DICTIONARY BASED CACHE LINE COMPRESSION

AUTHORS –

DANIEL COHEN, SAREL COHEN, DALIT NAOR, DANIEL WADDINGTON, MOSHIK HERSHCOVITCH

OUTLINE

1. Motivation for research
2. Current State of the Art Memory Snapshotting
3. Our contributions
4. Methodology and Experiments
   A. Dictionary Based Compression
   B. Experiments Results
5. Conclusion and Future Work

VM SNAPSHOTS

1. **CPU State**
2. **Memory State**
3. **Disk State**
4. **Other Devices State**

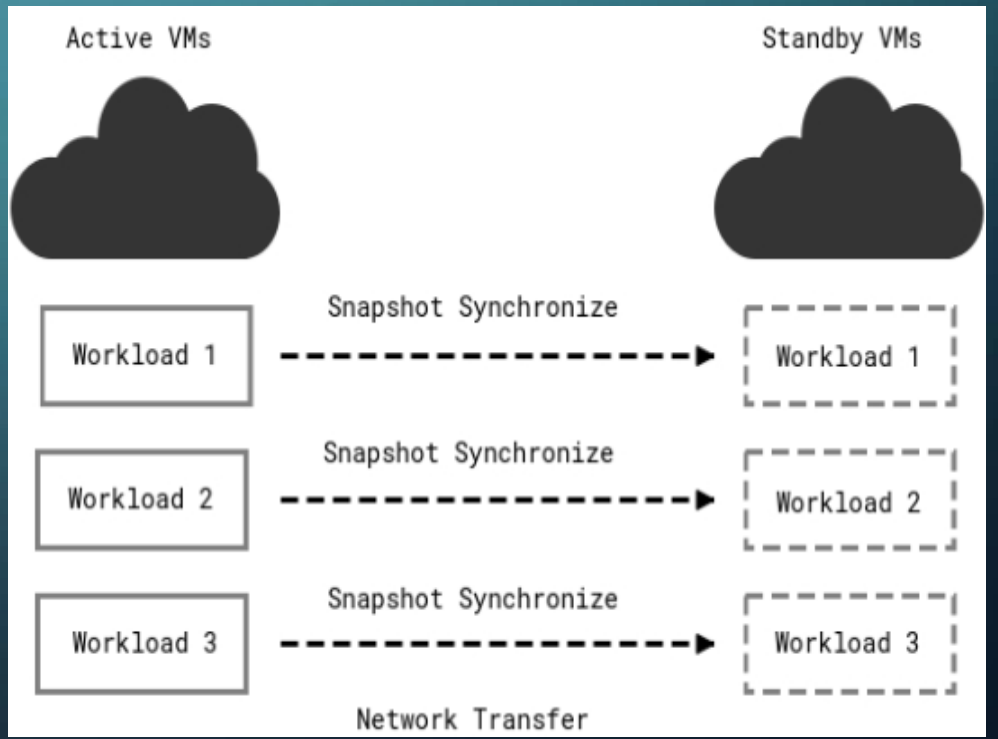
- In this work we will be focusing exclusively on the Memory State of a VM

HIGH AVAILABILITY WITH VMS

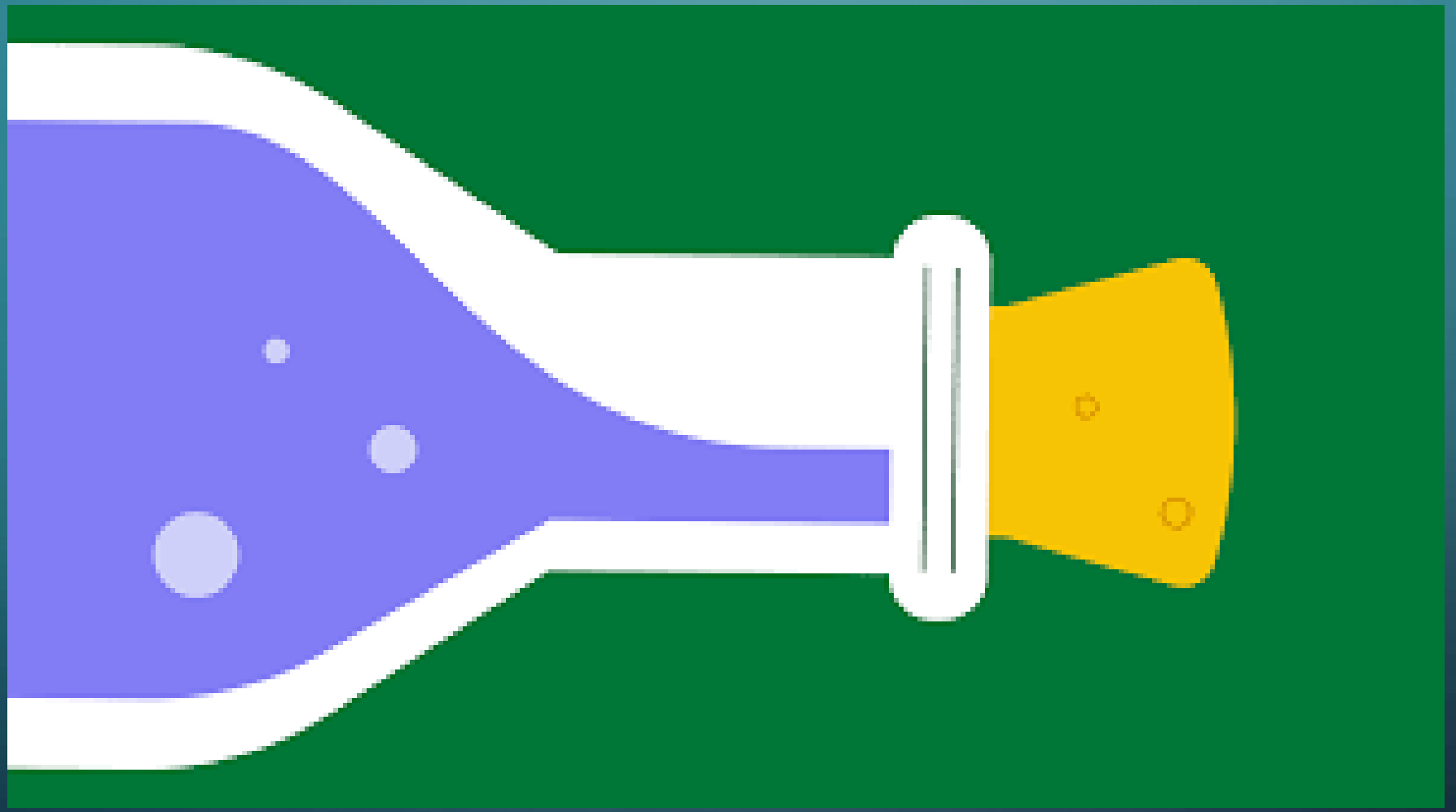
High Availability Applications

- Ensure continuous service by minimizing downtime.
- Utilize continuous snapshotting to maintain up-to-date replicas.

- **Active-Standby Environment**

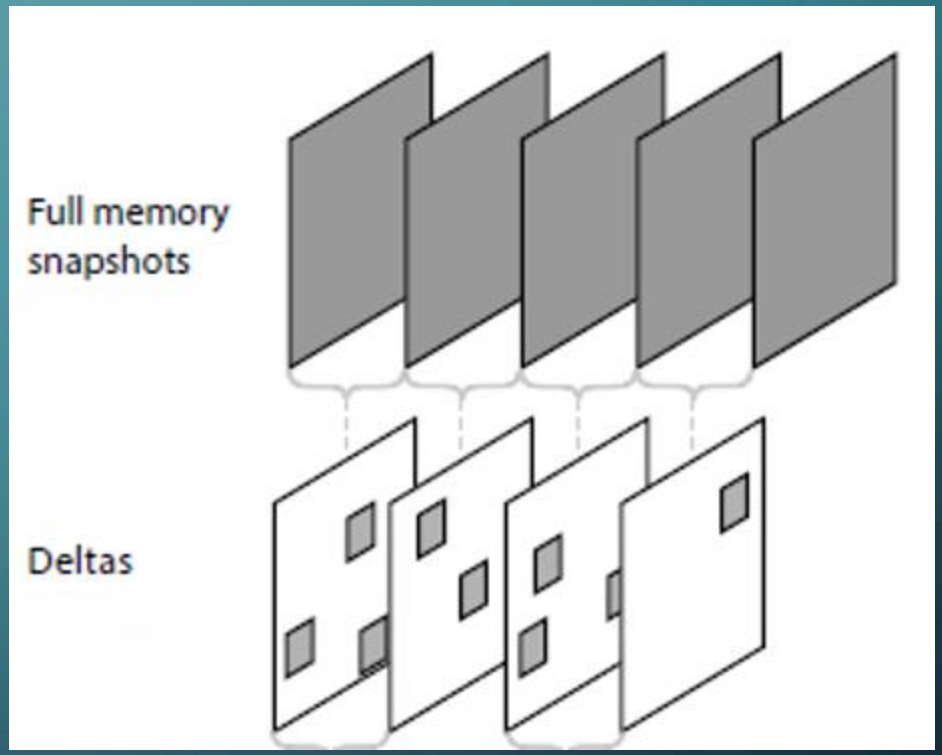


SO WHAT'S THE PROBLEM?



SNAPSHOT DELTAS

- **Concept of Snapshot Deltas:**



- **Traditional Methods**
  - Use a 4KB bitmap to track changes and capture deltas.

STATE OF THE ART SNAPSHOT DELTA CAPTURE

- **Improved Snapshot Deltas by leveraging CXL**
- What is CXL? (Compute Express Link)
- Track changes at a much finer granularity of 64 Bytes!
- Decreased Snapshot sizes with cache line deltas.

- Daniel Waddington, Moshik Hershcovitch, Swaminathan Sundararaman, and Clem Dickey. 2022. A case for using cache line deltas for high frequency VM snapshotting. In Proceedings of the 13th Symposium on Cloud Computing (SoCC '22).

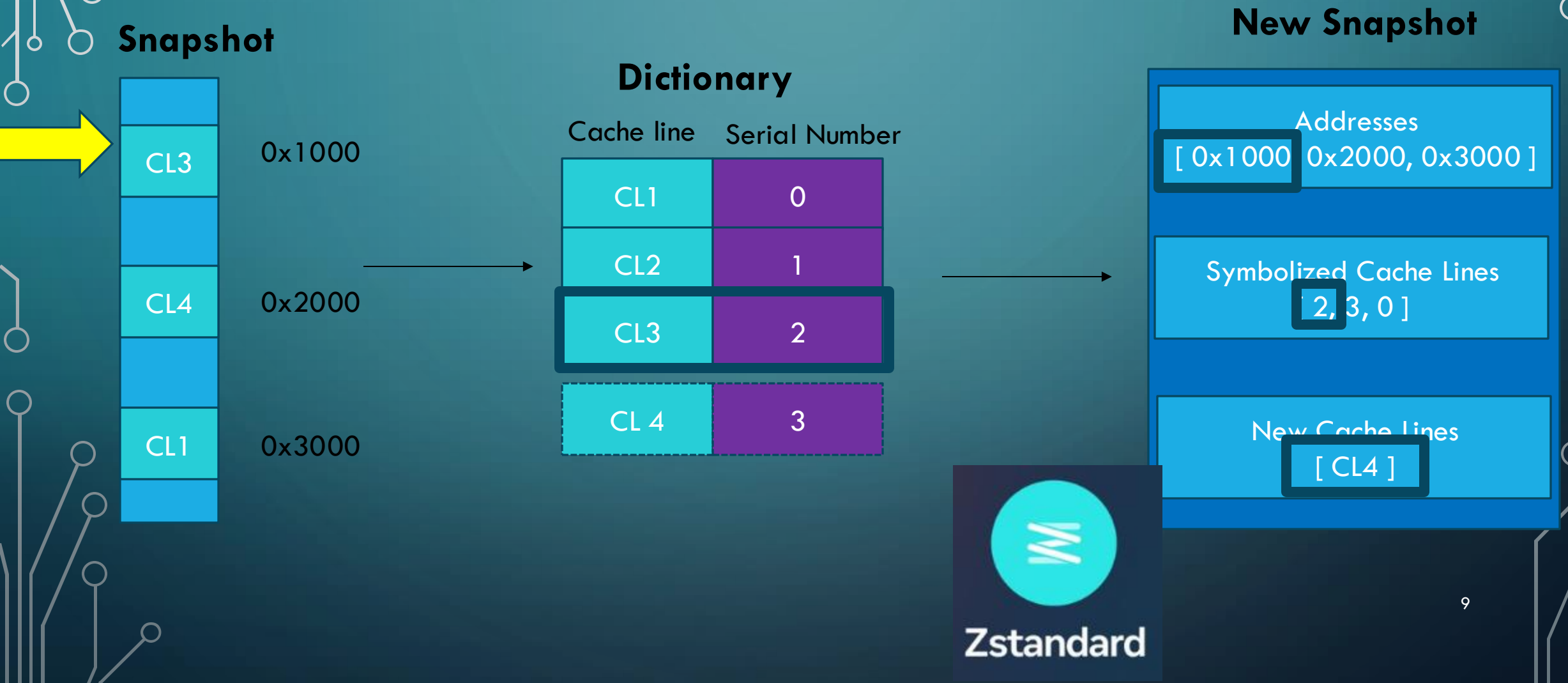
OUR CONTRIBUTIONS

- **What did we contribute?**
  1. Evaluation of several Compression Algorithms

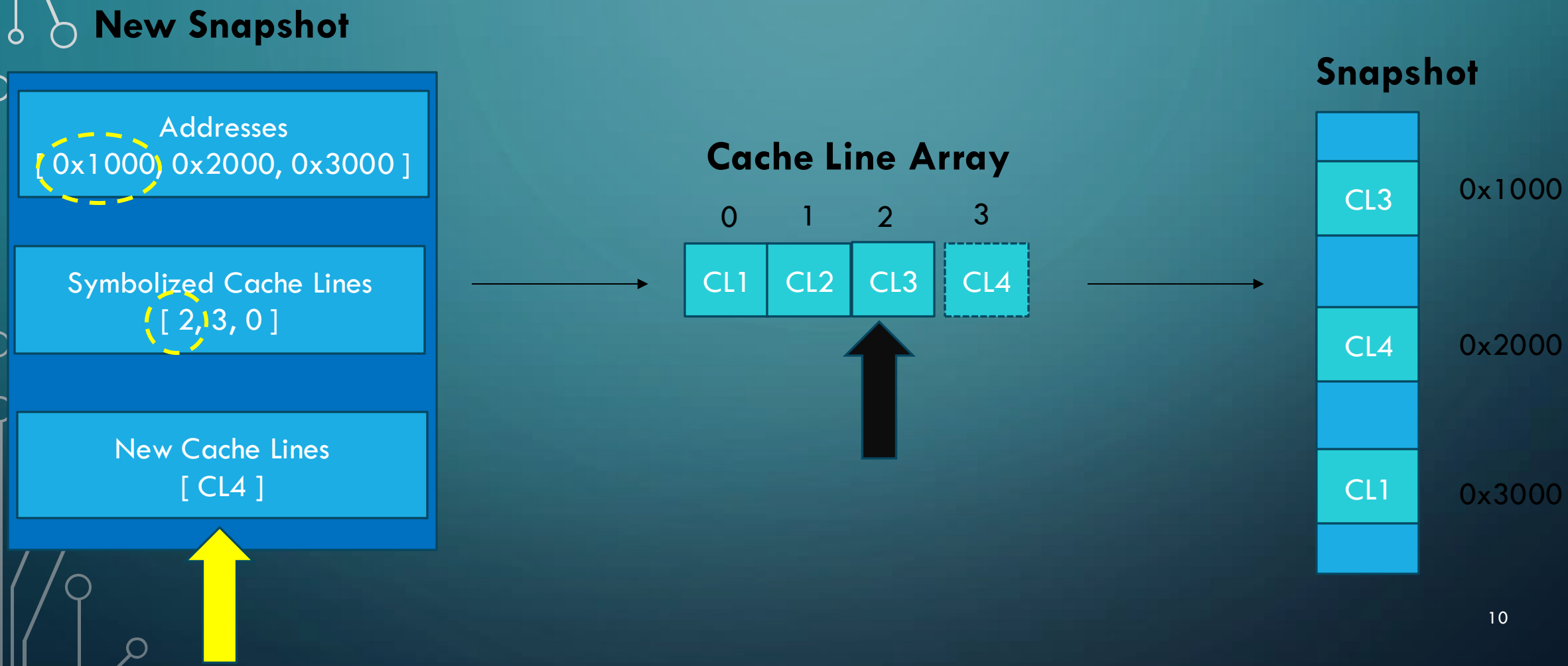
     Daniel Cohen, Sarel Cohen, Dalit Naor, Daniel Waddington, and Moshik Hershcovitch. 2023. Cache Line Deltas Compression. In Proceedings of the 16th ACM International Conference on Systems and Storage (SYSTOR '23)

  2. Dictionary Based Compression
     - Byte Grouping
     - Cache Line Partitioning

DICTIONARY BASED COMPRESSION – ACTIVE VM



DICTIONARY BASED COMPRESSION – STANDBY VM

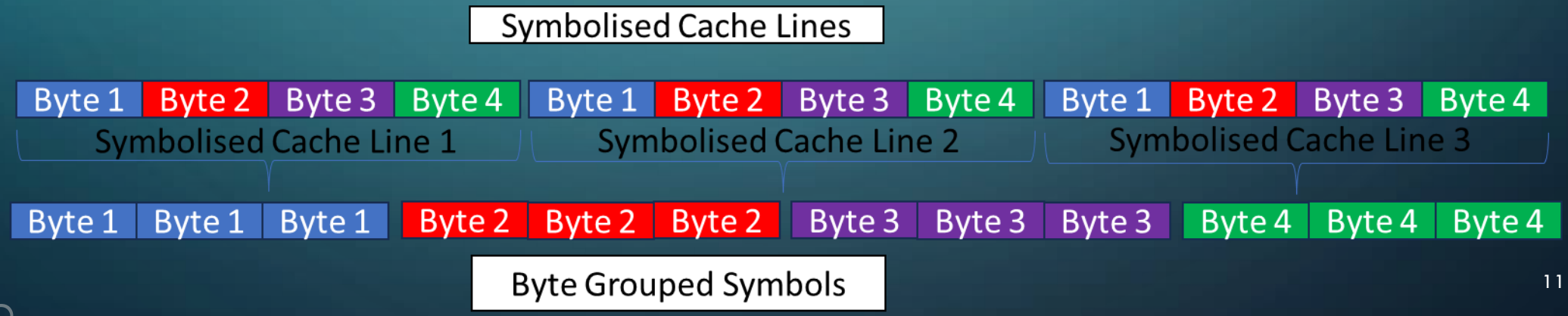


BYTE GROUPING

- **What is Byte Grouping?**
- **Why do we need it?**
- **How does it work?**

Addresses
[0x1000, 0x2000, 0x3000]

Symbolized Cache Lines
[2, 3, 0]



EXPERIMENTS

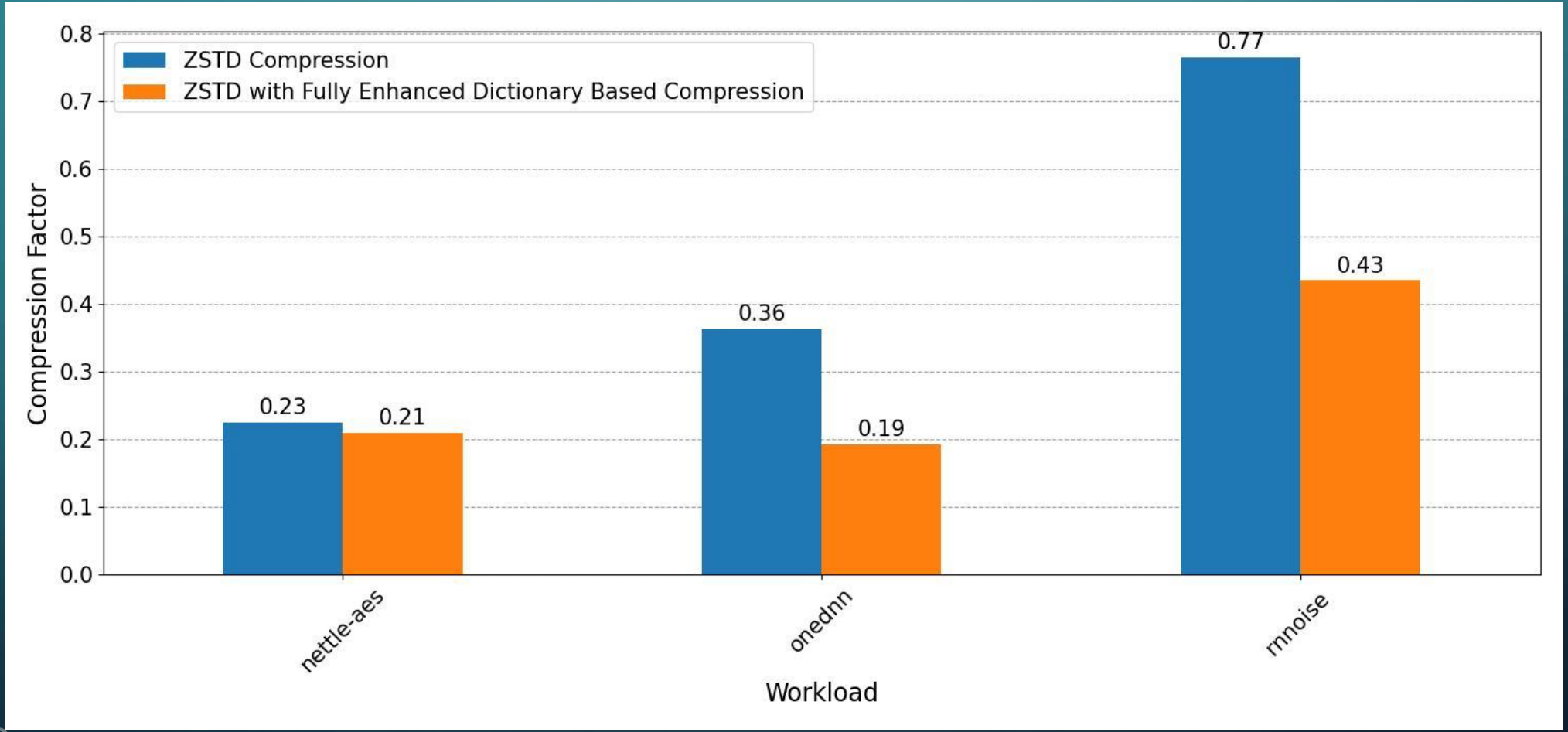
- Our Workloads:

- How does each dataset look like?
  - 600 Snapshots Deltas taken every 200ms

- Compression Factor
  - Proportion of the original size

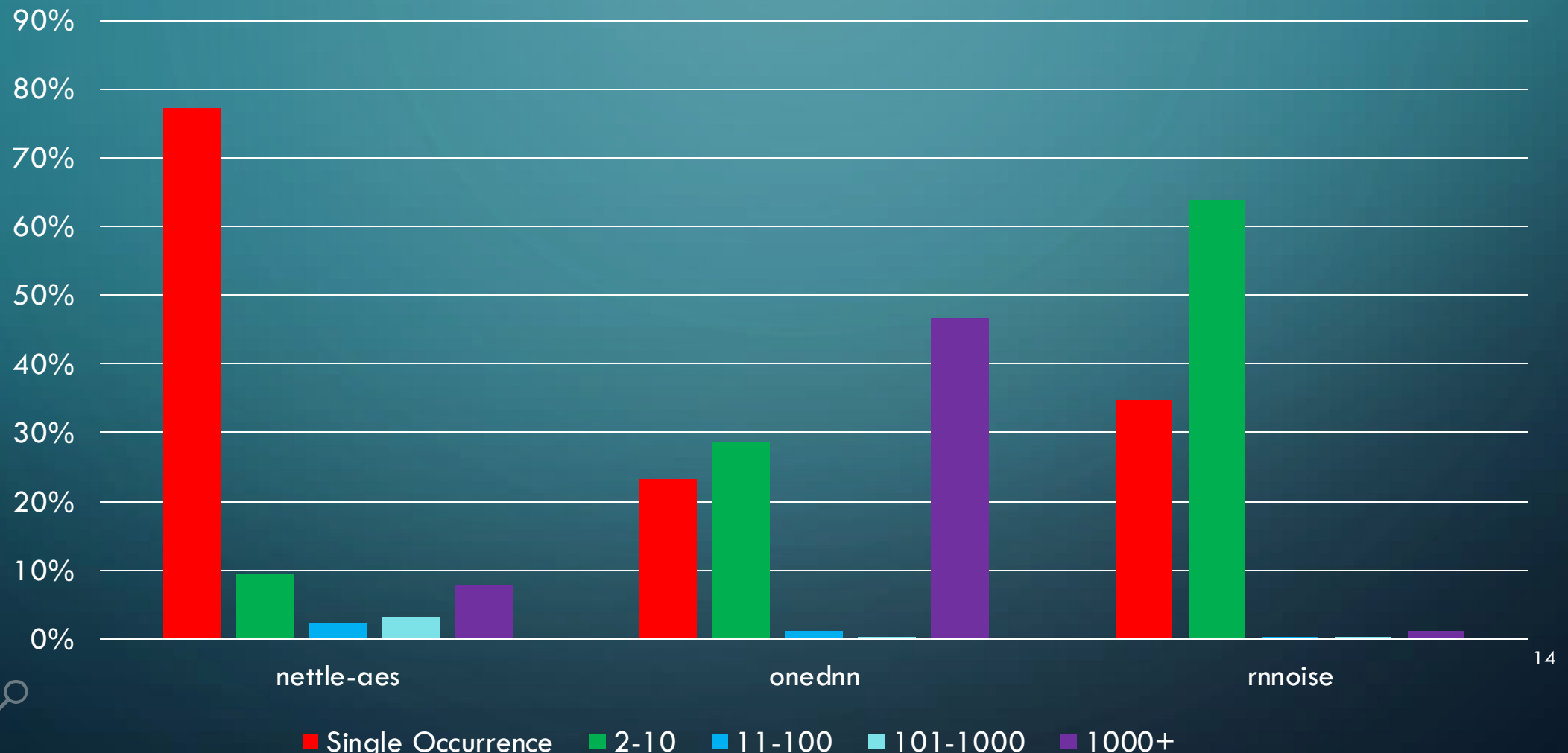
Test	Description
sqlite	Simple SQLite database benchmark
ebizzy	Workload resembling web-server
leveldb	Key-value store that uses Snappy compression
influxdb	InfluxDB time-series database
memcache	Memcache in-memory cache put workload
build-gcc	Compile the GCC compiler
quantlib	Quantitative finance for modeling, trading, and risk management
ngspice	SPICE circuit emulator
py-imgseg	Python image segmentation (skimage)
dolfyn	Computational Fluid Dynamics (CFD) simulation
himeno	Linear solver of pressure Poisson
py-3drotate	Python 3D matrix rotation (numpy)
nettle-aes	AES cryptography from the Nettle library
py-graph-spn	Python weighted graph spanning tree
py-feature	Python logistic regression feature selection
py-faces	Python face recognition using eigenfaces and SVMs
als	MLlib Alternating Least Squares (ALS) matrix factorization
rnnoise	Recurrent neural network for audio noise reduction
genetic	Genetic algorithm using the Jenetics library
onednn	Deep neural network training

DICTIONARY BASED COMPRESSION - RESULTS



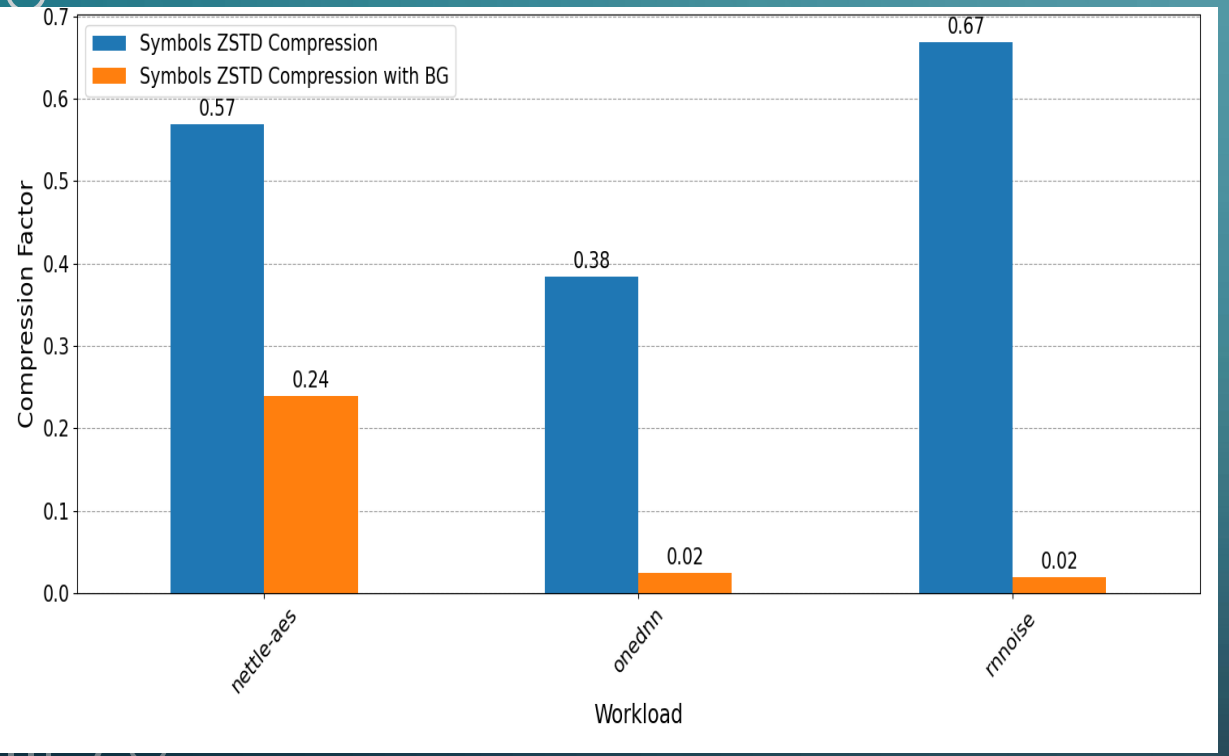
CACHE LINE OCCURRENCES

- **What influences the Performance of our Dictionary Based Compression?**

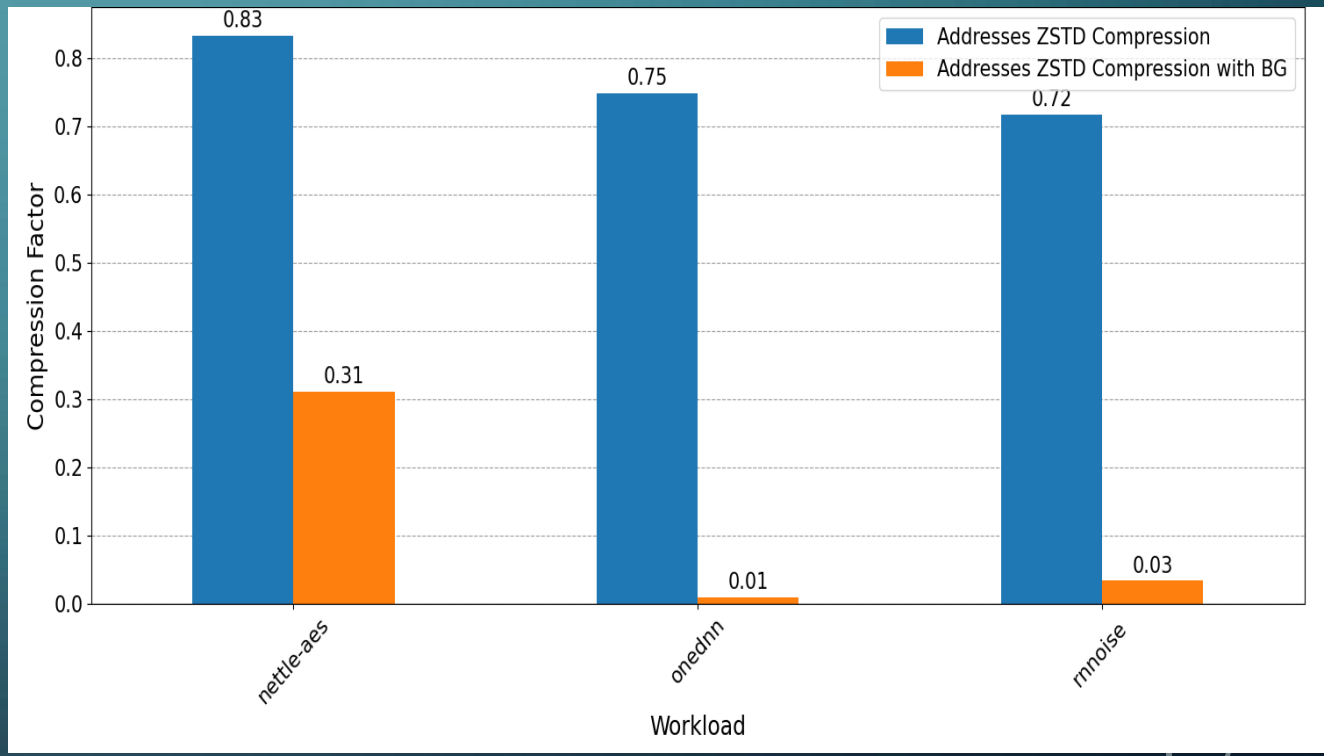


COMPRESSION WITH BYTE GROUPING

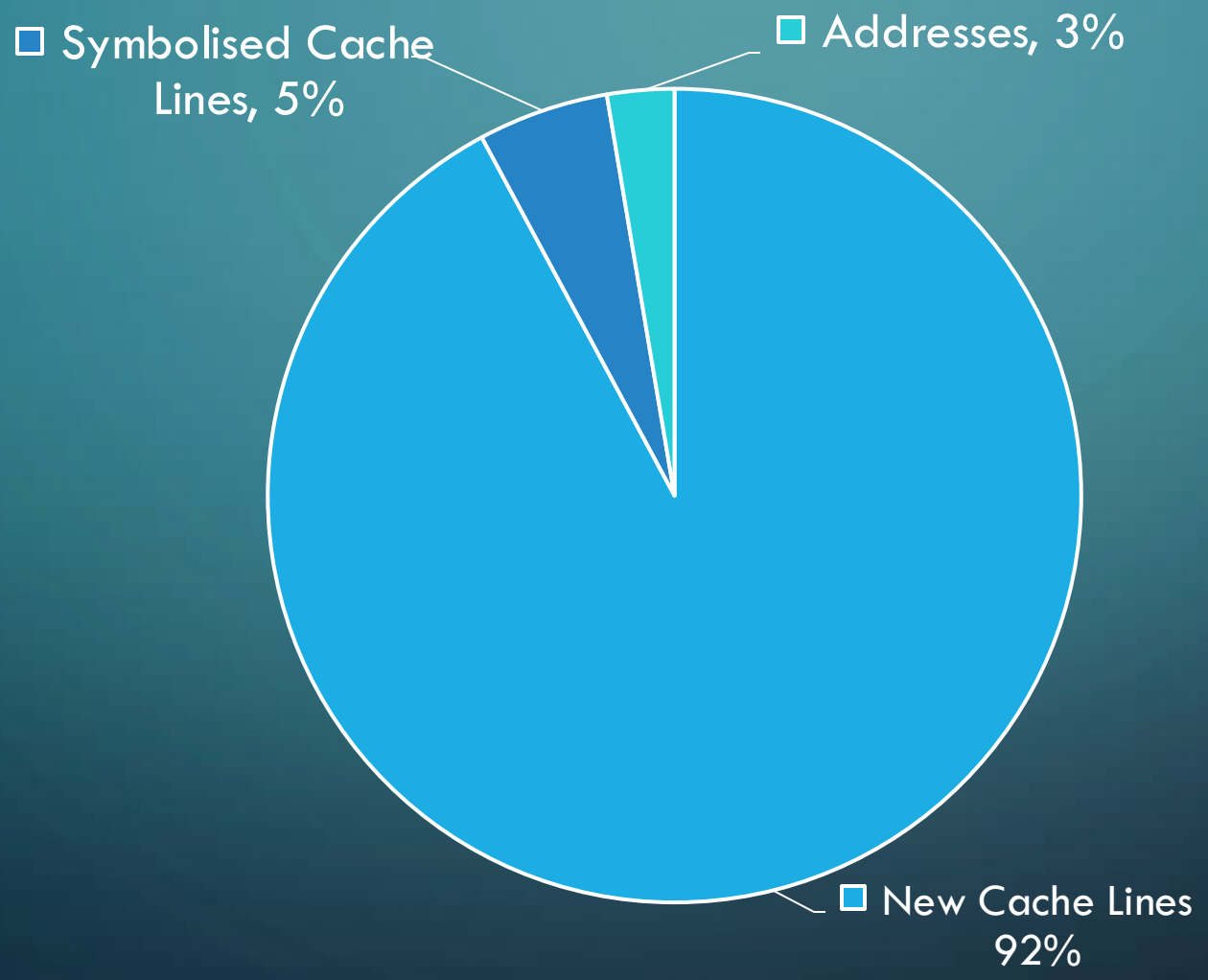
SYMBOLS COMPRESSION



ADDRESSES COMPRESSION



SIZE PERCENTAGES OF NEW SNAPSHOT COMPONENTS



CONCLUSION AND FUTURE WORK

- Our work builds upon the idea of representing memory snapshots deltas using cache lines.
- We found out whether cache lines deltas are compressible beyond standard compression and presented how we can use a dictionary-based method to improve on those results.
- The most prominent factor in our reduced snapshot mainly consists of the unique cache lines we have to transfer in each snapshot.
- Natural directions for future work:
  1. Measure the trade off between computation time and snapshot reduction
  2. Investigate which workloads respond better to our compression
  3. Limiting dictionary sizes

QUESTIONS?