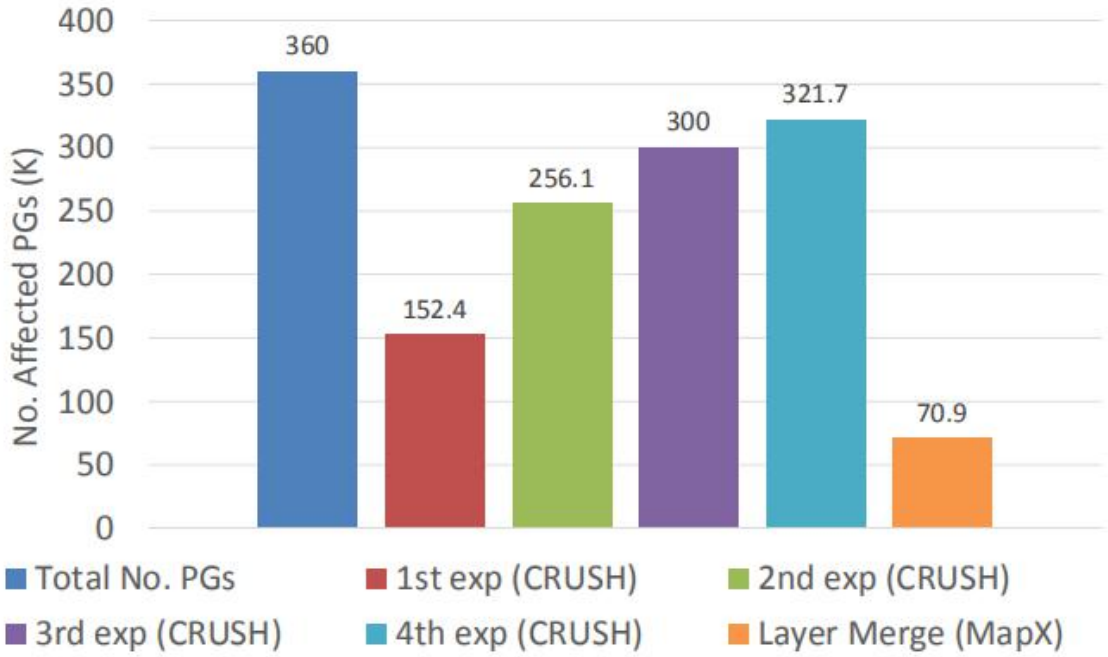


Figure 8: Number of affected PGs in layer merging in MAPX (after four expansions). Since CRUSH does not support merging, for reference we measure the number of affected PGs after each expansion in CRUSH.

Conclusion

- **MAPX:a novel extension to CRUSH that embraces the best of both decentralized and centralized methods**