ISSN No:-2456-2165

Design and Implementation of Sequential Read Ahead in Zoned Namespaces for Solid State Drives

Adnan Asad Vohra, Dr. Srividya P.
Department of Electronics and Communication Engineering
RV College of Engineering, Bengaluru, India – 560059

Abstract:- In the day and age of data and information, the ability to retrieve the information becomes of paramount importance. The cutting edge technology in terms of data storage is currently Solid State Devices which use NAND gates to store data. This relatively new method of storing data presents numerous avenues of research and breakthroughs. The concept of Zoned Namespaces in SSD firmware is one such major avenue of ongoing research. The objective of this paper is to understand the need for higher data accessing speeds and envisioning the advancements made possible by improving basic read and write speeds in SSD's. The goal is to allow for writing sequential data in namespaces where the data related to each other holds a granularity of a single zone. The idea is achieved by implementing sequential read ahead where the sequential data is read ahead of time by anticipating host read request to that data. This gives the host, cache hit on requested data which greatly improves performance. The pre-fetched data is cleared from cache once the data has been read or any disabling condition occurs thus not hampering normal functioning of the drive. The implementation was tested on a 8 TB form factor SSD. The results for reads were 70 MB/s for ZNS before SRA and 275 MB/s after SRA enablement. Thus a very significant increase is observed which proves that the objective was achieved.

Keywords:- *Sequential Read Ahead, Zoned Namespaces, Cache Memory, NAND, Static Random Access Memory, Queue Depth, Non-Volatile Memory Express.*

I. INTRODUCTION

Flash Translation Layer (FTL) is a key technology in a Solid--State Disk (SSD) system to manage the data transfer. Different mapping granularity in FTL will cause a change in the read/write performance and the size of mapping table. A zone is a range of logical block addresses that is managed as a single unit. Namespaces are a quantity of non-volatile memory that may be formatted into logical blocks. When formatted, a namespace of size n is a collection of logical blocks with logical block addresses from 0 to (n-1). Combining both these terms gives rise to Zoned Namespaces (ZNS), a namespace that is divided into zones and is operated by the Zoned Command Set. The data in the zones are sequential which results in highly cohesive data inside a zone.

Sequential Read Ahead (SRA) is the mechanism of reading sequential data before it has been requested by anticipating read request. This is done to improve response time by having higher cache hit ratio. The data is pre-fetched from the NAND and kept in SRAM cache for faster access times. For most cases of read the function should work properly while additional cases need to be added for defragmented zones and avoiding reading of holes between two zones. Flash Translation Layer Based on Grouping Pages was studied and the conclusion obtained was that using pg-FTL algorithm to divide the mapping table into three levels, the size of mapping table will fall dramatically because of the channel parallelism in pg-FTL. There was also an improvement under read requests comparing to the page mapping, especially sequential read requests [1]. On Investigating Hybrid SSD FTL Schemes for Hadoop Workloads the results presented included logical-physical mappings, I/O request size analysis, erase and merge counts and sensitivity to some important FTL parameters. Correlation was done to the workload behavior for most of the results. GC, Erase counts and lifecycle greatly improved [2].

II. METHODOLOGY ADOPTED

A. Design

The design of the project is carried out by first identifying the states of the SRA function. Next the conditions for reaching those states are listed mainly the enabled and disabled states. The interdependencies are also taken into account while also considering corner cases which might results in minor unwanted behavior of the system. The penalty for cache usage is also taken into account so as to not use too much RAM unnecessarily which can make the whole device slow thus countering what was intended.

The design of SRA shall be expandable from single stream to multi-streams, and multi-namespaces. When host issues low Queue Depth (QD) small command size sequential reads, after a ramp-up time, SRA shall be able to provide cache hit of host reads to boost performance number. If the Read across Zone Boundaries bit in the Identify Namespace data structure is set to ‘1’, then read operations are allowed to cross zone boundaries. If the Read across Zone Boundaries bit is cleared to ‘0’ the read operations are not allowed to cross zone boundaries.

SRA will consist of five states which are as shown in Table 1 –

Current SRA state	Valid Next SRA States
SRA_DISABLED	SRA_RAMPING
SRA_RAMPING	SRA_ENABLED, SRA_DISABLING
SRA_ENABLED	SRA_PAUSED, SRA_DISABLING
SRA_PAUSED	SRA_ENABLED, SRA_DISABLING
SRA_DISABLING	SRA_DISABLED

Table 1:- Valid state transitions of SRA

Conditions for enabling SRA –

- Host is not issuing large host read command
- Host is not issuing high QD host read commands
- There is no host write command in-flight
- There is no unmap command in-flight
- Host is issuing sequential read commands

The conditions for disabling SRA are – Any of the following condition (or conditions) can cause SRA going back to disabled, which is the default state

- New host write comes in
- New unmap command comes in
- Large host read command comes in
- High QD read commands come in
- Random read command comes in
- Sequential low QD read has been idle from host for X seconds

Once the data has been read and SRA is going back to SRA_DISABLED state the data present in cache has to be removed to empty the Random Access Memory (RAM). This process is called cache entry (CE) teardown. SRA cache entries that have been inserted into the L2P must be torn down by calling the CE rollback processor. Any other mechanism for tearing down the CE is invalid. CEs can either be torn down individually, in the case of Cache Entry Pruning, or collectively when the SRA transitions to state SRA_DISABLING.

B. Implementation

The implementation done is as follows in Fig. 2. The figure shown is a UML diagram in which the arrows indicates transitions from one function to the other in firmware while the task done by the functions is given numbered. The figure gives a series of steps which help execute SRA.

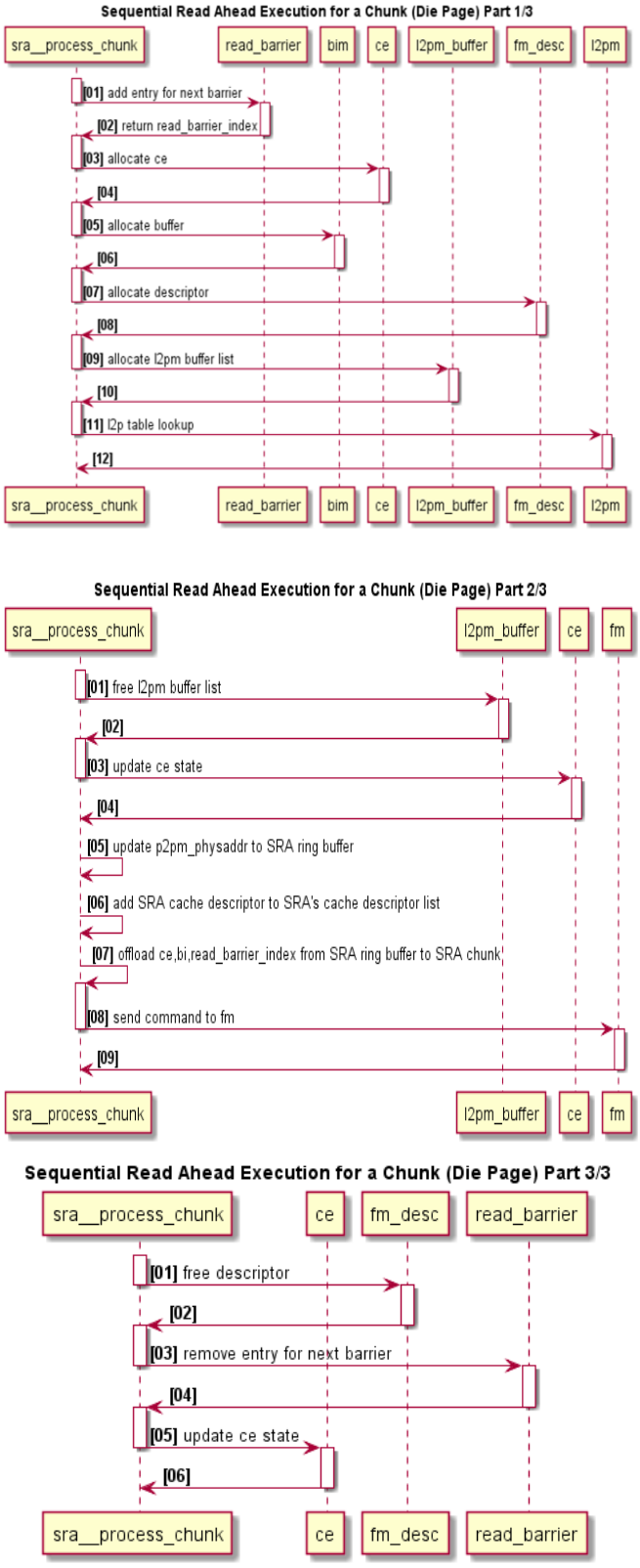

Fig.2: UML Diagram of Sequential Read Ahead

ISSN No:-2456-2165

The program code was written in C language and for the testing scripts, Python was used. The firmware logic was tested on an 8 TB SSD drive.

III. RESULTS AND ANALYSIS

The random read and write speeds for conventional SSD and zoned SSD's are shown below in Fig. 3 and Fig. 4. respectively. For sequential data which is specifically important for zoned namespaces the read and write speeds get boosted up to 1.5 GB/s at a Queue Depth (QD) of 128 and a block size of 512 KB as shown in Fig. 5.

```
32+0 records in
32+0 records out
131072 bytes (131 kB, 128 KiB) copied, 0.00203747 s, 64.3 MB/s
32+0 records in
32+0 records out
131072 bytes (131 kB, 128 KiB) copied, 0.00203319 s, 64.5 MB/s
32+0 records in
32+0 records out
131072 bytes (131 kB, 128 KiB) copied, 0.00110792 s, 118 MB/s
```

Fig 3:- R/W speeds for random data in conventional SSD

```
32+0 records in
32+0 records out
131072 bytes (131 kB, 128 KiB) copied, 0.000209066 s, 627 MB/s
read: Success
Zone LBAs Trimmed passed
write: Success
Zone state change to IMP_OPENED passed.
read: Success
IMP_OPEN Zone LBA read passed
Resetting all zones with NS format
Reset all zones passed
```

Fig 4:- R/W speeds for random data in Zoned SSD

```
/dev/nvme0n1: (g=0): rw=write, bs=(R) 512KiB-512KiB, (W) 512KiB-512KiB, (T) 512KiB-512KiB, ioengine=libaio, iodepth=128
fio-3.15
Starting 1 process
Jobs: 1 (f=1): [W(1)][100.0%][w=1428MiB/s][w=2856 IOPS][eta 00m:00s]
/dev/nvme0n1: (groupid=0, jobs=1): err= 0: pid=1708: Mon May  4 21:42:51 2020
  write: IOPS=2863, BW=1432MiB/s (1501MB/s)(200GiB/143024msec); 91 zone resets
    slat (usec): min=13, max=116, avg=52.47, stdev= 9.45
    clat (msec): min=3, max=122, avg=43.59, stdev= 8.35
     lat (msec): min=3, max=122, avg=43.64, stdev= 8.35
    clat percentiles (msec):
     |  1.00th=[   11],  5.00th=[   36], 10.00th=[   40], 20.00th=[   41],
     | 30.00th=[   42], 40.00th=[   43], 50.00th=[   44], 60.00th=[   45],
     | 70.00th=[   46], 80.00th=[   47], 90.00th=[   48], 95.00th=[   52],
     | 99.00th=[   75], 99.50th=[   80], 99.90th=[   84], 99.95th=[   87],
     | 99.99th=[  107]
   bw (  MiB/s): min= 1231, max= 1575, per=99.99%, avg=1431.72, stdev=48.21, samples=286
   iops        : min= 2462, max= 3150, avg=2863.45, stdev=96.43, samples=286
  lat (msec)   : 4=0.01%, 10=0.96%, 20=1.58%, 50=91.16%, 100=6.30%
  lat (msec)   : 250=0.01%
  cpu          : usr=13.00%, sys=7.52%, ctx=313248, majf=0, minf=11
  IO depths    : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.2%, 16=0.4%, 32=0.7%, >=64=98.6%
     submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.1%
     issued rwts: total=0,409600,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=128

Run status group 0 (all jobs):
  WRITE: bw=1432MiB/s (1501MB/s), 1432MiB/s-1432MiB/s (1501MB/s-1501MB/s), io=200GiB (215GB), run=143024-143024msec
```

Fig 5:- Write speeds for sequential data in Zoned SSD

```
/dev/nvme0n1: (g=0): rw=read, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=libaio, iodepth=1
fio-3.15
Starting 1 process
Jobs: 1 (f=1): [R(1)][52.1%][r=260MiB/s][r=66.7k IOPS][eta 06m:15s]
/dev/nvme0n1: (groupid=0, jobs=1): err= 0: pid=1938: Thu May 28 22:21:13 2020
  read: IOPS=66.9k, BW=262MiB/s (274MB/s)(104GiB/408364msec)
    slat (nsec): min=1947, max=42642, avg=2960.05, stdev=755.10
    clat (nsec): min=648, max=1049.4k, avg=10930.50, stdev=4183.97
     lat (usec): min=10, max=1052, avg=13.97, stdev= 4.38
    clat percentiles (usec):
     |  1.00th=[   10],  5.00th=[   10], 10.00th=[   11], 20.00th=[   11],
     | 30.00th=[   11], 40.00th=[   11], 50.00th=[   11], 60.00th=[   11],
     | 70.00th=[   11], 80.00th=[   11], 90.00th=[   11], 95.00th=[   12],
     | 99.00th=[   25], 99.50th=[   34], 99.90th=[   87], 99.95th=[  100],
     | 99.99th=[  106]
   bw (  KiB/s): min=255896, max=278792, per=99.99%, avg=267775.77, stdev=6475.24, samples=816
   iops        : min=63974, max=69698, avg=66943.92, stdev=1618.78, samples=816
  lat (nsec)   : 750=0.02%, 1000=0.01%
  lat (usec)   : 2=0.01%, 4=0.01%, 10=5.36%, 20=93.43%, 50=0.99%
  lat (usec)   : 100=0.15%, 250=0.05%, 500=0.01%, 750=0.01%
  lat (msec)   : 2=0.01%
  cpu          : usr=27.84%, sys=23.58%, ctx=27331829, majf=0, minf=15
  IO depths    : 1=100.0%, 2=0.0%, 4=0.0%, 8=0.0%, 16=0.0%, 32=0.0%, >=64=0.0%
     submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     issued rwts: total=27339008,0,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=1

Run status group 0 (all jobs):
   READ: bw=262MiB/s (274MB/s), 262MiB/s-262MiB/s (274MB/s-274MB/s), io=104GiB (112GB), run=408364-408364msec
```

Fig 6:- Read speeds for sequential data in Zoned SSD

Additionally the reads with SRA shown in Fig. 6. enabled show a significant increase in the read speeds reaching up to 275 MB/s at a QD of 1 and block size of 4 KB. These reads are performed at a low I/O depth and 4K block size for SRA to be enabled. The jump from 70 MB/s which was observed without SRA enablement to 275 MB/s after the implementation of SRA helps understand and appreciate the speed increase in reading data. The results for higher QD reads for both the cases gradually converge as the QD keeps increasing since SRA does not have a significant impact at a higher QD.

To improve I/O performance, this technique is employed in operating systems, where more of a file than was requested is read into memory with the assumption that subsequent reads are likely to need that data. Higher read ahead increases throughput at the expense of memory and Input/output Cycles per second (IOPS). Lower read ahead increases IOPS at the expense of throughput. Higher queue depths increase IOPS but can also increase latency. Lower queue depths decrease per-I/O latency, but might result in lower maximum IOPS. Thus there exists a tradeoff between SRA and QD where SRA is enabled with low QD and disabled when the IO depth of the workload is high.

IV. CONCLUSION

The advancements made by the concept and implementation of Sequential Read Ahead in Zoned Namespaces brings a new benchmark in terms of I/O speeds for Solid State Drives. It does so by addressing the fact that data is now being stored in larger chunks and need to be accessed simultaneously in a quick manner for example large applications, movies, games, data stored in servers and data centers. All these data are cohesive in nature and are generally accessed in sequential fashion. These areas of data storage greatly benefit from the concept of zoned storage and read ahead mechanisms to cope with the growing demand for faster access times and response times of systems.

The read and write speeds obtained in the results prove that the access speeds have significantly been improved and will thus result in faster and better systems. It will also enable applications to take advantage of these speeds and push data storage to a new boundary where even larger data will be accessible in a short span of time resulting in more complex application and heavier data storage and usage around the globe.

ACKNOWLEDGMENT

I am indebted to my guide and co-author, Dr. Srividya P, Associate Professor, RV College of Engineering for the wholehearted support, suggestions and invaluable advice throughout my project work and also helped in the preparation of this paper.

I also express our gratitude to my panel members Dr. Srividya P, Associate Professor and Dr. Shilpa D.R., Associate Professor, Department of Electronics and Communication Engineering for their valuable comments and suggestions during the phase evaluations.

My sincere thanks to Dr. K S Geetha, Professor and Head, Department of Electronics and Communication Engineering, RVCE for the support and encouragement. I express sincere gratitude to our beloved Principal, Dr. K. N. Subramanya for the appreciation towards this project work. I thank all the teaching staff and technical staff of Electronics and Communication Engineering department, RVCE for their help.

Lastly, I take this opportunity to thank my family members and friends who provided all the backup support throughout the project work.

REFERENCES

[1]. Li Wang, Min Zhu, Chunling Yang, Xiaoming Qiu, Research on the Flash Translation Layer Based on Grouping Pages– **Sixth International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC), IEEE, 2016**

[2]. Hyeran Jeon, Kaoutar El Maghraoui, Gokul B. Kandiraju, Investigating Hybrid SSD FTL Schemes for Hadoop Workloads, **IBM T.J. Watson Research Center, United States, 2013**

[3]. Hu Y, Jiang H, Feng D, et al. Performance impact and interplay of SSD parallelism through advanced commands, allocation strategy and data granularity. **International Conference on Supercomputing, Tucson, USA, 2011**

[4]. Dan M, Wang Y, Yang Y. A wear-leveling algorithm based on IO request prediction. **International Conference on System Science, Engineering Design and Manufacturing Informatization. IEEE, 2012**

[5]. C. L. Abad, N. Roberts, Y. Lu, and R. H. Campbell. A storage-centric analysis of mapreduce workloads: File popularity, temporal locality and arrival patterns. **Workload Characterization (IISWC), 2012 IEEE International Symposium**

[6]. NVMe base specification, Revision 1.4, 2019