Multi-Segment L2P Table Lookup Acceleration: Algorithm and Implementation

Eelin Tseng

Director, SSD ASIC Development

Silicon Motion Technology Corp.

FMS the Future of Memory and Storage

SiliconMotion

Legal Notice and Disclaimer

- The content of this document including, but not limited to, concepts, ideas, figures and architectures is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Silicon Motion Inc. and its affiliates. Silicon Motion Inc. assumes no responsibility or liability for any errors or inaccuracies that may appear in the informational content contained in this document.
- Nothing in these materials is an offer to sell any of the components or devices referenced herein.
- Silicon Motion Inc. may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Silicon Motion, Inc., the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.
-
- Silicon Motion, the Silicon Motion logo, MonTitan™, the MonTitan™ logo are trademarks or registered trademarks of Silicon Motion Inc.

Agenda

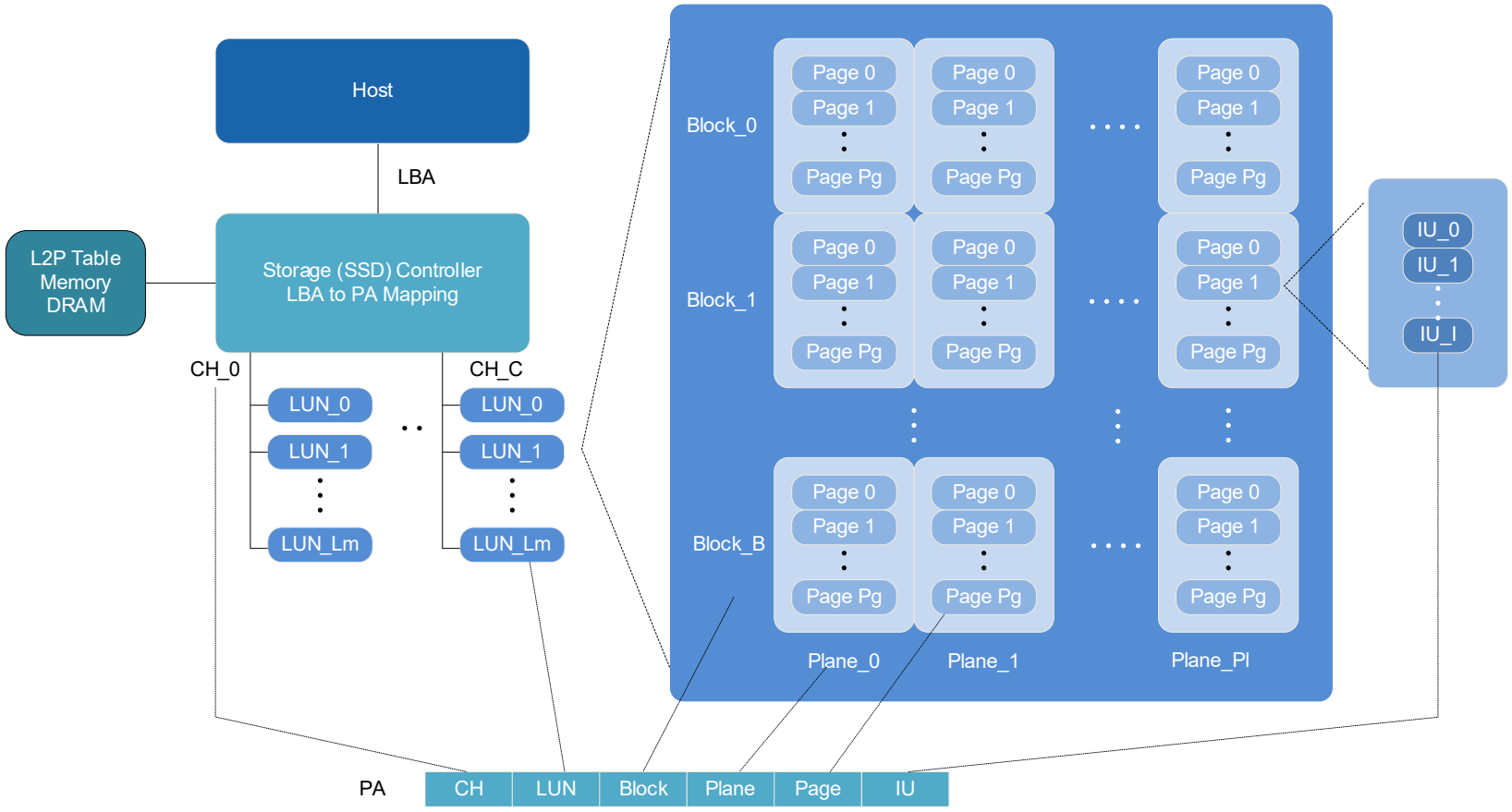
- Why multi-segment L2P Lookup Acceleration
- L2P Lookup Algorithm
- L2P Lookup Acceleration Implementation
- Application
- Summary

Why multi-segment L2P Lookup Acceleration

Program Statement

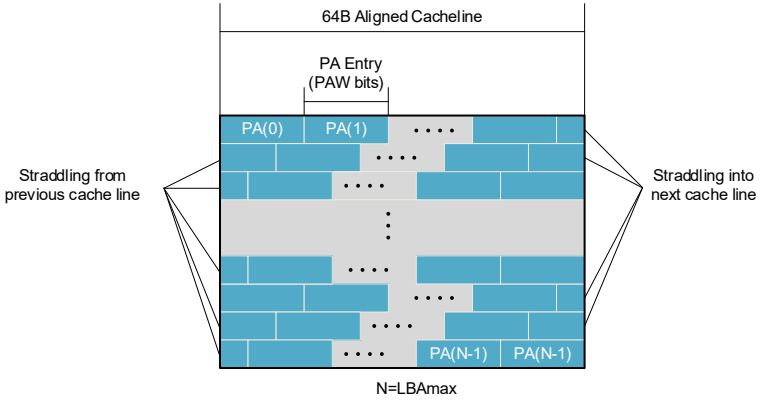
- In SSD controller, logical block address (LBA) needs to be mapped to NAND physical address through a mapping table (L2P Table). The minimum mapping table size is O(n*lg(n)) where n is user capacity
- FW based lookup consumes many CPU cycles and reduces performance drastically when the physical address bit number is not aligned to byte boundary.
- Multi-segment L2P table adds more complexity due to specific segment boundary alignment.
- HW acceleration is vital for L2P lookup performance.

Physical Address



L2P Table: Single- and Multi-Segment

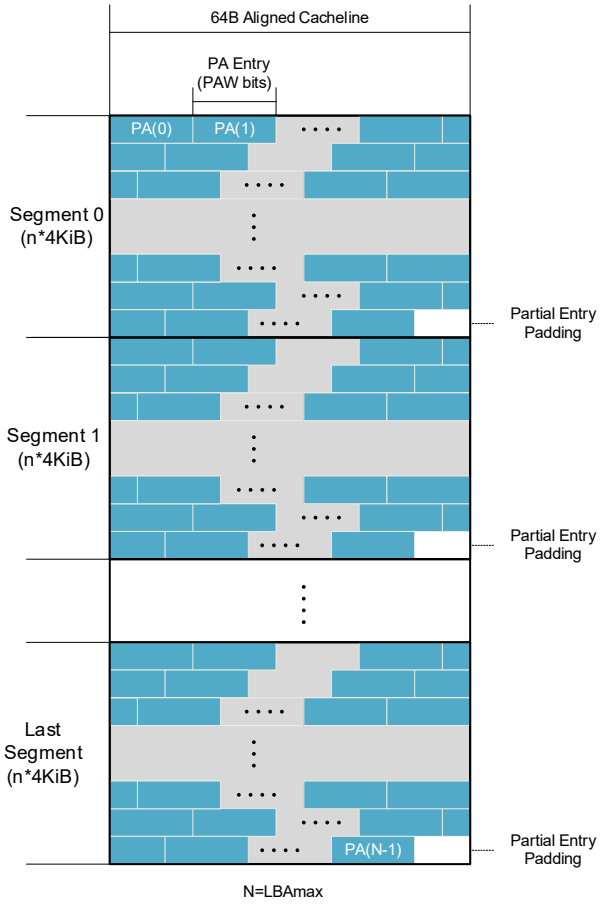
Single-Segment



Entire PAs are stored in a single memory region without a hole.

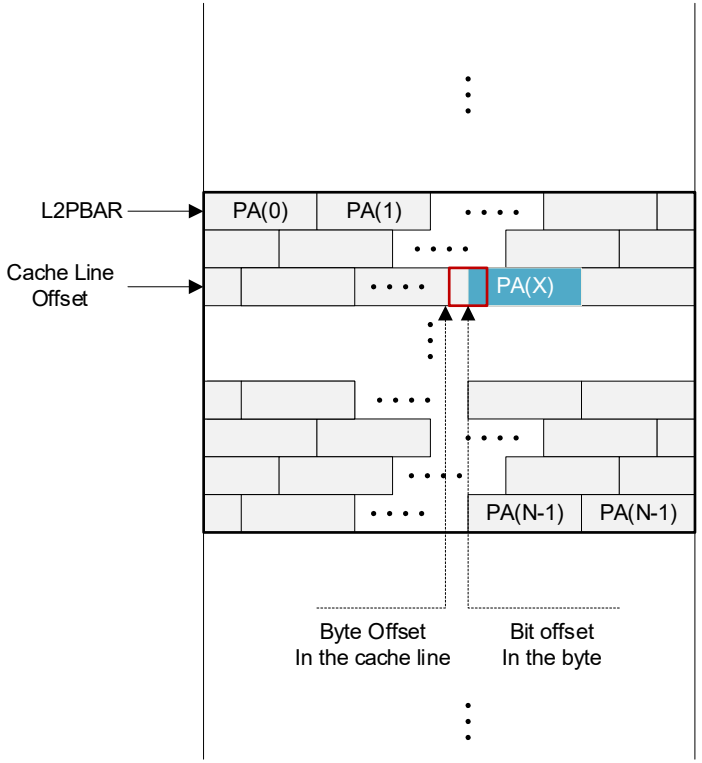
For a given LBA, the associated PA(LBA) bit address offset to the base address is LBA*PAW, where PAW is the size of PA address

Multi-Segment



- L2P table memory region is divided to multiple segments with equal length.
- The first PA entry in a segment is always aligned to beginning of the segment.
- No PA straddling between two adjacent segment boundary.
- A segment is a unit for L2P table journaling

L2P Table: Lookup

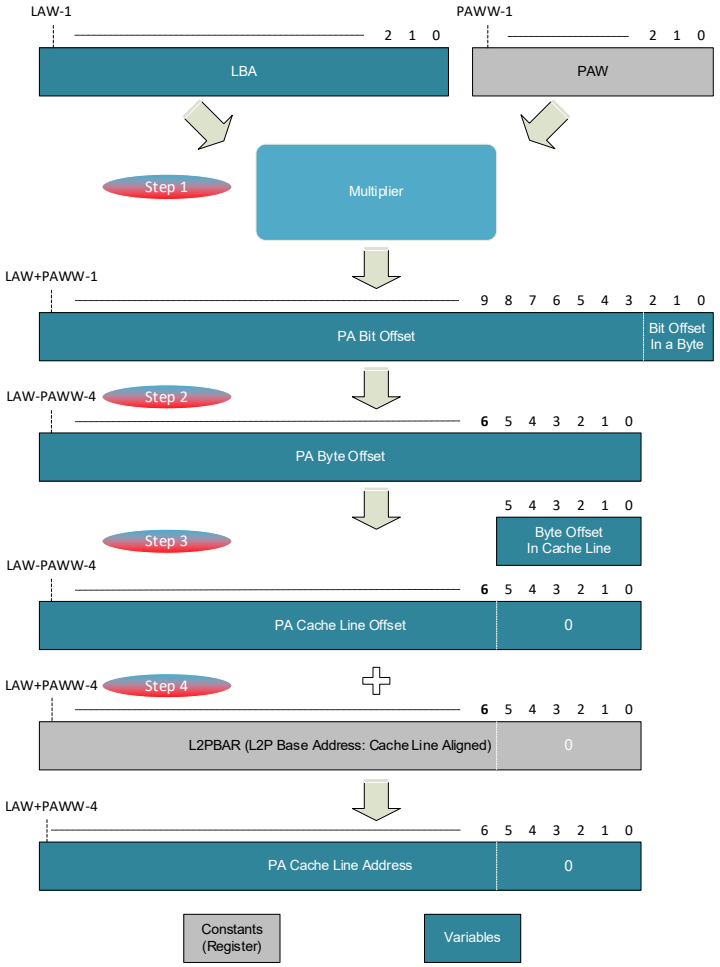


The L2P table lookup operation is for a given LBA find out address offset of PA(LBA), then read/write PA from or to the address offset.

The address offset includes 64B cache line address offset to the base address, byte address offset to the cache and bit address offset to the byte

Algorithms

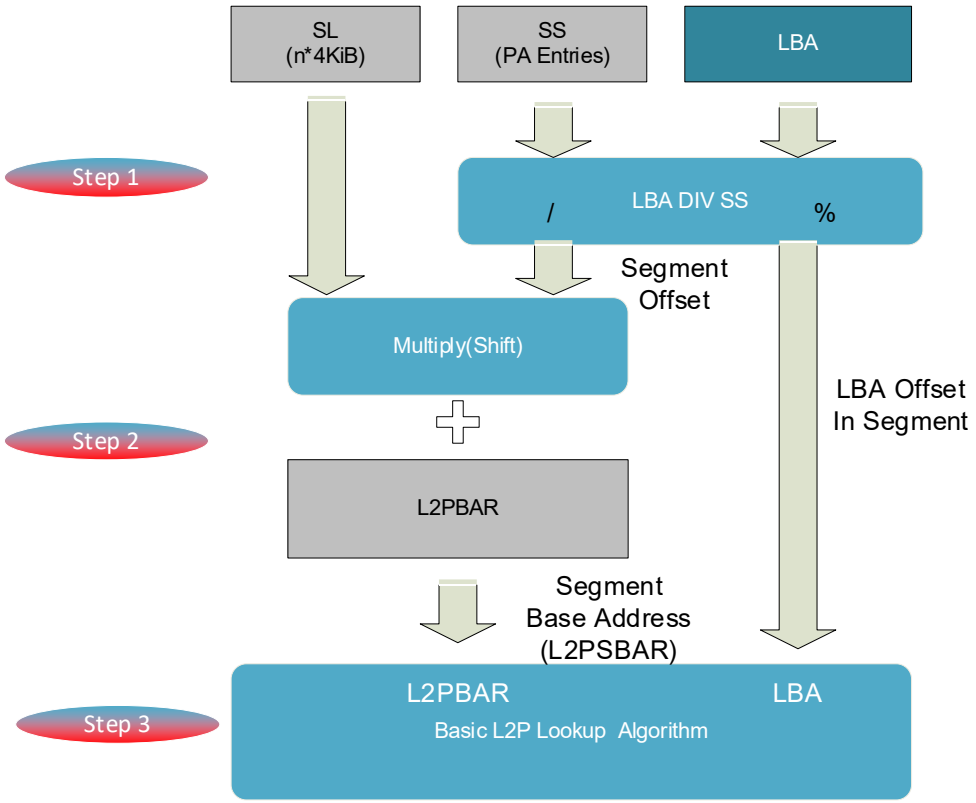
Algorithm: Single-Segment



Entire operation includes one multiplier, 2 shifts and one add arithmetic operations to find out the PA address with

- Cache line offset to the base address,
- Byte offset to the cache line and,
- Bit address to the byte

Algorithm: Multi-Segment



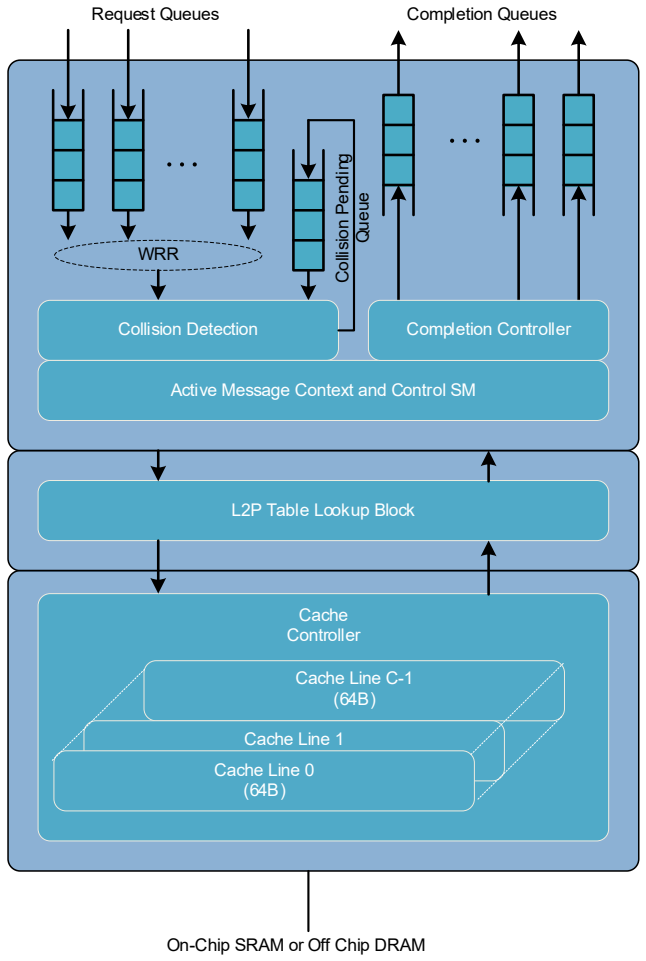
The multi-segment L2P table lookup algorithm is first to find:

- The segment address offset.
- LBA offset in the segment

Then use the segment address offset as a new base address and LBA offset in the segment as a new LBA to find PA bit address with single-segment lookup algorithm

Implementation

L2P Lookup Accelerator



- Queued request and completion interface
  - Requests from and completions to SMP cores in the controller. Used as a sync point for SMP cores.
  - No in-order limitation in the same queue or between queues.
  - Read and write L2P table for a LBA range.
    - Read: Request message carries LBA range. Completion message carries PAs.
    - Update: Request message carries both LBA range and PAs
- Collision detection for overlapped LBA ranges
  - No two collided LBA ranges in active state.
- Multiple active requests
  - For performance optimization.
- Cache Management
  - Access L2P table memory region in granularity of 64B cacheline.
  - Optimized for temporal locality
  - PA alignment in cache

Application

Usage Example of bit-pack L2P Entry for Large SSD

- Problem for large size SSD drive (64TB/128TB)
  - BOM cost for DRAM dies
  - PCB placement problem for DRAM dies in U.2, E3.S, and E1.L.
- The efficient way to reduce L2P Table size is to combine
  - large IU size and bit-pack L2P Entry (32/33/34/35/36bit PA per entry)

  L2P Size = ((SSD Size)/IU Size)(PA bits/8)*
- L2P Size for different IU sizes shown as the table below

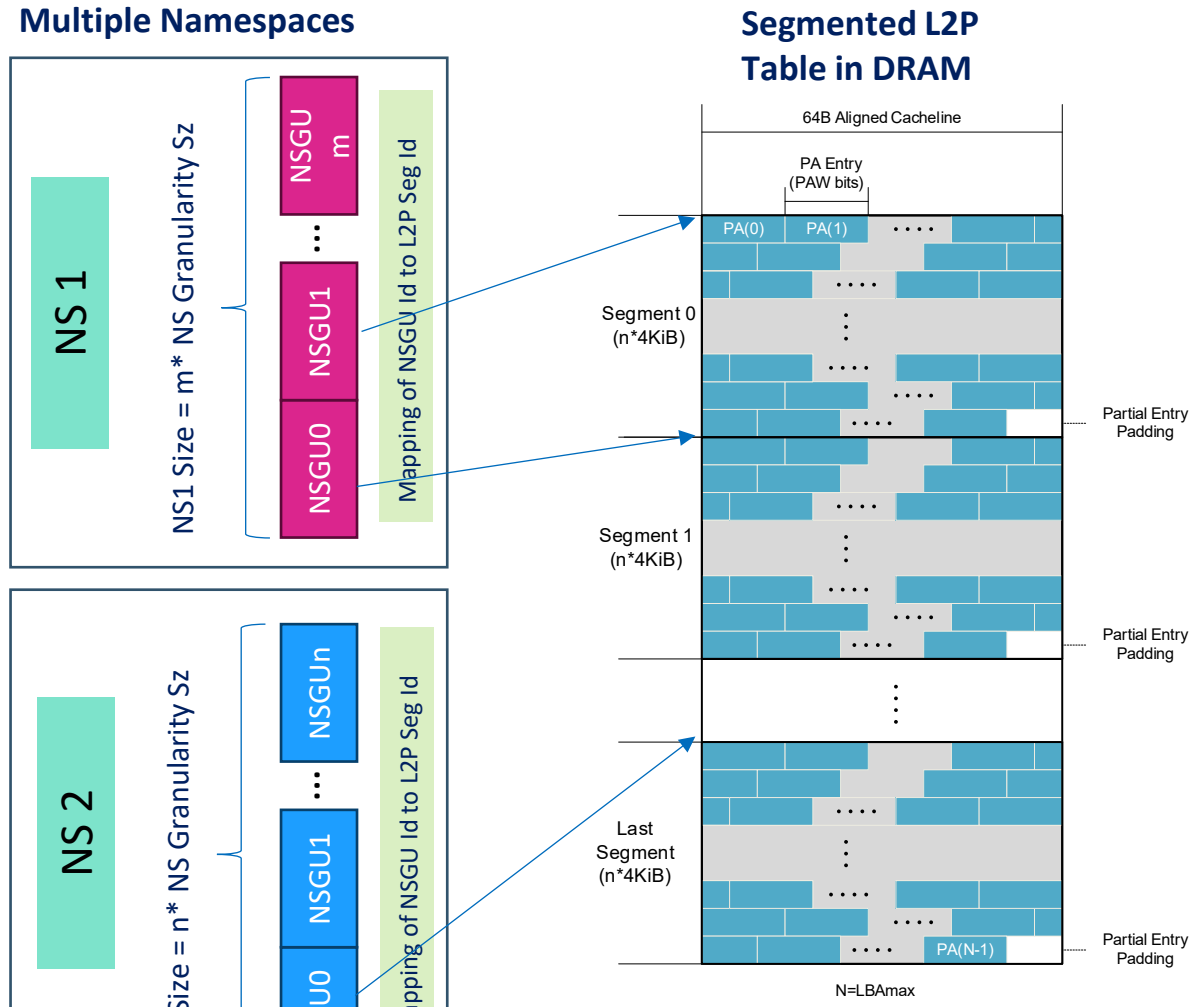
	4K IU		8K IU		16K IU	
SSD Size TB	***PA bits***	***L2P Size (GiB)***	***PA bits***	***L2P Size (GiB)***	***PA bits***	***L2P Size (GiB)***
16	33	15.01	32	7.28	32	3.64
32	34	30.92	33	15.01	32	7.28
64	35	63.66	34	30.92	33	15.01
128	36	130.97	35	63.66	34	30.92

Usage Example of Atomic L2P Access Operation

L2P Accelerator Atomic Access Operations	Use case
L2P Entry Read (start LBA, 1-8 of LBAs) Return the status & the corresponding 1-8 of PPAs in the Completion Message	Used in processing Host Read IO
L2P Entry Write (start LBA, 1-8 of LBAs, PPA list) : 1-8 PPAs of the consecutive 1-8 LBAs Return the status	Used in processing TRIM/De-allocate
L2P Entry Read First then Write (start LBA, 1-8 of LBAs, new PPA list) Return the status and old PPA list	Used in processing Host Write IO
L2P Entry Compare PPA and swap (start LBA, 1-4 of LBAs, old PPA list, new PPA list)	Used in processing backend write, Garbage Collection write

- Atomic operation means one operation which includes a sequence of L2P Table accesses to 1 or multiple L2P entries must be performed without being interrupted by other operations that has any overlapped L2P entry.
- The benefit of atomic L2P access is to reduce the LBA range lock use in IO control path and therefore reduce IO latency.

Usage of Multi-segment L2P Table



- In NVMe 2.0 multi-Namespace configuration, each NS is usually created with *n* of NS Granularity Units (NSGU) and one map table between NSGUs to assigned L2P segments.
- L2PAC engine supports configurable segment size. No any L2P PA spreading over consecutive two segments can save CPU time to calculate the L2P entry offset during L2P lookup.

Summary

- L2P Table and Lookup Algorithm for single and multiple segment memory regions with minimal memory size.
- HW implemented L2P table lookup accelerator for multi SMP core requests
- Application for SSD L2P table lookup.

Meet us at booth #315

SiliconMotion

Driving AI Innovation in Flash Storage

Scan to learn more!