MASSé: A high-performance storage solution for 3D XPOINT™ and Flash SSDs

Jack Zhang Intel Corporation; KuanKuan Guo, Changlong Chen Bytedance Inc.

Abstract

MASSé is the **M**edia **A**ware **S**mart **S**torage **E**ngine. While Solid-State-Drives (SSDs) have become commonplace in data-center and cloud, innovative new storage media technologies such as 3D XPOINT™ media or Intel® Optane™ media [1,4,22,35] and Quad-level Cell (QLC) flash [33,34] have emerged to address the ever-increasing need for better performance and lower total cost of ownership (TCO). Our observation is that existing storage algorithms and data structures - from application, through kernel, and to filesystem - have not evolved to take full advantage of the high-impact capabilities of these new storage media technologies. Consequently, applications are not fully benefitting from the performance capabilities of 3D XPOINT™ media, and the migration from today's mainstream TLC (Triple-level Cell) SSDs to low-cost QLC SSDs has been slowed. **MASSé** includes optimizations designed to take full advantage of the latest Solid-State-Drive media innovations, allowing applications to achieve better performance with 3D XPOINT™ media, as well as speed the migration to more cost-effective QLC SSDs. **MASSé** is lightweight opensource software that resides in user space, independent from applications and the kernel system. It can be integrated as a plug-in module into any storage system, enabling a tiered storage architecture that is more capable of delivering on the performance capabilities of 3D XPOINT™ media and the low-cost advantages of QLC SSDs.

Introduction

NAND Flash (media) --The key component of Solid-State Drives has been innovated from Single-level cell (SLC), Multi-level cell (MLC), Triple-level cell (TLC), to today's Quad-level cell (QLC) [34], with Penta-level cell (PLC) under development, As number of levels increases, storage density increases, while performance (speed and reliability) and costs decrease, the result is, larger, cheaper but potential lower performance SSDs. While software applications, storage engines, and filesystems work well enough for flash SSDs. However, software typically treats all flash SSDs the same, with no distinction between MLC, TLC or QLC media. The introduction of high performance, low latency 3D XPOINT™ media, in the form of persistent memory and SSDs, adds a new dimension to the challenges of designing storage subsystems that can take full advantage of today's innovative storage media. In fact, SSD data placement methods can have huge impacts in critical storage metrics, such as random write IOPS, latency, Quality of Service (QoS), and endurance. Common user complaints include "Why do I not see x number of times improvement over flash SSDs when dropped in an 3D XPOINT™ SSD?" and "Re-shaping writes into larger datasets and sequentially sending to a QLC SSD requires additional software investments, and implementations differ from application to application…is there a generic solution that supports this?"

We heard our customers' voice. **MASSé** – the **M**edia **A**ware **S**mart **S**torage **E**ngine - is designed to solve these problems.

MASSé (**Figure 1**) uniquely identifies and classifies heterogeneous SSDs by their media type, and builds inclusive data structures and algorithms, accordingly, helping to release maximum SSD capabilities to applications…**Media Aware**, it has intelligent module features such as data placements, IO re-shaping, key-value/virtual filesystem/virtual block APIs, workload pattern AI engine (under developments) etc….**Smart**, it is a replacement of filesystem and managing raw SSD blocks without modifying SSD firmware and kernel modules…**Storage Engine**, with all of the features comprising this innovative solution…**MASSé**.

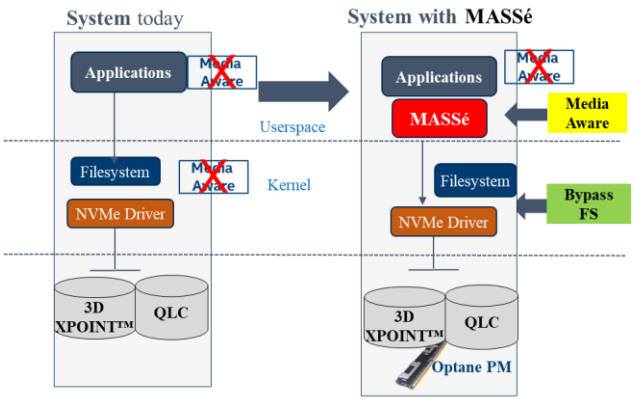

Figure 1 MASSé in system

MASSé is built on a key-value store framework and uses a key-value database to manage barebone SSDs. It was also known as the Smart Key-Value Data Store (SKVDS) or Smart Storage Engine (SEE) at Github repository [5]. This paper presents **MASSé** design strategies, tiered storage implementations, and evaluations which includes performance

matrices on different medias includes: QLC, TLC, 3D XPOINT™ SSDs and the tiered storage system of 3D XPOINT™ plus QLC or TLC SSDs. We also compare **MASSé** with the popular industry key value store **RocksDB** [7] as additional evidence to show that media-aware storage engine is required for new media technologies. Last, we will introduce our case studies at ByteDance Ltd, and we demonstrate **MASSé** replacing EXT4 filesystem at an application called TerarkDB.

2 Design implementations

There is much research available [6,9,10,16] for understanding how flash SSD works, as well as SSD optimization strategies [8,13,14,15], such as, no random writes, sequential large block size writes, 4KB block size alignments, separate read and write operations, balance workloads, reduce the number of threads for better latency, reduce IO path and lockless, reduce write pressure for better tail latency, and more.

Generally speaking, they are good guidelines for optimizing storage media through the TLC SSD generation, but not enough for QLC SSDs, and some of these guidelines are completely wrong for 3D XPOINT™ SSD.

2.1. MASSé design guidance

It is important to understand certain media characteristics:

3D XPOINT™:

1. Media is write-in-place, no garbage collection needed.
2. Media has low access latency
3. Media has much higher endurance, most applications will not encounter media wear-out
4. Media offers same bandwidth on both random write and sequential writes, where sequential bandwidth is 2200MB/s, equal to random write bandwidth which is 550K IOPS on 4KB block size = 550K x 4K = 2200MB/s datasheet [32].
5. Writes and reads are handled by hardware, no firmware involved
6. Distribution of hot spot avoids data migration by firmware.

QLC Flash:

1. Media is erase-before-program, need perform garbage collection
2. Media is subject to much low endurance, and delivers reduced write performance
3. Media requires larger than traditional 4KB IU table, 16KB/64KB.
4. Bandwidth for sequential write operations is much higher than for random write, where sequential bandwidth is 1600MB/s, random write bandwidth on 4KB size is 11K IOPS x 4K = 440MB/s, see datasheet [31]
5. Media is subject to high latency and unpredictable QoS on read when there are intensive writes. This is common to any NAND flash SSDs, but QLC even worse.

Then we define our **MASSé** design rules accordingly:

1. **3D XPOINT™ SSD**: do write-in-place, keeping frequently update data on these SSDs. Do not prioritize sequential writes, and avoid datasets less than 4K sequential writes, avoiding hotspots.
2. **QLC SSD**: Use large datasets (16KB/64KB) append writes to QLC by **single** writer in asynchronous mode
3. Plus, implement all strategies developed/understood for TLC SSD.

3D XPOINT™ SSD + **QLC SSD** tiered storage strategies:

1. **MASSé** uses 3D XPOINT™ SSD as a write pad on write path only due to cost sensitivity.
2. Persist small datasets to 3D XPOINT™ SSD and Merge them to large datasets in memory before sequentially flushing to QLC SSD.
3. Garbage collection (GC) uses 3D XPOINT™ SSD as GC write pad to temporarily hold valid data read from QLC SSD.
4. Read directly from QLC SSD, no read cache is needed.

2.2.Architecture

MASSé includes lightweight software components, shown in **Figure 2**, in order to minimize CPU and memory overheads. Unlike OpenChannel [2,21,23] and Zone Namespace Storage (ZNS) SSDs [3,29], **MASSé** uses the standard NVMe driver, with no requirements for special SSD firmware.

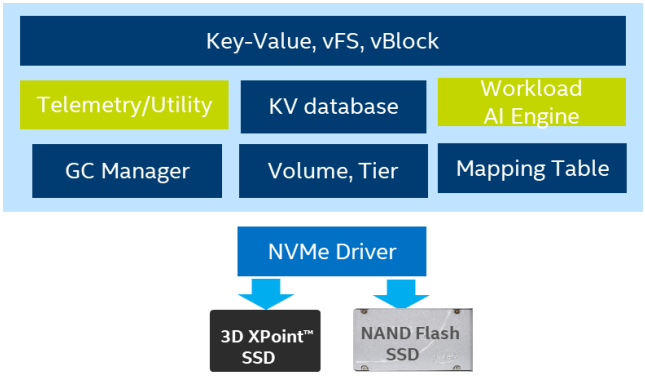

Figure 2 Architecture

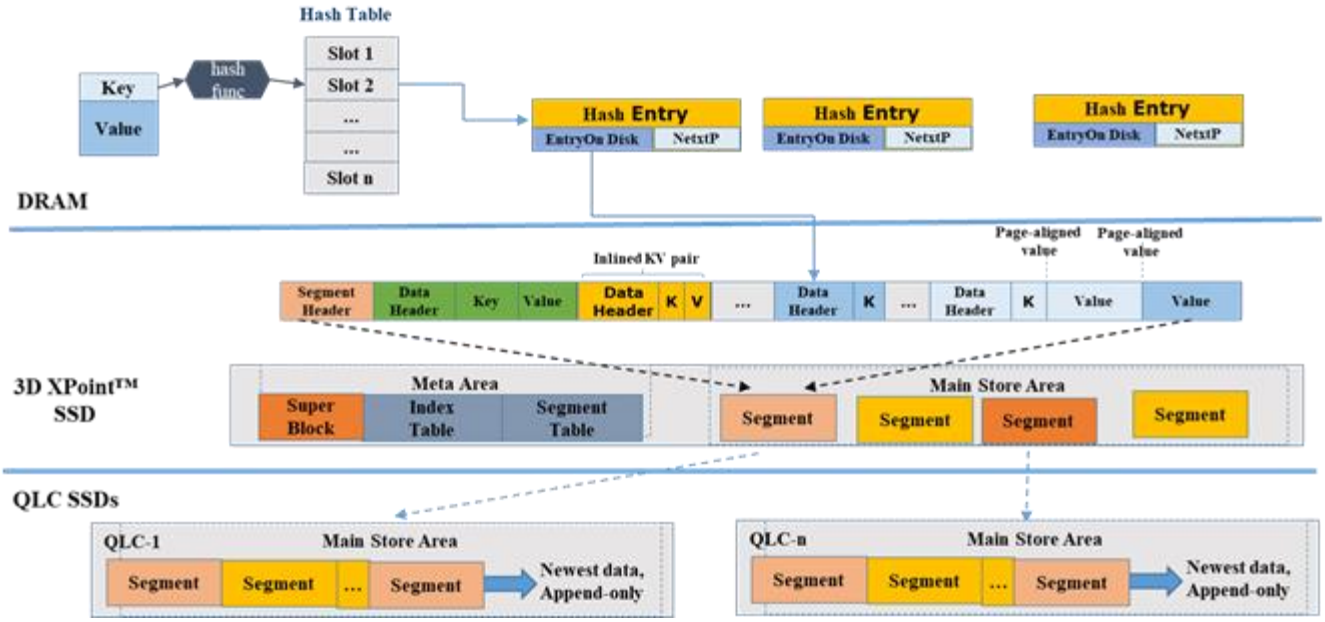

Figure 3 Tiered Storage System

Also, since **MASSé** is a user space software, it has no overheads on Linux kernel complexity and distro compatibility that other kernel-based design [17] has, and it is more efficient and flexible to integrate with applications.

The foundation is built on key-value or object store, plus basic virtual File and virtual Block APIs. The key components are hash-based mapping table in memory, volume and tier manager, garbage collection manager, and key-value database. We are also a developing utility with a telemetry component, which will help on-line SSD debug, performance analysis etc., as well as workload AI engine for IO patterns training and inference. However, as these topics are out-of-scope for this paper, we will discuss in a future.

2.3.Data Layout

As **Figure 3** shown, the data are placed among DRAM, **Fast** tier-3D XPOINT™ SSD, and **Capacity** tier-QLC SSD. DRAM stores hash mapping table, 3D XPOINT™ SSD stores engine metadata and temp data, QLC SSD stores all data.

We use "Segment" to hold both metadata and key-value data, the default segment size for 3D XPOINT™ SSD is 8KB, QLC is 256KB, they can be re-configured on-fly. For the capacity layer data layout shown in **Figure 4**,we divide capacity layer by number of Logic Band (LBand). This equates to the usable capacity/LBand size. The LBand size is configurable and corelated to SSD erase band, and each LBand has multiple segments pointed by header and tail pointer. New segment is always appended to tail, the garbage collection (GC) starts from header for valid data relocations, each time ONLY one LBand is open for write, but it allows multiple LBand to be opened for GC and read, and the number of reserved LBands is configured based on balance of performance and cost considerations.

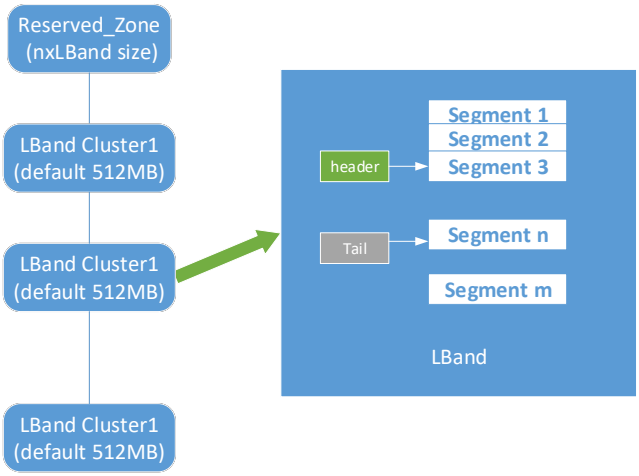

Figure 4 Capacity-tier Data Layout

Under these data placements, and data flow algorithm shown in **Figure 5**, the Write Amplification Factor (WAF) of QLC SSD is close to 1, this means NO garbage collection or less data reloaction happens inside QLC SSD. Several immediate benefits are apparent:

1. Improved IO performance
2. Predictable read latency and QoS
3. SSD DRAM size reduction by using larger indirection table, demonstrating the impact of large capacity QLC SSDs.
4. Reduced controller and firmware complexities.
5. Potentially reducing SSD cost by reducing SSD spare space requirements.

2.4.Data Flow

Figure 5 shows a data placement algorithm for tiered storage, where 3D XPOINT™ SSD is used as write pad and QLC SSD as data storage. Small datasets are first written to fast tier 3D XPOINT™ SSDs while merging in DRAM, and as soon as data is persisted, it completes the write operation. When merging data in DRAM reaches a defined size (default is 256KB), they are flushed to QLC SSD by asynchronous single thread, guaranteeing the sequential write operation. Garbage collection is only on QLC SSD and is triggered by thresholds based on pre-defined and on-fly calculations. GC operation includes reading valid data from QLC, persisting to fast tier (GC is done after persistent), and merging in DRAM along with the host write data.

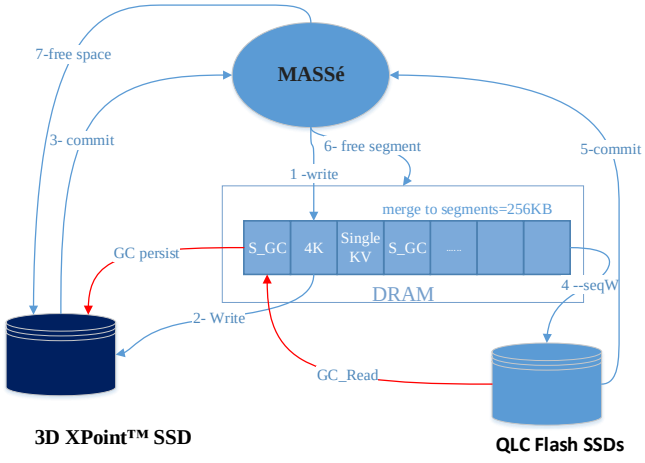

Figure 5 Write Flow

Data retravel is simple, **MASSé** finds key entry in mapping table that contains data physical address, read data from either 3D XPOINT™ SSD or QLC SSD.

Insert(K, V) //K is not present in the DB
d<-CalculateDigest(K) //Calculate Digest
s<-KeySlice(d) //2 Bytes from Digest
//allocate free space in Fast Tier
new_seg_num<-FindEmptySegmentInFT()
Loc<-WriteKVToFT(K, V, new_seg_num)
//KV Location points to Fast Tier
UpdateKVLocationInHashTable(s, Loc)
return

Get(K)
d<-CalculateDigest(K) //Calculate Digest
s<-KeySlice(d) //2 Bytes from Digest
Loc<-SearchForKeyInHashTable(s)
If(key-found)
If(Loc==FT)
V<-ReadKVFromFT(K, V, Loc)
return V
else
V<-ReadKVFromMT(K, V, Loc)
Return V
else
return not-found

Delete(K)
d<-CalculateDigest(K) //Calculate Digest
s<-KeySlice(d) //2 Bytes from Digest
Loc<-SearchForKeyInHashTable(s)
If(key-found)
DeleteKeyFromHashTable(s)
MarkDiskLocationInvalid (Loc)
return
else
return not-found

//merge small k-v, flush to capacity tier QLC
CapacityTierMergeWrite(K, V, d)
// check if the current segment buffer can accommodate this KV pair
if(CheckFreeSpace(segbuffer, K, V))
AddKVToSegmentBuffer(segbuffer, K, V)
else
new_seg_num<-FindEmptySegmentInCapacityT()
Loc<-WriteSegmentBufferToMT(segbuffer, new_seg_num)
for K in seg
s<-KeySlice(d) //2 Bytes from Digest
UpdateKVLocationInHashTable(K, d, Loc)
//KV Location points to MT
segbuffer<-CreateEmptySegmentBuffer()
AddKVToSegmentBuffer(segbuffer, K, V)

Table 1 Data Flow Algorithms

2.5.In-Memory Mapping Table

The mapping table is maintained in memory using a hash data structure. The digest is computed by RIPEMD-160 [26], and we use simplified and complex entries to reduce

total table size as well as solving collision issues. The simplified entry uses two-byte key slices of digest, the complex has complete 20-byte digest.

The design is as follows, but first this assumption: two distinct keys K1 and K2 have same hash value h(K1) = h(k2), and if so, the key slices S(h(K1)) and S(h(k2)) are different. These are the simplified entries shown in **Figure 6**, then we relax this assumption to allow the key-slices to collide as well, the full digest complex entry is used, the result is about ~15GB memory per 1 billion entries. The detailed calculation is:

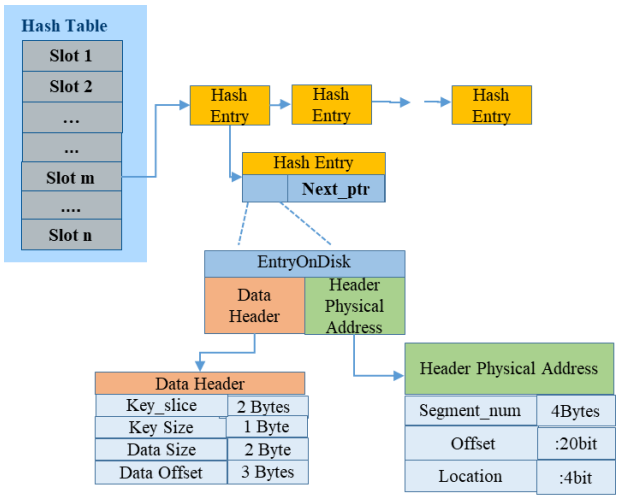

Figure 6 Simplified Entry Structure

a) # of random KV entries = 10^9

b) # of hash-slots = 2^{25}

c) If the keys have perfect distribution across hash-table slots and no key-slice collisions, there are no additional entries required: 15-bytes x1B = 15 GB.

d) However, expected number of such dual-collisions for m buckets and n inserts [27] is:

$$expected\ collisions = n - m\left(1 - \left(\frac{m-1}{m}\right)^n\right)$$

e) m = number of different bin-ids that a KV pair can have (the bin-ids for two KV pairs must be the same to be considered a collision)

$= 2^{\text{number of bits per key-slice}}$

$= 2^{16}$ in our case

f) n is average number of KV entries placed in a hash-table slot = total number of KV entries in the DB divided by number of hashslots $= 10^9/2^{25}$ in our case, n~=30, applying the formula above, the expected dual-collisions per slot are ~0.00655 (!), and therefore the total number of dual-collisions for the entire hash-table are 2^{25}*0.00655 ~= 220k.

g) Whenever a dual-collision happens, the complex entry ~48B. This adds 48B * 220k ~= 10 MB.

h) The total expected structure size for the proposed approach is therefore 15 GB + 10 MB = 15.01 GB, compared to the baseline 48GB… i.e., ~3x improvement.

The two-level index mapping table similar to design [30] is currently under developments for additional reducing table size while keeping less effects on performance.

2.6. Garbage Collection (GC)

No GC is required for 3D XPOINT™ SSDs in the fast tier, however, since we append sequential writes to QLC SSD in the capacity tier, we need to relocate valid data and free up space in the capacity tier. The basic operation shows in **Figure 5**, Segment Status Table (SST) is used to record segment space states for GC operations, **UseState** has free, used or reserved, **FreeSize** records free space remaining in the segment, and **DeathSize** records invalid key-value size in the segment. When a key-value is deleted or updated, the KV stored in previous segment is marked as invalid. Also, the LBand header keeps the KV invalidity statistics data. GC manager picks up most invalidity band for data relocations.

In general, GC is triggered by number of free Lbands left, however, in order to gain most efficient GC, we added intelligent algorithms, which separate GC threads by read, merge and write. The GC manager dynamically adjusts the number of GC threads by invalidity data, free Lbands, write intensity etc... As described in the previous chapter, fast tier is used as a GC write pad, similar to host write pad. After GC data persist to fast tier, the space can be reclaimed before actual data written back to capacity tier. This expediates the space freeing process.

2.7.Power Loss Imminent

Because our mapping table is in memory, Power Loss Imminent (PLI) is a must to have for data integrity. Per the previous description of data placement architecture, it already has recovery mechanism for unexpected power loss, because the meta data (hash entry) are persisted along with KV to capacity drive, so the mapping table can be re-built by reading these meta data back. However, this process can take a very long time for large capacity drives. For example, an 8TB QLC drive with 1GB/s read bandwidth will require more than 2 hours for recovery. This may not be acceptable for some applications. Therefore, we implement a simple algorithm to solve this, using the same segment concept to merge, update, and persist latest hash entries in fast tier, hash segment shown

in **Figure 7** is aligned to 4KB size for performance, merging multi hash entries and persisting them to fast tier index hash table area. Our test shows the worst case for above 8TB recovery is less than 200 seconds, a dramatically improved system recovery time.

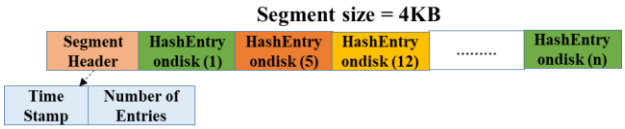

Figure 7 Hash Entry Segment

2.8. Other APIs

Besides most demanded application-friendly [19] key-value APIs, **MASSé** also developed virtual file and virtual block APIs.

2.8.1. Virtual File

Virtual File (vFile) is an abstraction of POSIX filesystem, the purpose is to replace **non** media aware filesystem by **MASSé** so the applications that rely on filesystem can also get performance benefits. In general, vFile translates IO related system calls to key-value APIs and re-defines file and directory data structures be compliant to POSIX filesystem.

we evaluate other user space filesystem [18,28] from architecture and performance perspectives, we believe the advantages of media aware in vFile-**MASSé** lead to better performance especially with tiered storage 3D XPOINT™ SSD plus QLC SSD.

2.8.2. Virtual Block

MASSé can also be presented as a virtual block device: /dev/virtualnvmeXn1, where –X is the virtual device number. It represents **Fast** tier (/dev/nvemYn1) and **Capacity** tier(/dev/nvmeZn1), and the application can manage it as raw device and still get benefits from **MASSé**'s efficient media aware management. We tested this option by using Flexible IO synthetic benchmark, observing no overheads on this virtual block APIs.

3 Case Study – ByteDance's TerarkDB

TerarkDB is a replacement of RocksDB with better compaction strategies and better compression, which delivers optimized read/write amplifications and tail latency, TerarkDB has been deployed in ByteDance's products such as NewSQL, MyRocks, ABase databases etc.,

The **MASSé** with vFile, showed in **Figure 8**, is a replacement of EX4 filesystem currently used by TerarkDB, the early tests (in next chapter) show TerarkDB with **MASSé** have much better performance than with EXT4.

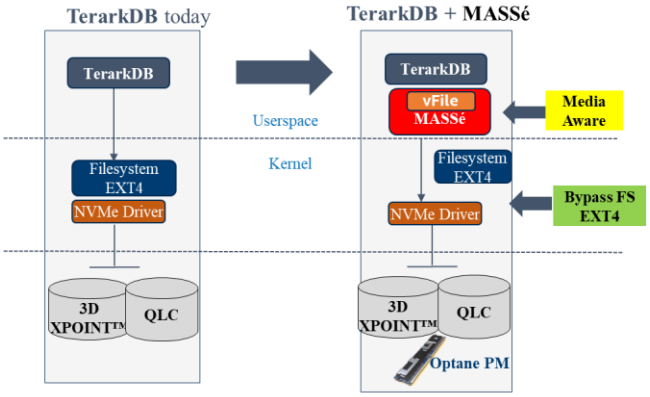

Figure 8 TerarkDB+ MASSé

Table 2 shows basic vFile IO translations from file system calls to KV APIs.

```
MASSé _pwrite(int fd, const void *buf, size_t count,
off_t offset)
  uint32_t write_blocks
  uint64_t bytes_written
  char K[key_count + 1]

  K[], write_blocks = ConvertToKeyArray()
  while(write_blocks > 0) {
    db->Insert(KV)
    write_blocks--
  }
  return bytes_written;

MASSé_pread(int fd, void *buf, size_t count, off_t off-
set)
  uint32_t read_blocks
  uint64_t bytes_read
  char K[key_count + 1]

  K[],read_blocks = ConvertToKeyArray()
  while(read_blocks > 0) {
    db->Get(K);
    read_blocks--
  }
  return bytes_read
```

Table 2 File Read/Write to KV Read/Write

We continue deep dive in this subject, the research interests include build standard abstract filesystem, support all POSIX filesystem IO operations, support multi-namespace etc.

4 Evaluation

We have following evaluations to prove **MASSé** is an effective and efficient storage solution for heterogeneous SSDs:

1. Compare performance on media aware engine (MASSé) vs media un-aware engine (RocksDB)
2. Demonstrate performance efficiency on SSD with different media under MASSé
3. Compare Bytedance's TerarkDB performance with MASSé vs with Filesystem EXT4.

4.1.Test configurations

CPU	Intel(R) Xeon(R) Gold 6142M CPU @ 2.60GHz
Memory	384GB
Storage	**3D XPOINT™ SSD:** Intel Optane™ P4800X 375GB U.2 **QLC SSD**: Intel P4320 7.68TB U.2 **TLC SSD:** Intel P4510 8TB U.2

Table 3 Hardware Configurations

4.2.MASSé vs RocksDB

In this evaluation, we want to show:

1. Performance delta between 3D XPOINT™ SSD and TLC flash SSD under RocksDB.
2. Performance gains **MASSé** over RocksDB under same media 3D XPOINT™ SSD.

The workload is abstracted from on-line index search and simulated and tested by db_bench.

workloads	Key=23B, Value=100B readrandomwriterandom 50/50 1Billion entries

Table 4 Simulation Workload

OP/Second shown in **Figure 9**, under Rocksdb, there is only an 11% improvement 3D XPOINT™ SSD over TLC flash SSD, while there is a 57% gain when **MASSé** replaces Rocksdb.

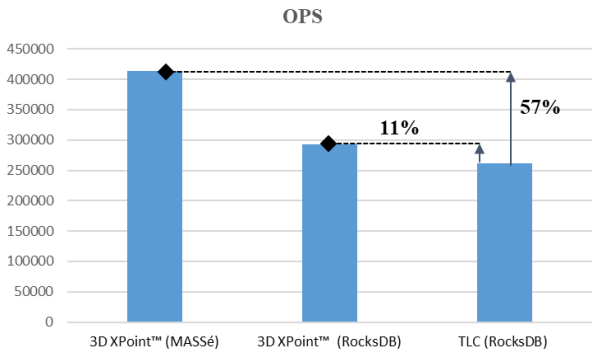

Figure 9 OP/S

Average latency shown in **Figure 10**, under Rocksdb, there is a 91% improvement when using high performance 3D XPOINT™ SSD to replace TLC flash SSD, increasing to a 1200% gain when replacing non-media aware Rocksdb with media aware **MASSé**

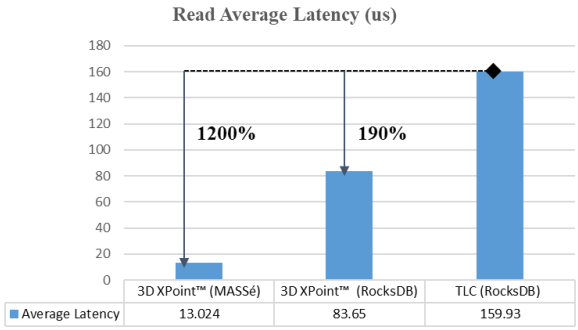

Figure 10 Read Average Latency

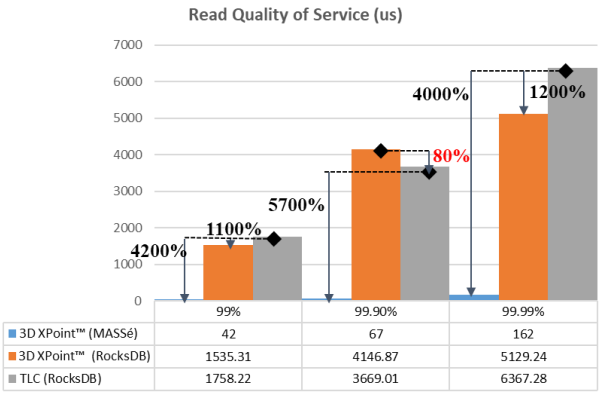

Figure 11 Read QoS

QoS shown in **Figure 11**, on 99% latency, under Rocksdb, there is no improvements when using high performance 3D XPOINT™ SSD to replace TLC flash SSD, however, there are 5700% gains when replacing Rocksdb with **MASSé.**

The conclusion is, **MASSé** releases 3D XPOINT™ SSD performance capabilities and delivers its actual value to applications.

4.3.MASSé on heterogeneous media

To further prove MASSé is an effective solution, we evaluated performance on different configurations, QLC SSD, TLC SSD, tiered storage on 3D XPOINT™ plus QLC and 3D XPOINT™ plus TLC SSD as well as 3D XPOINT™ SSD.

db_bench workloads:

workloads	Key=16B, Value=4096B readrandomwriterandom 50/50 1Billion entries

Table 5 Typical workload

Bandwidth shown in **Figure 12**, from QLC SSD at 471MB/s to 3D XPOINT™ SSD at 1291MB/s. High performance media gives you higher bandwidth, as you can see, where tiered storage 3D XPOINT™ + QLC SSD is better than TLC SSD. The 3D XPOINT™ SSD is ~5% of storage capacity, this means when evaluating the cost delta between 3D XPOINT™ SSD vs TLC and TLC vs QLC, you may not only achieve performance gains but also reduce TCO cost.

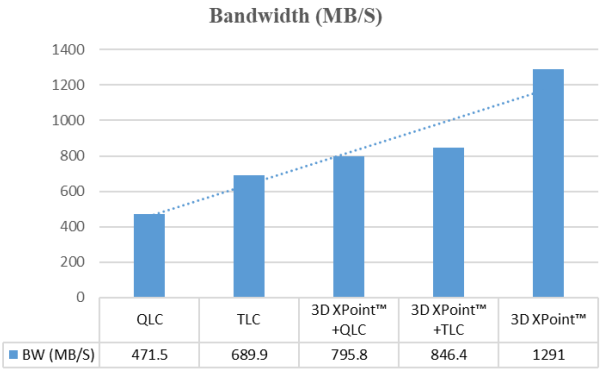

Figure 12 Bandwidth

Read latency and QoS in **Figure 13**: similar trending as bandwidth, where the high-performance media gives lower latency and improved QoS. As you see, **MASSé** is able to limit two 9 and three 9 latency under 1ms, which is almost impossible for flash SSDs under mix random read and write workloads. NAND latencies are typically unpredictable due to inside garbage collections. **MASSé** can better control QoS by only sequentially writing to flash SSDs. Since no GC or data relocation inside the NAND SSD is triggered, QoS is predictable and controlled.

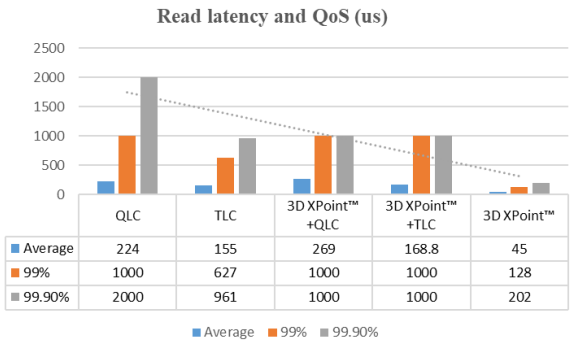

Figure 13 Read Latency and QoS

Write latency and QoS in **Figure 14**: Tiered storage 3D XPOINT™ plus QLC SSD or TLC SSD shows multiple times better QoS than without 3D XPOINT™ SSD. This is important when using QLC SSD because 1) QLC has much lower write performance 2) as the capacity of QLC SSD becomes larger, the indirection table gets bigger (from 4KB to 16KB/64KB), which requires a larger write buffer to merge small datasets. These will introduce much higher write QoS or tail latency if not properly handled and using 3D XPOINT™ SSD to persist these small datasets dramatically improves write QoS.

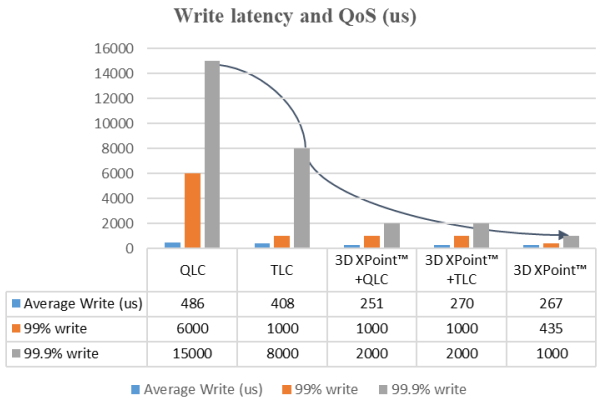

Figure 14 Write Latency and QoS

4.4.TerarkDB: MASSé vs EXT4

In addition to above tests on key-value, we extended our evaluations to ByteDance's production software TerarkDB. TerarkDB has been used and deployed in many on-line services, this allows us to further approve **MASSé** bringing media capabilities to applications, this evaluation compares TerarkDB performance with EXT4 vs with **MASSé.**

We run db_bench workloads on the same 3D XPOINT™ SSD twice, one test uses EXT4, the other one uses **MASSé.**

workloads	Key=20B, Value=100B readrandomwriterandom 70/30 10 M entries, no read cache

Table 6 TerarkDB Workloads

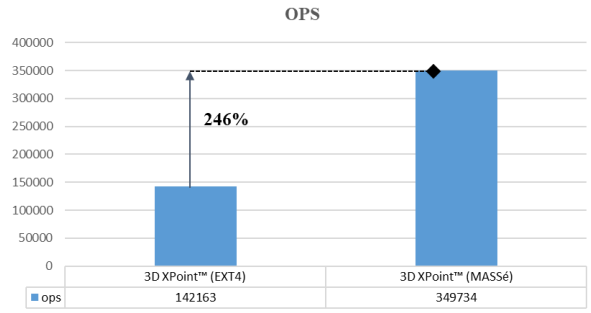

Figure 15 OP/S

MASSé is ~2.5 times better than EXT4 on OPS shown in **Figure 15**, and ~2-3 times better than EXT4 on average read latency and QoS shown in **Figure 16**.

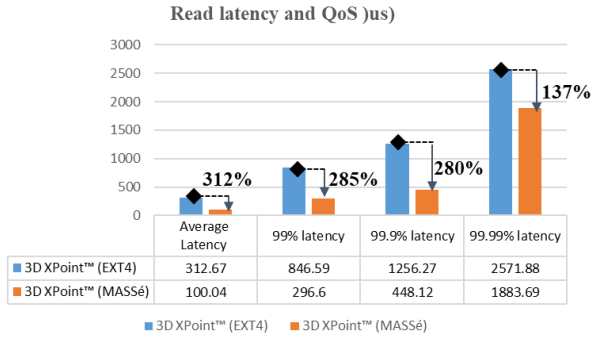

Figure 16 Latency and QoS

5 Conclusions

The **M**edia **A**ware **S**mart **S**torage **E**ngine – **MASSé**, is a high-performance and effective storage solution that releases the maximum power of heterogeneous SSD media. It is an inclusive design that reduces application burdens and encourages investments in new storage technologies. By making the combination of 3D XPOINT™ SSDs and QLC SSDs more effective, **MASSé** meets the growing demands of cloud and datacenter to improve performance while reducing cost.

MASSé is still in early design. Future activities include **MASSé** integration into customers' applications and production qualifications, as well as performance improvements and key-value store feature enhancements, such as workload AI engine, telemetry, two-level mapping table optimization, and more. As **MASSé** migrates to the opensource community, we expect **MASSé** to become a more popular storage engine which will play a critical role in optimizing the effectiveness of new media storage technologies.

Reference

[1] 3D XPOINT™ A Breakthrough in Non-Volatile Memory Technology https://www.intel.com/content/www/us/en/architecture-and-technology/intel-micron-3d-xpoint-webcast.html

[2] Matias Bjørling, Javier Gonzalez, Philippe Bonnet. LightNVM: The Linux Open-Channel SSD Subsystem. In 15th USENIX Conference on File and Storage Technologies, FAST 2017

[3] Zoned Namespaces (ZNS) SSDs https://zonedstorage.io/introduction/zns/

[4] 3D XPOINT™--Wikipedia https://en.wikipedia.org/wiki/3D_XPoint

[5] SKVDS Github https://github.com/TeamSKVDS

[6] CAI, Y., HARATSCH, E. F., MUTLU, O., AND MAI, K. Error patterns in MLC NAND flash memory: Measurement, characterization, and analysis. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)* (2012), IEEE, pp. 521–526

[7] RocksDB A persistent key-value store https://github.com/facebook/rocksdb

[8] DEAN, J., AND BARROSO, L. A. The Tail at Scale. *Commun. ACM 56*, 2 (Feb. 2013), 74–80

[9] AGRAWAL, N., PRABHAKARAN, V., WOBBER, T., DAVIS, J. D., MANASSE, M., AND PANIGRAHY, R. Design tradeoffs for ssd performance. In *USENIX Annual Technical Conference (ATC)* (2008), pp. 57–70

[10] Hao, M., Soundararajan, G., Kenchammana-Hosekote,D., Chien, A.A., and GUNAWI, H. S. The tail at store: a revelation from millions of hours of disk and SSD deployments. In *14th USENIX Conference on File and Storage Technologies (FAST)* (2016), pp. 263–276

[11] Hybrid Layout Key-Value Data Store Github: https://github.com/Intel-bigdata/HLKVDS

[12] ZHANG, J., ZHOU, Y., YANG, WY., PORTER, B., A New Key-Value Data Store for Heterogeneous Storage Architecture. Storage Developer Conference 2017.

[13] CHEN, F., LUO, T., AND ZHANG, X. CAFTL :A Content-Aware Flash Translation Layer Enhancing the Lifespan of Flash Memory based Solid State Drives. *9th USENIX Conference on File and Storage Technologies (FAST)* (2011)

[14] KANG, J.-U., HYUN, J., MAENG, H., AND CHO, S. The Multi-streamed Solid-State Drive. In *6th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage)* (2014)

[15] KIM, J., LEE, D., AND NOH, S. H. Towards SLO Complying SSDs Through OPS Isolation. In *13th USENIX Conference on File and Storage Technologies (FAST)* (2015), pp. 183–189.

[16] CHEN, F., LEE, R., AND ZHANG, X. Essential roles of exploiting internal parallelism of flash memory based solid state drives in high-speed data processing. In *IEEE 17th International Symposium on High Performance Computer Architecture* (2011), IEEE, pp. 266–277.

[17] Micron Heterogeneous-memory Storage Engine https://github.com/hse-project

[18] L.Paul, V.Vishal SPDK Blobstore: A Look Inside the NVM Optimized Allocator. Storage Developer Conference SINA 2017

[19] BJØRLING, M., BONNET, P., BOUGANIM, L., AND DAYAN, N. The necessary death of the blockdevice interface. In *6th Biennial Conference on Innovative Data Systems Research (CIDR)* (2013), pp. 1–4.

[20] SWANSON, S., AND CAULFIELD, A. M. Refactor, Reduce, Recycle: Restructuring the I/O Stack for the Future of Storage. *IEEE Computer 46*, 8 (2013)

[21] ZHANG, JC. LU, YY., SHU, JW., QIN, XJ. FlashKV: Accelerating KV Performance with Open-Channel SSDs

[22]. Micron 3D XPoint™ https://www.micron.com/products/advanced-solutions/3d-xpoint-technology

[23] WANG, P., SUN, G., JIANG, S., OUYANG, J., LIN, S., ZHANG, C., AND CONG, J. An efficient design and implementation of LSM-tree based keyvalue store on open-channel SSD. *Proceedings of the Ninth European Conference on Computer Systems (EuroSys)* (2014), 1–14

[24] JOSEPHSON, W. K., BONGO, L. A., LI, K., AND FLYNN, D. DFS: A File System for Virtualized Flash Storage. *ACM Transactions on Storage 6*, 3 (Sept. 2010), 1–25

[25] VIOLIN MEMORY. All Flash Array Architecture, 2012

[26] RIPEMD-160 Wikipedia https://en.wikipedia.org/wiki/RIPEMD

[27] Expected number of hash collisions https://stackoverflow.com/questions/9104504/expected-number-of-hash-collisions

[28] Storage Performance Development Kit (SPDK) –Blob-store Filesystem (BlobFS) https://spdk.io/doc/blobfs.html

[29] NVMe specifications on Zoned Namespaces(ZNS) https://nvmexpress.org/new-nvmetm-specification-defines-zoned-namespaces-zns-as-go-to-industry-technology/

[30] SILT: A Memory-Efficient, High-Performance Key-Value Store, Hyeontaek Lim, Bin Fan, David G. Andersen, Michael Kaminsky

[31] Intel QLC P4326 datasheet: https://www.intel.com/content/www/us/en/products/memory-storage/solid-state-drives/data-center-ssds/d5-series/d5-p4326-series/d5-p4326-15-36tb-2-5inch-3d2.html

[32] Intel Optane SSD datasheet: https://www.intel.com/content/www/us/en/products/memory-storage/solid-state-drives/data-center-ssds/optane-dc-ssd-series/optane-dc-p4800x-series/p4800x-750gb-2-5-inch.html

[33] QLC, what can we expect from the technology? https://www.architecting.it/blog/qlc-nand/

[34] WiKi Multi-Level Cell https://en.wikipedia.org/wiki/Multi-level_cell

[35] Intel® Optane™ media https://www.intel.com/content/dam/www/public/us/en/documents/technology-briefs/optane-memory-media-technology-brief.pdf