@Large Research
Massivizing Computer Systems

VU UNIVERSITY AMSTERDAM

TropoDB: Design, Implementation and Evaluation of a KV-Store for Zoned Namespace SSDs

https://github.com/atlarge-research/TropoDB

Krijn Doekemeijer

k.doekemeijer@student.vu.nl
www.krien.github.io

Nick Tehrany, Animesh Trivedi

Background: Increasing storage demands

Demands:

- More storage: IDC expects **175 zettabytes** of data by 2025!
- Faster storage

Challenges:

- How?
- Where?
- Fast, yet cost effective...?
- ...

Where to store?: Key-Value (KV) stores

- NoSQL databases
- Ubiquitous
- High write/read throughput
  - SLA requirements (P99…)

RocksDB

mongoDB®

Amazon DynamoDB

How to store?: KV-stores use flash SSDs

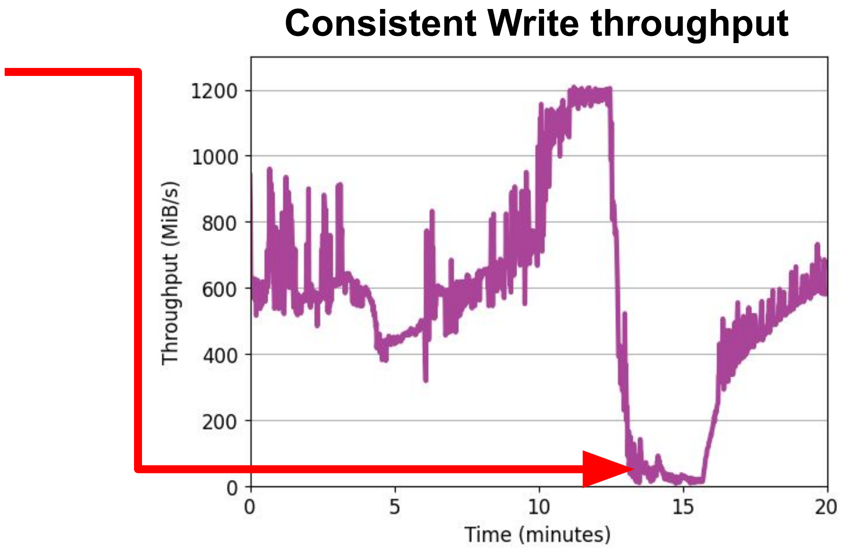
- KV-stores are optimized for **flash SSDs**

- Problem: most flash SSDs use the **block interface**…
    - The block interface does **NOT** reach **KV-SLA** demands

Problem: The block interface breaks SLAs for flash storage!

Block interface:

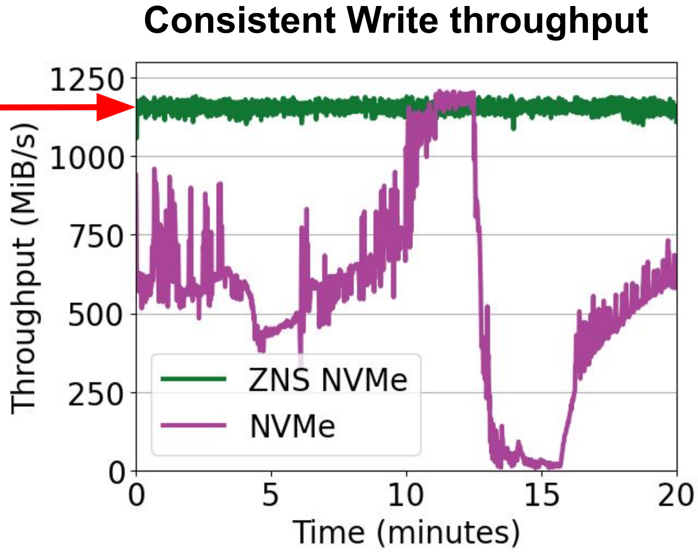
- High throughput fluctuation (**SLAs…**)
- Does not match flash
  - Requires **expensive** translator…
- KV-store SLAs?



Consistent Write throughput

Solution?: Meet Zoned NameSpaces

- Zoned NameSpaces (ZNS):
  - Matches flash closely
  - Stable throughput (**SLAs**)

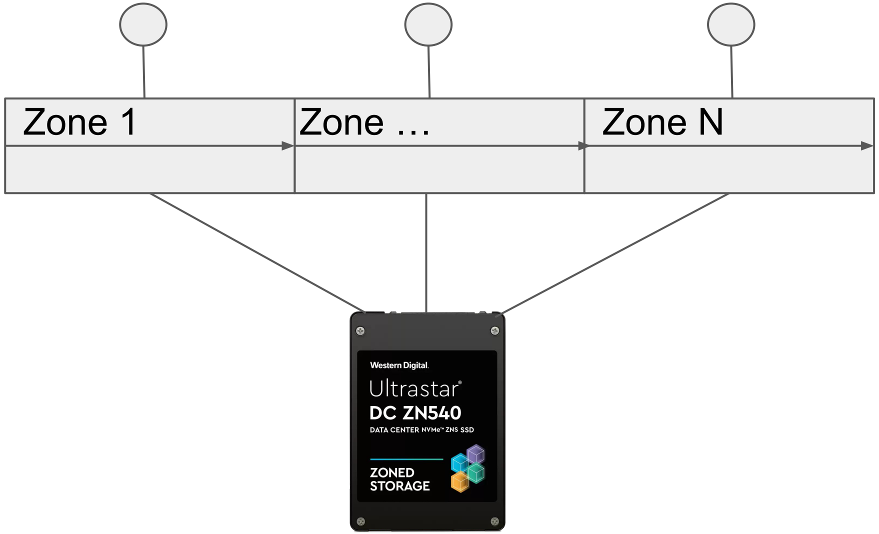


Note: in this experiment we use the **same SSD model**, except for the interface

ZNS: A new abstraction

Device is divided into **zones**

- I/O is issued to zones
- **Sequential writes only**
- **Applications** manage zone state

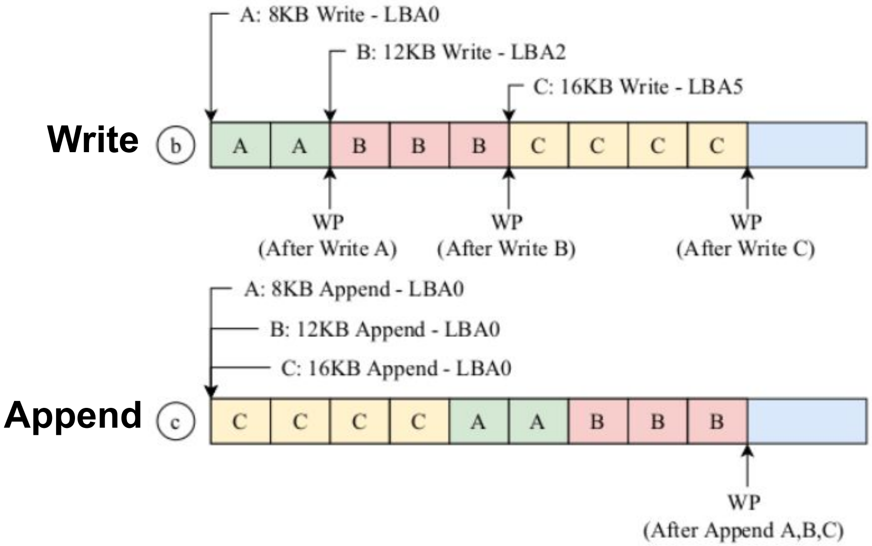


ZNS: Append operation

- Alternative for writes
- Leads to **higher throughput scalability**

Challenge:

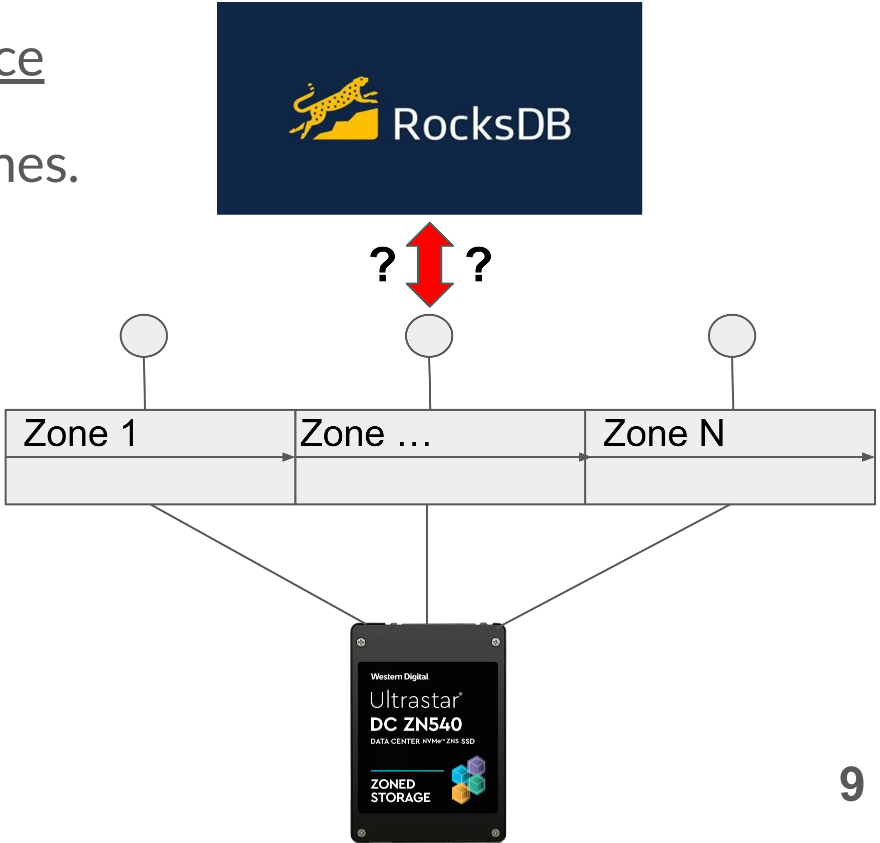
- Issued to zones, returns address
- Requires rewriting write traffic



Problem: Need to rethink KV-stores for ZNS

ZNS is a fundamentally different interface

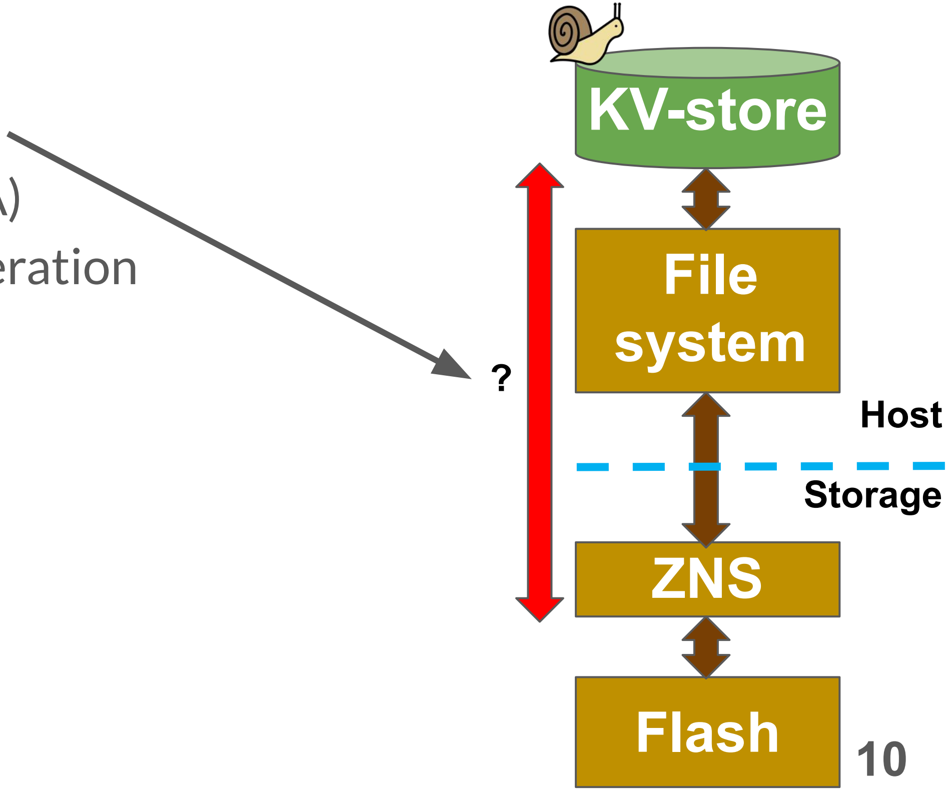
- KV-stores need to interface with zones.
- Random writes are **NOT** allowed
- How to leverage appends?
- …
- **KV-stores need to be rewritten…**



Problem: Need to rethink KV-stores for ZNS

Problems with available ZNS KV-stores:

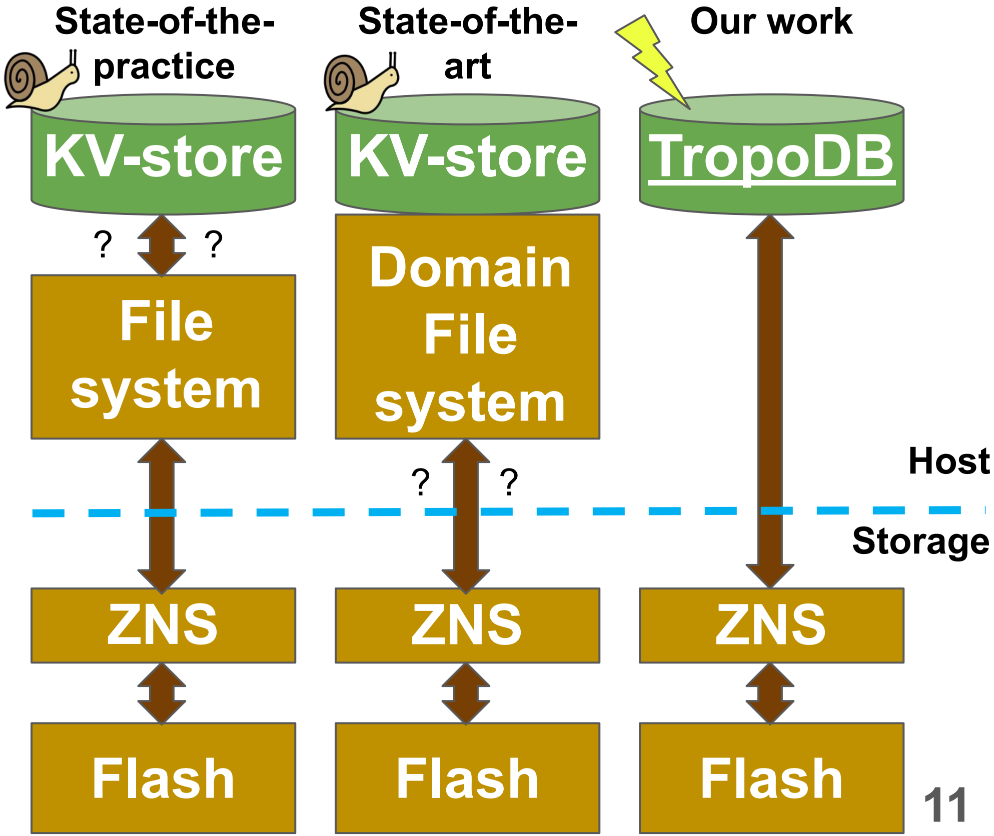
1. Semantic gap (how to communicate)
2. No control over data placement (SLA)
3. They do not use the **ZNS append** operation
   a. **Requires domain knowledge**



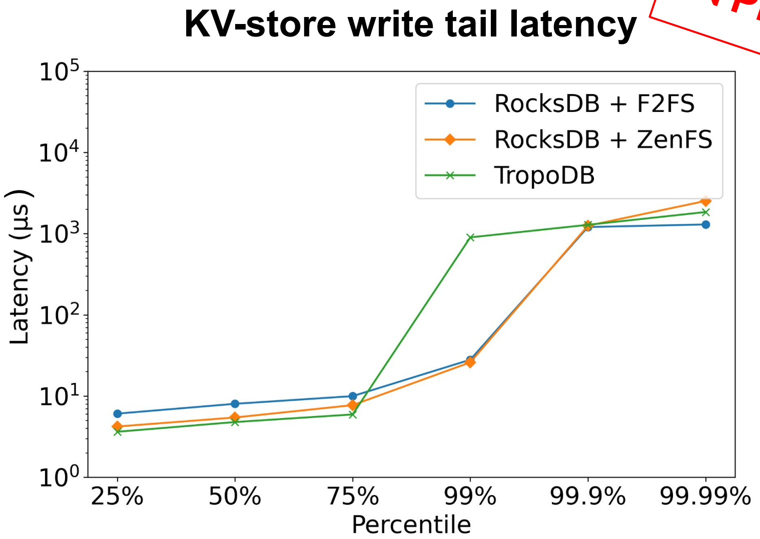
TropoDB: A new design

TropoDB's architecture:

(1) Remove all software layers
    (a) Address semantic gaps
(2) Use **ZNS appends** for all writes
    (a) Appends have better throughput scalability.
(3) Implemented in **RocksDB**
    (a) Do not reinvent the wheel



TropoDB: Work in Progress Results



IN PROGRESS

TropoDB: Take-home messages

- **ZNS** is a new interface for **flash storage** leading to better **QoS**
- TropoDB is an ongoing work for a **KV-store rewrite for ZNS**
- Goals:
  - Better QoS for a single device
  - Evaluate ZNS-specific optimisations
- **We highly value collaborations and new ideas!**
  - Like expanding it to multiple devices (ZNS over fabrics)

https://github.com/atlarge-research/TropoDB

TropoDB: Case-study ZNS appends

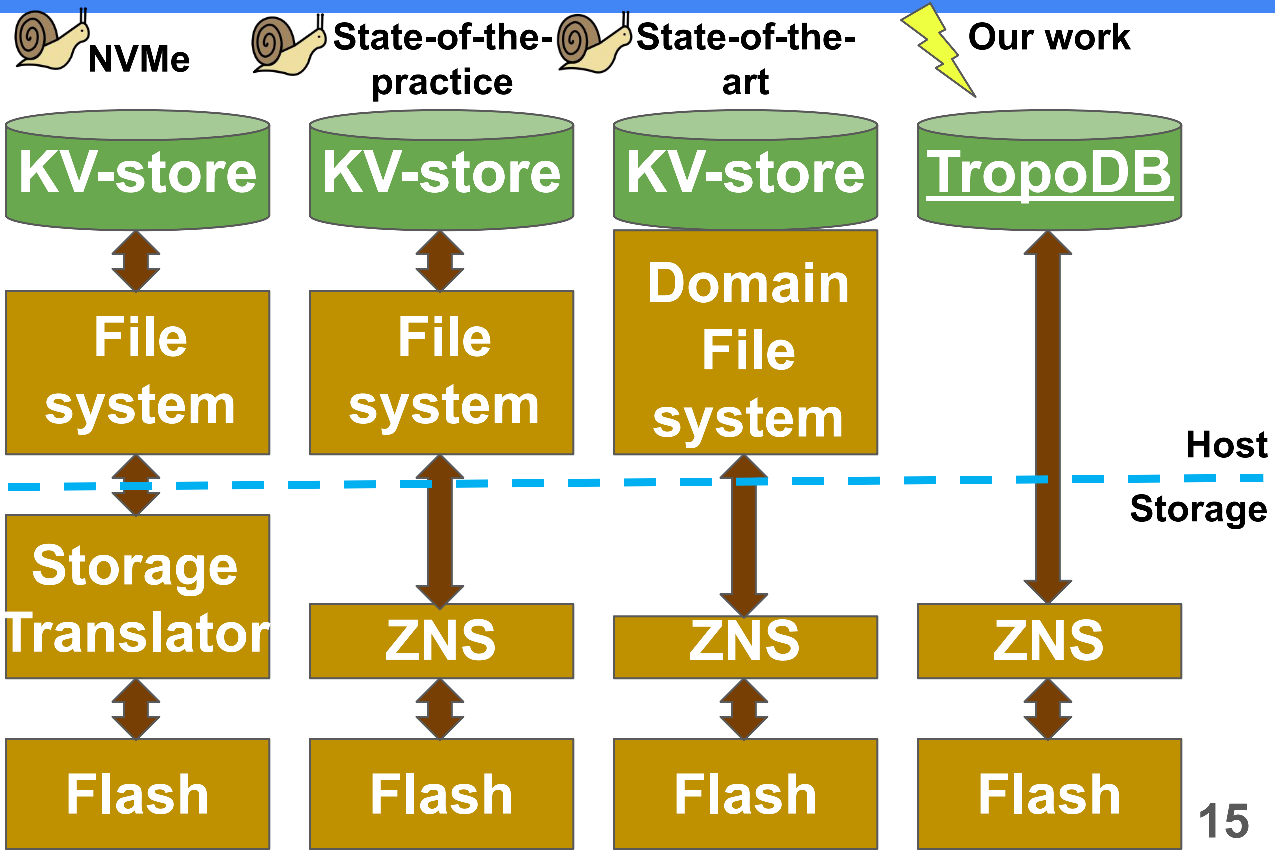
Use case: Write-ahead logs

- Write-heavy
- Many outstanding requests
- Little reads

TropoDB: Removing all layers

We need to remove all layers:

- Semantic gap



Problem: Research question

Research Question:
“How to leverage the unique design properties of NVMe ZNS devices to optimise a key-value store?”

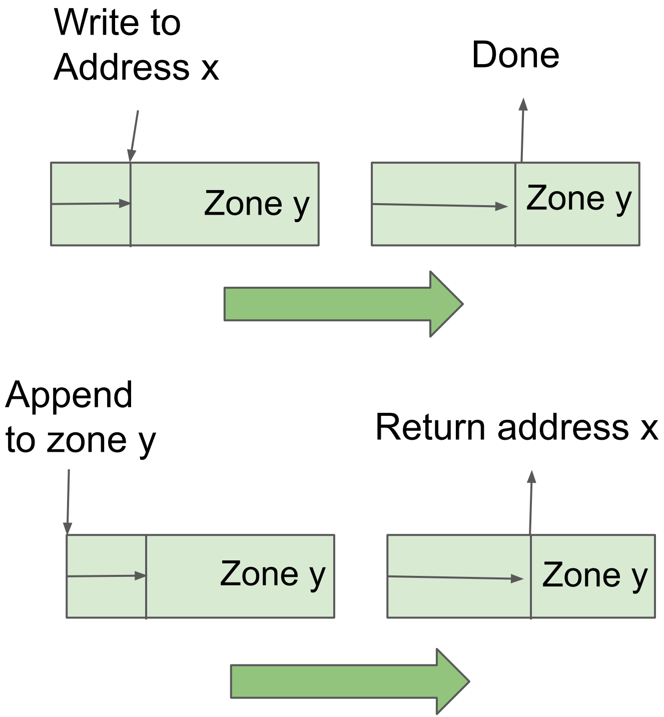
TropoDB: Case-study ZNS appends

Problem with writes

- **ZNS does not allow multiple writes to 1 zone!**

Solution? ZNS append operation

- Multiple appends allowed to 1 zone
- Appends are issued **to zones**
- Appends can be reordered
- Need to rewrite software


Write to Address x
Done
Zone y
Zone y
Append to zone y
Return address x
Zone y
Zone y