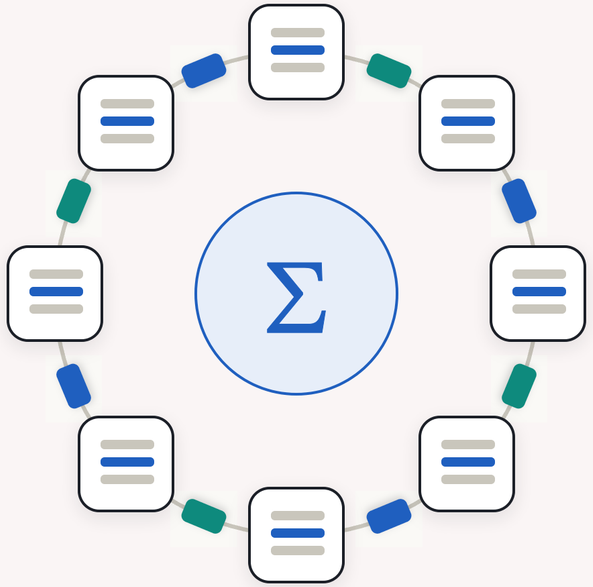
SC26 · CHICAGO · TECHNICAL PAPER

# hZCCL2

## Co-Designing Collective Communication with Vectorized Homomorphic Compression

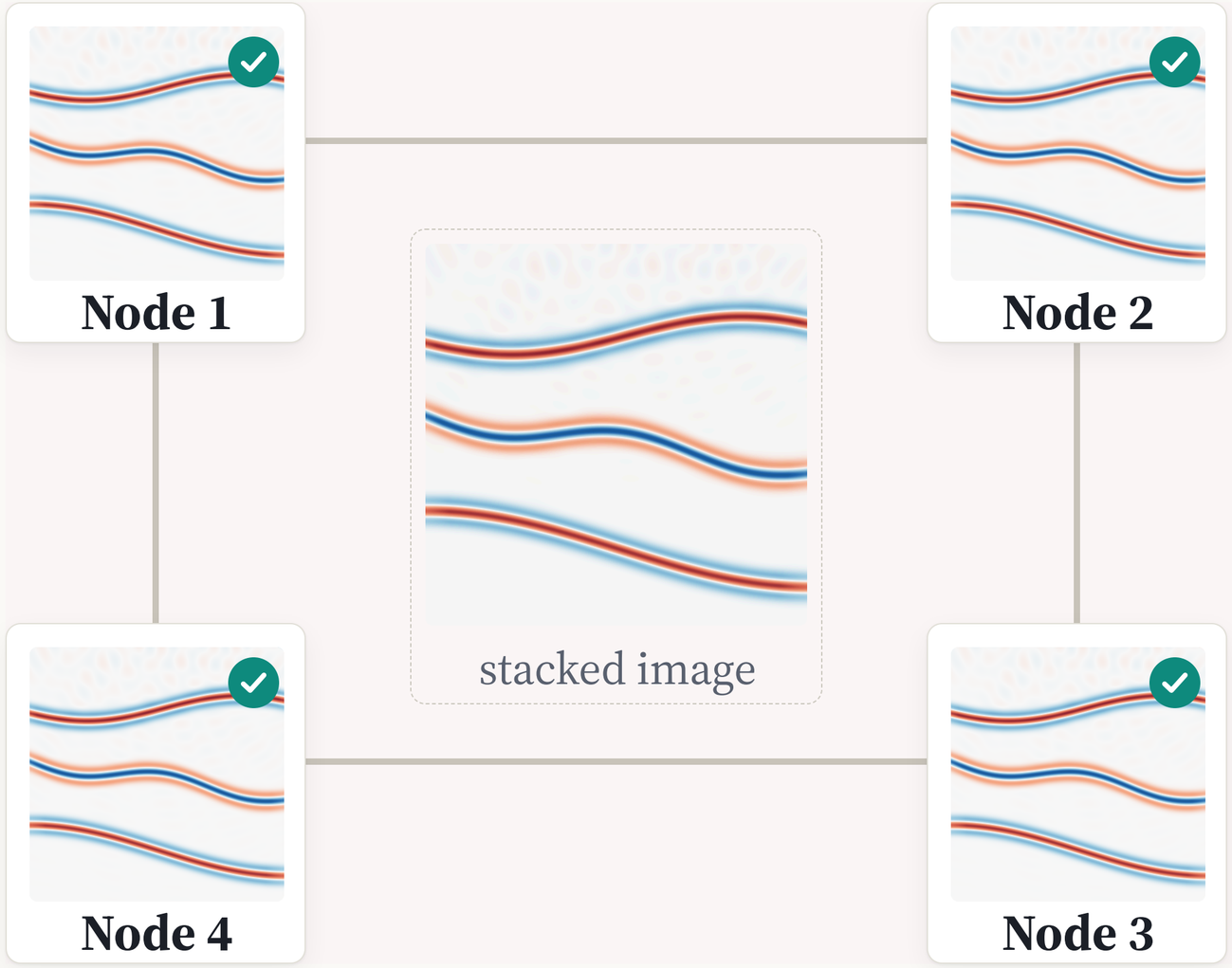



Jiefeng Zhou[1], Sheng Di[2], Yanfei Guo[3], Yuhao Guo[1],
Lingqi Zhang[4], Kai Zhao[5], Rajeev Thakur[2],
Franck Cappello[2], Jiajun Huang[1]

[1]University of South Florida [2]Argonne National Laboratory [3]NVIDIA
[4]RIKEN R-CCS [5]Florida State University

MOTIVATION

# Reducing large fields across nodes is network-bound




## The wire is the bottleneck

memory, one socket

**699.7 GB/s**

network link (100 Gb/s)

**12.5 GB/s** **56× slower**

100–600 MiB per node per call

> Results only need to stay within an **error tolerance**.

Same in climate, cosmology, AI training

MOTIVATION

# Lossy compression: 32 bytes become 7

One block of 8 values, error bound $\varepsilon = 10^{-3}$



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $x_i$ 32-bit floats | 0.8312 | 0.8472 | 0.8590 | 0.8655 | 0.8662 | 0.8604 | 0.8488 | 0.8322 |
| $q_i = \mathrm{trunc}(x_i/\varepsilon)$ error at most $\varepsilon$ | 831 | 847 | 859 | 865 | 866 | 860 | 848 | 832 |
| $\delta_i = q_i - q_{i-1}$ small integers | +16 | +16 | +12 | +6 | +1 | −6 | −12 | −16 |

as floats — 32 B

**compressed** — **7 B**, within $\varepsilon$ of every value

**Fewer bytes on the wire**, but encode and decode sit on the critical path.

MOTIVATION

# Homomorphic: add compressed data directly

**DOC**: decompress–operate–compress, at each of the $P-1$ ring hops




sequential rebuild: $q_i = q_{i-1} + \delta_i$

**Homomorphic** (hZCCL, SC24): add the compressed blocks directly




**Why it works** $\delta_i^A + \delta_i^B = (q_i^A + q_i^B) - (q_{i-1}^A + q_{i-1}^B)$

MOTIVATION

# Three gaps hold compressed collectives back

## 1 Compression is too slow

memory, one socket

**700 GB/s**

prior compressor

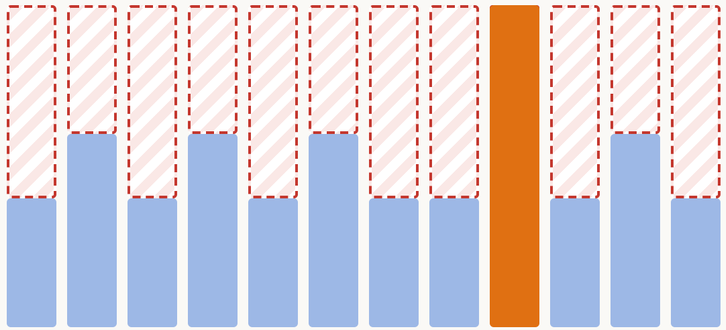
**154 GB/s**

**78% unused**

Prior compressors pack bits **one value at a time** and leave most of the bandwidth unused.

## 2 One outlier inflates a block

one **outlier** sets the width
**hatched**: wasted bits



Each value gets the width of the **largest**. Big blocks, which SIMD wants, waste the most.

## 3 Alltoall is not compressed

| | |
|---|---|
| **Bcast** | prior work ✓ |
| **Scatter** | prior work ✓ |
| **Allreduce** | prior work ✓ |
| **Alltoall** | missing ✗ |

Alltoall drives **FFTs** and **mixture-of-experts** training, and compressed sizes are unknown in advance.

MOTIVATION

# hZCCL2: one block format, end to end

## Compressor

**VecBA-SP: speed**

**OFLE: ratio**

SIMD from load to store

→

## Homomorphic add

$$\hat{A} \oplus \hat{B} = \widehat{A + B}$$

on compressed blocks,
no decompression

→

## Collectives

**Allreduce computes**

**Bcast · Scatter · Alltoall**

transport compressed data

**One block-aligned format** in every stage: no conversions, no copies

**5.3×**
faster compression

**+37.6%**
compression ratio

**8.1×**
faster reduction

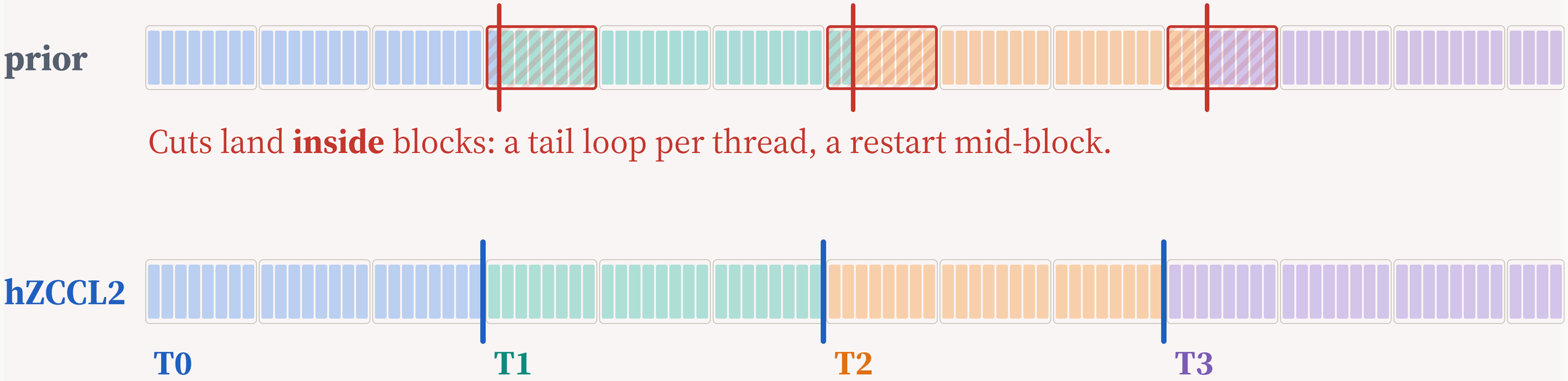
**39.5×**
faster RTM stacking

DESIGN

# Split the work in whole blocks, not values

$N$ values, blocks of $B$, $T$ threads: each thread takes

$$T_i = \lfloor N/(B\,T) \rfloor \times B$$




Whole blocks only: branch-free SIMD loops and parallel decode. **Alone 3.5× faster**, before any SIMD.

DESIGN

# One SIMD pass replaces a scalar loop

**1** **Prior: one scalar loop**

quantize, predict, pack: one value at a time

$\delta_i = q_i - q_{i-1}$ chains every iteration

**2** **Block-aligned: three simple stages**

quantize + delta → sign, magnitude → pack bits

simpler loops, more parallel instructions

**3** **Each stage in SIMD**

16 lanes → 16 lanes → 16 lanes

16 values per instruction (AVX-512)

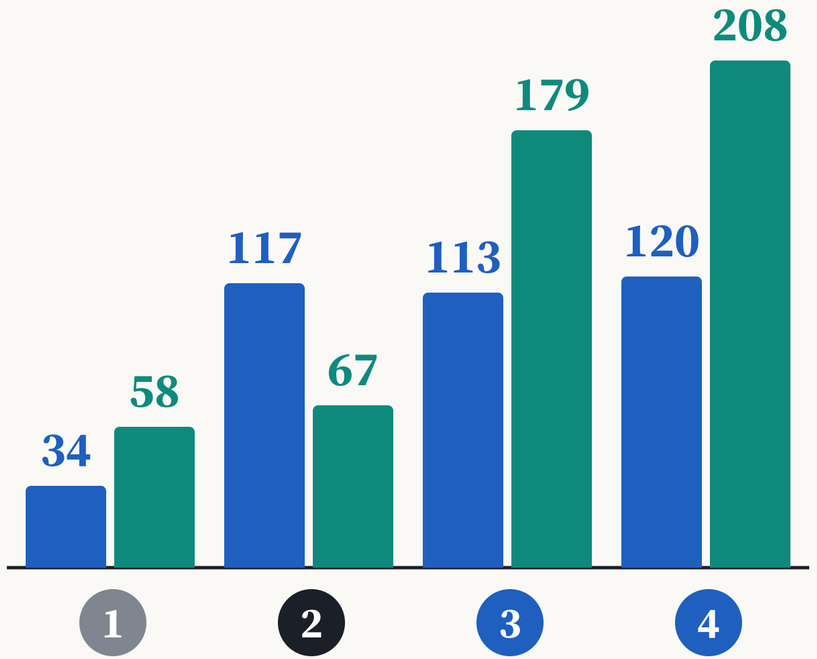
**4** **VecBA-SP: one fused SIMD pass**

load → quantize → delta → sign → max, in registers

no buffers between stages, same output bytes

**Throughput**, GB/s (HACC)

**compress** · **decompress**

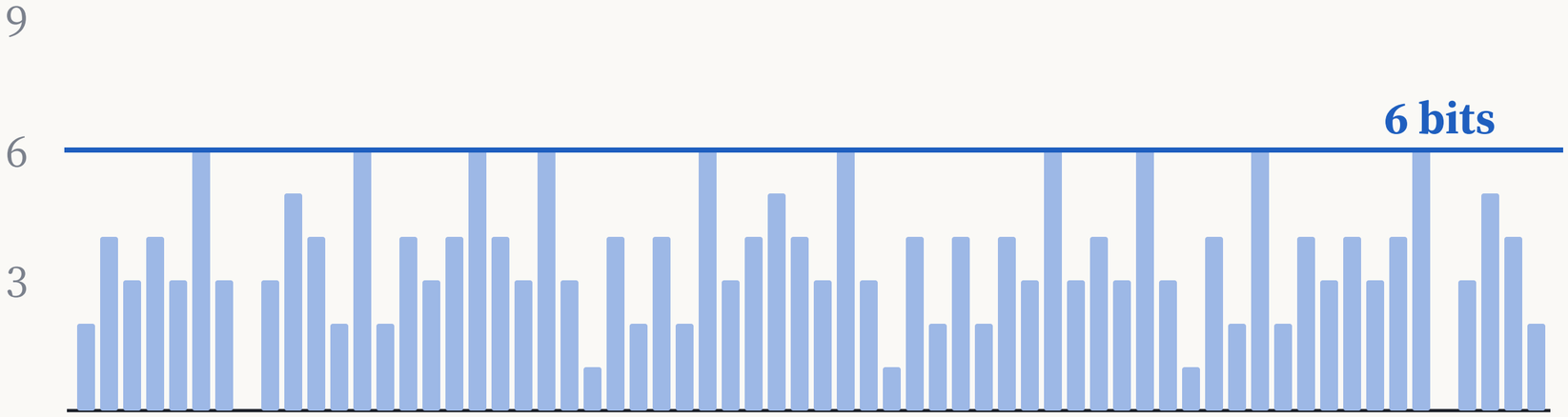



**fusion: +16% decompress here, up to +42%**

DESIGN

# Outliers move to a side stream

bits needed by each of 64 values in a block




## OFLE splits the block

6 bits each, plus a side stream

**outlier stream**

2 outliers: high bits and positions

**61 B** instead of 81

### VecBA-SP

**fastest**: one fused pass

when compression time dominates

### OFLE

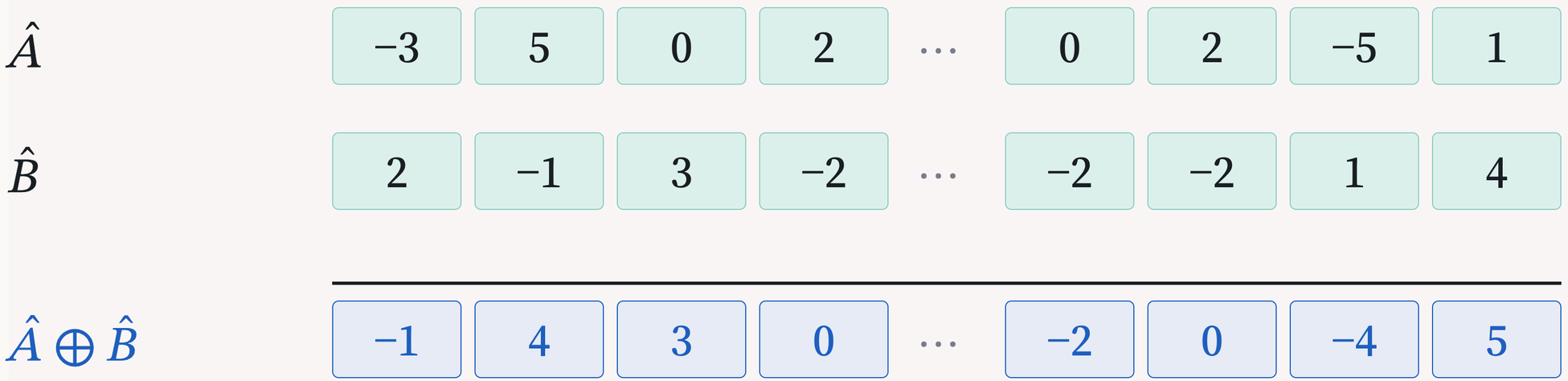
Hurricane ratio **9.2 → 11.8**, **+28%**, at 88% of the speed

when bytes on the wire dominate

DESIGN

# Adding compressed blocks: exact and SIMD-wide

Two compressed blocks, 16 lanes at a time (8 shown)




**one SIMD add**
for 16 lanes

Most blocks of high-ratio data are **zero**: a header check skips them.

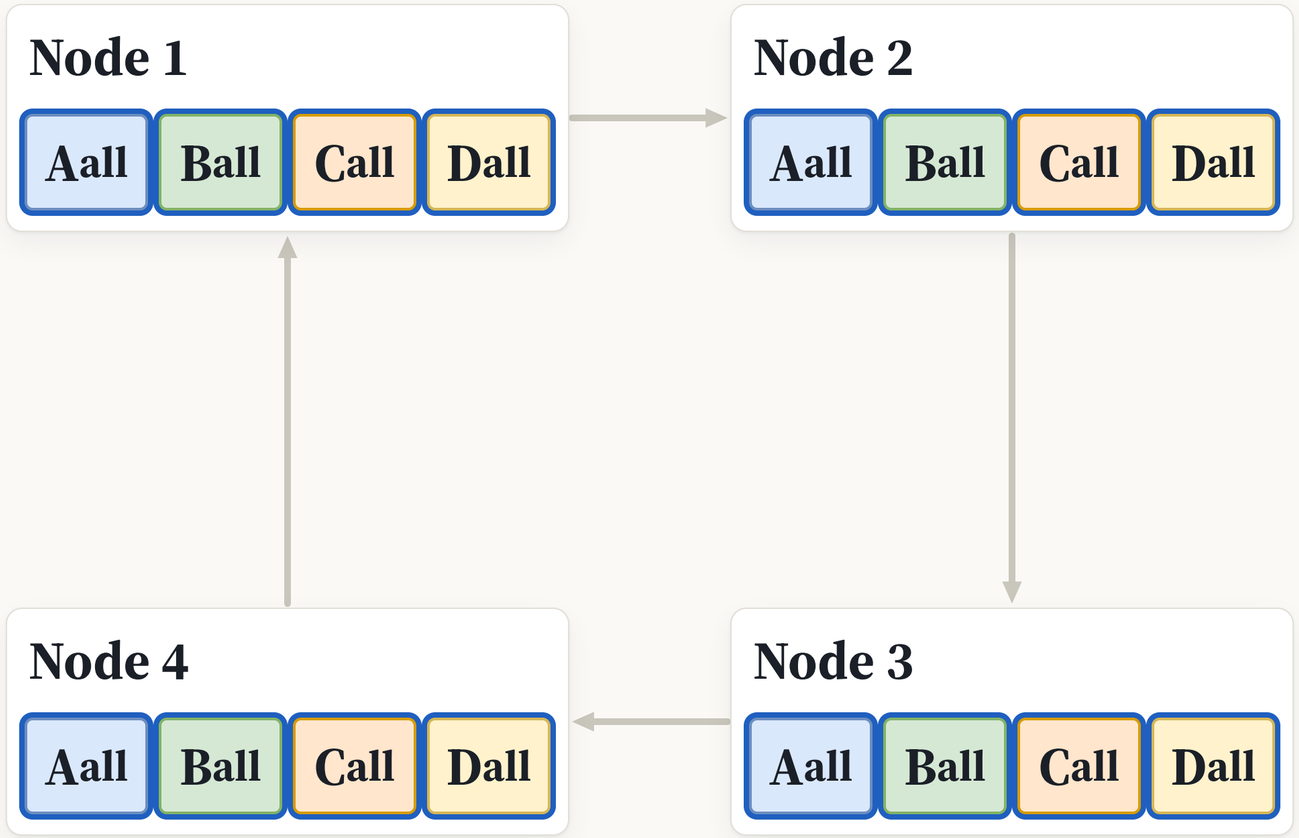
**Exact on the integer codes**

$$\delta_i^A + \delta_i^B = (q_i^A + q_i^B) - (q_{i-1}^A + q_{i-1}^B)$$

**Error of one reduction over $P$ ranks**

$$|\hat{s} - s| \le P\,\varepsilon$$

# Allreduce never leaves the compressed domain

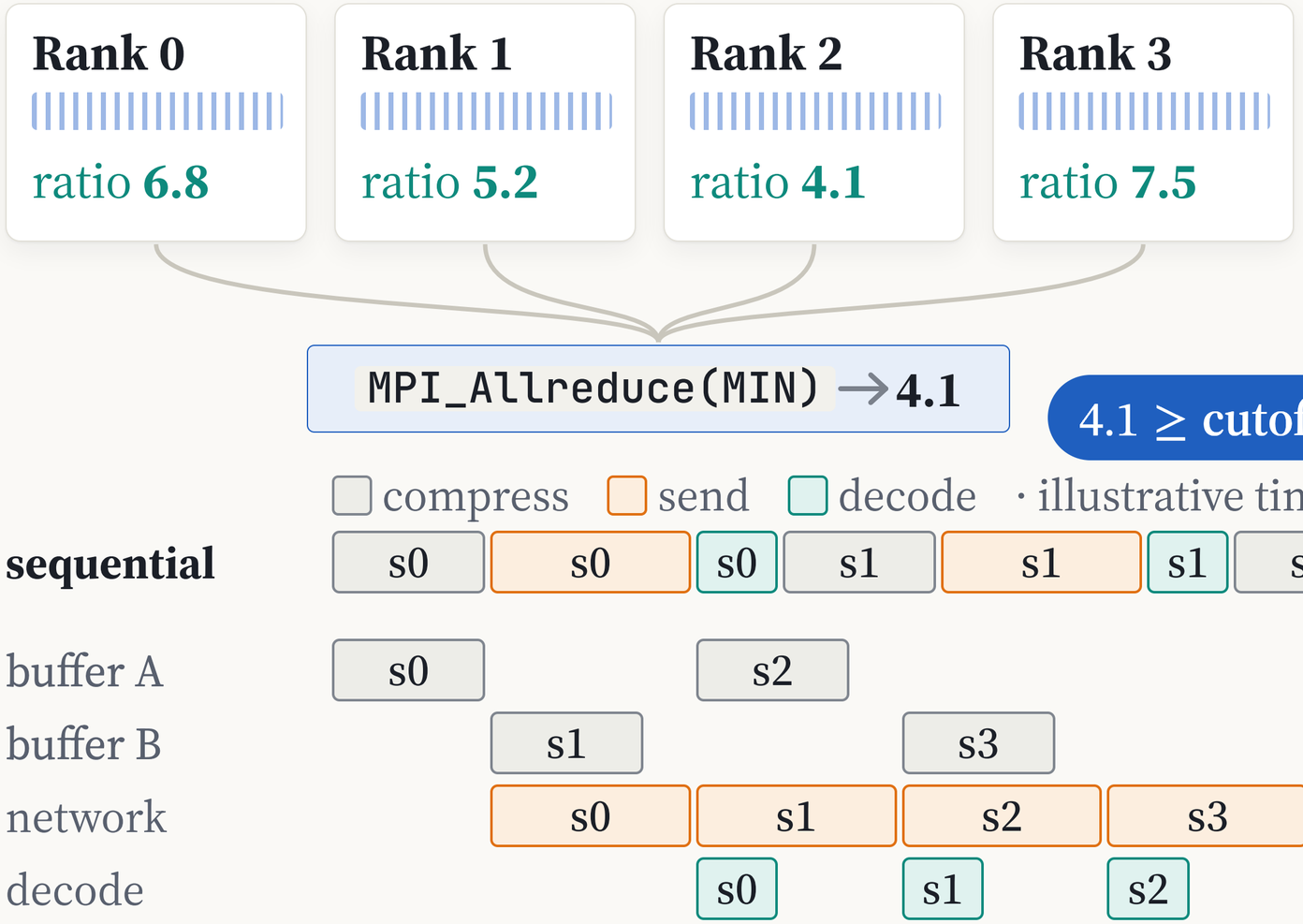



1. **Compress lazily**
   only what the first step needs
2. **Reduce-scatter**
   add received segments **directly**, still compressed
3. **Allgather**
   forward reduced segments as they are

**No decompress or recompress** in the ring; DOC needs $P - 1$ per node.

COLLECTIVES

# The first compressed Alltoall on CPUs




One small Allreduce lets all ranks **agree** before anyone compresses.

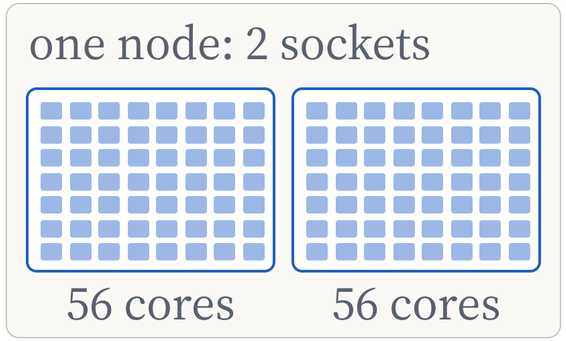
each rank samples 4,096 values; example ratios

EVALUATION

# 3,584 cores, eight real datasets

## Platform

**32** SPR nodes
**3,584 cores**




128 GB HBM2e per node

100 Gb/s Omni-Path

MPICH 4.3.2, GCC 13.2

## **Datasets**, FP32

| Dataset | Domain | Size |
|---|---|---|
| HACC | Cosmology | 6.3 GB |
| Hurricane | Weather | 1.3 GB |
| NYX | Cosmology | 3.1 GB |
| RTM | Seismic | 95.3 GB |
| SCALE | Climate | 6.3 GB |
| JetIn | Turbulence | 6.2 GB |
| HCCI | Combustion | 0.7 GB |
| QWEN-3.5 | AI weights | 9.1 GB |

## Compared

**native MPI**
MPICH 4.3.2

**hZCCL**
prior work, SC'24

**hZCCL compressor**
Alltoall baseline

**VecBA-SP · OFLE**
hZCCL2

## Protocol

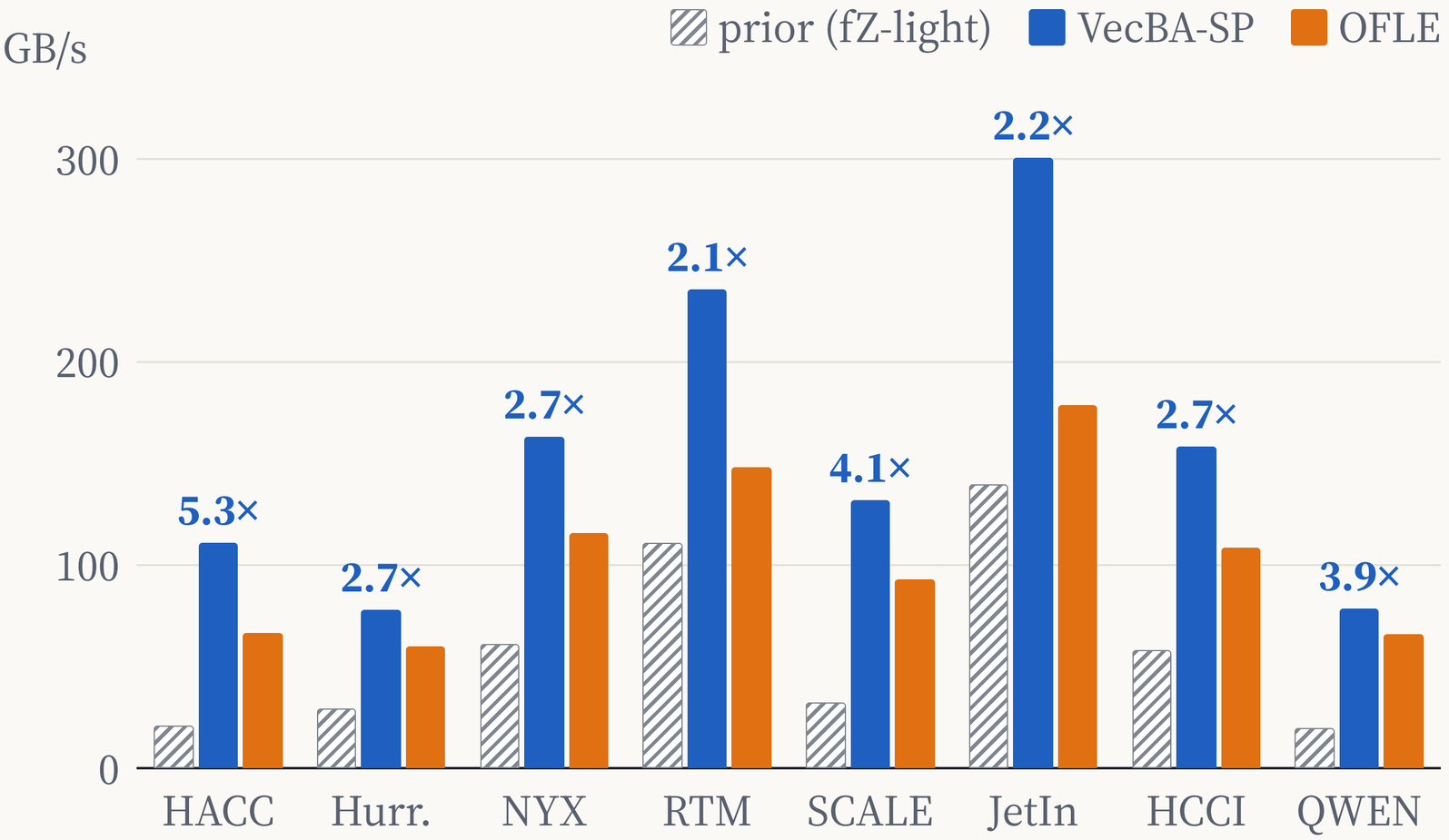
$B = 128$, relative $\varepsilon = 10^{-4}$

112 threads per node

best of 10 runs

EVALUATION

# Compression runs 2.1–5.3× faster




Compression, $B = 128$, relative $\varepsilon = 10^{-3}$, one socket (Fig. 5 in the paper)

**2.1–5.3×**

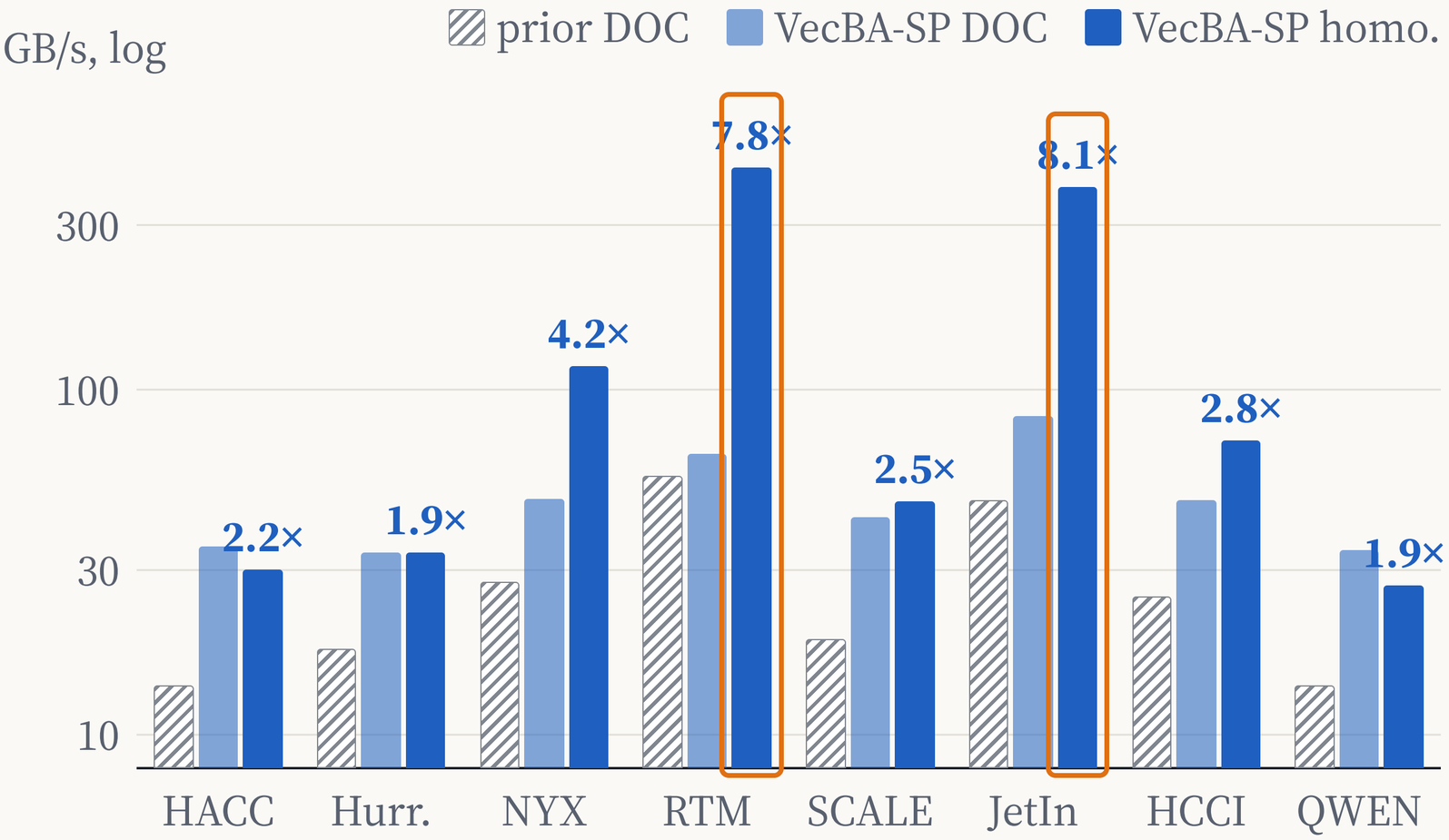
faster compression than the prior compressor, on all 8 datasets

**+37.6%**

more ratio with **OFLE** on Hurricane at $B = 256$; OFLE runs at 59–84% of VecBA-SP speed

EVALUATION

# Homomorphic reduction: 1.9–8.1× faster than DOC




**1.9–8.1×**

faster than decompress–operate–compress, at the tightest bound

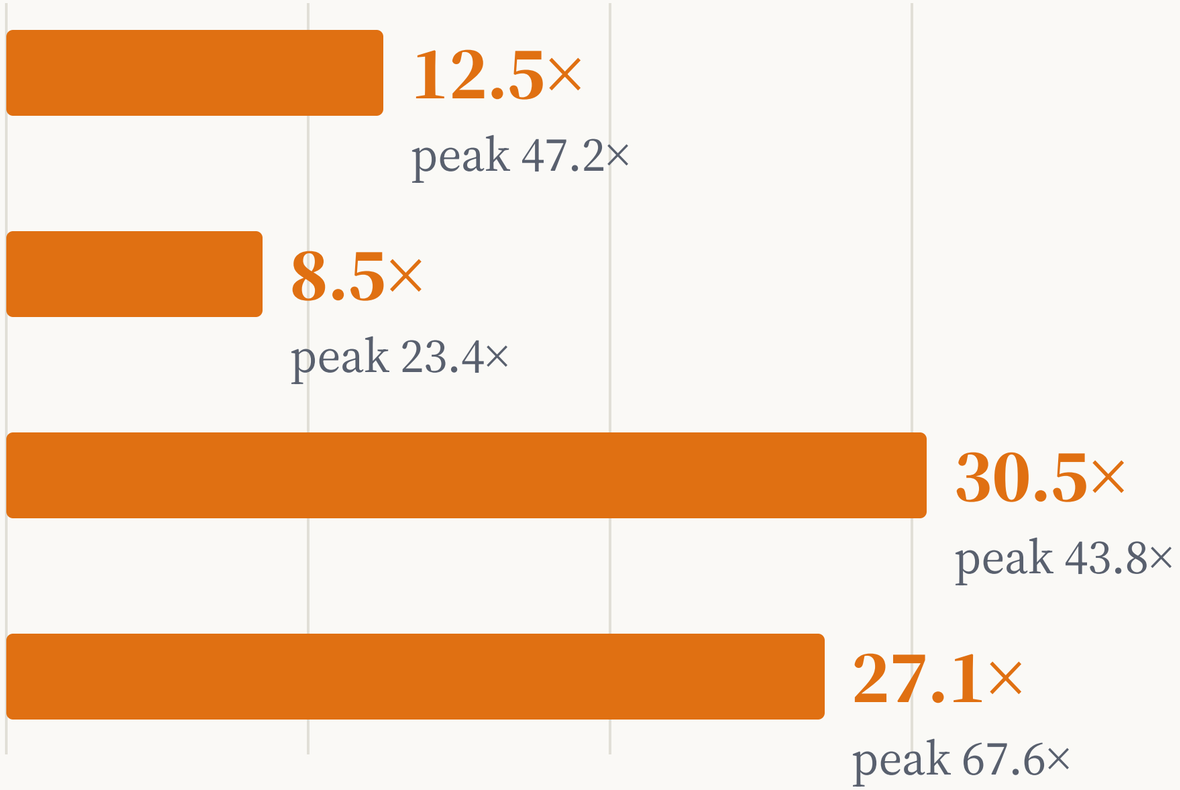
Largest on high-CR **JetIn** and **RTM**: mostly zero blocks

$B = 128$, relative $\varepsilon = 10^{-4}$, one socket (Fig. 7 in the paper)

EVALUATION

# Collectives run 8.5–30.5× faster than MPI

**Speedup over native MPI**, geometric mean




**vs hZCCL**

up to **1.28×**

up to **1.83×**

up to **1.61×**

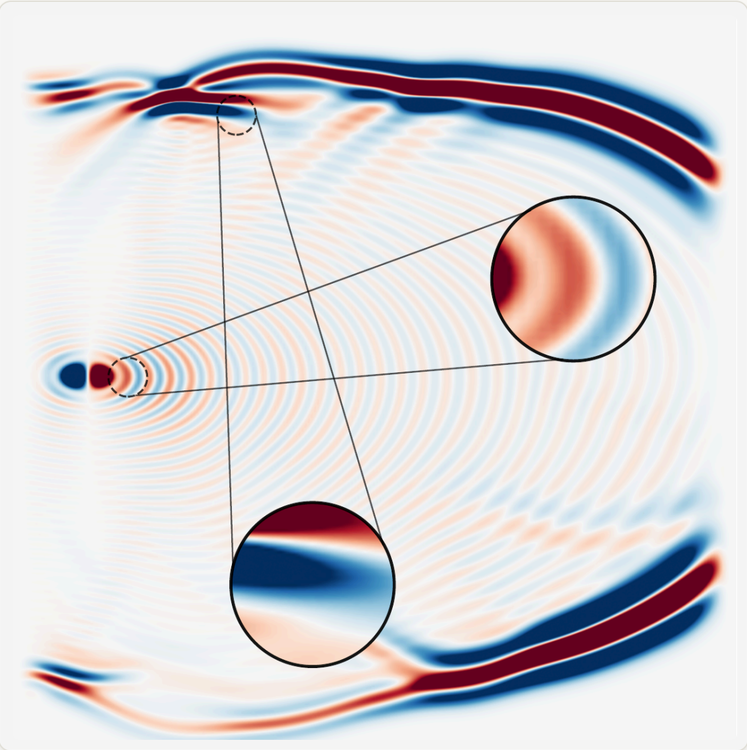
up to **2.04×***

*vs hZCCL's compressor

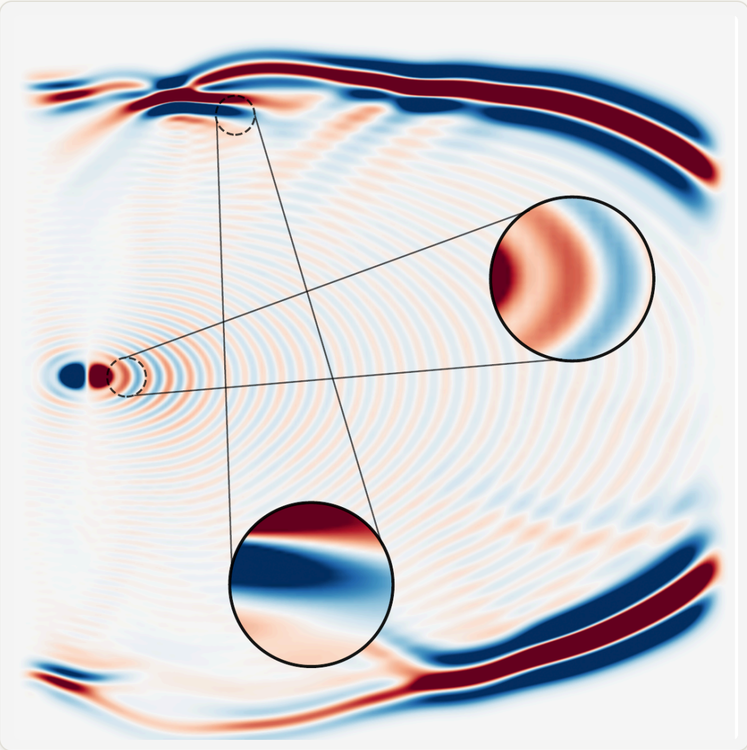
32 nodes, 3,584 cores · RTM data, 100–600 MiB · OFLE

EVALUATION

# Same seismic image, 39.5× faster



MPI · lossless



hZCCL2 · lossy

Lossy collectives change the data. Is the image still right?

## No visible difference

PSNR **118.6 dB**, NRMSE $1.18 \times 10^{-6}$ vs lossless, absolute $\varepsilon = 10^{-5}$

# 39.5×

faster than MPI

**1.21**× faster than hZCCL

CONCLUSION

# Three things to remember

**5.3×**
faster compression

**SIMD-native compression** keeps up with the memory; OFLE adds up to 37.6% ratio on outlier-heavy data.

**8.1×**
faster reduction

**One block-aligned format** lets collectives add compressed data on SIMD lanes, exactly, without decompressing.

**39.5×**
faster RTM stacking

**Four compressed collectives**, including the first CPU compressed Alltoall: 8.5–30.5× over MPI, same image.

**Next:** GPU-resident data · automatic choice between VecBA-SP and OFLE · larger scale.
Code, scripts and data are in the artifact.

# Thank you!

## Questions?

**hZCCL2:** Co-Designing Collective Communication with Vectorized Homomorphic Compression

Jiefeng Zhou · `jiefeng@usf.edu`

Backup: prior work · gain over hZCCL · typical vs peak · error growth · modes & fallback · portability & scaling · message sizes · Bcast & Scatter · AVX-512 registers · OFLE format · lossless backend

BACKUP

# Prior work: what each framework covers

| | Compressor design | | | Collective co-design | | | |
|---|---|---|---|---|---|---|---|
| | SIMD encoding | Outlier-aware FLE | Lossless backend | Homo-morphic | Bcast / Scatter | All-reduce | All-toall |
| SZx | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| fZ-light | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| C-Coll (IPDPS'24) | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✗ |
| hZCCL (SC'24) | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✗ |
| **hZCCL2 (ours)** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

Table I in the paper: the first CPU homomorphic framework with a compressed **Alltoall**.

BACKUP

# Where the speedup over hZCCL comes from

## Allreduce

### 100 MiB

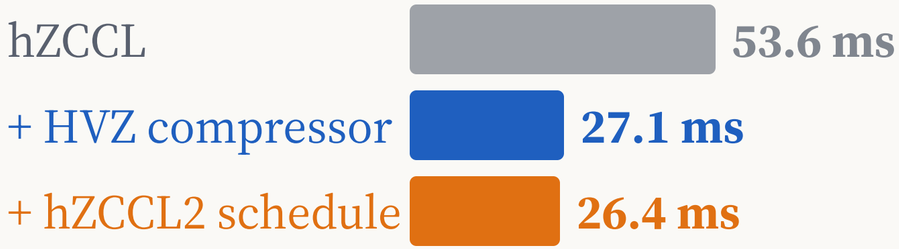



### 600 MiB




## Alltoall

### 100 MiB




### 600 MiB




Compressor + operator: **1.17–1.69×** (Allreduce), **1.41–2.07×** (Alltoall). Schedule: **1.00–1.07×** (Table XI).

BACKUP

# Typical vs peak speedup, and how we measured

| OFLE, 32 nodes, 100–600 MiB | Geo-mean | Peak |
|---|---|---|
| Bcast | **12.5×** | 47.2× |
| Scatter | **8.5×** | 23.4× |
| Allreduce | **30.5×** | 43.8× |
| Alltoall | **27.1×** | 67.6× |

Same allocation and MPICH 4.3.2 for every method; latency = min over 10 runs of the slowest rank, because native MPI varies a lot between runs.

BACKUP

# Error accumulation in reductions

**Exact on integer codes**

$$|\hat{s} - s| = \left| \sum_{i=1}^{P} e_i \right| \le P\,\varepsilon$$

deterministic, one reduction over $P$ ranks

95.44% within $\pm 2\sqrt{P}\,\sigma$

independent, centred errors

- The bound covers **one** reduction; repeated ones need their own budget.
- Strict tolerance: set $\varepsilon = \mathrm{tol}/P$.
- Skylake Allreduce: 92.98 → 62.87 dB from 4 to 128 nodes, **same for hZCCL and hZCCL2**.
- RTM stacking: PSNR 118.6 dB.

BACKUP

# Which mode, and when we fall back to MPI

## VecBA-SP

few threads, throughput-bound: 1-thread stacking 12.9× vs 10.7× over MPI

## OFLE

outlier-heavy, wire-bound: Hurricane +37.6% ratio at $B = 256$

## Native MPI

sampled ratio below the cutoff on any rank, or small messages

| | ratio | | compress GB/s | |
|---|---|---|---|---|
| | **VecBA-SP** | **OFLE** | **VecBA-SP** | **OFLE** |
| Hurricane | 9.2 | **11.8 (+28%)** | 59 | 52 |
| HACC | 3.9 | **4.4 (+13%)** | 120 | 63 |
| RTM | 178 | **190 (+7%)** | 122 | 125 |

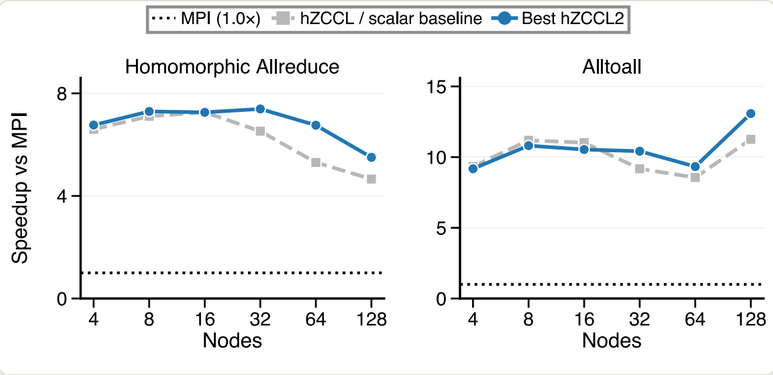
Table IX, $B = 128$, relative $\varepsilon = 10^{-4}$. The fallback checks compressibility, not accuracy; automatic mode choice is future work.

BACKUP

# Portability, and scaling to 128 nodes

| | HACC | | NYX | | RTM | |
|---|---|---|---|---|---|---|
| **GB/s, 112 threads** | **C** | **D** | **C** | **D** | **C** | **D** |
| scalar fused (fZ-light) | 33.5 | 58.0 | 58.5 | 132.6 | 100.4 | 350.3 |
| block-aligned, scalar | 116.8 | 66.7 | 107.1 | 180.3 | 120.6 | 414.5 |
| SIMD staged | 112.5 | 179.3 | 108.8 | 215.7 | **125.1** | 417.5 |
| SIMD fused (VecBA-SP) | **119.6** | **208.0** | **109.8** | **306.8** | 122.2 | **424.8** |

Table VIII: the layout alone gives **3.5×** on HACC. Format, outlier encoding, reduction and scheduling are ISA-independent (AVX-512, AVX2, scalar); ratio does not depend on SIMD width. Intel only.

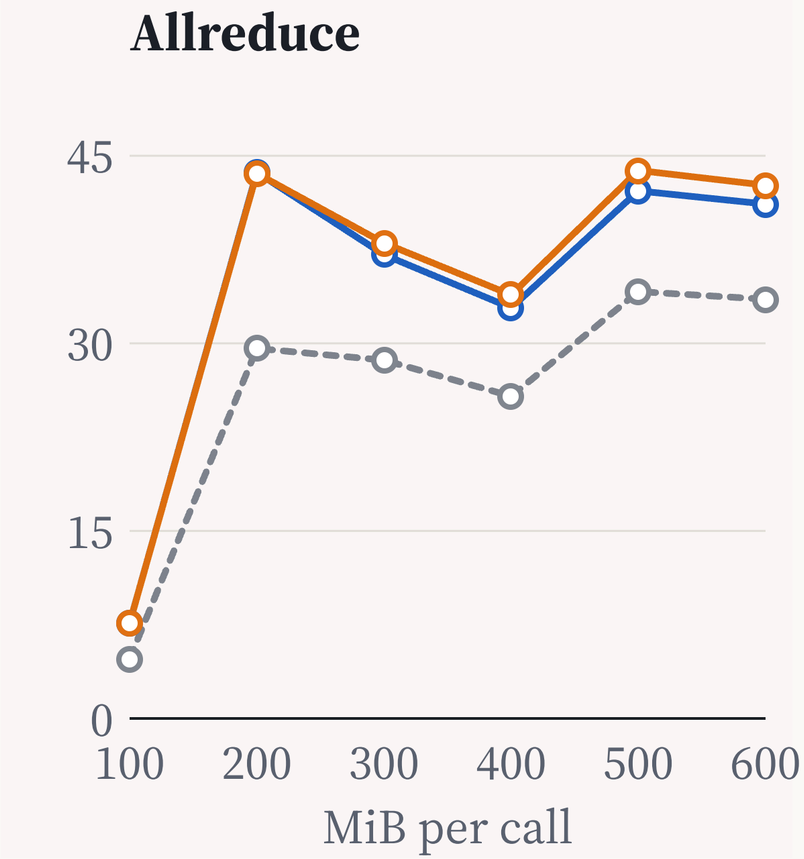


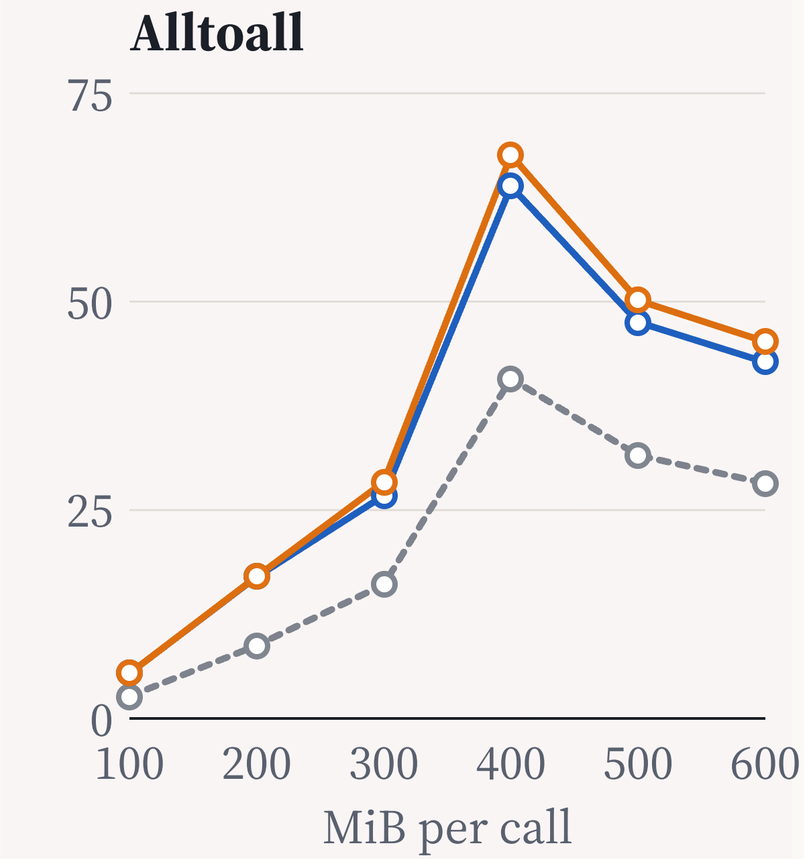

## Skylake, DDR4, 4–128 nodes (Fig. 9 in the paper)

- **Allreduce** 1.13–1.27× over hZCCL at 32–128 nodes
- **Alltoall** 1.09–1.16× over the scalar baseline
- **Bcast** near parity; hZCCL faster for **Scatter**

BACKUP

# The speedups hold across message sizes




**67.6×**

peak, Alltoall · 43.8× Allreduce (Fig. 8)

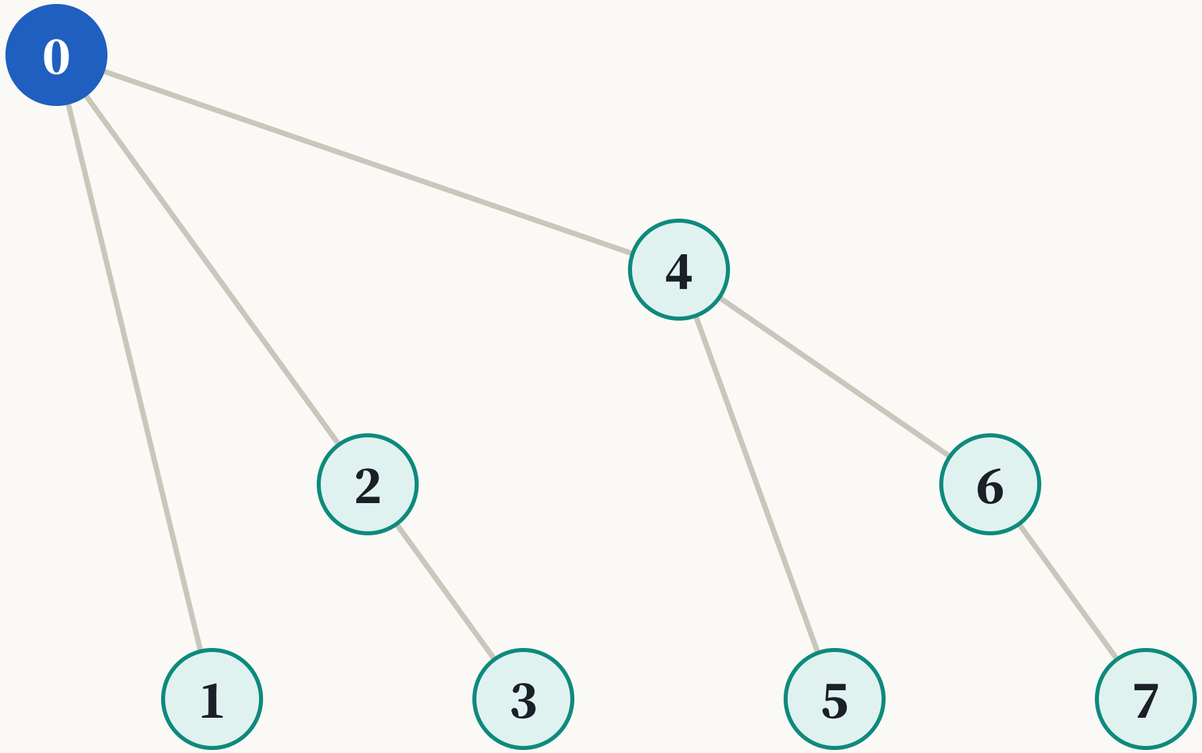
OFLE wins at large messages: **fewer bytes** cut wire time

1 thread: more mixed (see paper)

--- hZCCL — VecBA-SP — OFLE × over MPI, 112 threads

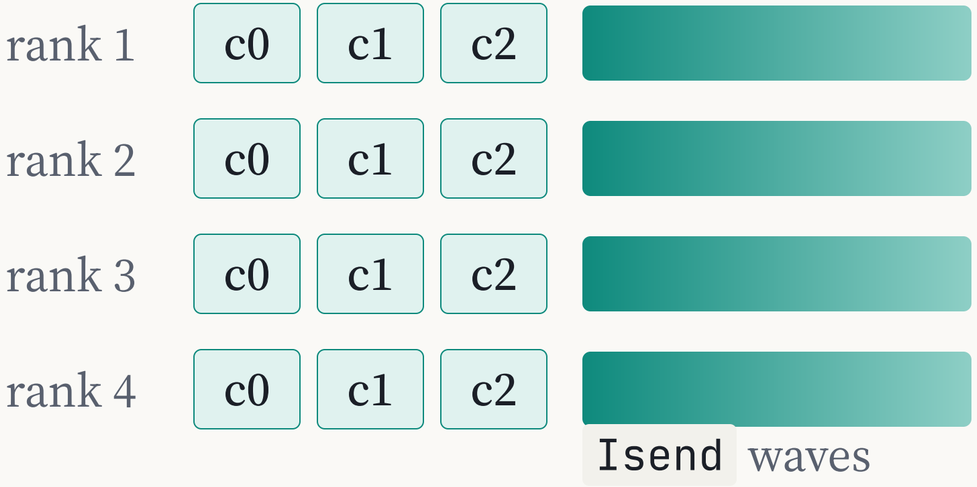
BACKUP

# Bcast and Scatter: the transport path




**Bcast**: compress once at the root, send down a binomial tree; inner ranks forward while decoding.

**Scatter**: parallel packing at the root




`collapse(2)` : all threads compress at once

> Bcast, Scatter, Alltoall **transport** compressed data; only Allreduce **computes** on it. All fall back to native MPI when compression cannot pay.

BACKUP

# VecBA-SP inside the AVX-512 registers

| lane | 0 | 1 | 2 | 3 | … | 13 | 14 | 15 | |
|---|---|---|---|---|---|---|---|---|---|
| v_data | 0.789 | 0.790 | 0.792 | 0.789 | … | 0.814 | 0.821 | 0.832 | vector 4 of 4 |
| v_quant | 789 | 790 | 791 | 789 | … | 813 | 821 | 831 | |
| v_prev | 793 | 789 | 790 | 791 | … | 810 | 813 | 821 | prior **793** |
| v_diff | −4 | +1 | +1 | −2 | … | +3 | +8 | +10 | |
| v_abs | 4 | 1 | 1 | 2 | … | 3 | 8 | 10 | $w = 32 - \mathrm{clz}(11) = 4$ |
| sign | 1 | 0 | 0 | 1 | … | 0 | 0 | 0 | |
| v_max | 11 | 11 | 11 | 9 | … | 10 | 11 | 10 | |

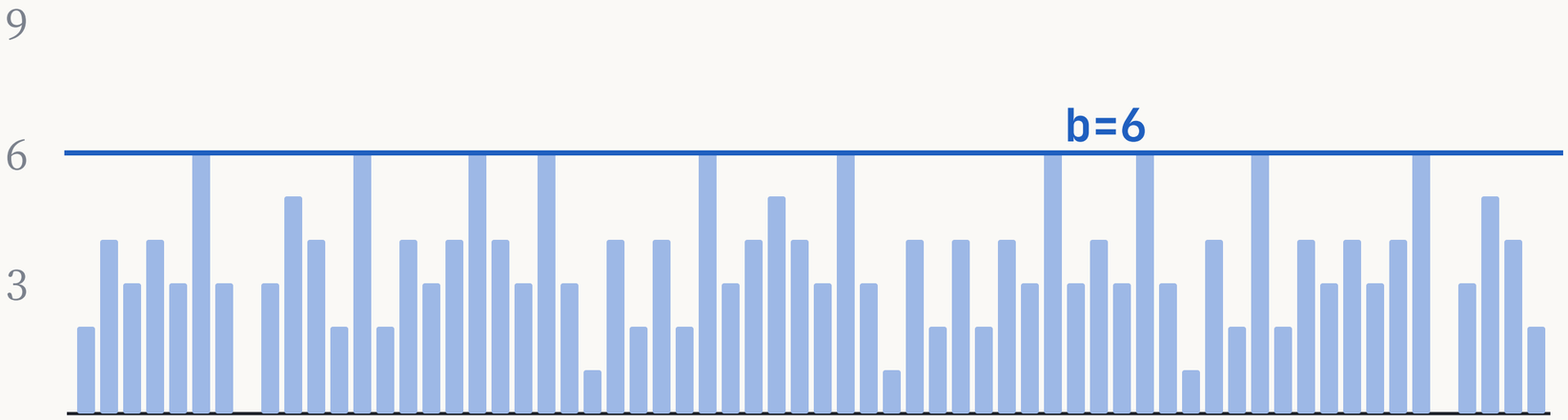



= **41 B** instead of 256 B

BACKUP

# OFLE: threshold search and block format

bits needed by each of the 64 values




## Threshold modes

**dynamic** $t = 0$
best of 4 (default)

**fixed** $t = 1..31$
REL or ABS

**bypass** $t \geq 32$
normal blocks only

| header 2 B | pos. 2 B | signs 8 B | main: 64 × 6 bits = 48 B | high 1 B | = 61 B |
|---|---|---|---|---|---|

header **1** **00110** 01 0 **0000010**: outlier flag, width 6, 3 extra bits, 2 outliers
vs **81 B** with FLE · positions: index list when sparse, bitmap when dense
normal blocks keep the 1-byte header, shared with VecBA-SP

BACKUP

# Optional lossless backend: storage, not the wire

| | Hurricane | | NYX | | RTM | |
|---|---|---|---|---|---|---|
| **OFLE + backend** | **CR** | **GB/s** | **CR** | **GB/s** | **CR** | **GB/s** |
| none (raw) | 11.7 | **47.1** | 20.8 | **59.4** | 185 | **62.1** |
| + zlib | **23.9** | 0.2 | **131** | 0.4 | **8946** | 9.9 |
| + Blosc | 16.5 | 4.1 | 65.5 | 7.9 | 4244 | 27.9 |
| + ZFP (reversible) | 15.7 | 1.2 | 44.3 | 2.0 | 1508 | 20.6 |

Table X: encode GB/s, $B = 128$, relative $\varepsilon = 10^{-4}$. A lossless stage wraps FLE output for storage and checkpoints; collectives use raw FLE.