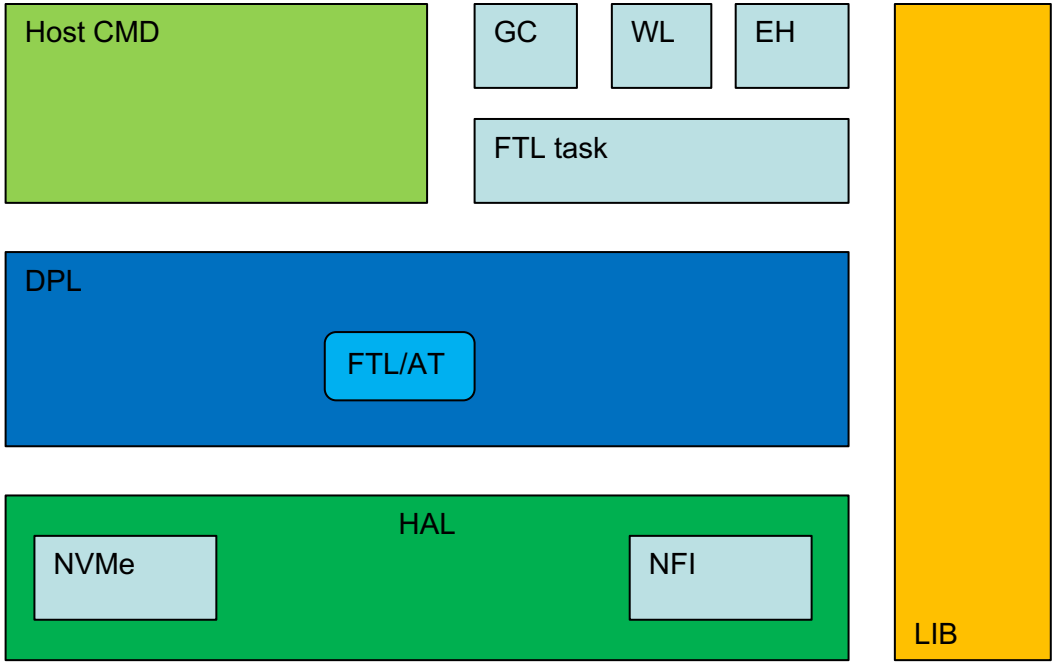
Key-Value Store Friendly SSD Interface Design and Optimization -- base on RocksDB

teng.yang@starblaze-tech.com

Starblaze Technology

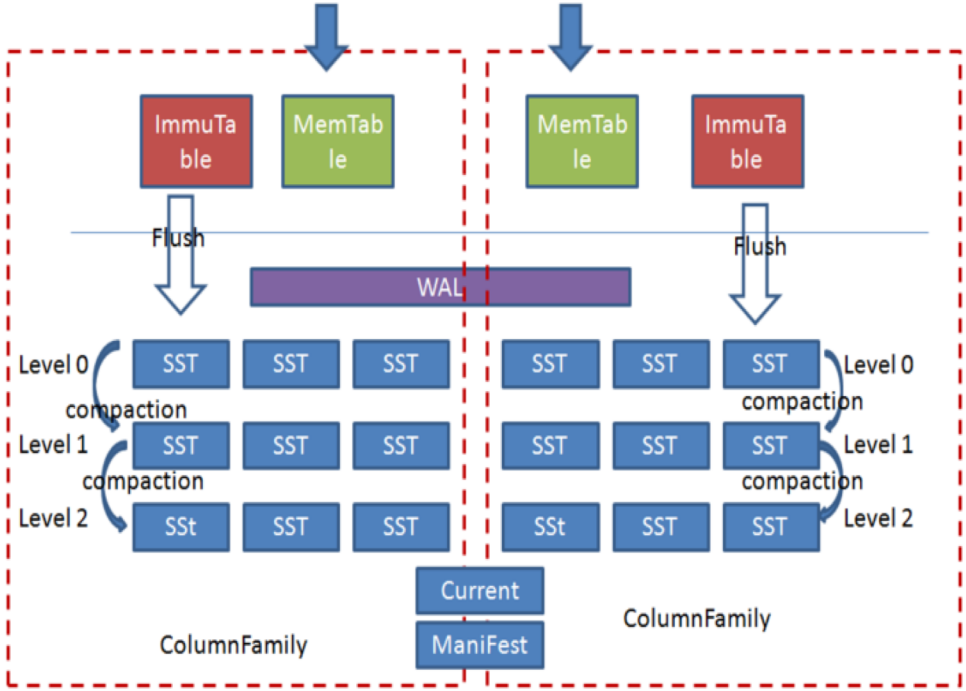
Background 1: Conventional SSD

- FTL
  - Garbage Collection
  - L2P translation
  - Wear-Leveling
  - Read retention
  - Read Disturb
  - others



Background 2: RocksDB

- RocksDB is a typical Key-Value Store System
- Key mechanisms
  - Immutable is flushed into files (SSTable)
  - SSTable files compaction for removing invalid KV



Limit with Conventional SSD on RocksDB

- High software stack Consumption
  - Compaction in RocksDB : space collection in logic level
  - Garbage Collection in SSD: space collection in physical level
- Predictable latencies cannot be guaranteed – 99 percentiles
  - garbage collection, wear-leveling and other ftl task
- Read Bandwidth may be drop
  - multiple write streams(multiple user thread)
- unavoidable Write Amplification
  - GC,WL lead to write amplification

Solution 1 – NVMe SSD feature

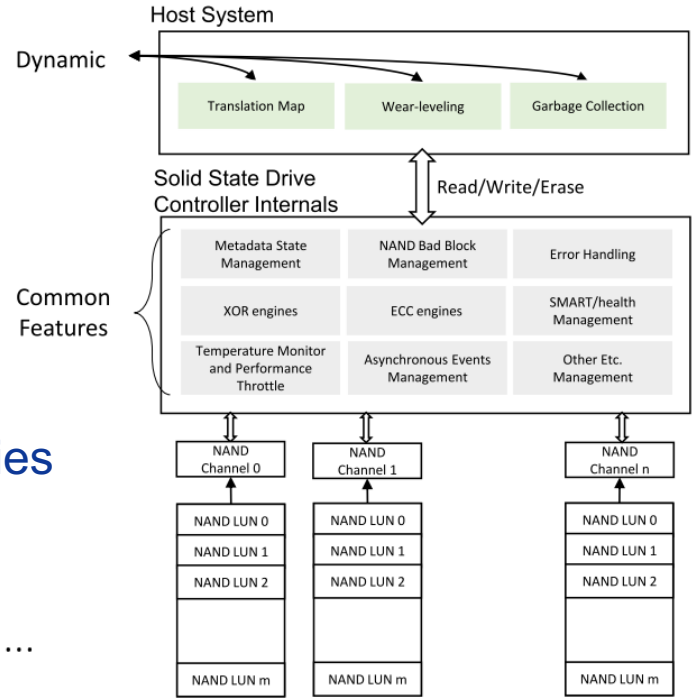
- Stream
  - expose a block I/O interface to the application
- Data Set Management
  - mark retired data => garbage collection be more efficiency

Solution 1 – Limit

- Garbage Collection / Wear Leveling
  - QoS
  - WA
- open block count for stream may be not enough
  - user threads count be limited

Solution 2 - Open Channel SSD

- Host in control
  - garbage collection
  - wear-leveling
  - Translation Map
- Device maintain
  - ssd offload engines and responsibilities
  - SSD geometry



Open Channel SSD

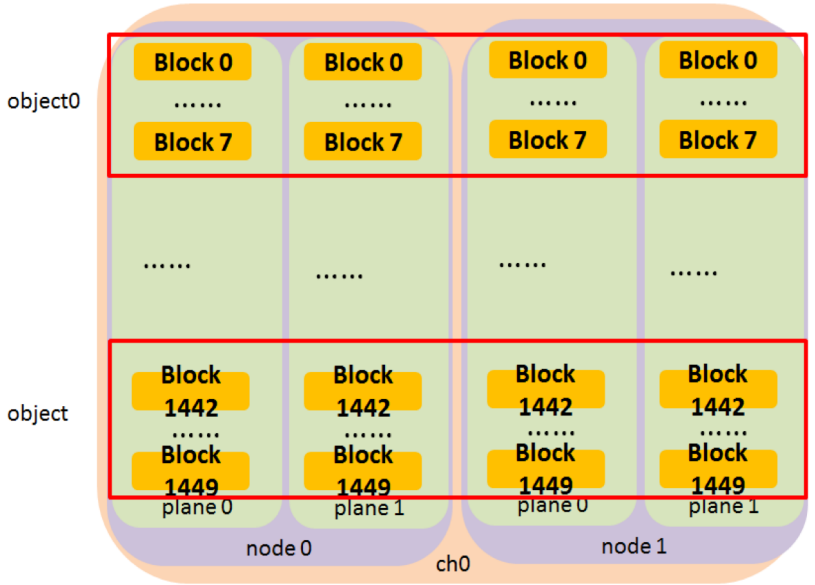
Solution 2 - Limit

- NAND is too complex to handle
  - different nand => different Physical Page Addresses (PPA)
- Application developer must know FTL very well
  - garbage collection, wear-leveling and other FTL knowledge

=> the interface of Open Channel SSD is not friendly enough

Solution 3 - Object SSD

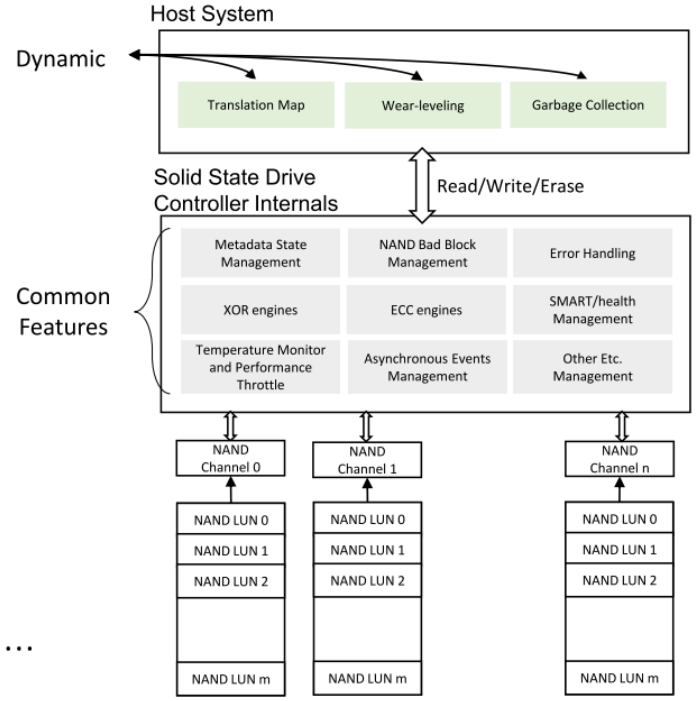
Commands	FW
Object read	Super block read
Object write	Open block append program
Object create	Block management/WL
Object Erase	Block Erase
Object Seal	Block management
Object inquiry	SMART



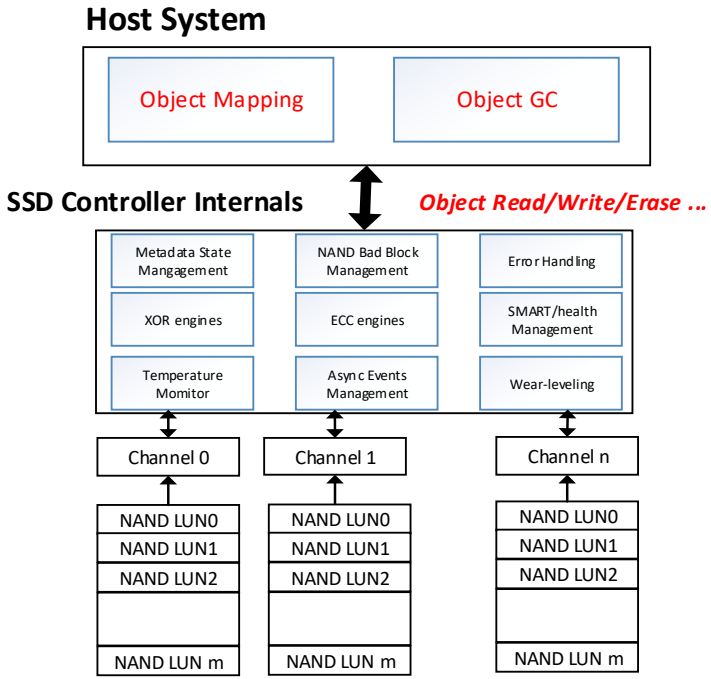
Flash Memory Summit

Object SSD

STARBLAZE



Open Channel SSD



Object SSD

Optimize RocksDB with Object SSD

- Fit SSTable size in RocksDB to object size
- SSTable is directly flushed into objects(replace of flush to file)
- **reuse RocksDB's compaction** – remove gc inside ssd
  - Achieve predictable latency - no 99 percentiles
  - Avoid write-amplification introduced by the FTL
  - Improve the steady state of the device
- multiple user threads corresponding to multiple objects
  - io isolate with multiple objects

Benefit RocksDB with Object SSD

- software stack consumption low
  - Compaction + GC => Compaction
- QoS is much better
  - GC inside ssd is removed
  - WL is very slightly in Object(block) Level
- Write Amplification is much smaller
- Better read bandwidth
- RAM costs down:
  - Mapping table(block level) size is less than 1/1000
- **Friendly interface**
  - object interface can be used like API
  - firmware focus on nand

Conclusion

- Object SSD provide a friendly interface to host side.
- Object SSD did help to solve the problem of RocksDB with Conventional SSD

- Come by Starblaze Booth #649 for more info
- This work is co-worked with Prof. Dejun Jiang at Institute of Computing Technology, Chinese Academy of Sciences. For detailed questions, he can be reached using email: jiangdejun@ict.ac.cn