A Case for using Cache Line Deltas for High Frequency VM Snapshotting

*Daniel Waddington
Moshik Hershcovitch
Swaminathan Sundararaman
Clem Dickey

IBM Research Almaden

Disclaimer

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY.

WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED.

IN ADDITION, THIS INFORMATION IS BASED ON IBM’S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE.

IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION.

NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, OR SHALL HAVE THE EFFECT OF:

• CREATING ANY WARRANTY OR REPRESENTATION FROM IBM (OR ITS AFFILIATES OR ITS OR THEIR SUPPLIERS AND/OR LICENSORS); OR

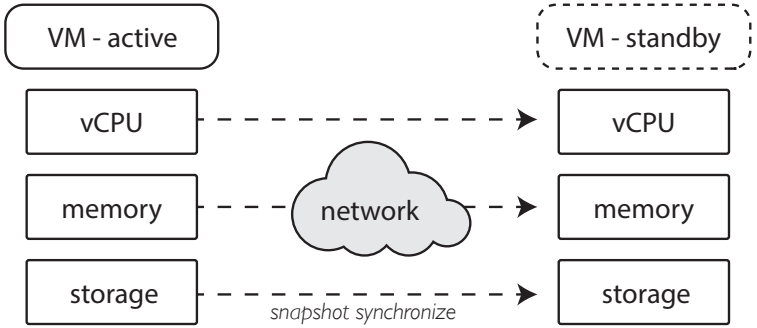
• ALTERING THE TERMS AND CONDITIONS OF THE APPLICABLE LICENSE AGREEMENT GOVERNING THE USE OF IBM SOFTWARE.

IBM’s statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM’s sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Outline

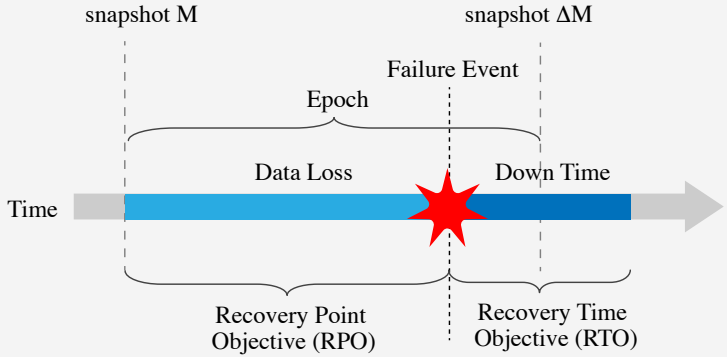
- Problem context - VM snapshotting
- New opportunities to reduce copy-amplification with emerging CXL hardware
- Case study and quantitative evaluation of gains from leveraging CXL-based Intelligent Memory Controller

Continuous VM snapshotting for high-availability



Active VM is periodically suspended, caches flushed, IO flushed and changes (deltas) copied across the network to a standby instance

State-of-the-art hypervisors use 4KiB page granularity to identify memory changes



Continuous synchronization of active and standby virtual machine instances – on failure the standby machine “takes over”

Compute, memory, network and storage all need to be coherent (achieving this is outside the scope of the paper)

Minimizing snapshot epoch reduces RPO and the amount of volume data that must be retained for replay in the network

CXL (Compute eXpress Link)

CXL is an open standard industry-supported cache-coherent interconnect for processors, memory expansion, and accelerators

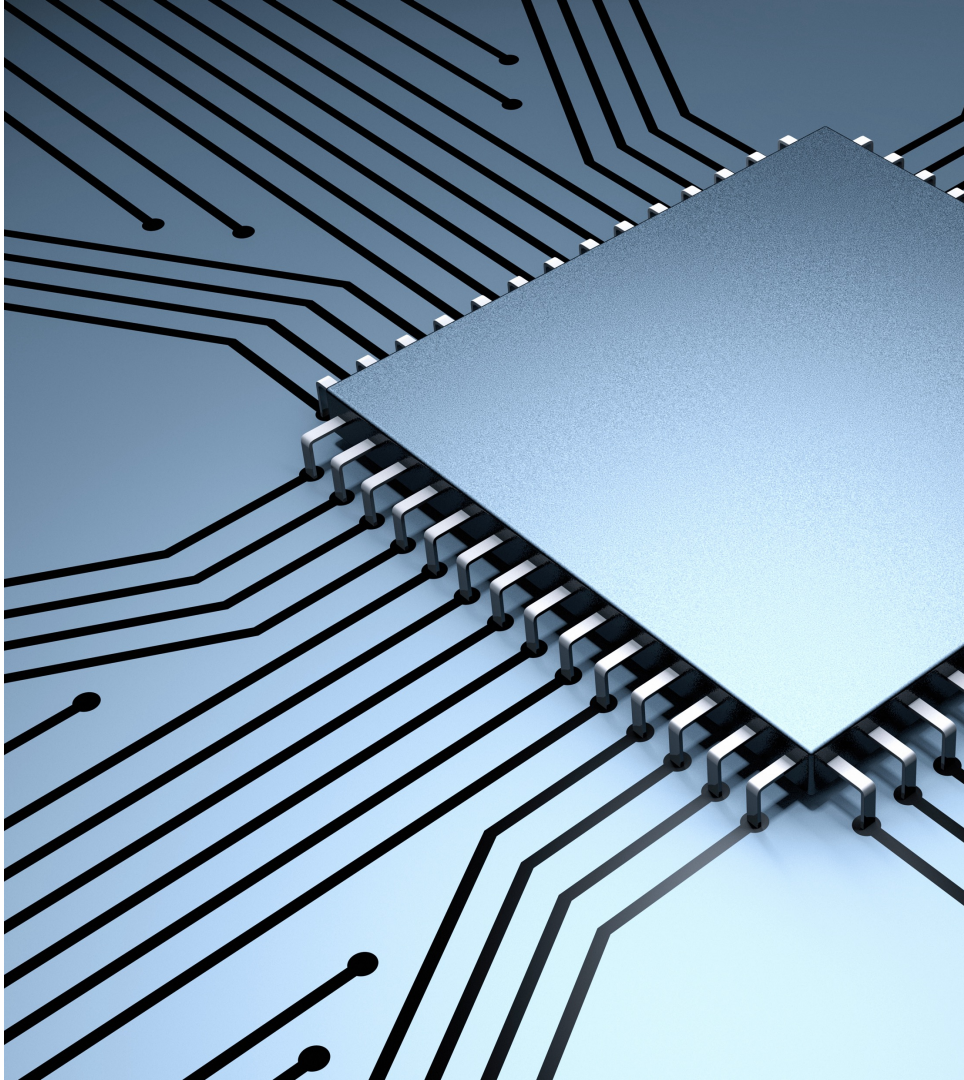
- https://www.computeexpresslink.org/

Specification version 2.0 ratified

CXL leverages a PCIe 5 feature that allows alternate protocols to use the physical PCIe layer

Intel Sapphire Rapids and AMD Genoa expected to support CXL

IBM OpenCAPI and Open Memory Interface to be absorbed by CXL

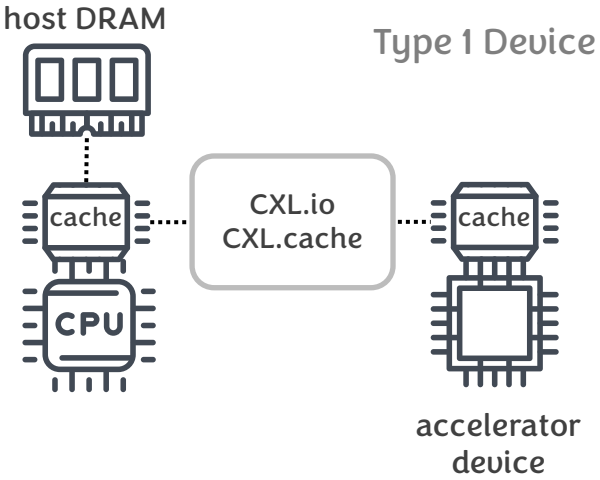


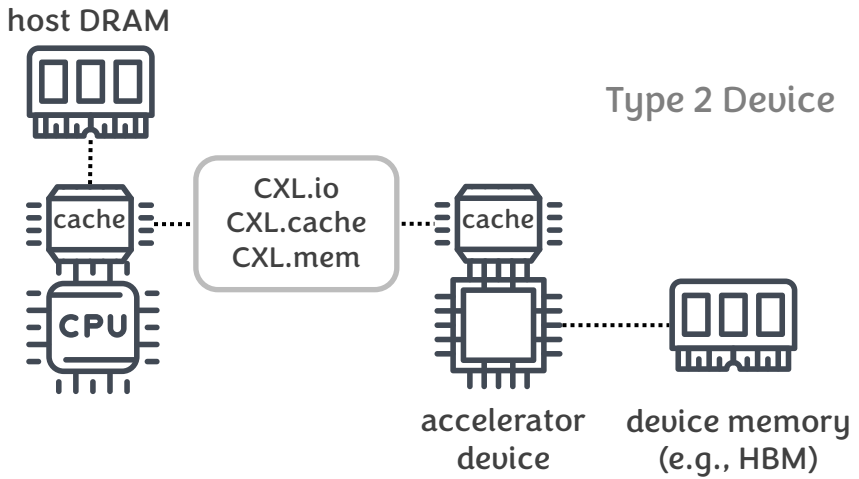
CXL (Compute eXpress Link)

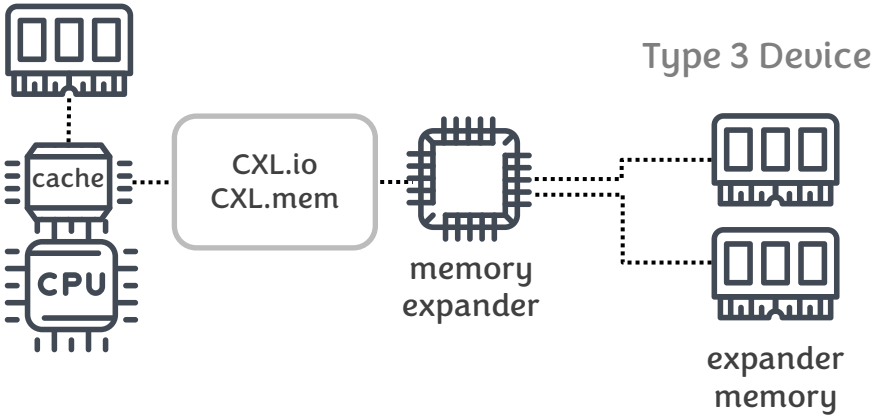
Type 1: Devices coherently access host memory

Type 2: Devices share their own memory with host

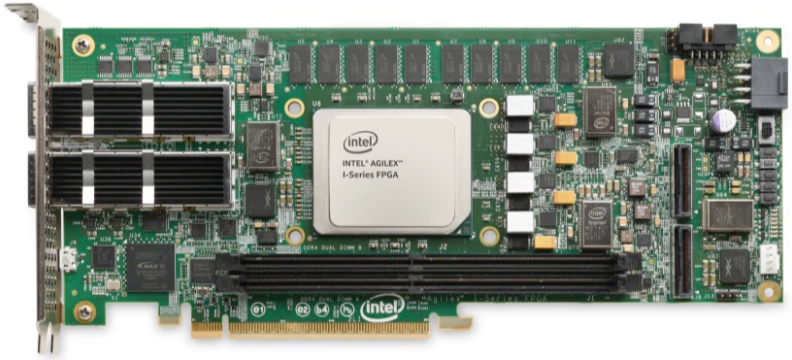
Type 3: Memory expansion device

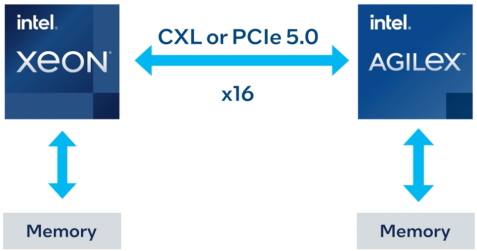






Intel Agilex Platform



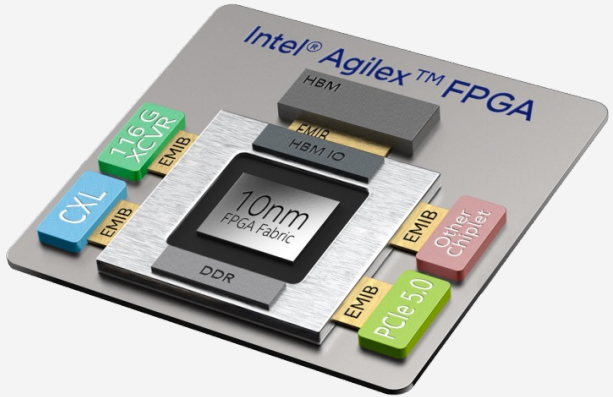


I-Series FPGA / SoC accelerator

Combination of hard and soft IP

Intel CXL hard IP supports Type 1, Type 2, and Type 3 configurations and specification revisions 1.1 and 2.0 (R-tiles)

Other vendors IP can also be used with the platform

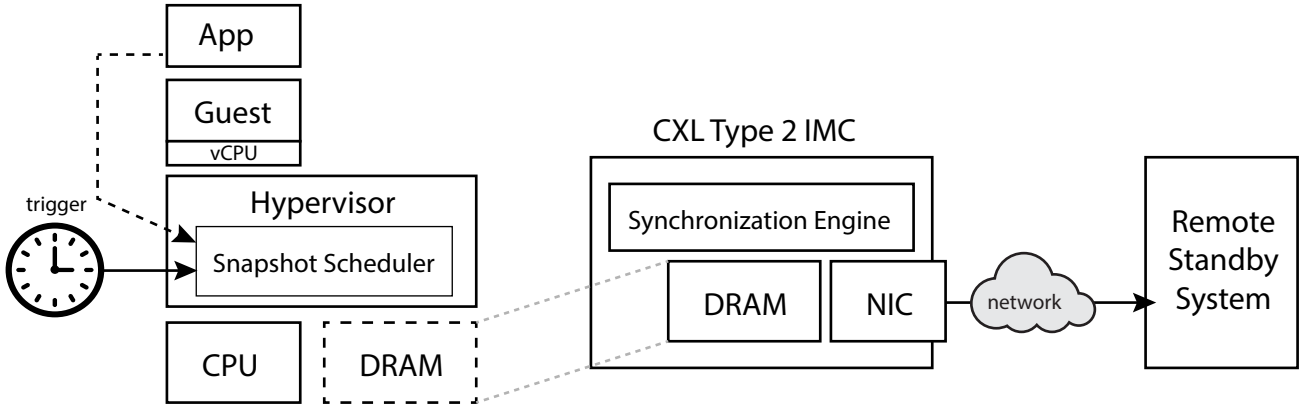


https://www.intel.com/content/www/us/en/products/details/fpga/agilex/i-series.html

Hypothesis

We can apply CXL-based technology to the continuous snapshotting problem

We can improve performance by reducing time to identify modified (dirty) memory since last epoch and reducing copy-amplification and ultimately the volume of data to transmit across the network



This work examines the reduction in copy-amplification through finer granularity and compression

Method

Run a broad set of real workloads on QEMU (600x 200ms epochs)

Take full memory snapshots at each epoch

Off-line …

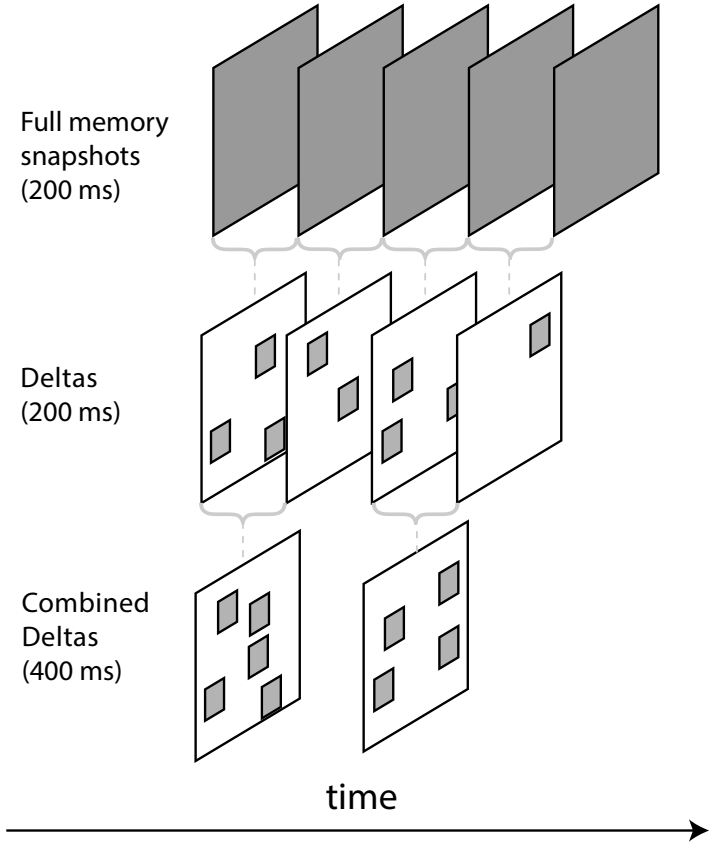
Derive deltas (4KiB and 64B)

Optionally apply compression (RLE, Zlib) – individual cache line or packed cache lines

Combine deltas to examine impact of larger epochs

Derive *mean page-write density* and *total transfer size*

$$WD_{\mu} = \overline{\forall p \in P_e : \frac{\text{modified-CL-count(p)}}{64}}$$



Benchmarks

DeathStarBench (Gan et al.) microservices benchmark

https://github.com/delimitrou/DeathStarBench

Phoronix Test Suite

https://openbenchmarking.org/

Test	Description
dsb-hotel	DeathStarBench - hotel booking app
dsb-social	DeathStarBench - social media app
dsb-media	DeathStarBench - media distribution app
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
savina	Savina concurrency benchmark for Reactors.IO

py-imgseg	Python image segmentation (skimage)
py-fft	Python FFT signal processing on audio (scipy)
gromacs	Molecular dynamics compute (water_GMX50)
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
forest	MLlib Random forest classifier
bayes	MLlib multinomial naive Bayes classifier
genetic	Genetic algorithm using the Jenetics library
onednn	Deep neural network training
rnnoise	Recurrent neural network for audio noise reduction
cobweb	Akka unbalanced cobwebbed tree

Categories:

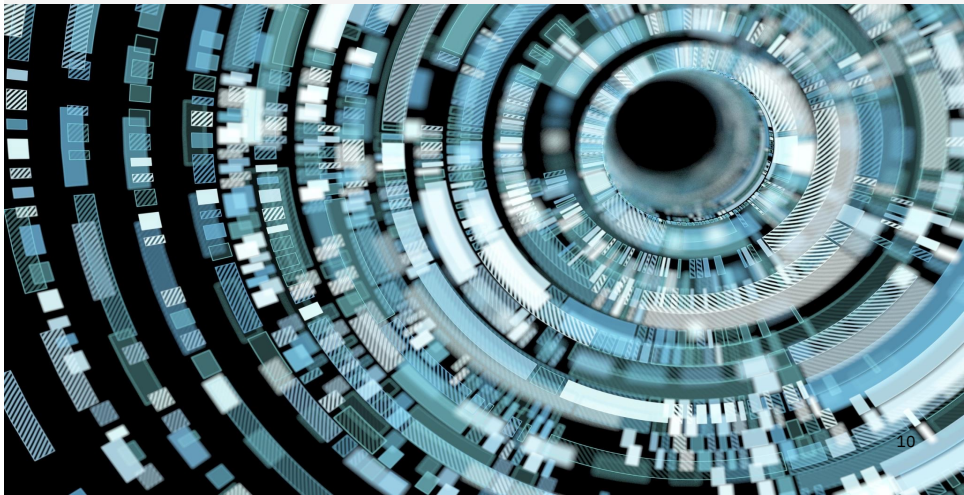
Microservices

Cloud

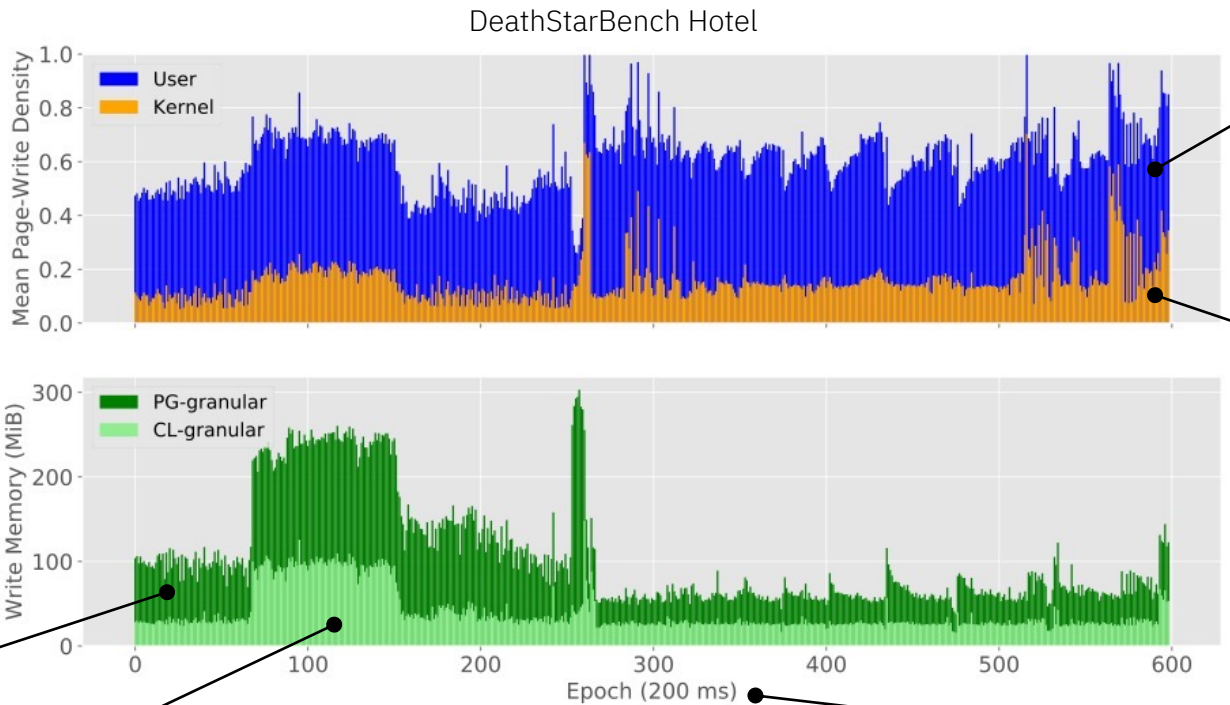
Enterprise

Numerical

AI/ML



Plot interpretation



mean write density of modified user-space pages (not stacked)

mean write density of modified kernel-space pages

total write volume at page granularity

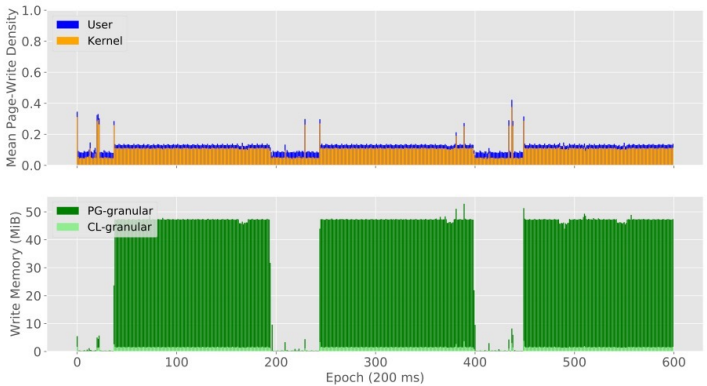
total write volume at cache line granularity

epoch of 200ms

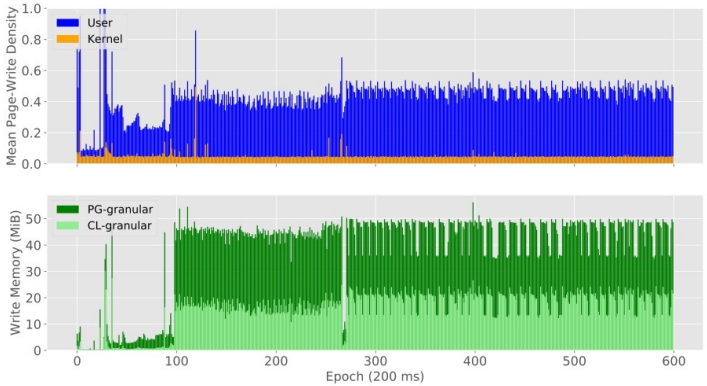
Test	Epoch Count	WD_μ	WD_σ	CL_{mod} MiB	PG_{mod} MiB	Amp $Factor$
dsb-hotel	600	0.45	0.08	20615.69	46275.91	2.24

Test	Memory Footprint Reduction (MiB)	Duration (seconds)	Copy Bandwidth Reduction (MiB/s)
dsb-hotel	25660.22	120	213.84

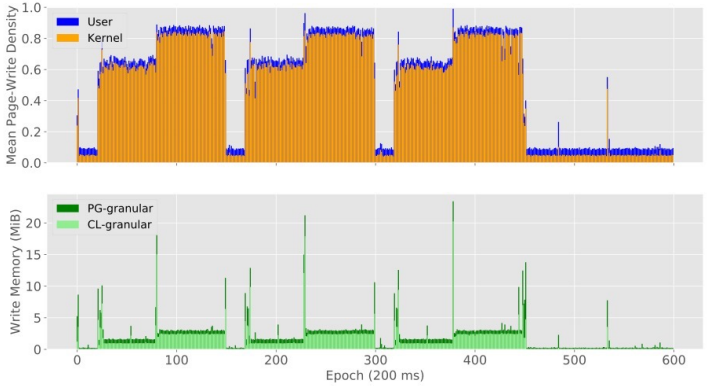
Sample plots



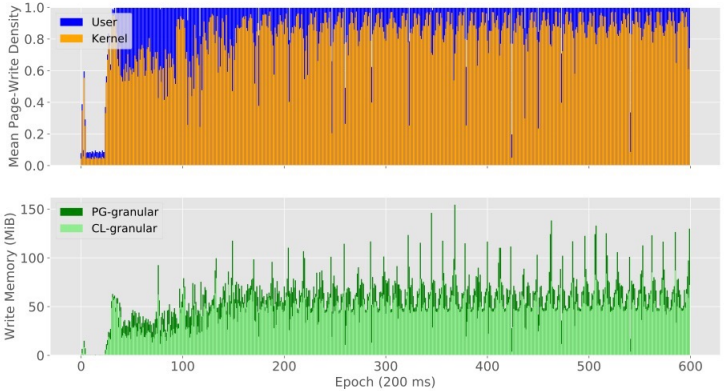
memcache: 44.6 amp. factor, 157.8 MiB/s transfer reduction



ngspice: 2.44 amp. factor, 113 MiB/s transfer reduction



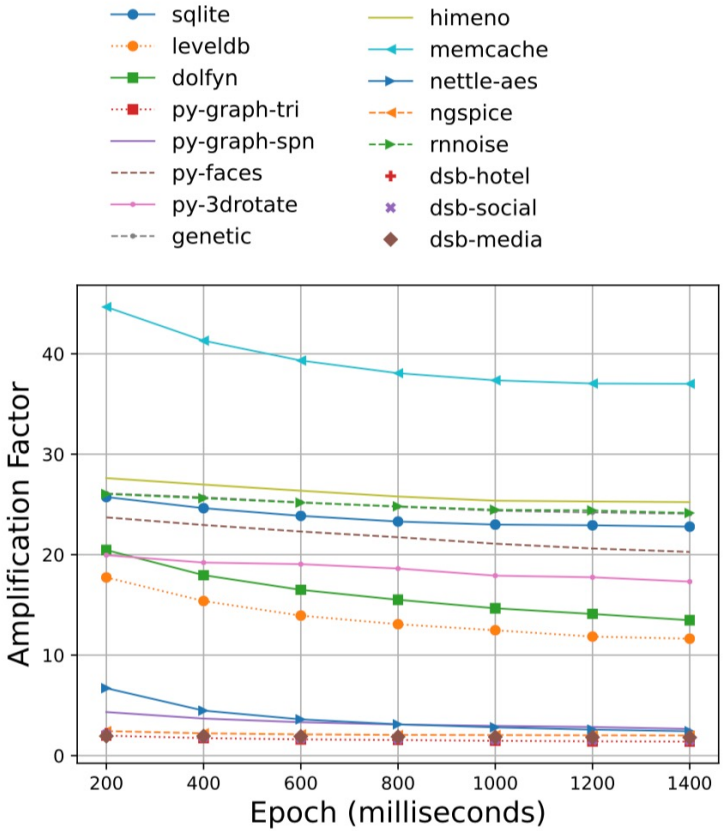
dolfyn: 20.46 amp. factor, 0.36 MiB/s transfer reduction



forest: 1.47 amp. factor, 68 MiB/s transfer reduction

Change in copy-amplification with change in epoch

Does epoch size impact amplification?



Summary of results

Measured amplification factors ranging from 1.02 to 44.6

Mean amplification factor across all experiments at 200ms epoch is 9.34

Measured copy-reduction bandwidth at 200ms epoch ranging from 0.19 MiB/s to 393.5 MiB/s

Mean copy-reduction across all experiments is 57 MiB/s
(no compression)

Max observed reduction in copy-amplification by increasing epoch to 1.4 second epoch (7x) is 17% (memcache)

Zlib compression of packed cachelines reduces data volume to 34% of original size

XOR RLE compression of packed cachelines reduces data volume to 56% of original size

Conclusions

Most applications write pages at 100-200 MB/s for a 200ms epoch (i.e., up to 1GB/s)

Worst-case spikes can create 500MB/epoch (2.5GB/s)

By using cache line granularity (enabled by CXL technology) – data volume can be reduced by ~10x depending on the workload

This data volume can be reduced approximately by half again by using fast XOR-RLE compression

Combining cache line deltas and compression brings worst case to ~125MiB/s

Expanding epoch (which impacts volume of state that would need to be held in network) does not significantly impact write-amplification but of course can reduce data transfer volume

A CXL-based FPGA prototype could bring value providing that the base CXL memory access slowdown is not overwhelming

Questions?