STORAGE DEVELOPER CONFERENCE

SDC22 Fremont, CA September 12-15, 2022

BY Developers FOR Developers

A SNIA. Event

A New Adapter for Zoned Namespace SSDs

Presented by

Hui Qi (Samsung R&D Institute, China, Xi'an)

Agenda

- **Overview of Zoned Namespace (ZNS) SSDs**
- **Zoned Device Software Ecosystem**
- **Introduction to the Adapter (xZTL)**
- **Application Benchmark (Conventional SSD vs ZNS SSD)**
  - RocksDB Benchmark
  - Percona Benchmark
- **Building ZNS Ecosystem Together**

Overview of ZNS SSDs

Overview of ZNS SSDs (1/2)

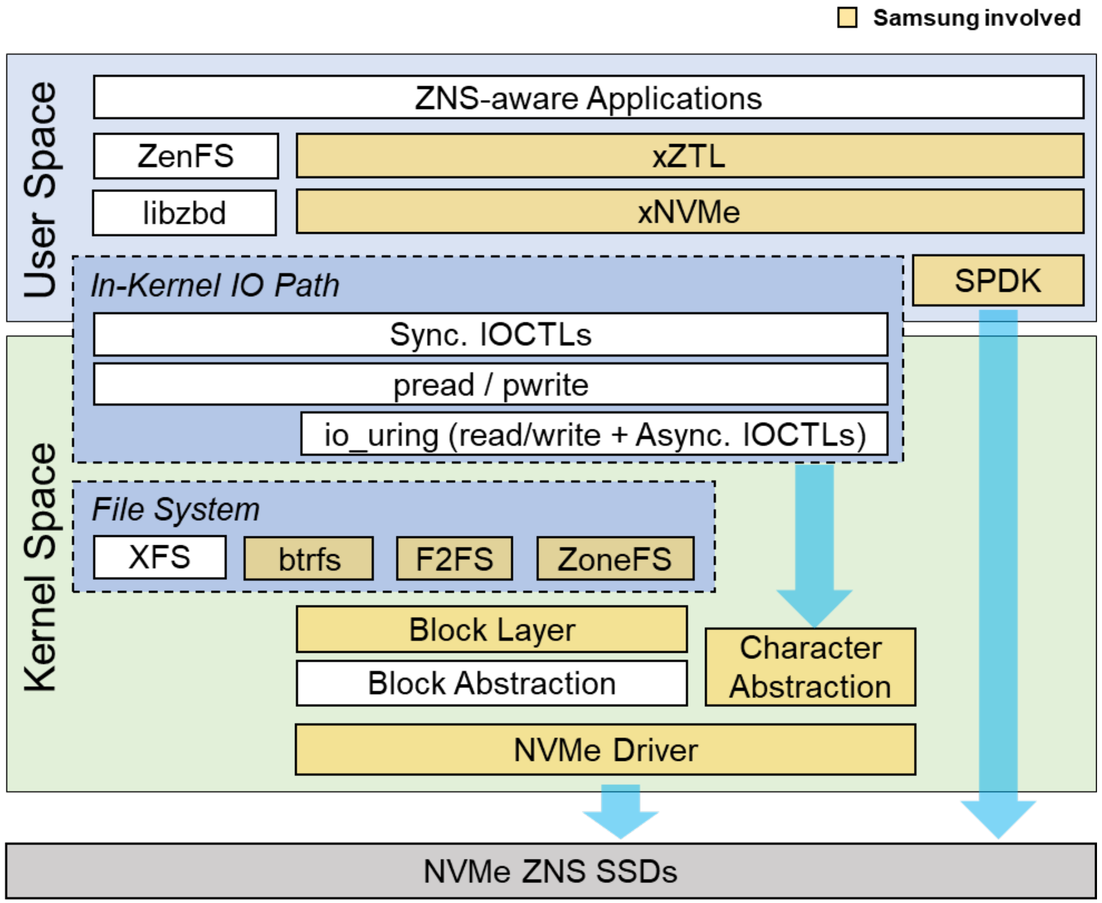
- **Benefits from ZNS SSDs**
  - Flash-friendly workloads are suitable for ZNS SSDs since sequential write is required within a zone
    - Log-structured
  - Garbage collection (GC) is not required
    - Write amplification (WA) by GC is eliminated and lifespan is extended
    - Over-provisioning (OP) space for GC is not necessary
    - Performance is predictable without the impact of GC
  - Zones fall into different groups to deal with different IO streams
    - Providing QoS for multi-tenant workloads
    - Improving the noisy neighbor problem by IO determinism
  - TCO is reduced
    - Less DRAM and no OP for internal SSD

Overview of ZNS SSDs (2/2)

- **ZNS Challenges**
  - Host needs to be in charge of zone resource management
  - Host must understand the data placement based on the characteristics and the constraints of ZNS

Zoned Device Software Ecosystem

Zoned Device Software Ecosystem



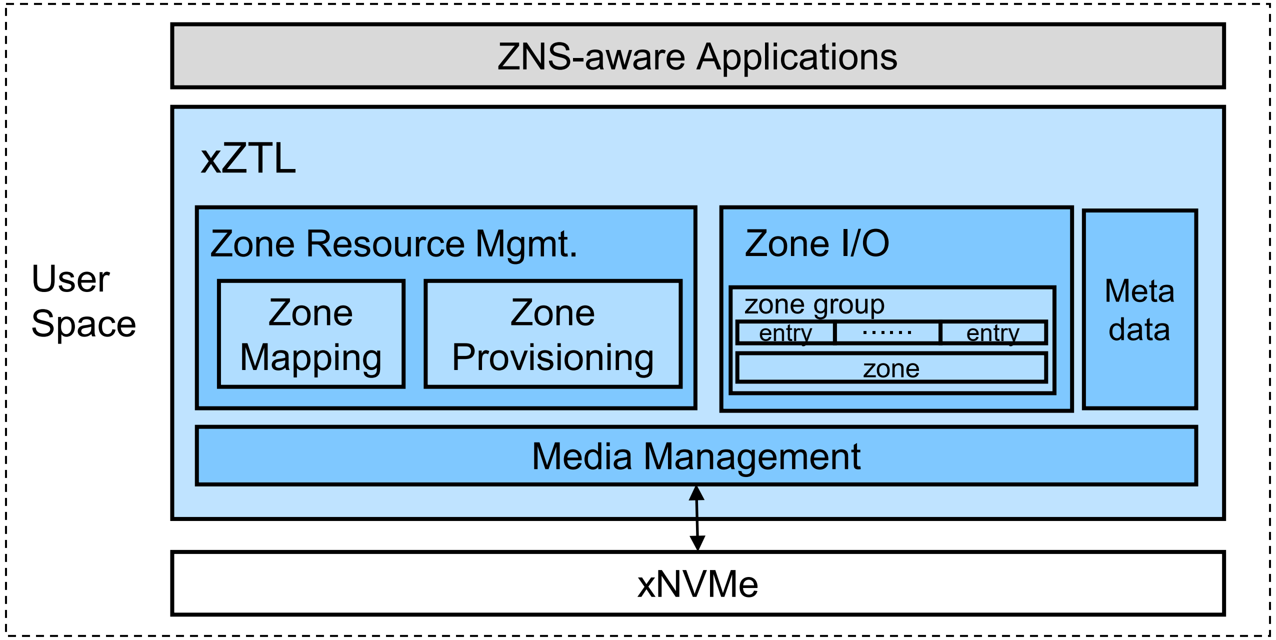
- **Linux Kernel improvement for ZNS SSDs**
  - Character device with full ZNS features (5.14)
  - Enabling in-kernel pass-through I/O path (5.19)

    https://www.snia.org/educational-library/enabling-asynchronous-i-o-passthru-nvme-native-applications-2021
  - Removing the constraint of power of two (PO2) in kernel and allowing non-PO2 zoned devices to access block layer (ongoing)
- **File systems are enabled for ZNS**
  - *btrfs, F2FS, ZoneFS*
- ***xNVMe* provides unified APIs for various IO paths on multiple platforms**
  - IO path: *psync, POSIX aio, libaio, io_uring, SPDK NVMe driver, IOCTLs*, etc.
  - Platforms: *Linux, FreeBSD, Windows*
- ***xZTL* enables the host to easily access ZNS SSDs via xNVMe**
  - Providing interfaces to access ZNS SSDs
  - Managing zone resources & data placement

Introduction to xZTL

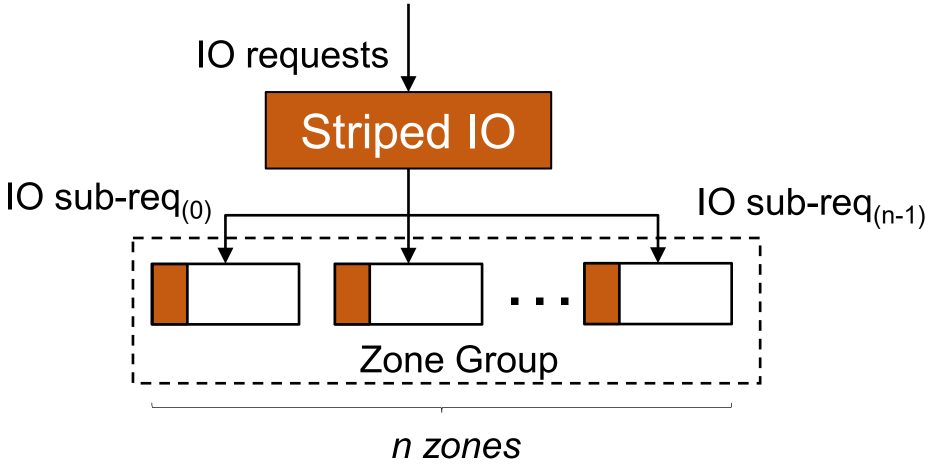
Introduction to xZTL (1/2)

- **Key Features**
  - Supporting both block abstraction and character abstraction
  - Accessing ZNS SSDs with various I/O paths on multiple platforms via xNVMe
  - Supporting I/O models of ZNS SSDs with small/large zones (striped I/O)
  - A user space library which can be applied to different applications (available on RocksDB for now)



Introduction to xZTL (2/2)

- **Functions of Modules**
  - Zone Resource Management
    - A set of zones is defined as a group which is a basic IO operation object. The number of zones in a group determines the max number of stripes for IO requests
    - Each zone is divided into size-fixed entries and an entry is a minimum IO processing unit
    - Provisioning groups and entries for IO requests



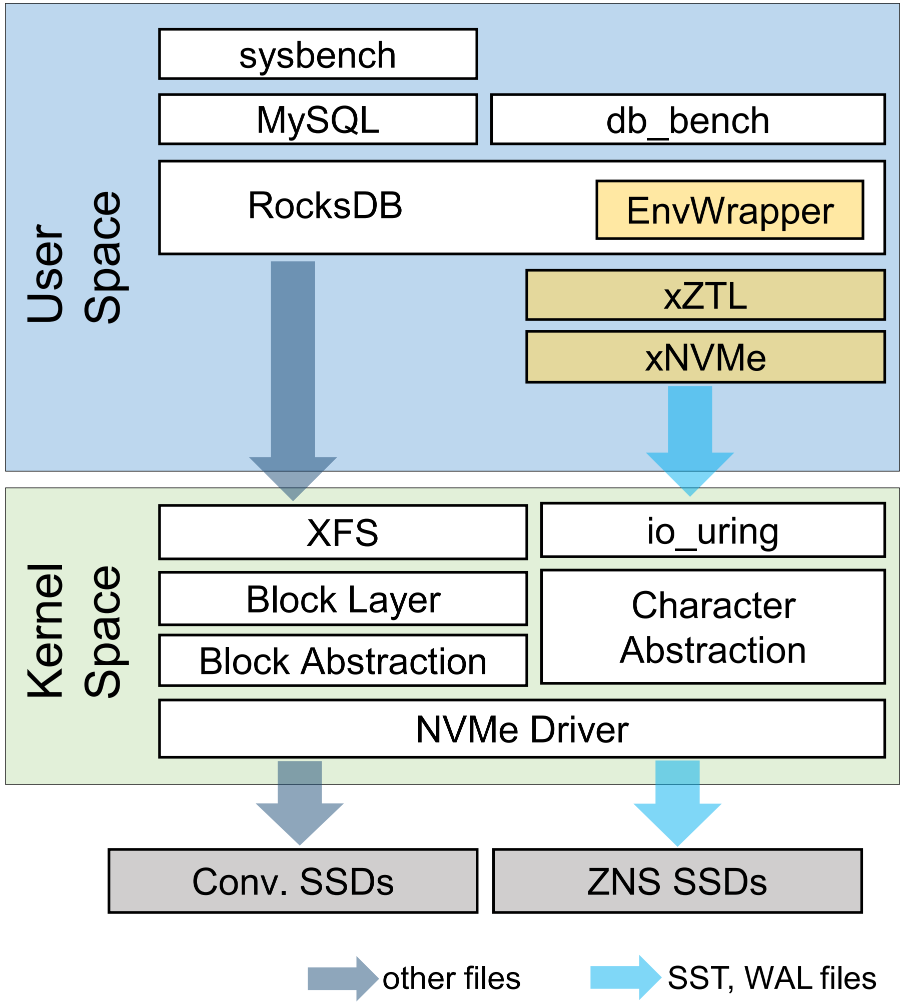
  - Zone I/O
    - Splitting I/O request into sub-requests in zones if I/O size exceeds the size of an entry
  - Metadata
    - Storing mapping table in zones for recovery (two fixed meta zones)
  - Media Management
    - Transferring data and sending commands to ZNS SSDs

Application Benchmark

Conventional SSD vs ZNS SSD

Application Benchmark

- **SSDs**
  - Conventional SSD
    - 3.84 TB (OP 7%)
    - TLC v6 NAND
  - ZNS SSD
    - 4.10 TB (OP 0%)
    - TLC v6 NAND
- **Server Configuration**
  - CPU: Intel(R) Xeon(R) 2.90GHz, 48 cores
  - DRAM: 256G
- **Benchmark Workload**
  - *db_bench* for RocksDB
  - *sysbench* for Percona (MySQL with RocksDB as backend storage)

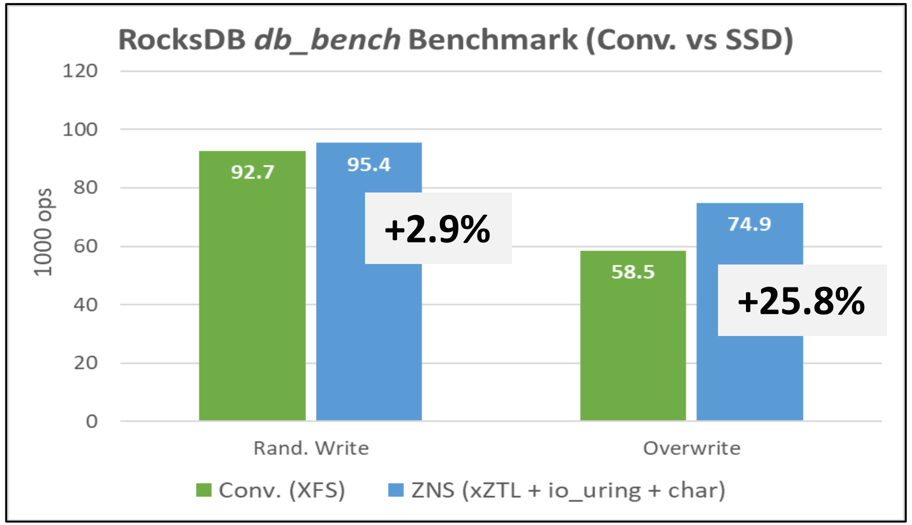


Application Benchmark

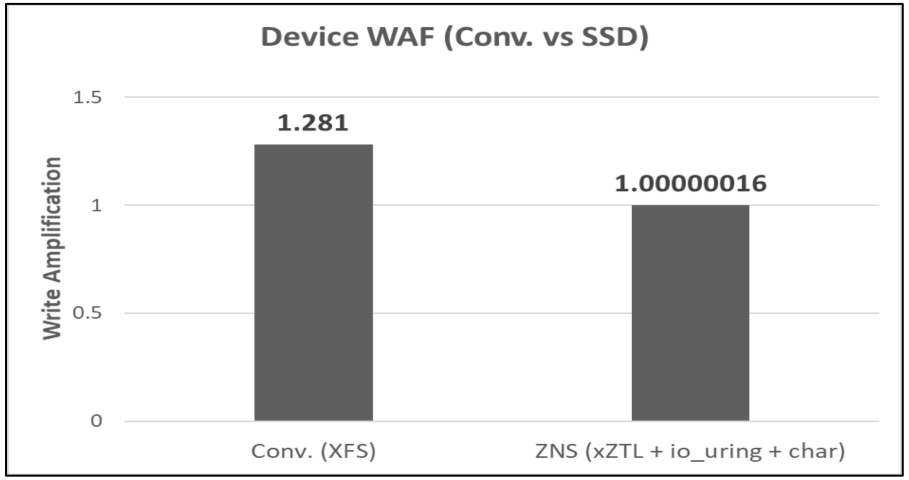
RocksDB Benchmark

RocksDB Benchmark

- ***db_bench* Workload**
  - Data size: 500 GB
    (Leave both Conv. & ZNS 1.2 TB capacity to benchmark)
  - Workload:
    - Random write once
    - Overwrite once
- **I/O Path**
  - Conventional SSD
    - XFS
  - ZNS SSD
    - xZTL + xNVMe + io_uring + character abstraction
- **Benchmark Result**
  - ZNS SSD delivers 2.9% higher random write and 25% higher overwrite
  - There is nearly no WAF for ZNS SSD



higher is better



closer to 1 is better

Application Benchmark

Percona Benchmark

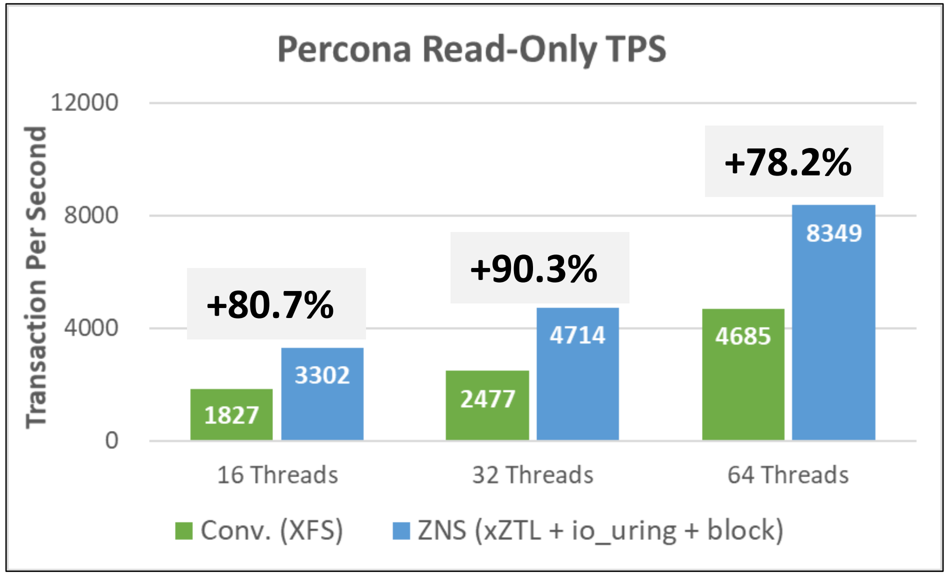
Percona Benchmark (1/4)

- **Sysbench Workload**
  - Data size: 25 tables with 50 million records per table
  - Workload: Read-Only, Write-Only, Read-Write with 16, 32, 64 threads separately

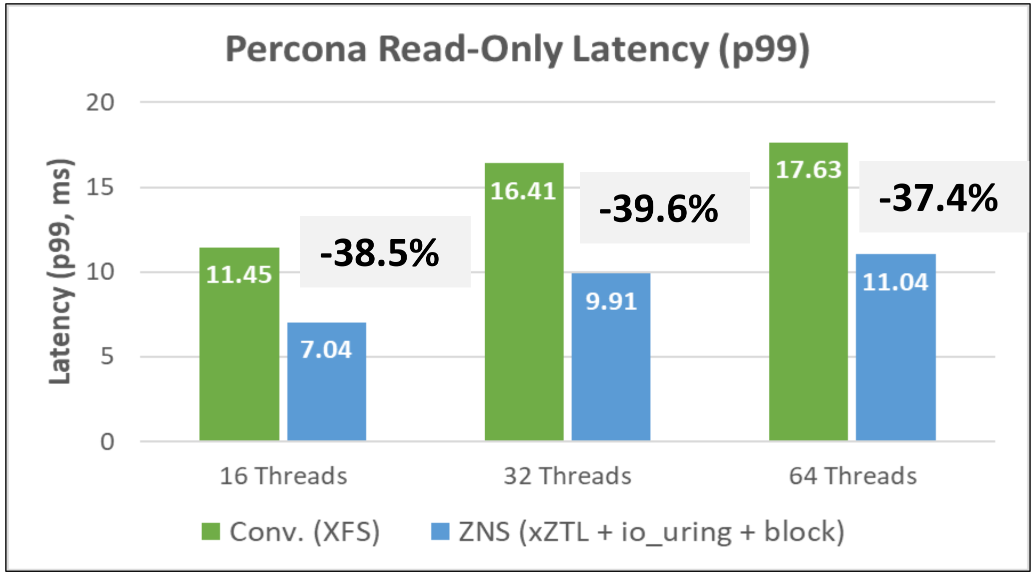
- **I/O Path**
  - Conventional SSD
    - XFS
  - ZNS SSD
    - xZTL + xNVMe + io_uring + block abstraction

Percona Benchmark (2/4)

- **Sysbench Workload for Percona Benchmark Results**
  - Read-Only workload:
    - Transaction per second (TPS) of ZNS SSD is 78%–90% higher than that of conventional SSD
    - The p99 latency is reduced around 38% compared with conventional SSD



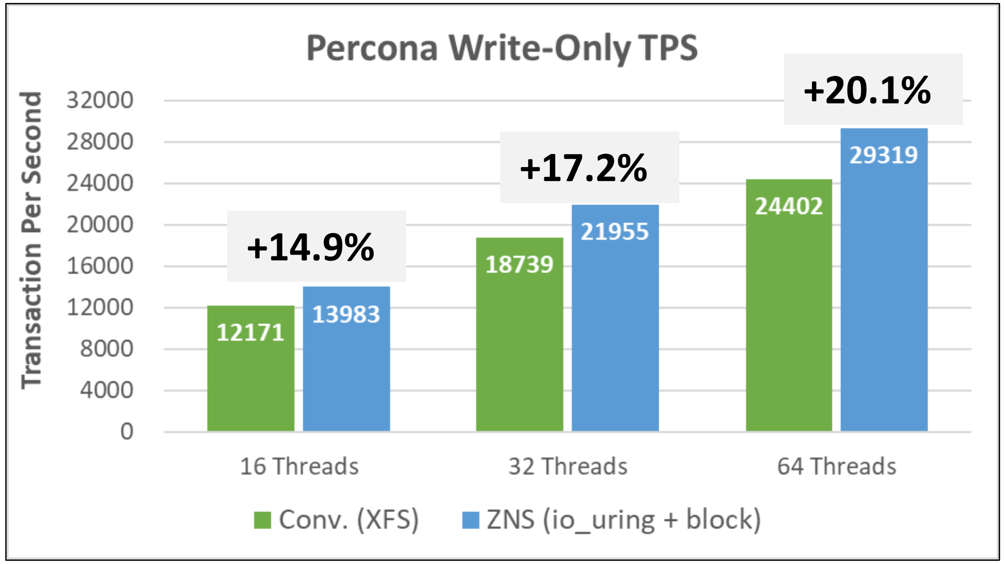
higher is better



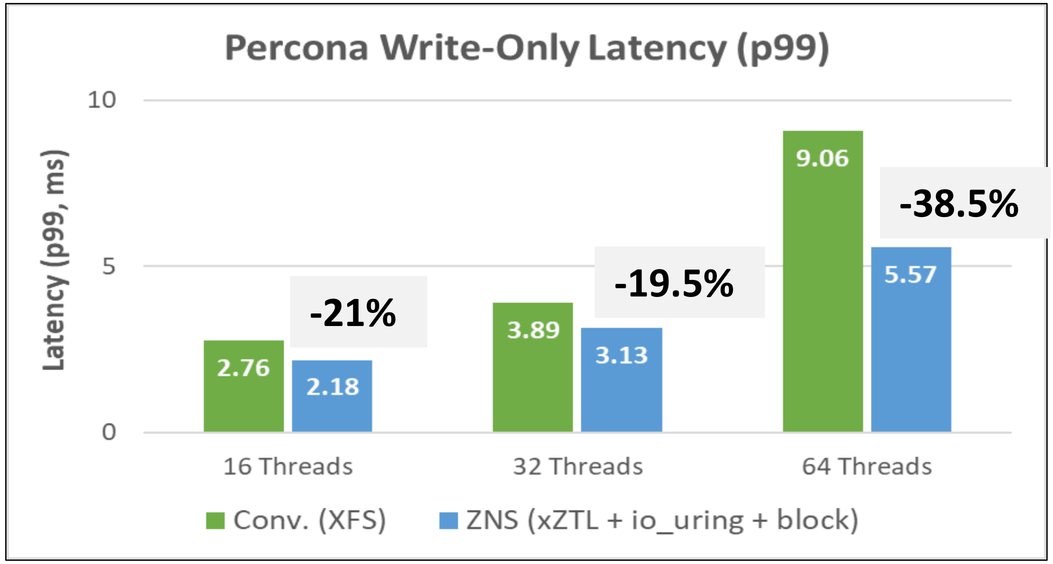
lower is better

Percona Benchmark (3/4)

- Write-Only workload:
  - TPS of ZNS SSD is 14%–20% higher than conventional SSD
  - The p99 latency is reduced 20%–38% compared with conventional SSD



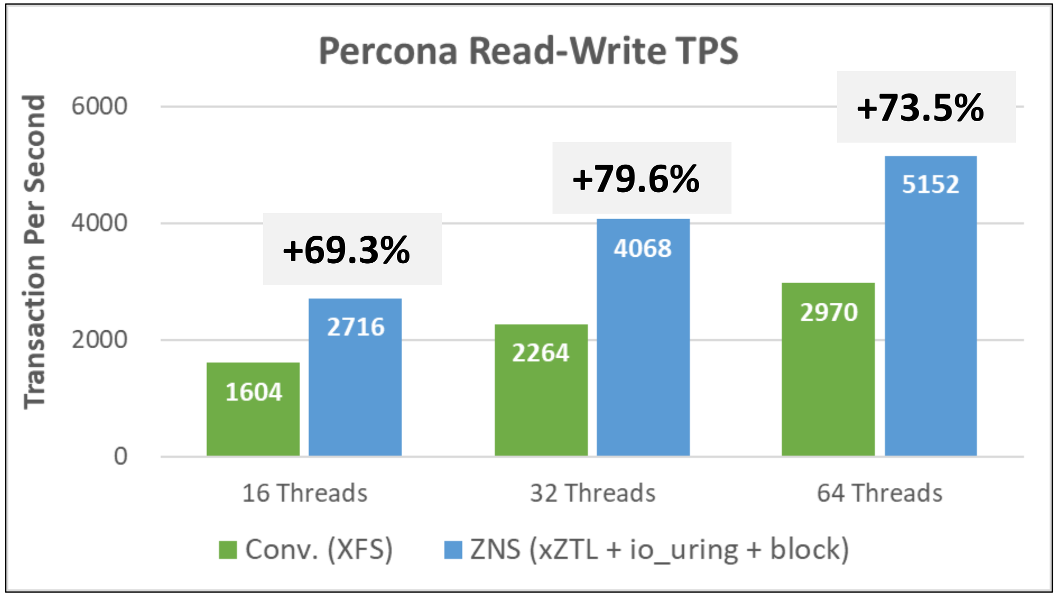
higher is better



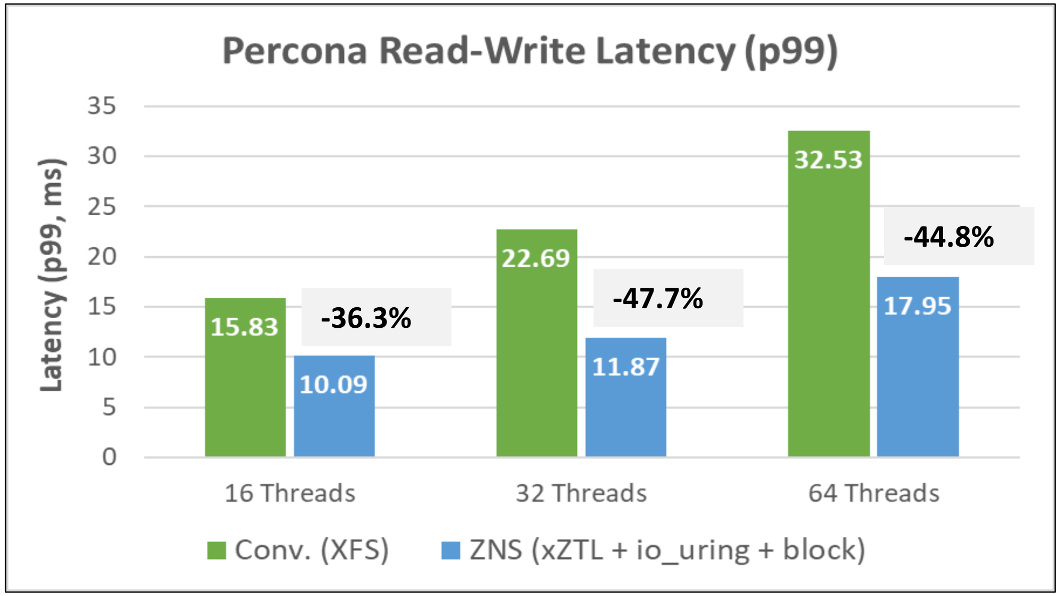
lower is better

Percona Benchmark (4/4)

- Read-Write workload:
  - TPS of ZNS SSD is 69%–79% higher than conventional SSD
  - The p99 latency is reduced 36%–47% compared with conventional SSD



higher is better



lower is better

Building ZNS Ecosystem Together

Building ZNS Ecosystem Together

- **The collaboration is open and your contributions are welcomed**
  - https://github.com/OpenMPDK/xZTL
  - https://github.com/OpenMPDK/xNVMe
  - Discord Channel: Samsung Memory Open-Source

- **Contacts**
  - Hui Qi <hui81.qi@samsung.com>
  - Jing Xia <j.xia@samsung.com>
  - Javier Gonzalez <javier.gonz@samsung.com>
  - Simon Lund <simon.lund@samsung.com>