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

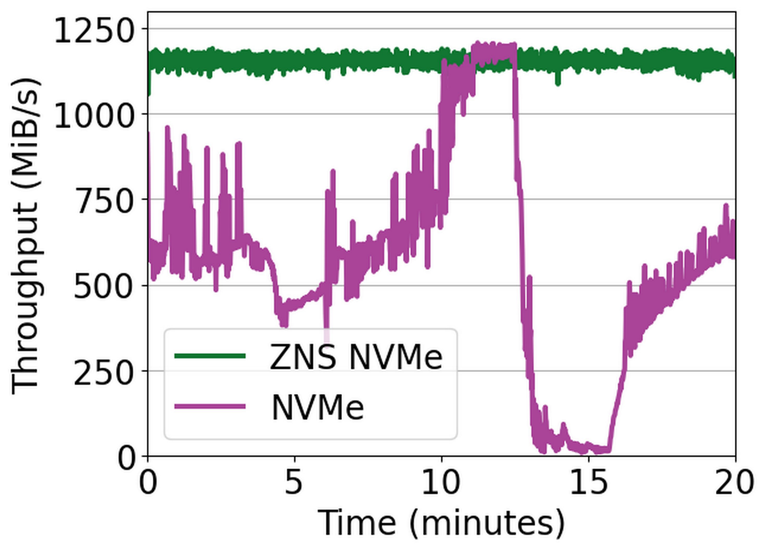
Krijn Doekemeijer (k.doekemeijer@student.vu.nl)[1], Animesh Trivedi[1]
[1]VU Amsterdam

1 Context

- The amount of data will reach **> 180 zettabytes** in 2025
- Data is frequently stored in **Key-value stores (KV)**

RocksDB mongoDB. Amazon DynamoDB

- KV-stores require physical storage to have:
  - **Low tail latency**
  - **High throughput**
- Data centers are transitioning to **NVMe flash storage**
- However, a **new** storage interface, **Zoned Namespaces**
  - has **better tail latency** than NVMe
  - has **better durability** than NVMe
  - Meets storage demands more closely



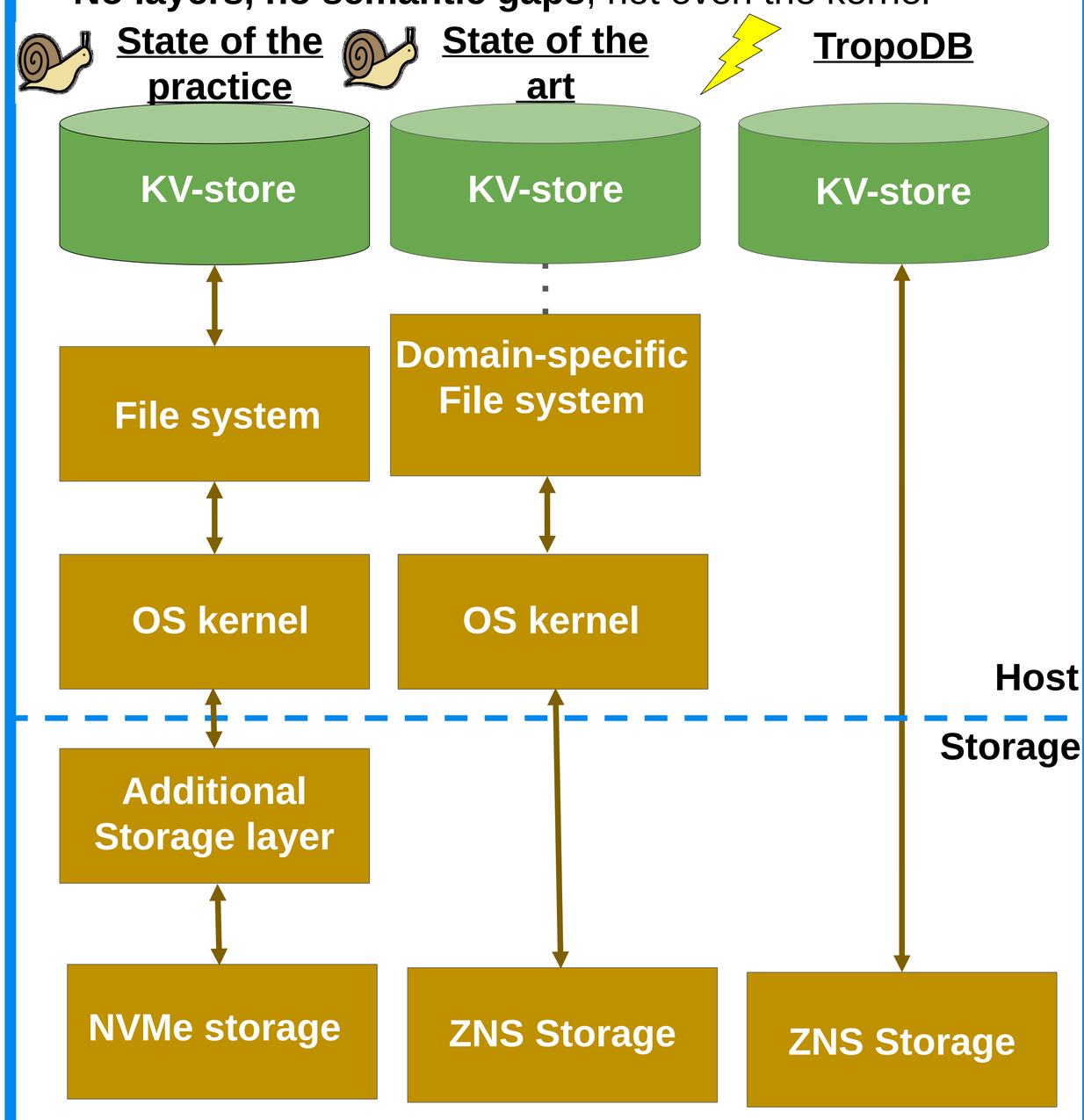
2 Problem

We want to use ZNS for KV-stores, but…

- ZNS is a completely **new abstraction**
- ZNS requires **rewriting software**
- **Current KV-stores** can not effectively make use of ZNS:
  1. They require generic **file systems**
  2. They have **no control** over **data placement**
  3. They **do not** use **ZNS-specific operations** (append)

3 TropoDB, Layerless solution

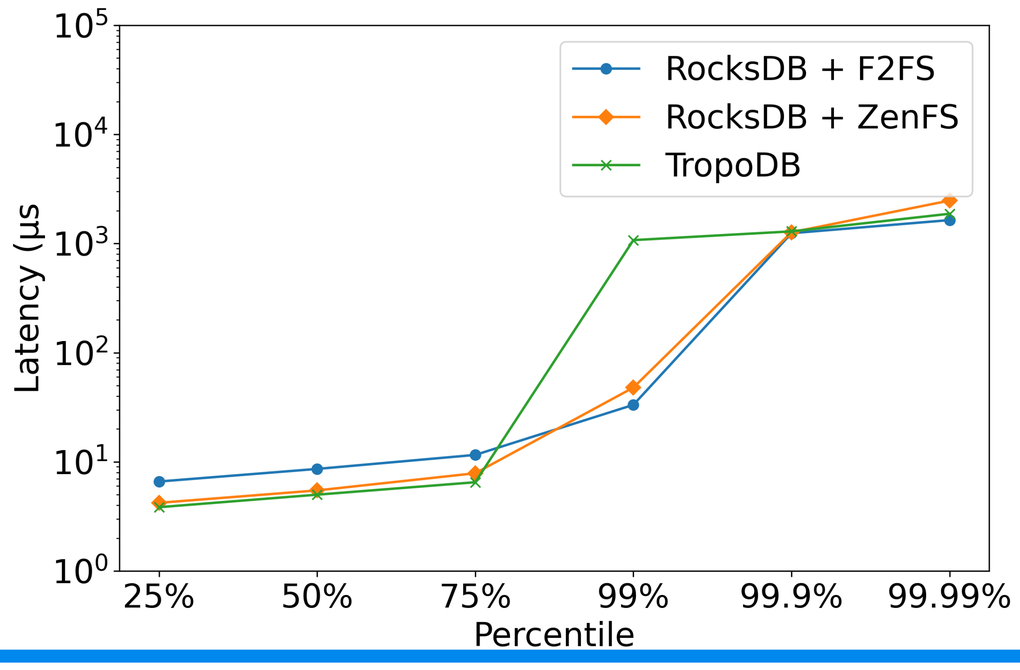
- A KV-store that issues **raw ZNS commands**
- **No layers, no semantic gaps**, not even the kernel



4 Evaluating TropoDB

- Evaluated on **real ZNS hardware**
- Comparing **TropoDB** to **state of the practice** and **state of the art**
- Write-heavy workload
- TropoDB comes *close* to the competition in tail latency!

1 TB of random overwrites of KV-pairs



5 What next?

TropoDB has shown moving to ZNS is viable. Thus potential directions are:

- Characterising ZNS performance characteristics
- Improving TropoDB with new ZNS insights
- Making TropoDB stable/competitive
- TropoDB as a research platform
- Creating/designing ZNS optimised file systems
- Creating ZNS schedulers for multi-tenancy

VU VRIJE UNIVERSITEIT AMSTERDAM

For more see:
https://github.com/atlarge-research/TropoDB

@Large Research
Massivizing Computer Systems