Intelligent Data Migration Policies in a Wo-CoW Tiered Storage Stack

Johannes Wünsche, Sajad Karim, Michael Kuhn, Gunter Saake, David Broneske

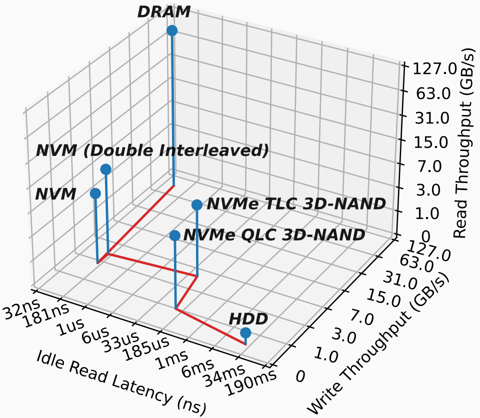
May 8, 2023

Otto-von-Guericke University Magdeburg

Faculty of Computer Science

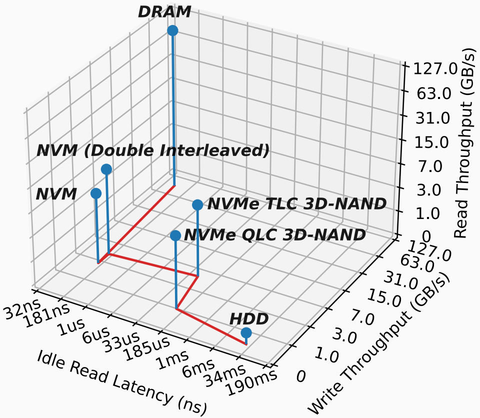
Motivation

- Storage Landscape with unique characteristics
- Storage Class Memory → characteristics ill-fitted to usual assumption of slow-to-fast monotonicity
- Different devices behave better with specific workloads



Motivation

- Different Layers handle multiple data stream with different degree of performance degradation e.g. multiple writes on Optane (Fedorova et al. 2022)
- Creating an holistic model might deliver us advantages for utilization

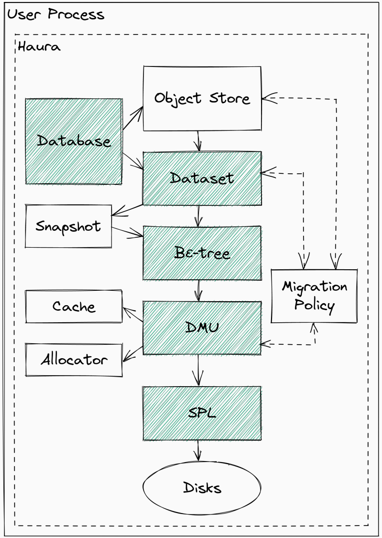


Research Questions

1. How are data migration decisions into a write-optimized storage stack consisting of B^{ε}-trees effected?
2. How can we combine data-structure-aware and data-structure-agnostic migration information?
3. Which effects does CoW have on data migration behavior?

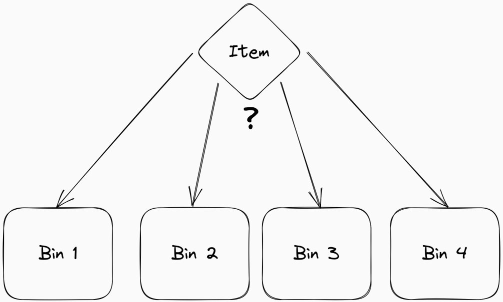
Haura

- Monolithic user-space tiered storage stack
- Write-optimized multi-tier B^{ε}-trees
- Copy-on-Write → crash consistent + cheap snapshots
- Interfaces: key-value + object store
- Experimental base extendable with data structures & policies
- Research stack focused on heterogeneous/tiered storage



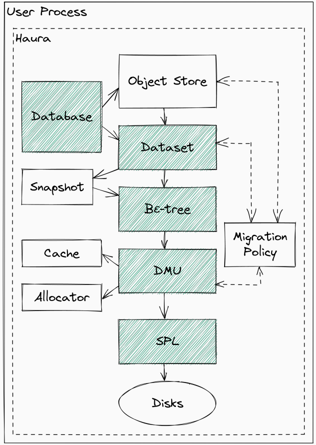
Data Placement

- Core: Generalized Assignment Problem (GAP) (Ross and Soland 1975)
- Special-cases: MKP (Kellerer, Pferschy, and Pisinger 2004), SSAP (Ghandeharizadeh, Irani, and Lam 2018)
- Tiered Storage Cost dependent on:
    - Item Frequency
    - Item Operation
    - Bin Type
    - *Time*



Migration Policies

Migration Policies



- Migration Policies detached from layer-based storage stack design
  - Knowledge Aggregation
  - Combine internal data representation impacts with multi-level hints
- Minimize Interference of Policy Computation with Query/Insert Flow
  - Message Passing Design between components
- Policy-agnostic

Migration Messages

Two message kinds:

Migration Messages

Two message kinds:

1. Node Message (*Internal Representation*) > `Write`, `Fetch`, `Remove`
    - Allow for fine-granular control over all tree layers
    - Small selection (4MiB) of entries
    - Identification via Pivots
    - Storage Preferences inherited when splitting or merging nodes

Migration Messages

Two message kinds:

1. Node Message (*Internal Representation*) > `Write`, `Fetch`, `Remove`
    - Allow for fine-granular control over all tree layers
    - Small selection (4MiB) of entries
    - Identification via Pivots
    - Storage Preferences inherited when splitting or merging nodes
2. Object Message (*External Semantics*) > `Open`, `Close`, `Write`, `Read`, `Migrate`
    - Granularity to specific chunks (keys) of an object
    - Effects on leaf nodes
    - Partial or Whole Object Migration
    - Multi-level hints (Ge et al. 2022)

- Combination of messages may be used and combined
- Two example policies implemented:
  1. LFU-based approach for nodes and objects (Mátáni, Shah, and Mitra 2021)
  2. RL-based approach by Vengerov (2008) (objects)
- Messages allow for a range of policies to be implemented
- Resource-intensive algorithms can be tolerated to a degree

Results

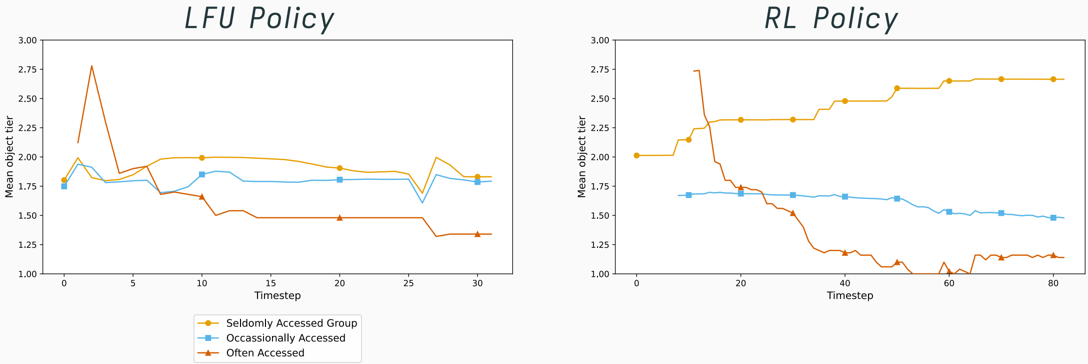
Results

- Multiple workflows with well-fitting distribution known beforehand
- 3 tier setup utilizing a debug DRAM layer, NVMe-SSD, HDD
- Generated Workflow
- Scenarios:
  1. Synthetic Distribution Classification
  2. Snapshot Distribution Classification
  3. Write-only impacts
  4. Application Checkpoints

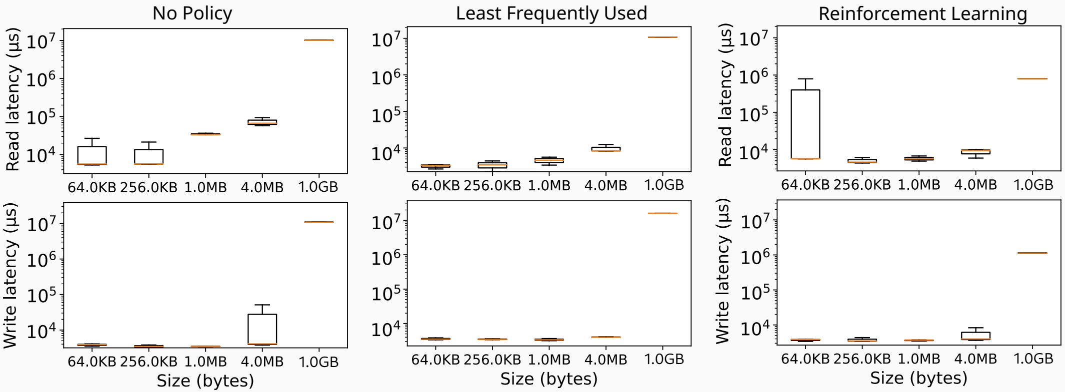
Synthetic Distribution

- Based on surveys of HPC systems (Meister et al. (2012), Welch and Noer (2013))
- Files from traditional file systems mapped to individual objects
- Majority of files small, bulk of data consisting of increasingly sized objects
- Noise-sensitive → Emitted operations read-only; all migrations will incur additional costs
- Random assignment of objects to three groups (`Seldom`, `Occassional`, `Often` accesses)
- Selection pre-defined to represent whole object range
- Latency measurement performed in the end with purged cache

Synthetic Distribution



Synthetic Distribution - Often Latency

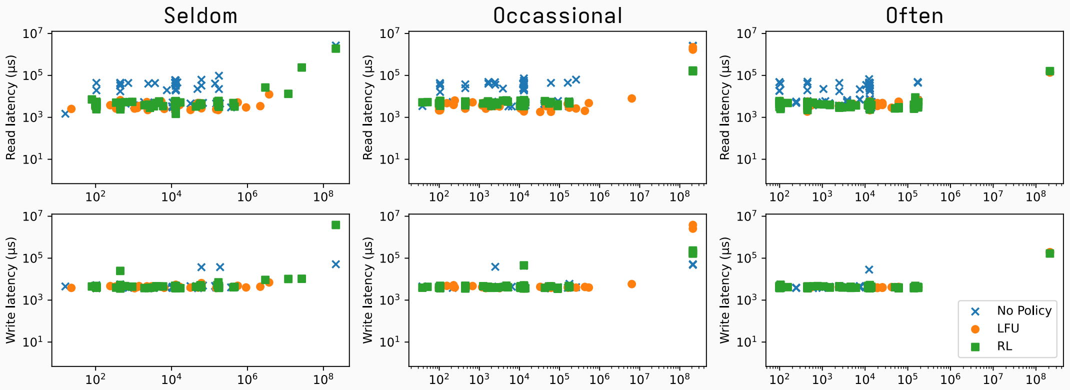


Snapshot Distribution

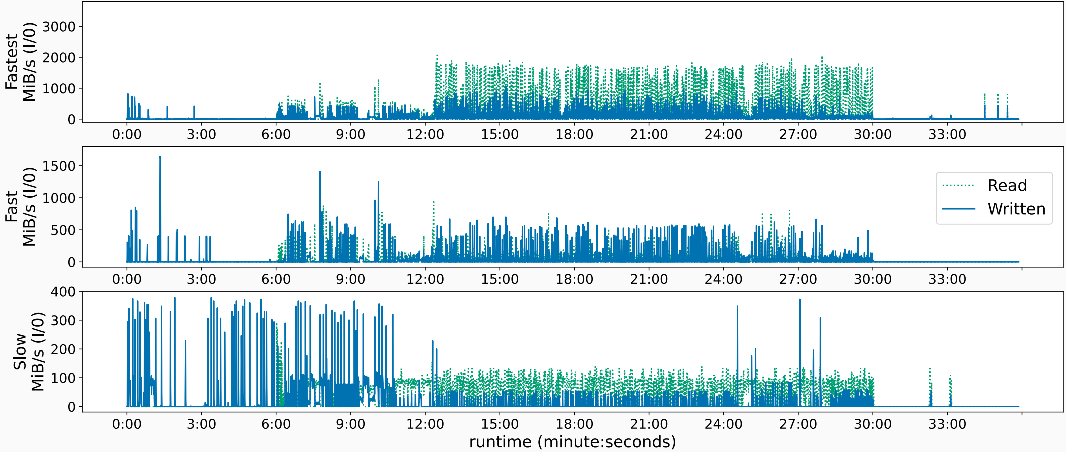
- Two applications[1] initiated and runs performed with checkpoints to produce data
- Build artifacts, configurations and logs
- Home directory archived and loaded into Haura
- Three Groups created as in Synthetic Distribution

[1] NWChem and XCompact3D

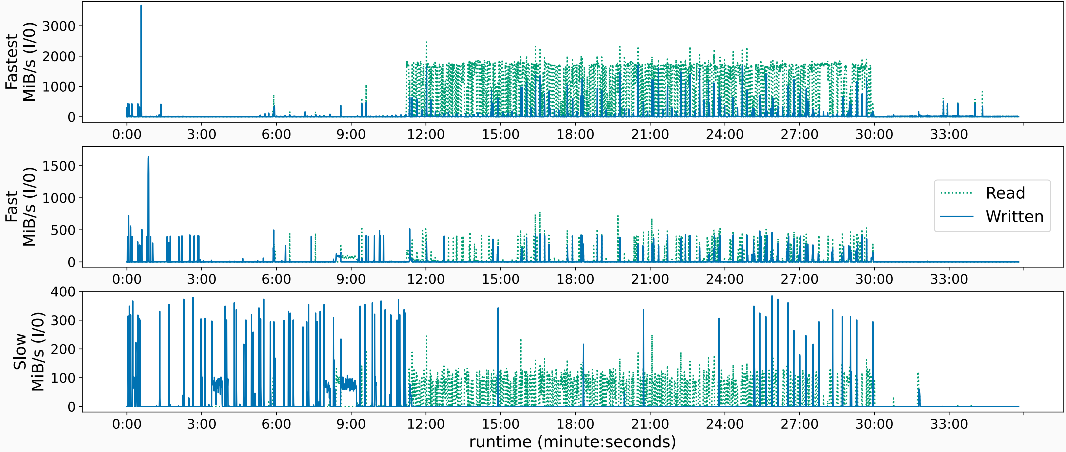
Snapshot Distribution



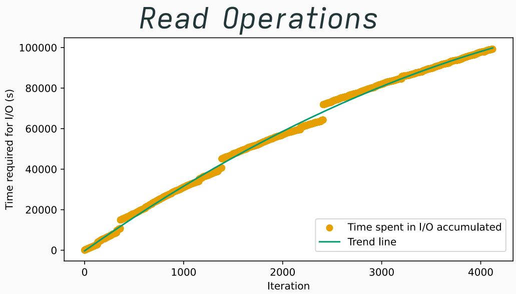
Snapshot Distribution - LFU



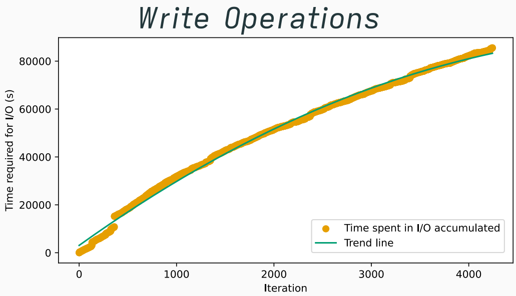
Snapshot Distribution - RL



Write-only



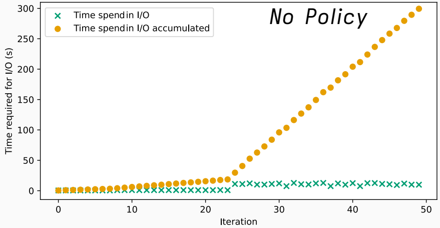
Read Operations
Time required for I/O (s)
100000
80000
60000
40000
20000
0
0
1000
2000
3000
4000
Iteration
Time spent in I/O accumulated
Trend line

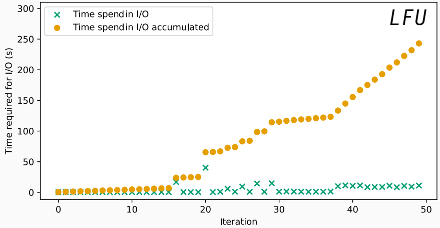


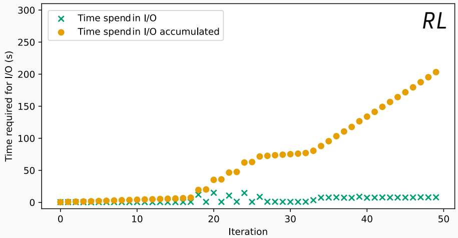
Write Operations
Time required for I/O (s)
80000
60000
40000
20000
0
0
1000
2000
3000
4000
Iteration
Time spent in I/O accumulated
Trend line

Application Checkpoints

x-axis representing size in bytes, x-axis and y-axis are in log-scale







Summary

- Introduction of *Haura* as a write-optimized storage stack with advantages to state-of-the-art LSM based approaches
- Message-based migration interface combining external and internal information with tolerance for slack in migration policy performance
- Simple threshold policy creates considerably more traffic than reinforcement learning policy
- Write-optimization alleviates latency concerns for small write operations almost entirely while preserving indexing performance[2]

→ Follow-up project SMASH: https://smash-spp2377.github.io/

[2] Flushing of data still needs to occur, but much later and accumulated to the actual residence device.

Appendix

Misc

- Lazy Promotion
  - Actual use initiates the migration process
  - Overlap in fetch costs
  - Reduction of additional writes and fragmentation
- Object I/O categorization
  - Skips user-knowledge of specific configuration
  - Matching of objects to fitting storage tier
  - (Seq, Rnd, Mix) + (Write, Read, Mix)
- JULEA-integration (Kuhn 2017)
  - Expose functionality to other processes

Bibliography

Fedorova, Alexandra, Keith A. Smith, Keith Bostic, Alexander Gorrod, Sue LoVerso, and Michael J. Cahill. 2022. "Writes Hurt: Lessons in Cache Design for Optane NVRAM." *CoRR* abs/2205.14122. https://doi.org/10.48550/arXiv.2205.14122.

Ge, Xiongzi, Zhichao Cao, David H. C. Du, Pradeep Ganesan, and Dennis Hahn. 2022. "HintStor: A Framework to Study I/O Hints in Heterogeneous Storage." *ACM Trans. Storage* 18 (2): 18:1–24. https://doi.org/10.1145/3489143.

Ghandeharizadeh, Shahram, Sandy Irani, and Jenny Lam. 2018. "The Subset Assignment Problem for Data Placement in Caches." *Algorithmica* 80 (7): 2201–20. https://doi.org/10.1007/s00453-017-0403-4.

Kellerer, Hans, Ulrich Pferschy, and David Pisinger. 2004. "The Multiple-Choice Knapsack Problem." In *Knapsack Problems*, 317–47. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-24777-7_11.

Kuhn, Michael. 2017. "JULEA: A Flexible Storage Framework for HPC." In *High Performance Computing - ISC High Performance 2017 International Workshops, DRBSD, ExaComm, HCPM, HPC-IODC, IWOPH, IXPUG, p^3MA, VHPC, Visualization at Scale, WOPSSS, Frankfurt, Germany, June 18-22, 2017, Revised Selected Papers*, edited by Julian M. Kunkel, Rio Yokota, Michela Taufer, and John Shalf, 10524:712–23. Lecture Notes in Computer Science. Springer. https://doi.org/10.1007/978-3-319-67630-2/_51.

Mátáni, Dhruv, Ketan Shah, and Anirban Mitra. 2021. "An O(1) Algorithm for Implementing the LFU Cache Eviction Scheme." *CoRR* abs/2110.11602. https://arxiv.org/abs/2110.11602.

Meister, Dirk, Jürgen Kaiser, André Brinkmann, Toni Cortes, Michael Kuhn, and Julian M. Kunkel. 2012. "A Study on Data Deduplication in HPC Storage Systems." In *SC Conference on High Performance Computing Networking, Storage and Analysis, SC '12, Salt Lake City, UT, USA - November 11 - 15, 2012*, edited by Jeffrey K. Hollingsworth, 7. IEEE/ACM. https://doi.org/10.1109/SC.2012.14.

Ross, G. Terry, and Richard M. Soland. 1975. "A Branch and Bound Algorithm for the Generalized Assignment Problem." *Math. Program.* 8 (1): 91–103. https://doi.org/10.1007/BF01580430.

Vengerov, David. 2008. "A Reinforcement Learning Framework for Online Data Migration in Hierarchical Storage Systems." *J. Supercomput.* 43 (1): 1–19. https://doi.org/10.1007/s11227-007-0135-3.

Welch, Brent, and Geoffrey Noer. 2013. "Optimizing a Hybrid SSD/HDD HPC Storage System Based on File Size Distributions." In *IEEE 29th Symposium on Mass Storage Systems and Technologies, MSST 2013, May 6-10, 2013, Long Beach, CA, USA*, 1–12. IEEE Computer Society. https://doi.org/10.1109/MSST.2013.6558449.