Implement Object Storage with SMR based Key-Value Store

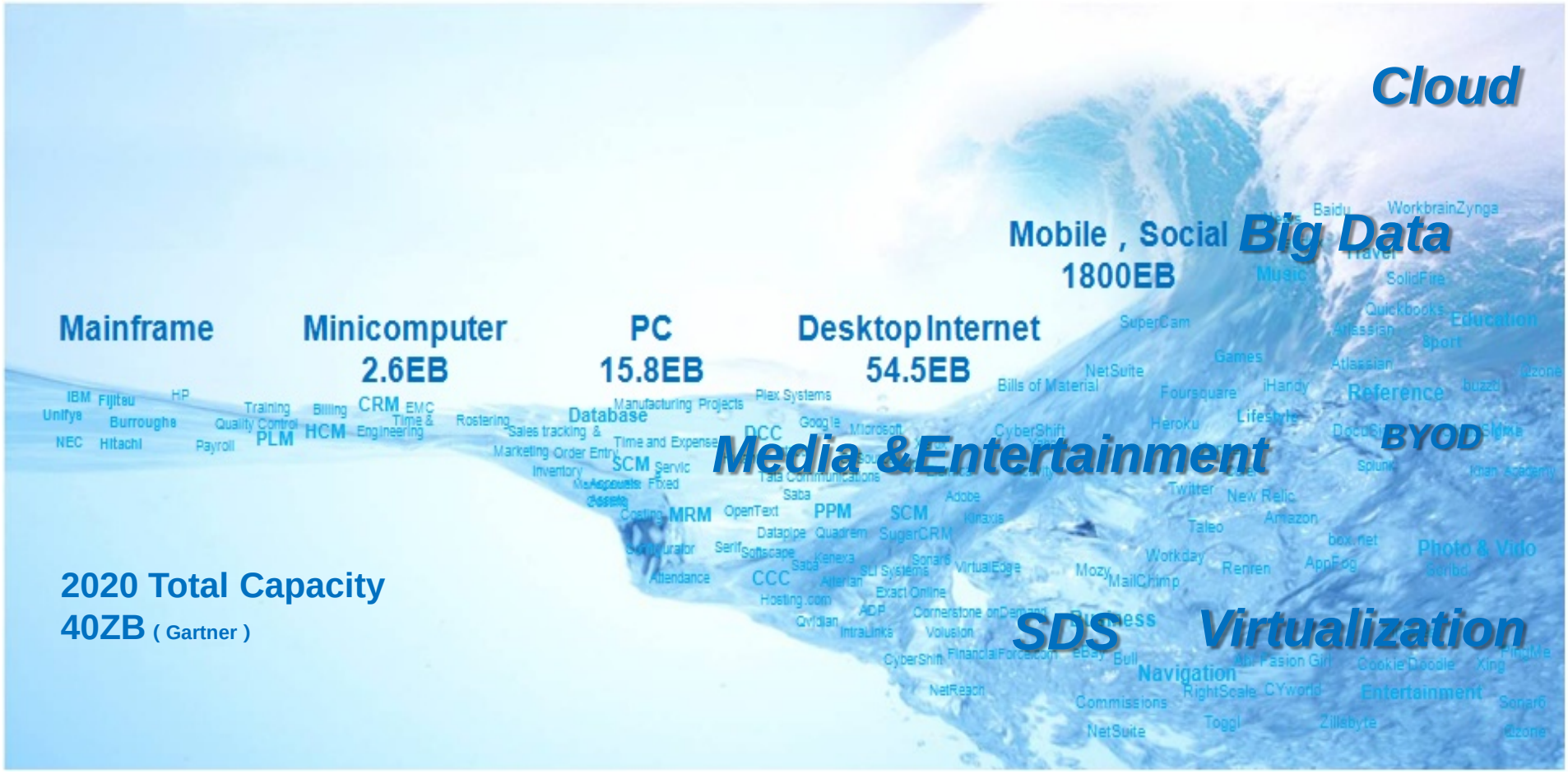
luoqingchao@huawei.com
thunder.zhang@huawei.com

Huawei Technologies Co.

Agenda

- Object Storage Market Overview
- Object Storage Design with SMR based Key-Value Store
- Summary Future Works

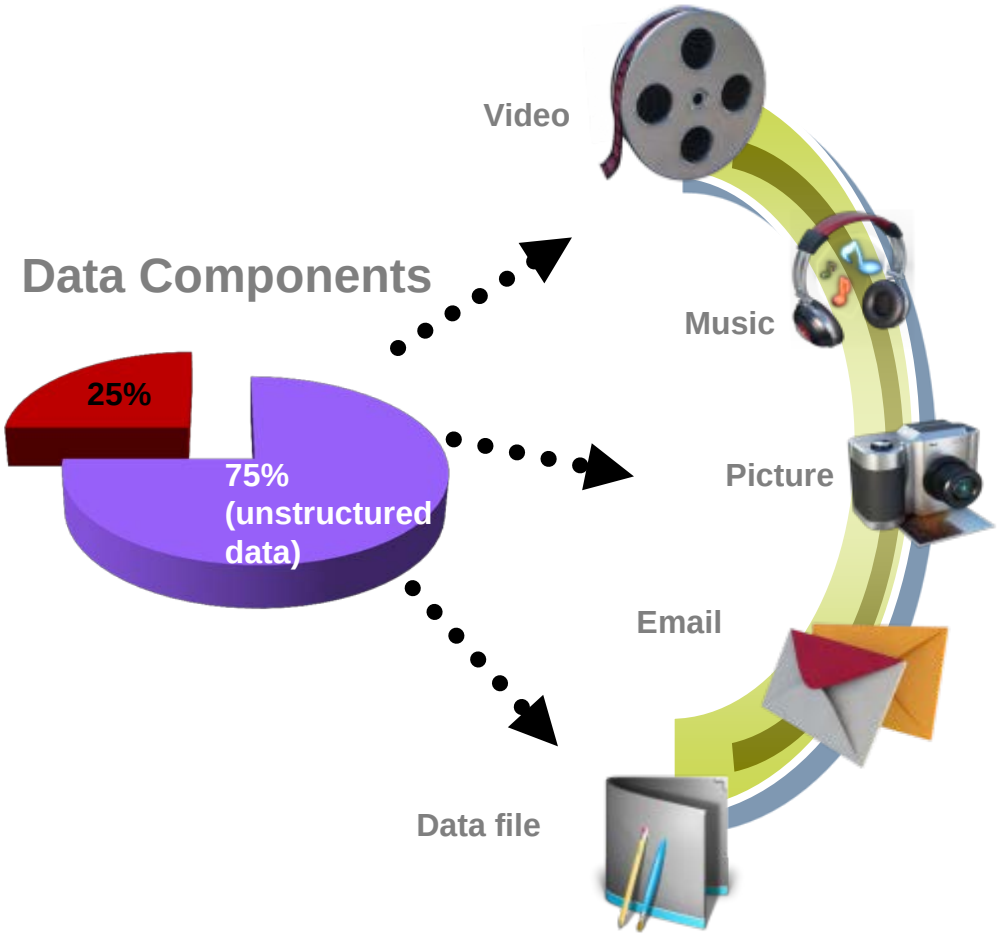
Massive Data Storage Trend



A Revolution That Will Transform How We Live, Work, and Think ……

——Kenneth Cukier

Components and Characteristics of Massive Data



- **Seldom** updated
- **Undefined** value
- **Large** capacity and **high** growth speed
- **Long** storage time

Object Storage Technology matches these requirements

SMR matches Object Storage Market

- **Object Storage Requirement**
  - Huge volume need large capacity drive.
  - Competitive TCO need cheap storage media
  - Write once few modification matches SMR write out-of-place feature

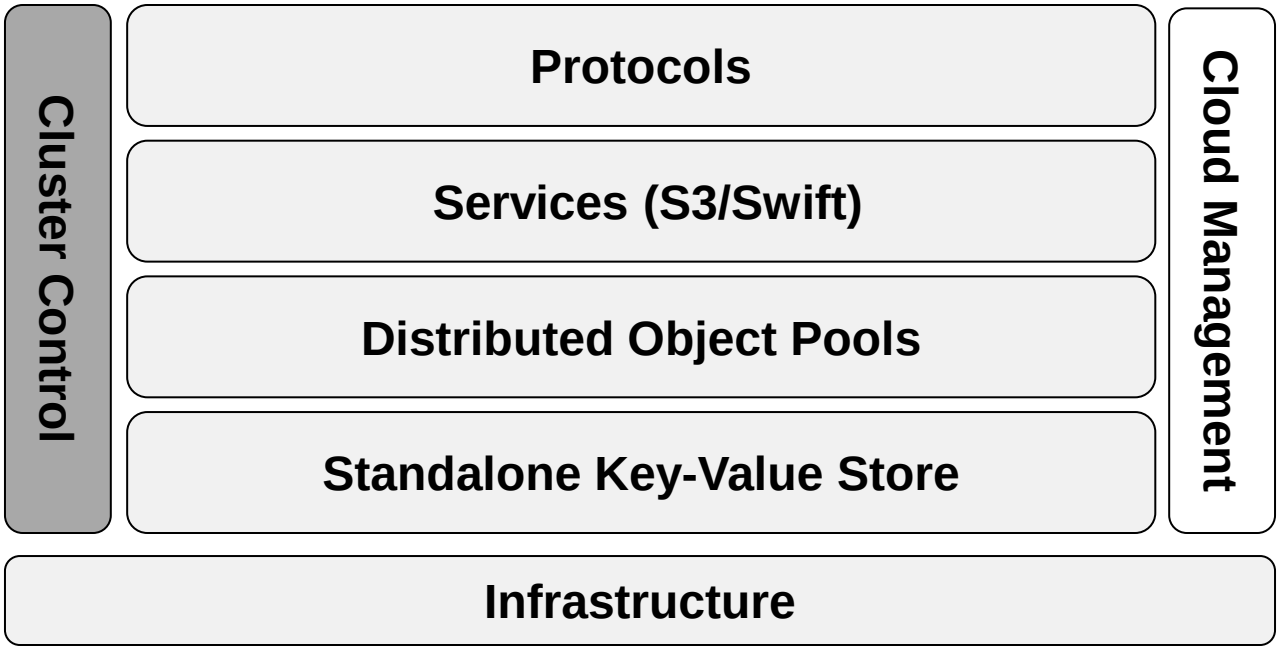
- **SMR Technology Background**
  - Type – Drive managed, Host Aware, Host Managed
  - Standard – ZBC, ZAC
  - Industry – all HDD vendors will release SMR 2015~2016

Huawei cooperation with HDD vendors on SMR:
http://events.linuxfoundation.org/sites/events/files/slides/SMR%20in%20Linux%20Systems%20-%20Vault.pdf
http://www.hgst.com/company/media-room/press-releases/HGST-Delivers-Worlds-First-10TB-Enterprise-HDD-for-Active-Archive-Applications

Agenda

- Object Storage Market Overview
- Object Storage Design with SMR based Key-Value Store
- Summary and Future Works

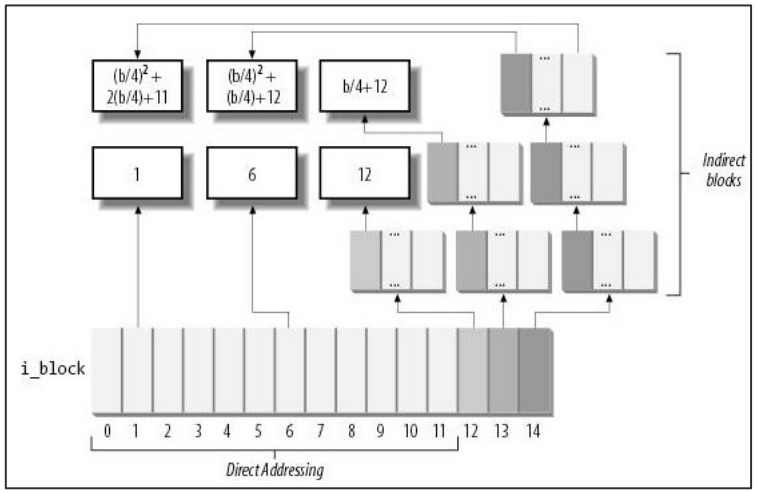
Huawei Object Storage Architecture



- Standalone Key Value Store, Provide simple KV access on HDD/SDD
- Distributed Object Pools, Provides redundant KV access, like replica and Erasure code
- Service, Provide S3/Swift like access

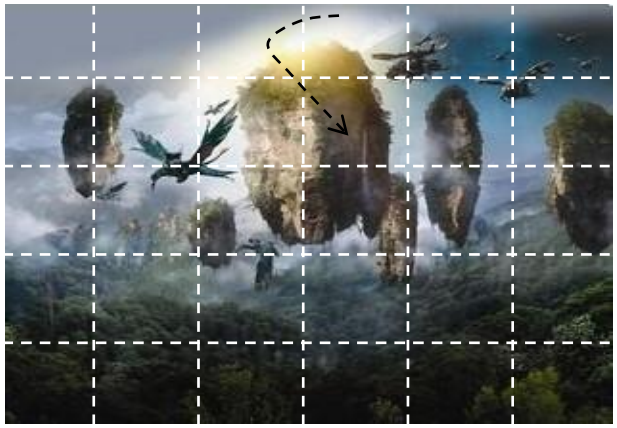
Why Key-Value(KV) based not Logical Block Address(LBA)

metadata



VS.

get() / put()



Complicated
(Huge Metadata)

Simple
(LBA)

Distribution Layer

Standalone Drive Layer

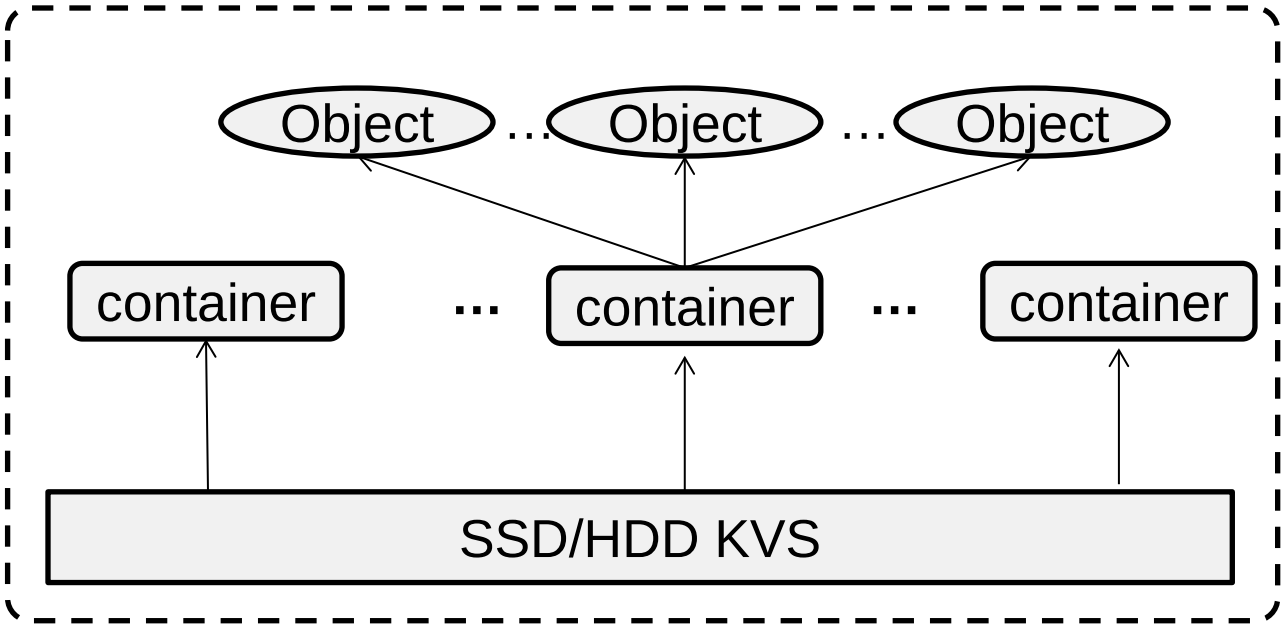
Simple
Name Policy

Complicated
(KV)

LBA=Logical Block Address

KV=Key Value

Huawei Key Value Store(KVS) Data Model



- One KVS has multiple containers/Pools
- Every Container has it own policy, like key size, value size, shared allocation/ reserved allocation, delete policy, etc…
- Access Object by KV API, Object can store metadata

Key Value Store(KVS) API

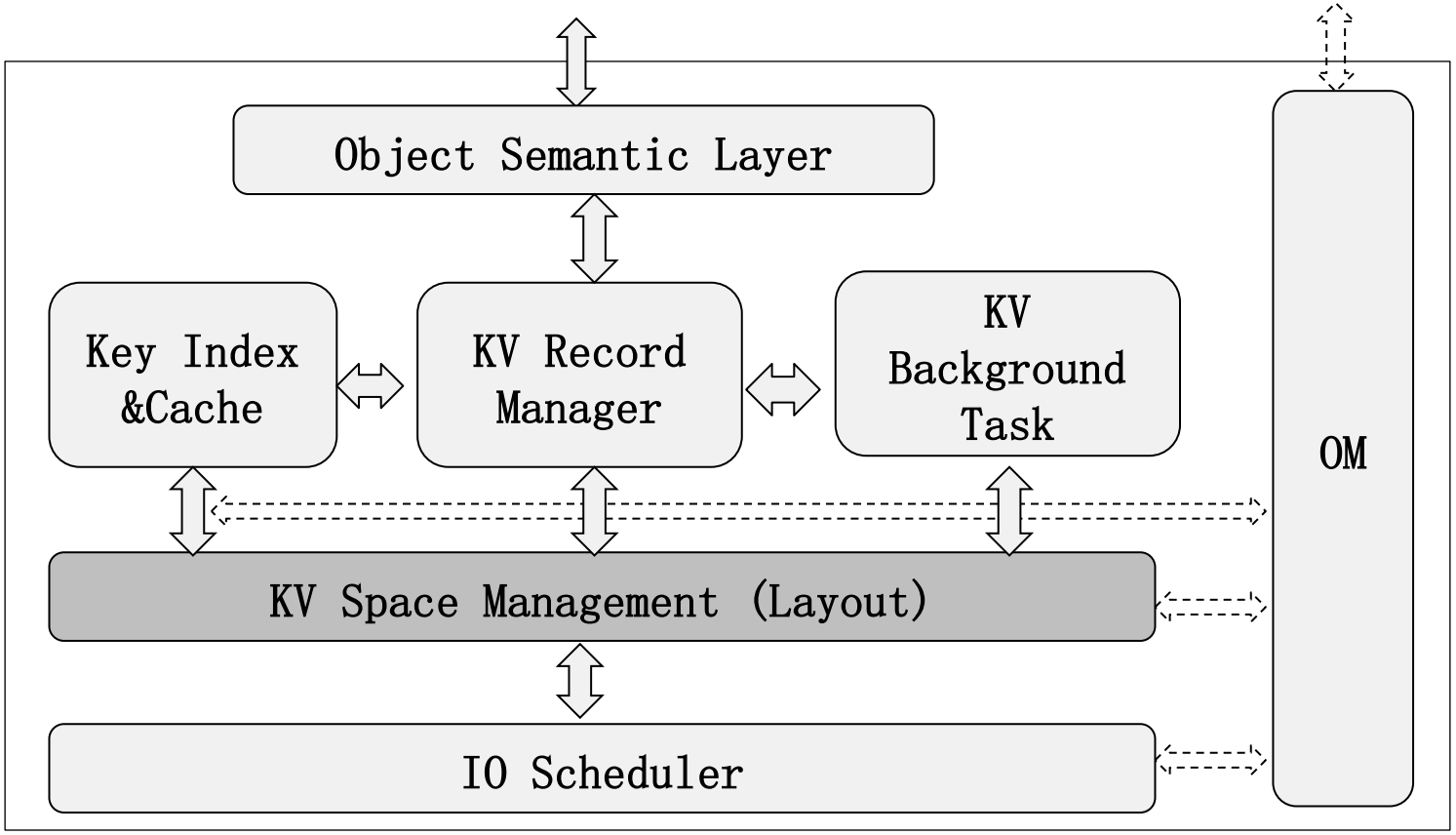
Pool (Container) operations:

- Create(name, config_file, pl_id)
- Destroy(name)
- Open(name, mode, pl_id)
- Close(pl_id)
- Set_prop(pl_id, prop, value)
- Get_prop(pl_id, prop, value)
- Get_stats(pl_id, stats)
- Xcopy(src, dest, flag, regex, regex_len)
- …

Object operations:

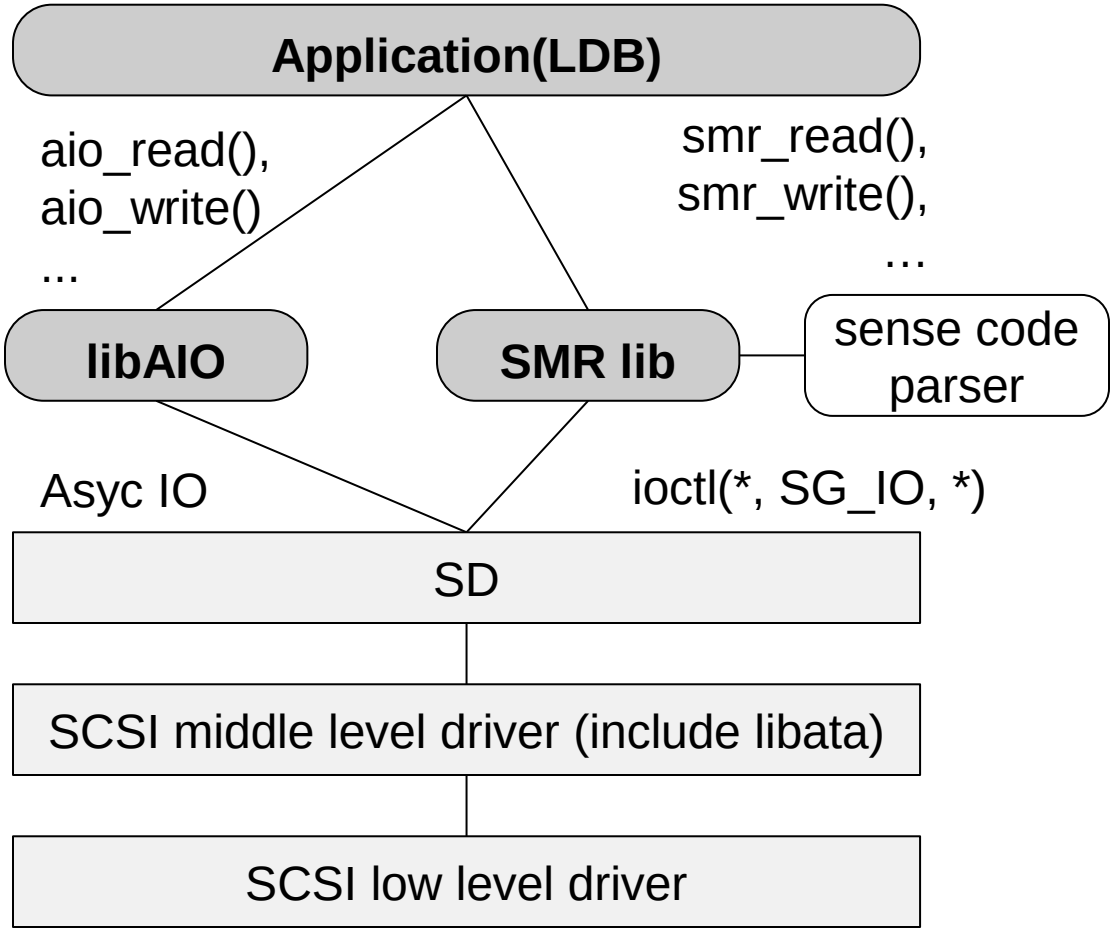
- put (pl_id, key, value, kv_props, put_opts)
- get(pl_id, key, value, kv_props, get_opts)
- del(pl_id, key, value, kv_props, del_opts)
- Iter_open(pl_id, flag, regex, regex_len, limit, *iter_id)
- Iter_next(pl_id, iter_id, kvarray)
- Iter_close(pl_id, iter_id)

KVS core --- LDB (Log structured DB) Modules



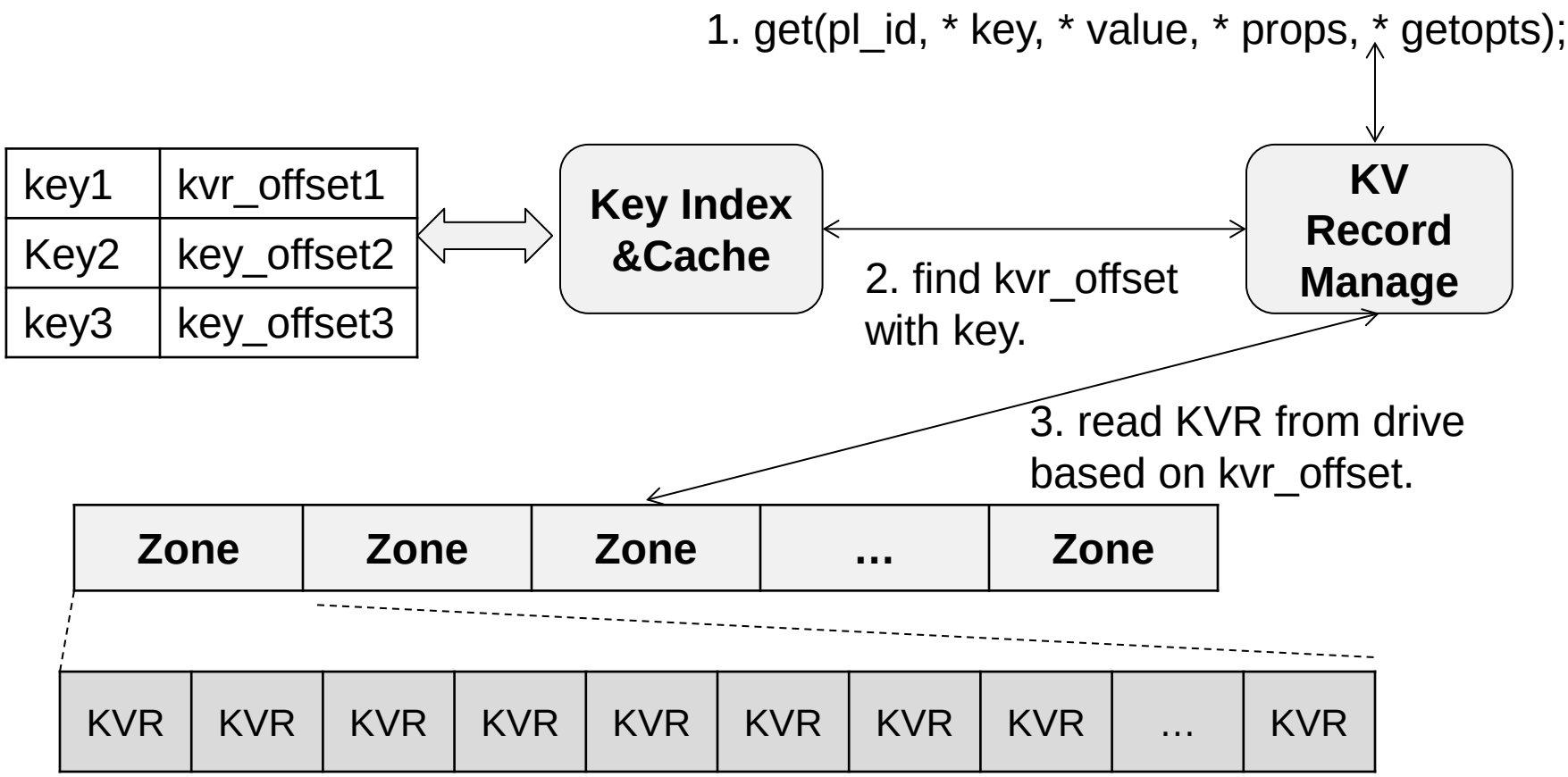
- OM, Operate & Maintenance
- Layout & Scheduler provides SMR write-out-of place allocation and IO stack.

SMR IO stack



- AIO for parallel Access
- SMR lib for SMR new commands in user space
- AIO only get EIO when error, SMR lib can get sense code then parse detail error

LDB KV access Overview



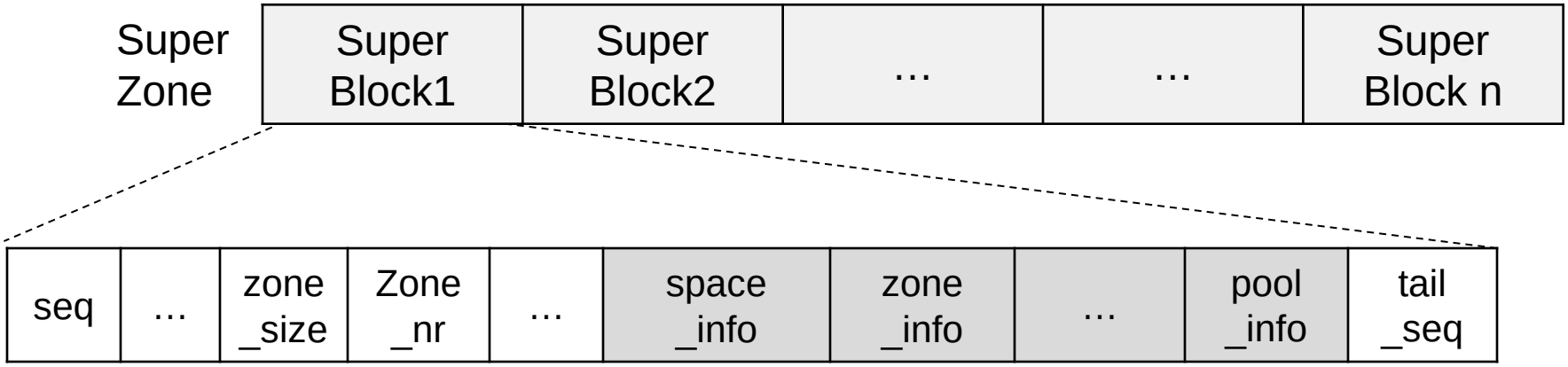
- Drive is divided into zones, and zone size align media characteristics (256M for SMR).
- Store KVR(Key Value Record) in Zone.

Zone based Layout

Super Zone	Data Zone	Date Zone	…	Data Zone	Reserved Zone

Zone Type	Sub Zone Type	Function
Super Zone	Super Zone	Store meta data information
Data Zone	Index Zone	Store memory index checkpoint to make boot faster
	KVR Zone	Store generic Object KVR information
	Tombstone Zone	Store deleted Object KVR information
Reserve Zone	Reserve Zone	For Add new functions in the future

Super Zone



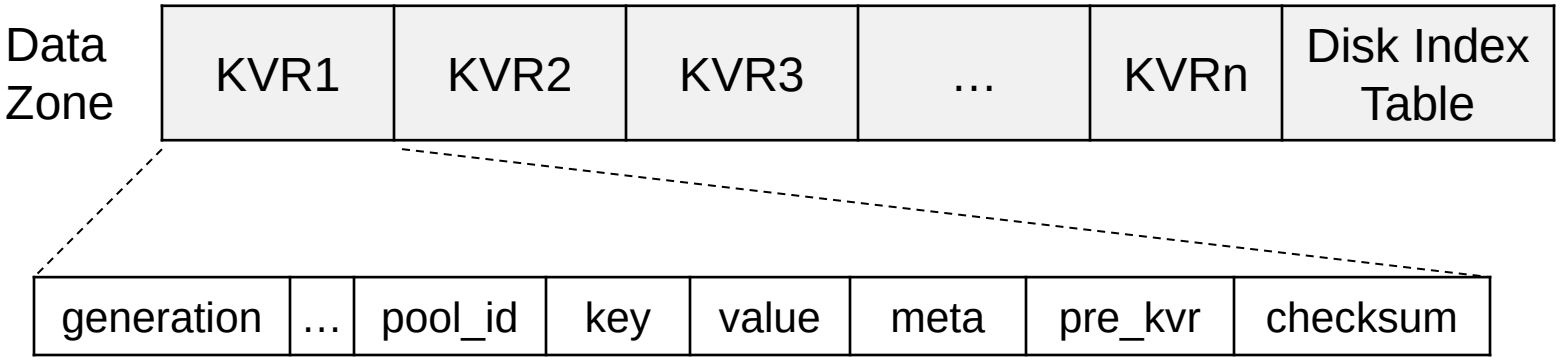
- One Super Zone has many super block (SB), every SB stores LDB metadata of one specified time, seq is used to record the sequence of SB.

- There are more than one super zone, and the super block is stored sequentially with log style, not overwrite; after write new super zone the old super zone can be reset and reuse again.

Detail Information of SB

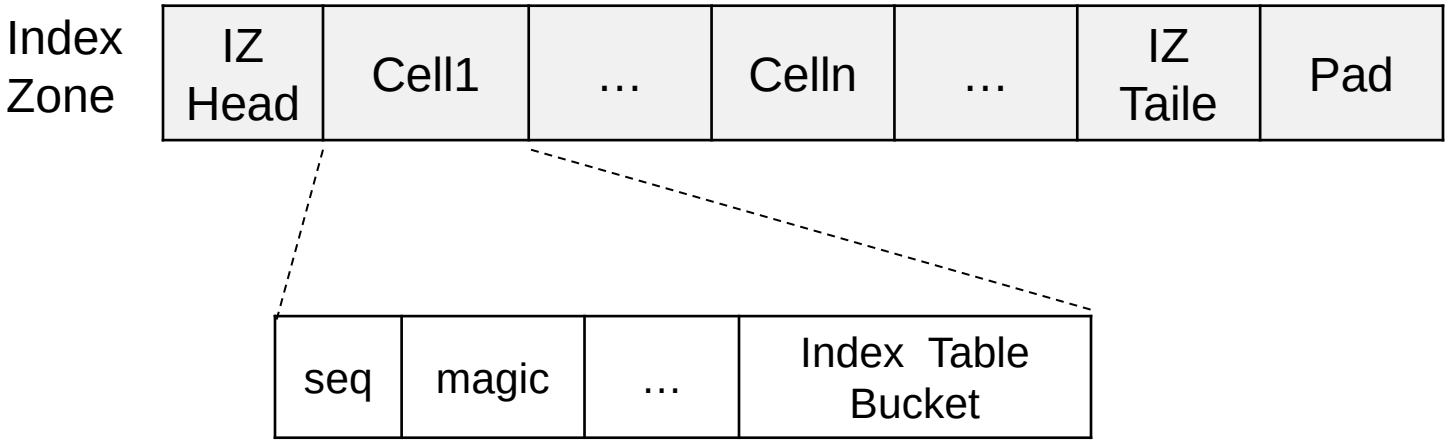
- **Space Info**
  - Allocation statistics
  - KVR statistics
  - garbage information
- **Zone Info**
  - Index zone info
  - Data zone info
  - Tombstone info
- **Pool Info**
  - Pool numbers
  - Pool name, id, capacity
  - Pool key hints, value hints, policies

KVR in Zone



- Each Object is stored as KVR, and KVR allocation is log style as SMR required write out-of-place.
- KVR has pool_id field, then multiple pool's KVR allocation can share one zone.
- Each KVR can store meta-data, upper layer application will leverage.
- KVR has pre_kvr field when delete/overwrite exist key, at that time it will generate a tombstone KVR in tombstone zone.
- At the end of each data zone, put all the KVR index together as disk index table, for recovery oriented design.

Index Zone Definition



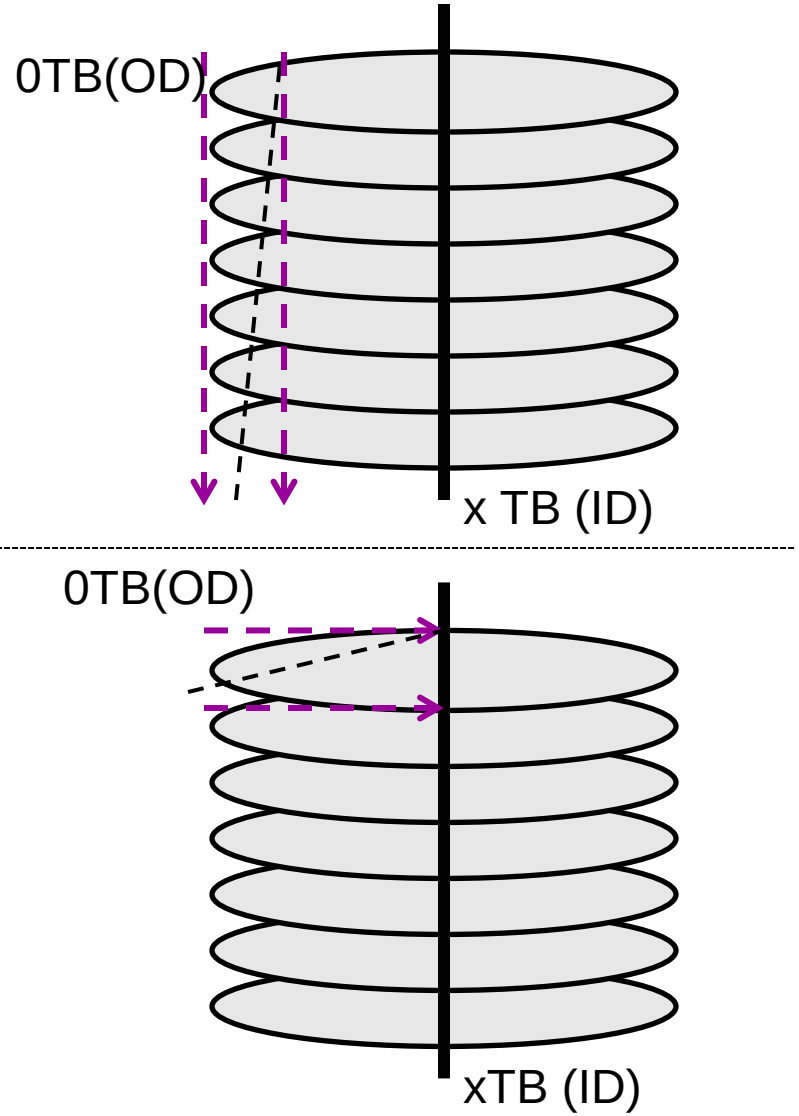
- IZ, Index Zone
- Seq, identify an fully index zone entry.
- magic, index zone entry magic number
- Bucket, Memory index is organized as bucket, each bucket is about1MB
- To make boot faster, memory index will make checkpoint and store into cell with log style.

SMR drive ZONE layout

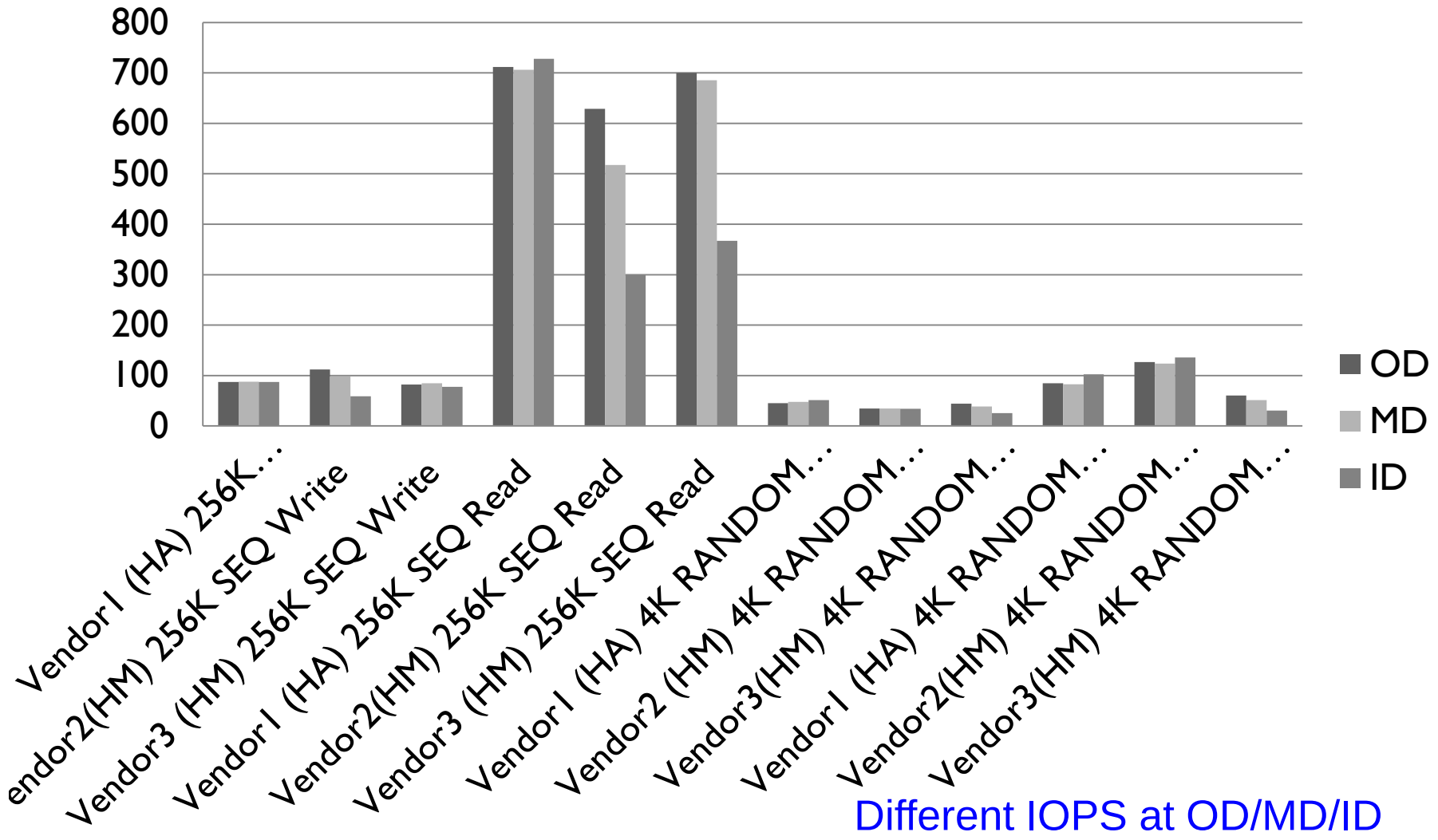
- SMR Drive Throughput depends on ZONE layout on CHS (Cylinder, Head, Sector). Two kinds of zone layout, performance differs at OD(Outer Disc), MD(Middle Disc), ID(Inner Disc)

- Access Zones at OD/MD/ID, then measure performance, random 4K accesses at the head of each zone. And test SMR related command latency.

	Vendor1 (HA)	Vendor2 (HM)	Vendor3 (HM)
report zone	20ms	24ms	388ms
open zone	X	X	X
close zone	X	X	X
write point reset	428ms	1353ms	456ms
finish zone	X	X	X

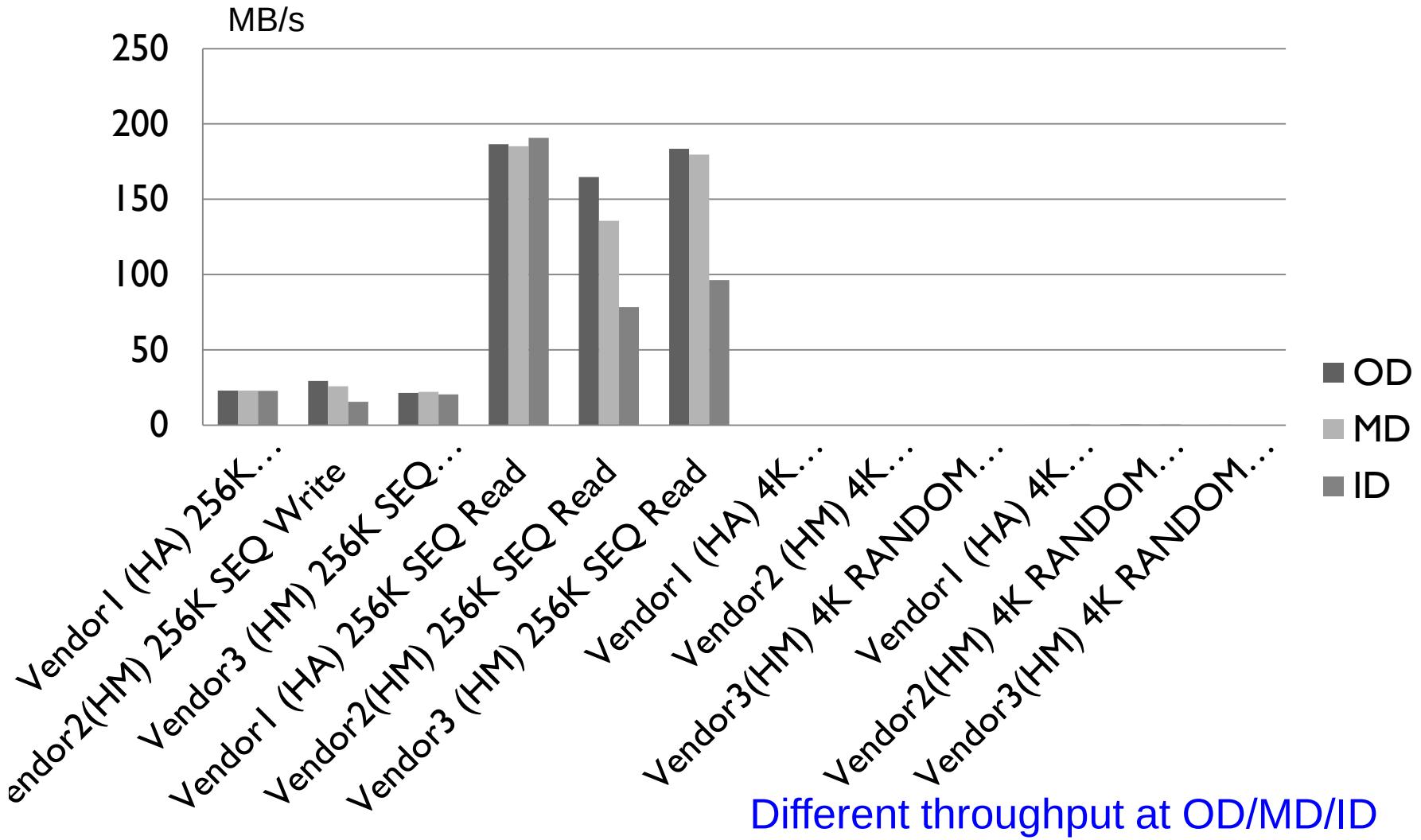


SMR drive IOPS



Different IOPS at OD/MD/ID

SMR drive Throughput

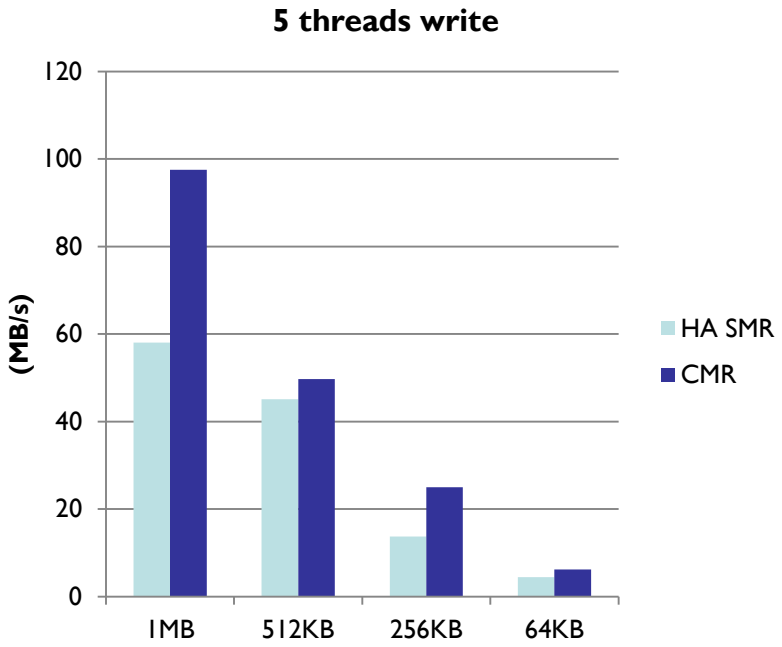


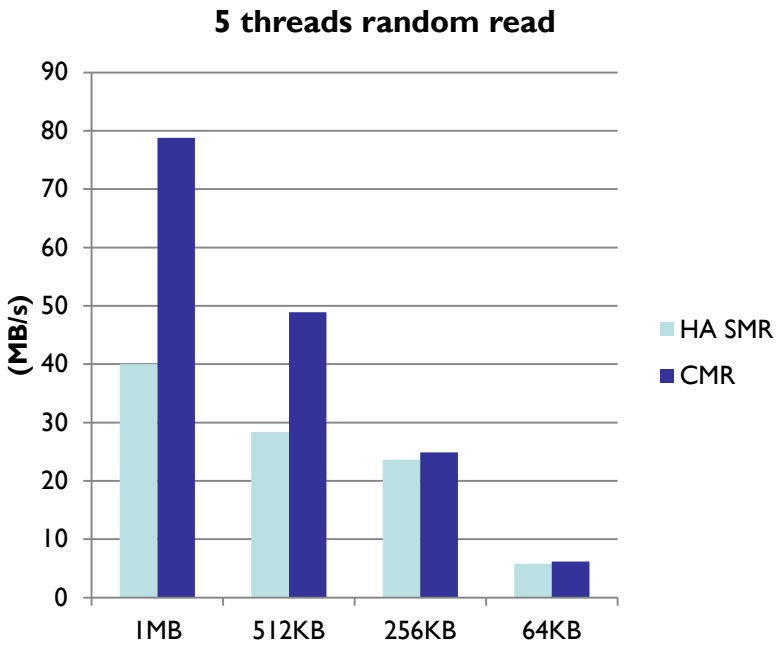
Different throughput at OD/MD/ID

LDB KV Throughput 5 threads (HA SMR vs. CMR)

Test Environment:

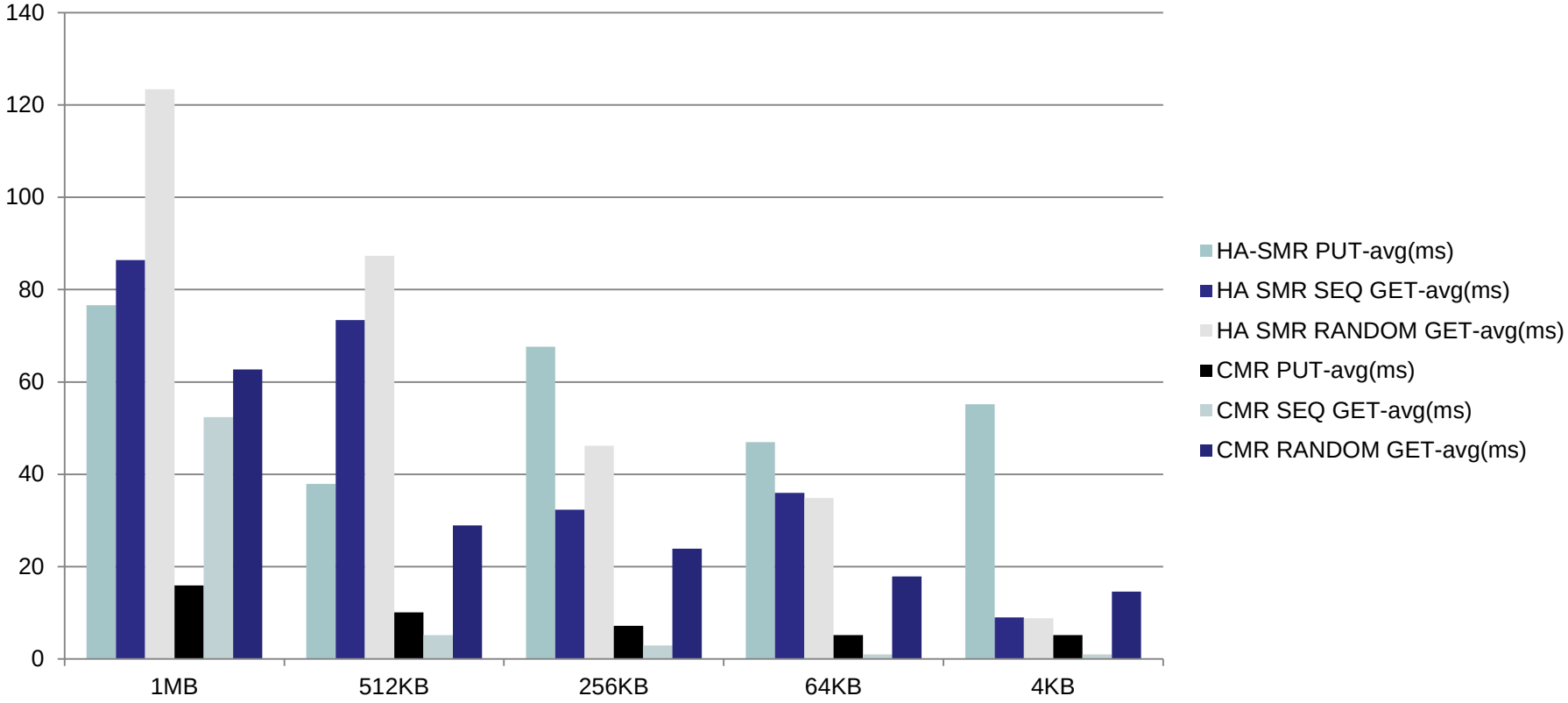
- ARM 7+1 GB
- Linux 2.6.34
- LDB
- KV test tool





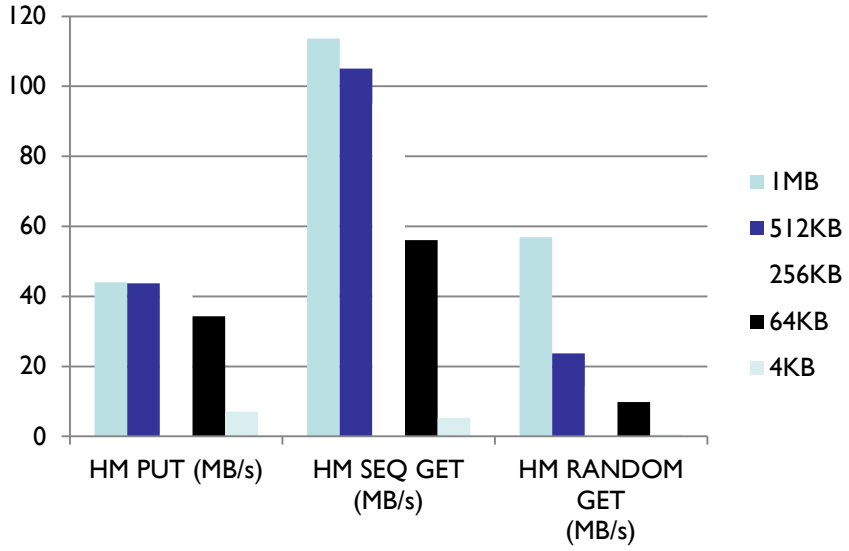
HA SMR throughput is half of CMR now

LDB KV Latency 5 threads (HA SMR vs. CMR)



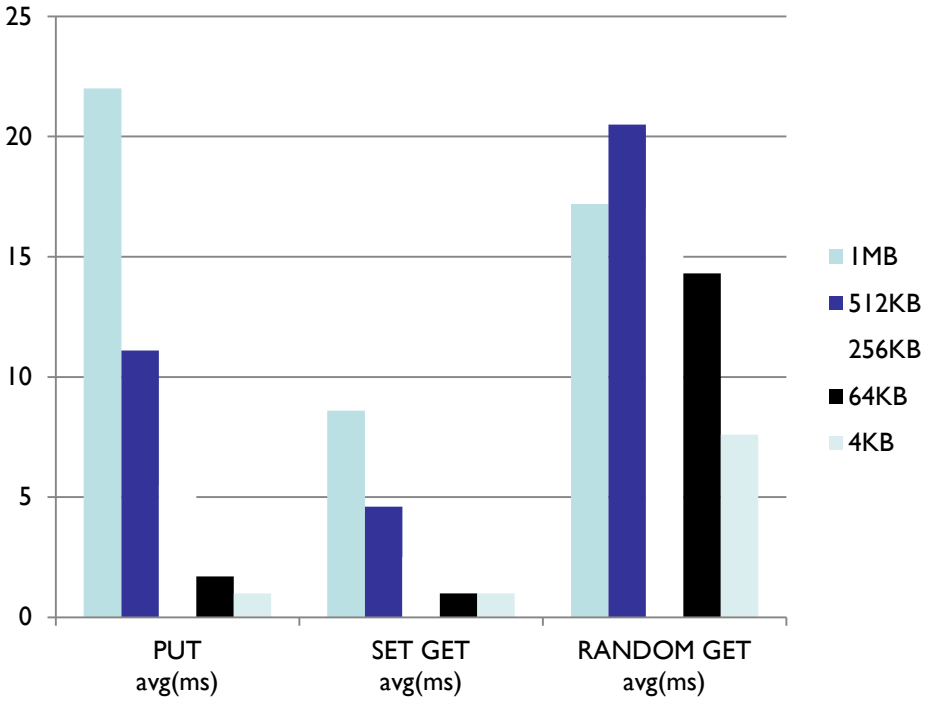
Now, HA SMR latency is higher than CMR(xx ms level), and lots of jitter

LDB KV Throughput & Latency over HM SMR



Test Environment:

- **X86+4 GB (HM need HBA FW & driver modification)**
- Linux 3.0.76
- LDB
- KV test tool



- HM SMR throughput is close to CMR, even better some time
- HM SMR latency like CMR, and there are few jitter

Agenda

- Object Storage Market Overview
- Object Storage Design with SMR based Key-Value Store
- Summary and Future Works

Summary

- LDB is log structure and in memory index design, most operations are “1 memory access + 1 disk access” , and all writes(include random) are sequential.

- Memory Index Table can be swap to disk depends on memory size configuration

- Recovery oriented design
  - atomic, write out-of-place not write-in-place
  - Super block tracks zone allocation and pool configuration
  - Memory index table checkpoint
  - KVR(s) in one zone are packed together

Future Work 1: SMR drive related

- SMR drives have new sense code, how to do SMR handle error?
- Check libata translate sense code well for ZAC?
- Standards for these sense code and translation…
- IOCTL is synchronous IO model, check the performance?
- Confirm IOCTL works well with NCQ?

Future Work 2: Integrate with Applications

- Applications Delete and update data will cause garbage collection (GC), and GC on SMR will use reset write pointer.
  - How to design efficient GC?
  - reset write pointer may cause FTI (Far Track Interference) and drives have new sense code, how to do SMR handle error?

- Application may put KVR with meta-data, how to implement application aware metadata processing?

Thank You