ZNS+: Advanced Zoned Namespace Interface for Supporting In-Storage Zone Compaction

Kyuhwa Han, Hyunho Gwak, Dongkun Shin, Joo-Young Hwang,

In 2021 USENIX Symposium on Operating Systems Design and Implementation

2021. 12. 15

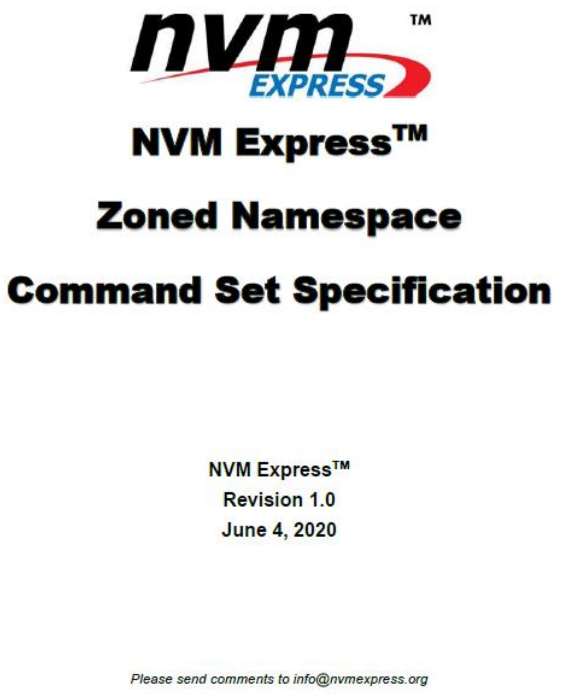
Presentation by Han, Yejin

hyj0225@dankook.ac.kr

Contents

1. Introduction
2. Background
3. Motivation
4. ZNS+: LFS-Aware ZNS
5. ZNS+-aware LFS
6. Evaluation
7. Conclusion

Zoned Namespace (ZNS) Storage



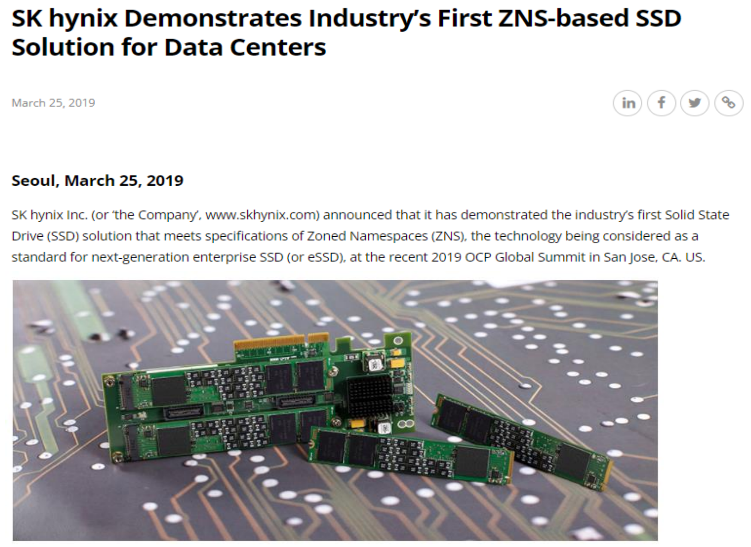
NVM Express™

Zoned Namespace

Command Set Specification

NVM Express™
Revision 1.0
June 4, 2020

Please send comments to info@nvmexpress.org



SK hynix Demonstrates Industry's First ZNS-based SSD Solution for Data Centers

March 25, 2019

Seoul, March 25, 2019

SK hynix Inc. (or 'the Company', www.skhynix.com) announced that it has demonstrated the industry's first Solid State Drive (SSD) solution that meets specifications of Zoned Namespaces (ZNS), the technology being considered as a standard for next-generation enterprise SSD (or eSSD), at the recent 2019 OCP Global Summit in San Jose, CA. US.

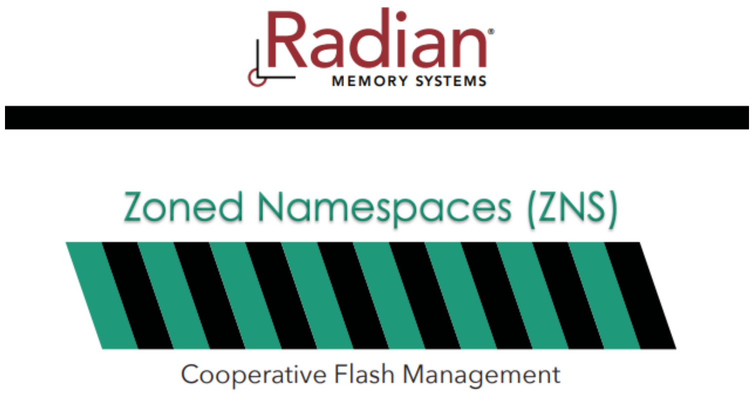
삼성전자, 차세대 기업 서버용 'ZNS SSD' 출시

2021/06/02

삼성전자가 ZNS(Zoned Namespace) 기술을 적용한 차세대 엔터프라이즈 서버용 솔리드스테이트드라이브(SSD)를 2일 출시했다.

신제품 ZNS SSD PM1731a는 6세대 V낸드 기반의 4TB, 2TB 용량 2.5인치 제품으로 출시되며, 올해 하반기부터 본격 양산될 예정이다.

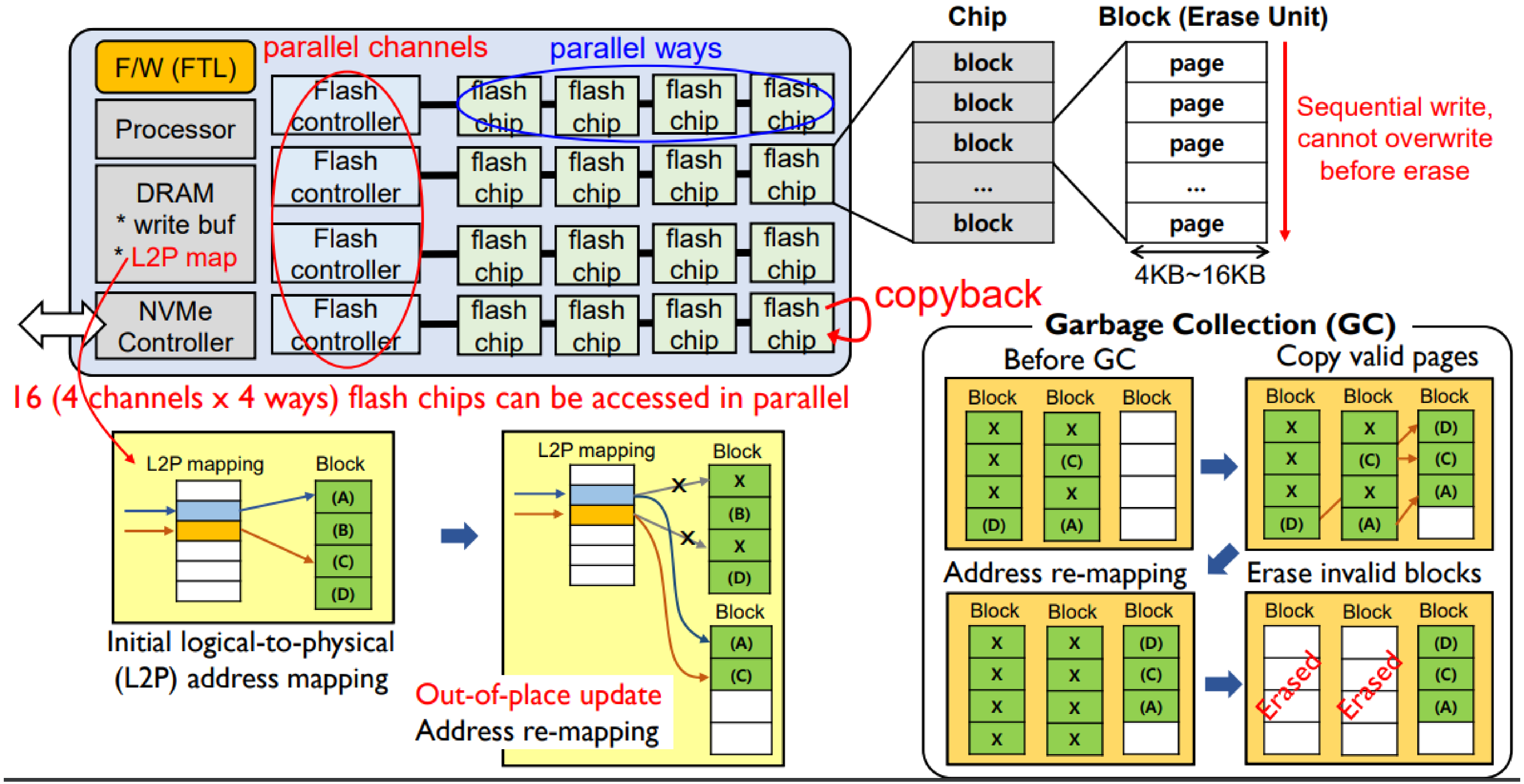
이번 SSD의 가장 큰 특징은 ZNS 기술이 적용됐다는 점이다. ZNS는 SSD 전체 저장 공간을 작고 일정한 용량의 구역(Zone)으로 나누고 용도와 사용 주기가 같은 데이터를



ZNS SSDs

Accelerating applications for data at scale

Conventional SSD Architecture

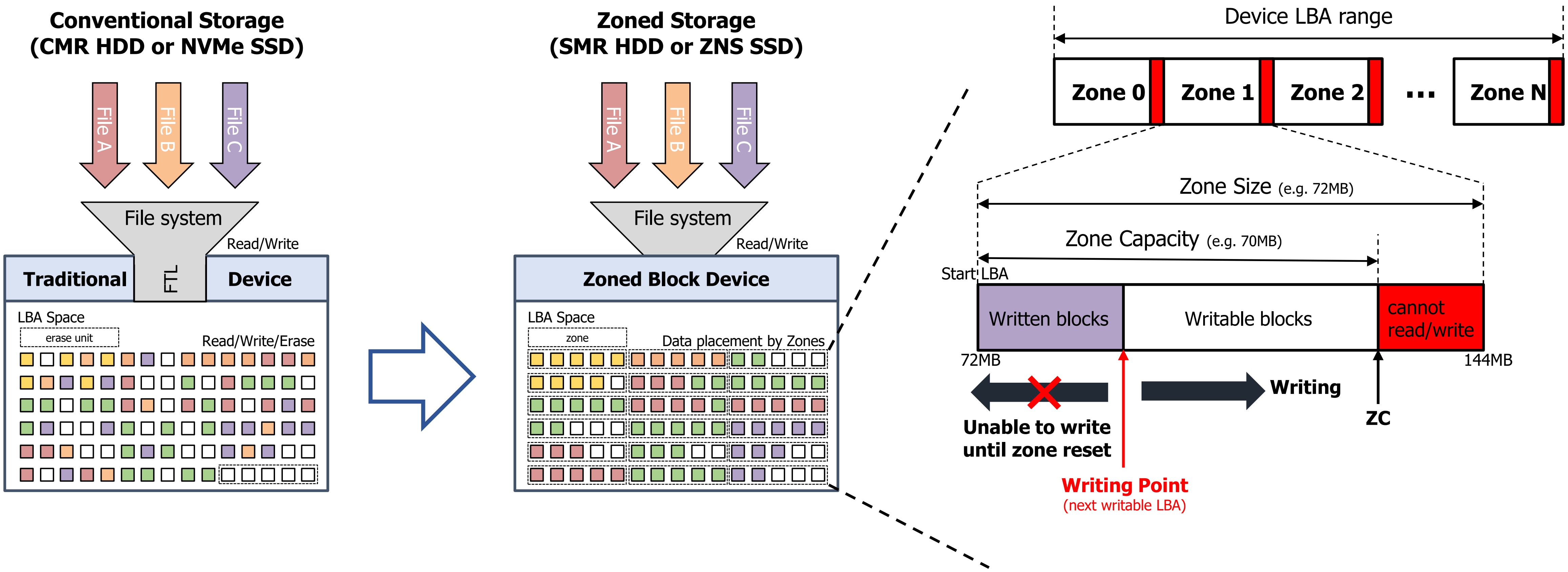


16 (4 channels x 4 ways) flash chips can be accessed in parallel

Initial logical-to-physical (L2P) address mapping

Out-of-place update
Address re-mapping

Zoned Namespace SSD



Hot/cold separation, Small Mapping table(DRAM), GC-less

Zone mapping in ZNS SSD

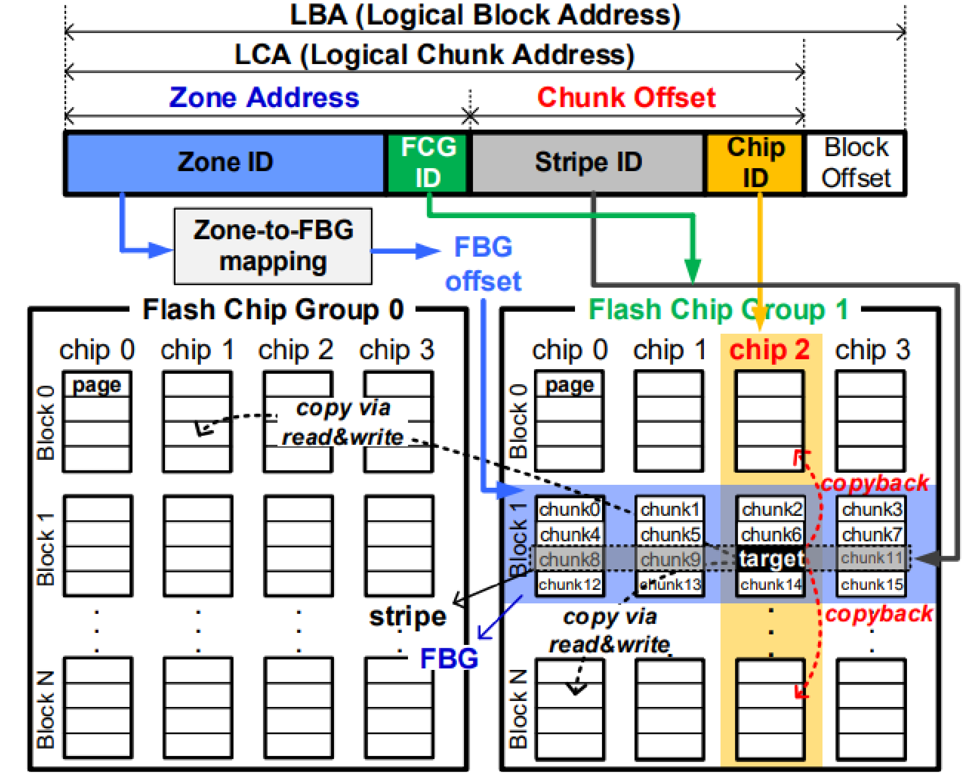


Figure 1: An example of zone and chunk mapping. With the zone address, the second FBG in the FCG 1 is selected. With the chunk offset, the third stripe in the selected FBG and the third flash page (chip 2) in the stripe are targeted.

Zone-to-FBG mapping

Zone	FBG	LogFBG	WP
0	5	-	32
1	6	15	6
2	7	-	124
⋮	❶ LogFBG allocation		⋮

F2FS Segment Management

- One of actively mainted Log-structured File Systems
- 6 types of segments: hot, warm and cold segments for each node & data
- Supproting both append logging and threaded logging
- Threaded logging is disabled in the patch version for ZNS

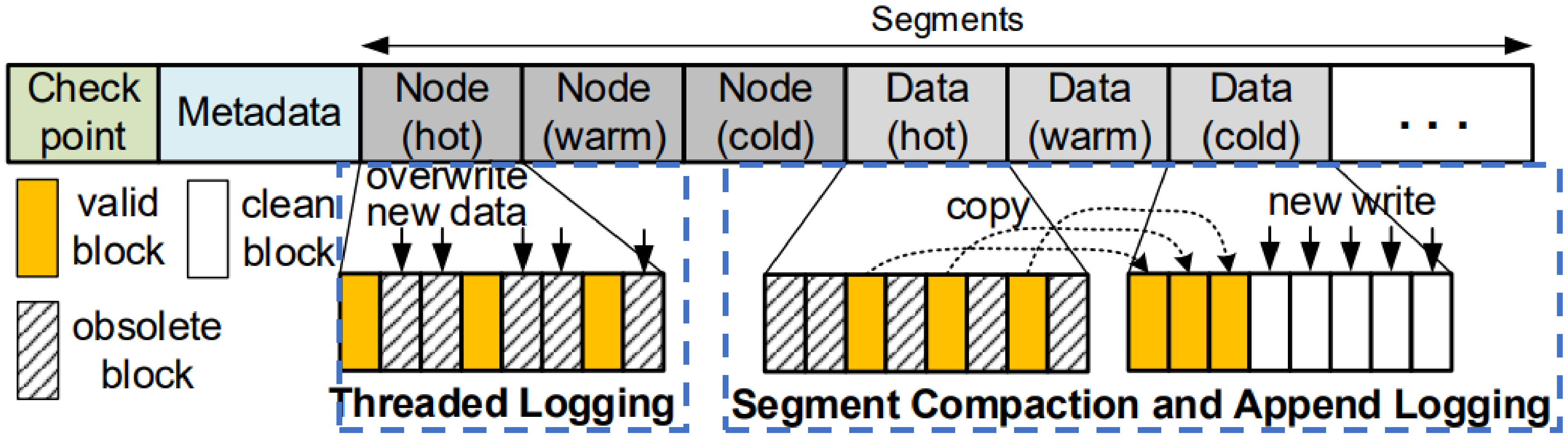
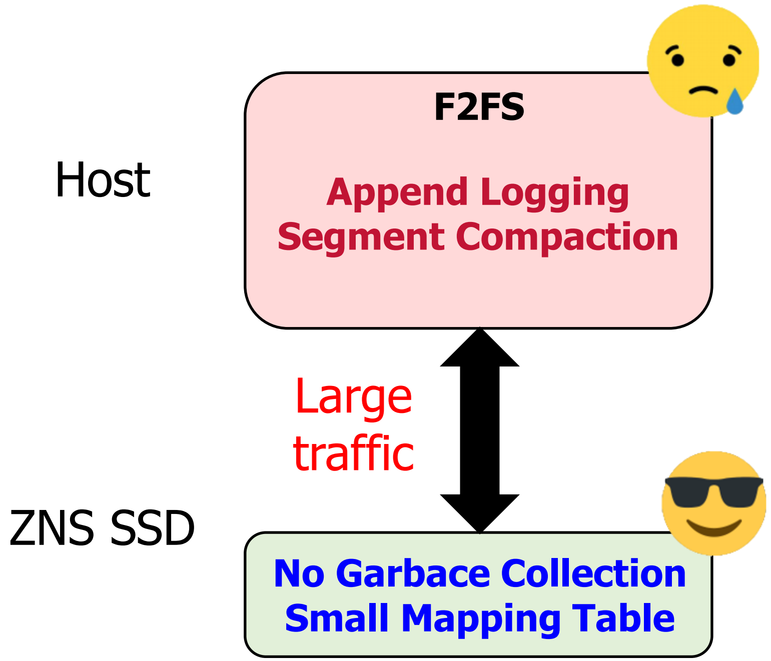


Figure 2: F2FS disk layout and logging schemes.

Increased Host Overhead to simplify SSDs

- Host-side GC overhead > Device-side GC overhead



Host

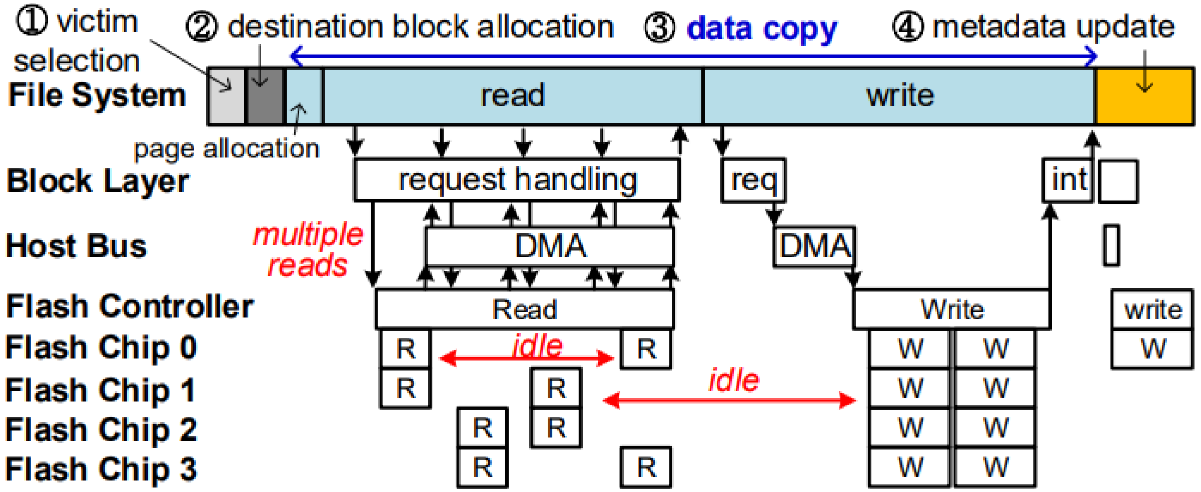
F2FS

Append Logging
Segment Compaction

Large traffic

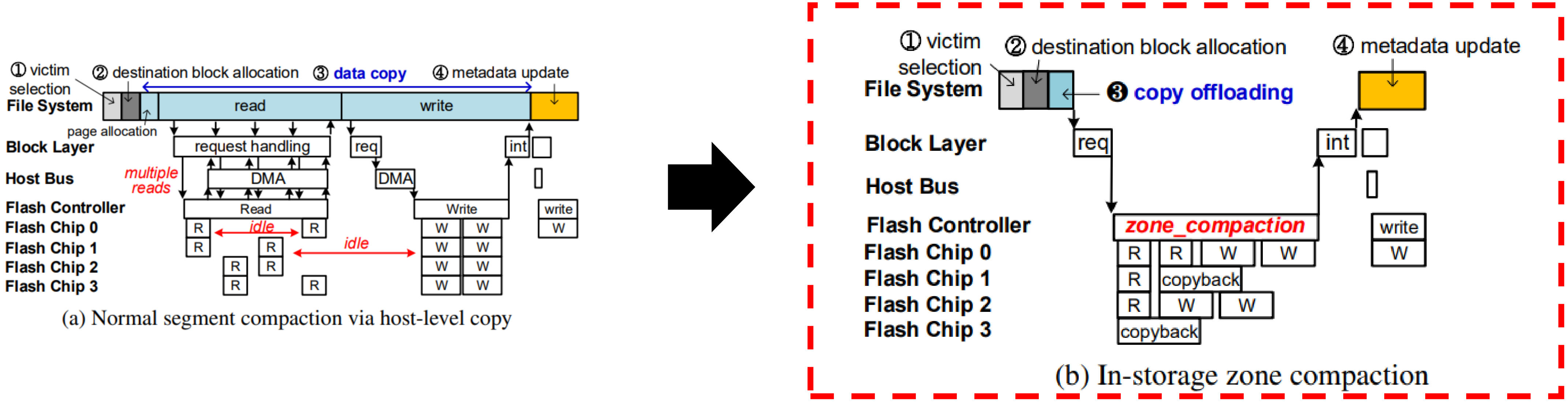
ZNS SSD

No Garbace Collection
Small Mapping Table



(a) Normal segment compaction via host-level copy

Internal Zone Compaction (IZC)

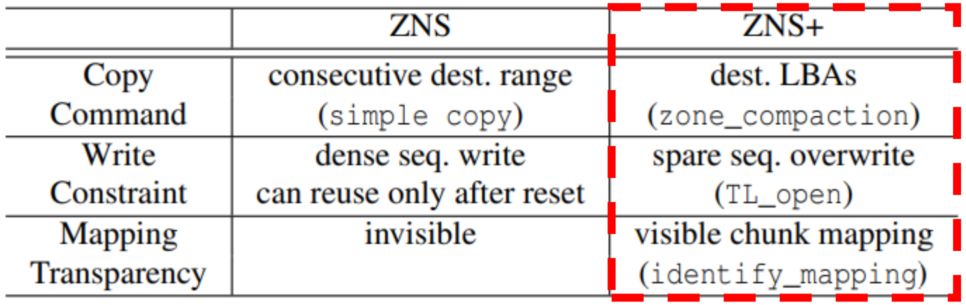


(a) Normal segment compaction via host-level copy

(b) In-storage zone compaction

Figure 3: Segmentation compaction on (a) ZNS vs. (b) ZNS+ interface (Time goes to the right.)

Table 1: Comparison between ZNS and ZNS+

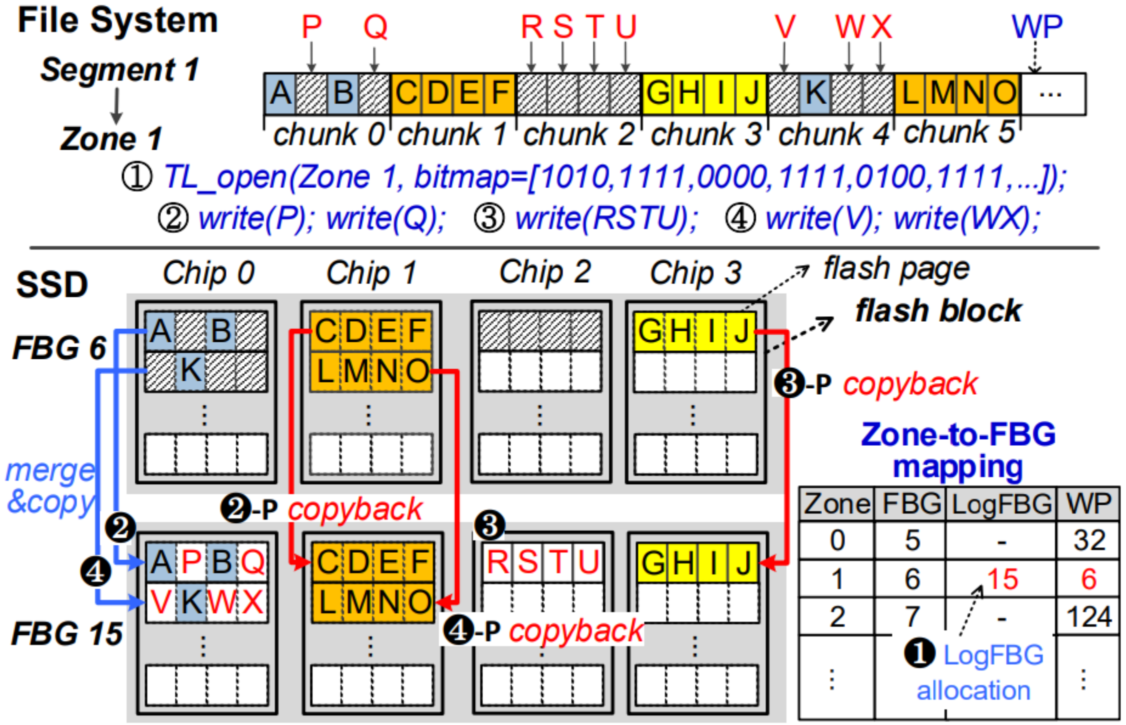


	ZNS	ZNS+
Copy Command	consecutive dest. range (simple copy)	dest. LBAs (zone_compaction)
Write Constraint	dense seq. write can reuse only after reset	spare seq. overwrite (TL_open)
Mapping Transparency	invisible	visible chunk mapping (identify_mapping)

Sparse Sequential Overwrite

- Threaded logging-based block reclamation

- LBA-ordered Plugging(LP)
- PPA-ordered Plugging(PP)



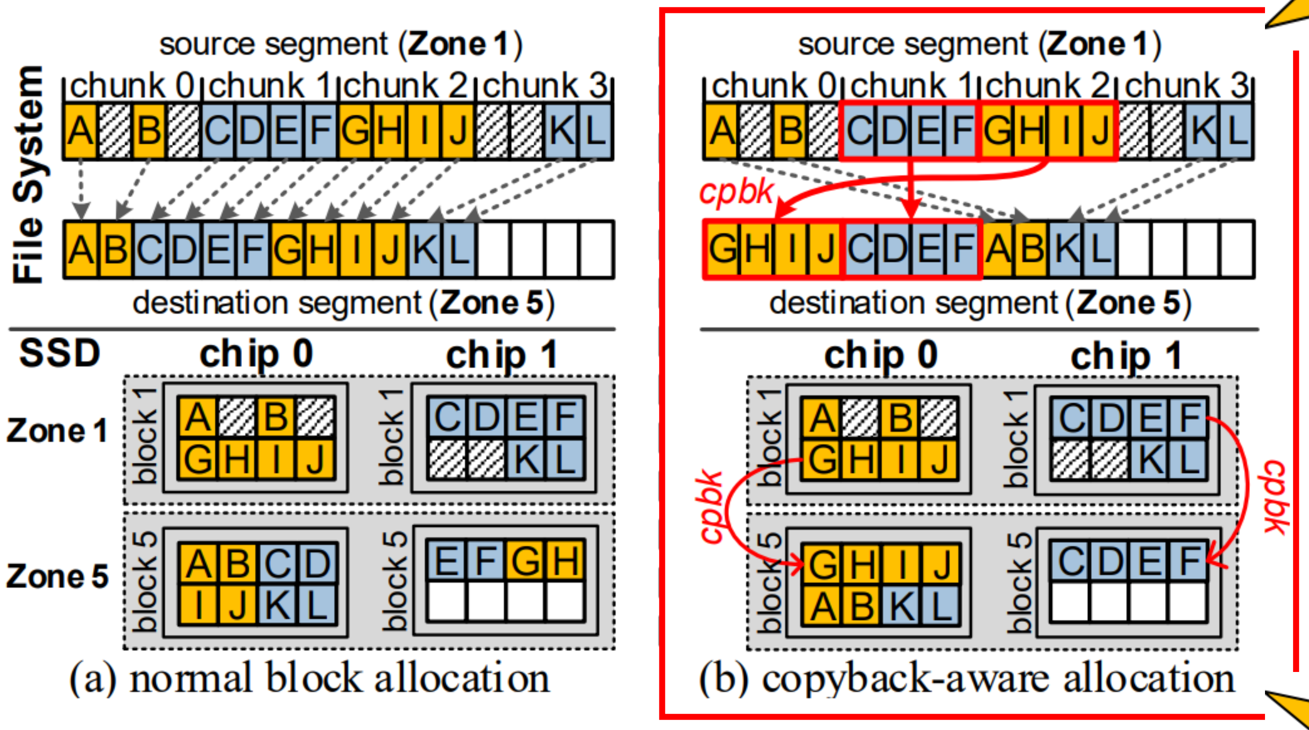
① TL_open(Zone 1, bitmap=[1010,1111,0000,1111,0100,1111,...]);
② write(P); write(Q); ③ write(RSTU); ④ write(V); write(WX);

Zone	FBG	LogFBG	WP
0	5	-	32
1	6	15	6
2	7	-	124
⋮			⋮

Figure 4: Skipped block plugging for threaded logging

Copyback-aware Block Allocation

I see.
Then, I will use a copyback-aware block allocation

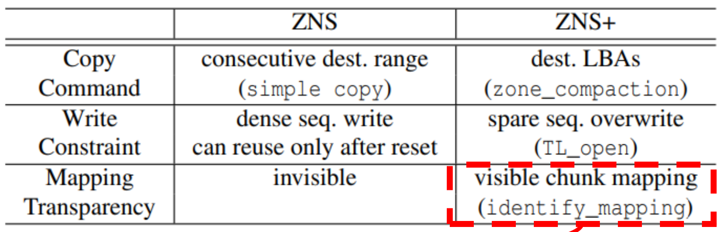


(a) normal block allocation

(b) copyback-aware allocation

Figure 5: Copyback-aware block allocation

Table 1: Comparison between ZNS and ZNS+



	ZNS	ZNS+
Copy Command	consecutive dest. range (simple copy)	dest. LBAs (zone_compaction)
Write Constraint	dense seq. write can reuse only after reset	spare seq. overwrite (TL_open)
Mapping Transparency	invisible	visible chunk mapping (identify_mapping)

Do you know I can copy data quickly within a flash chip? It's copyback.

Hybrid Segment Recycling (HSR)

How about the reclaiming efficiency of threaded logging? 😢

- Reclaiming Cost Imbalance: Only same type of dirty segment must be selected
- Pre-invalid block problem: accumulated as threaded logging continues without checkpointing

Solutions! 😎

- Periodic Checkpointing (when *pre-invalid blocks* > θ_{PI})
- **Reclaiming Cost Modeling**

Threaded logging cost $$C_{TL} = f_{plugging}(N_{pre\text{-}inv} + N_{valid}) \qquad (1)$$

Segment compaction cost $$C_{SC} = f_{copy}(N_{valid}) + f_{write}(N_{node} + N_{meta}) - B_{cold} \qquad (2)$$

Experimental Setup

- ZNS+ emulator based on FEMU
- Real ZNS+ implemented at Cosmos+ OpenSSD
- Modified F2FS 4.10
- Comparison: ZNS vs. IZC (internal zone compaction, no TL) vs. ZNS+ (IZC and TL)

The CASE of FEMU:
Cheap, Accurate, Scalable and Extensible Flash Emulator

Huaicheng Li, Mingzhe Hao, Michael Hao Tong,
Swaminatahan Sundararaman*, Matias Bjørling+, Haryadi S. Gunawi

THE UNIVERSITY OF CHICAGO ParallelM* CNEXLABS+

Cosmos/Cosmos+ OpenSSD

5. Evaluation

▪ Segment Compaction Performance

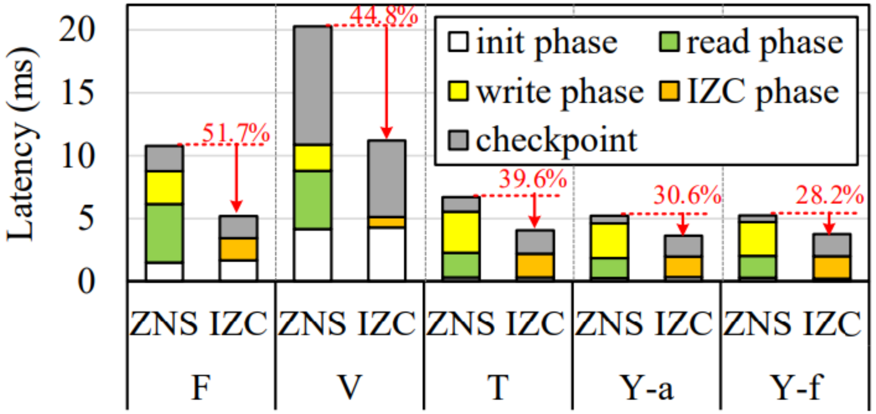


Figure 6: Average compaction time (F: fileserver, V: varmail, T: tpcc, Y-a: YCSB workloada, and Y-f: YCSB workloadf)

▪ Threaded Logging Performance

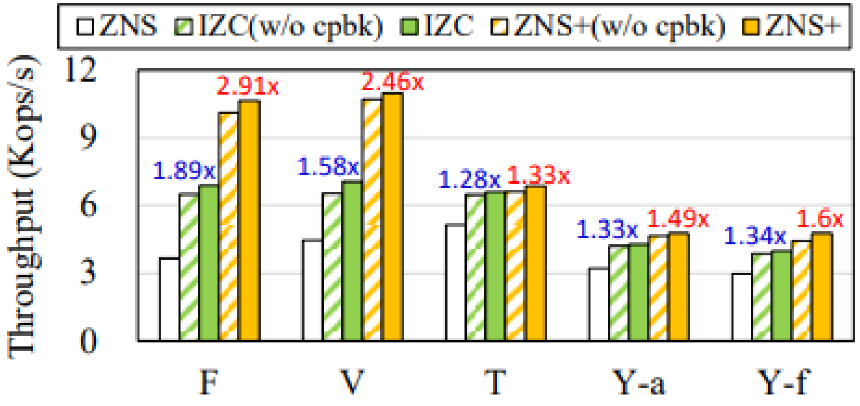


Figure 7: Effect of the threaded logging support

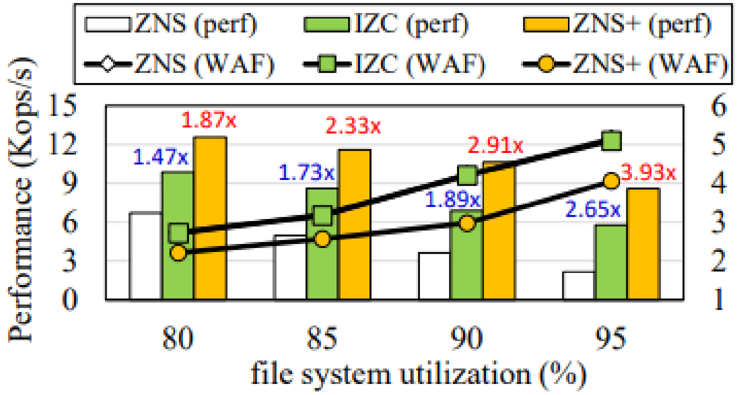
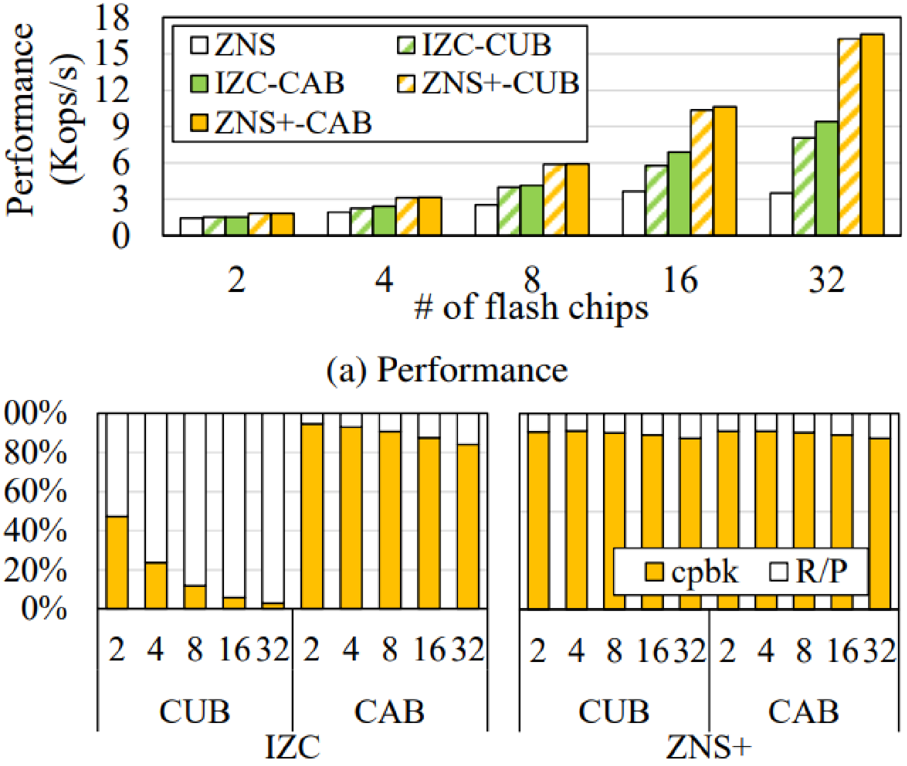


Figure 8: Performance at various file system utilizations (fileserver workload)

- **Copyback-aware Block Allocation**



(a) Performance

(b) Internal copy operation breakdown

Figure 11: Performance comparison for varying chip-level parallelisms (fileserver workload)

6. Conclusion

ZNS+

- ZNS+ offloads block copy operations to the SSD, supporting IZC(Internal zone compaction) , sparse sequential overwrite
- For ZNS+-aware filesystem, it allows the copyback-aware block allocation and HSR
- File system performance of ZNS+ system is better than the normal ZNS-based system

https://github.com/eslab-skku/ZNSplus

ZNS+: Advanced Zoned Namespace Interface for Supporting In-Storage Zone Compaction

Kyuhwa Han, Hyunho Gwak, Dongkun Shin, Joo-Young Hwang,

In 2021 USENIX Symposium on Operating Systems Design and Implementation

Thank You!

2021. 12. 15

Presentation by Han, Yejin

hyj0225@dankook.ac.kr