/* ACADEMY SOFTWARE FOUNDATION

open source days '26

/* ACADEMY SOFTWARE FOUNDATION

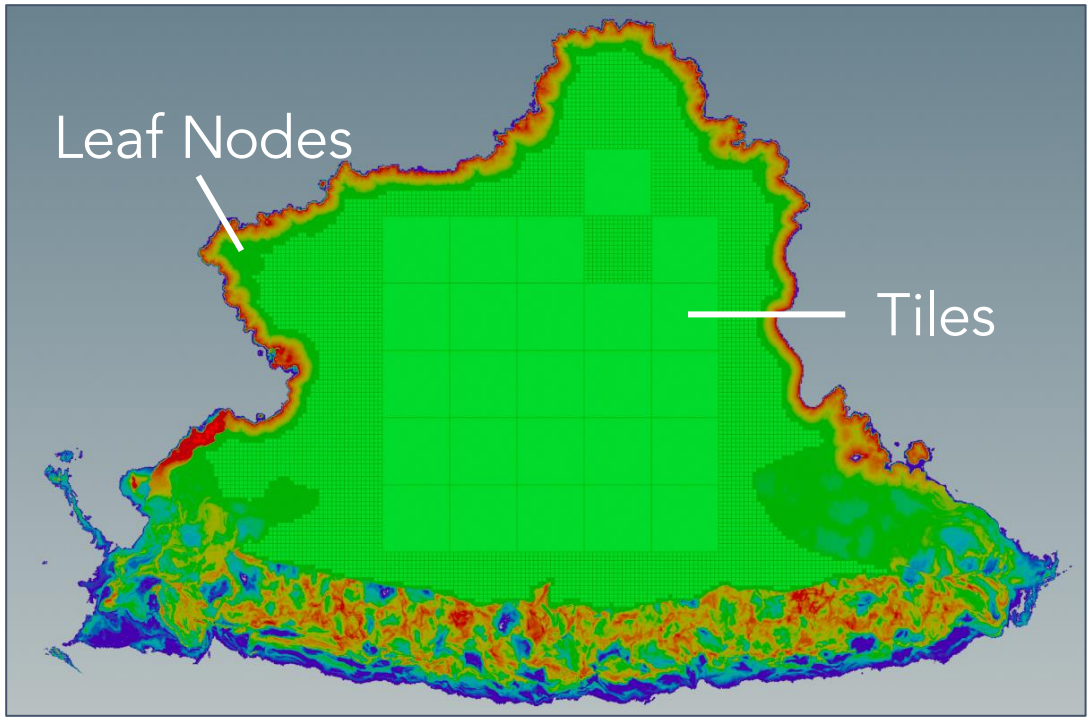
open Source days '26

A File Format for the Next Decade

Dan Bailey, ILM

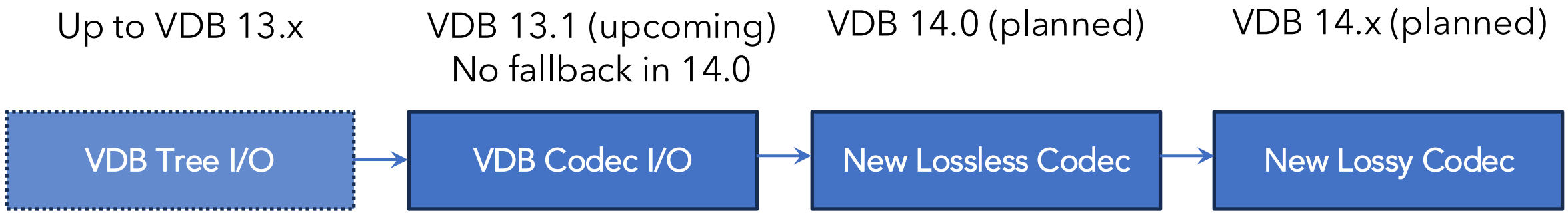
VDB

- Sparse, hierarchical volumetric data structure, tools and I/O
- Open-sourced in 2012, first ASWF project in 2018

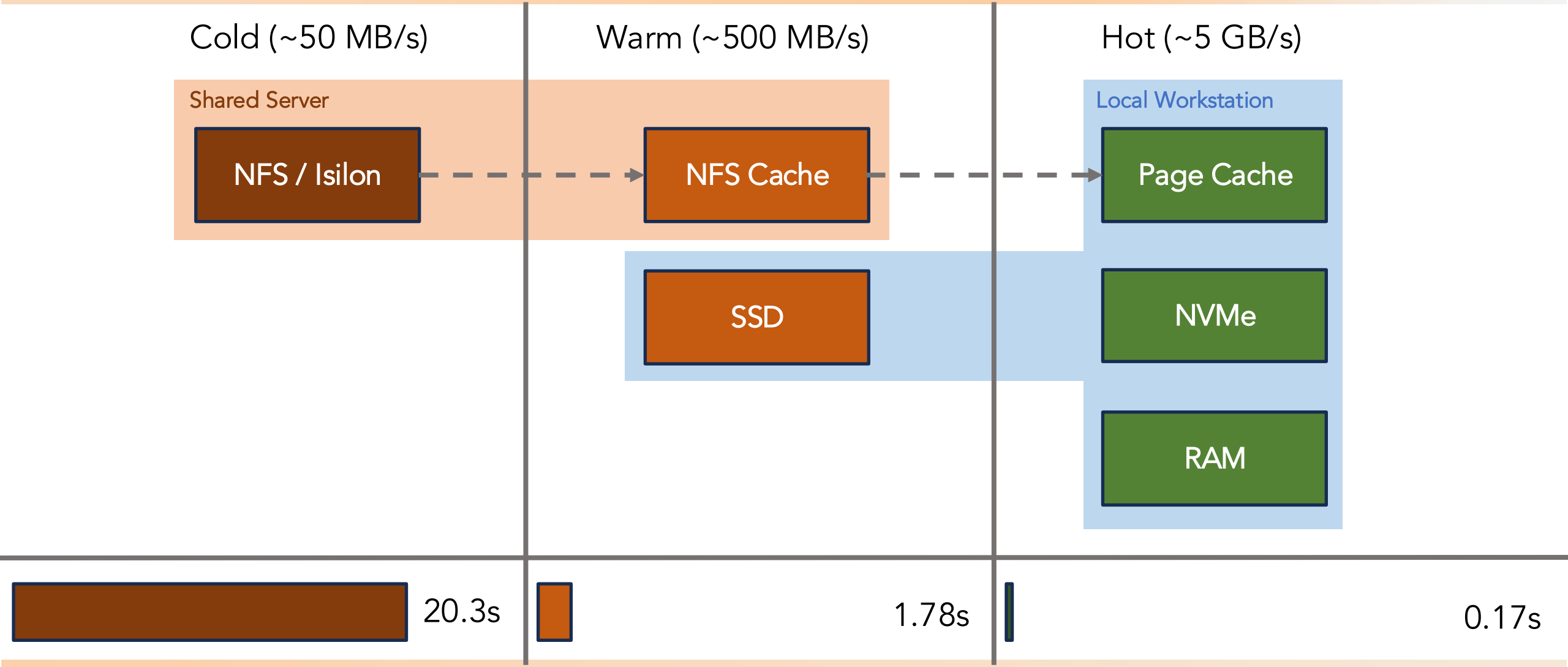


VDB I/O

- Existing I/O implementation has a number of limitations
- File format fallen behind, hard to explore new techniques



I/O Hardware and Caching

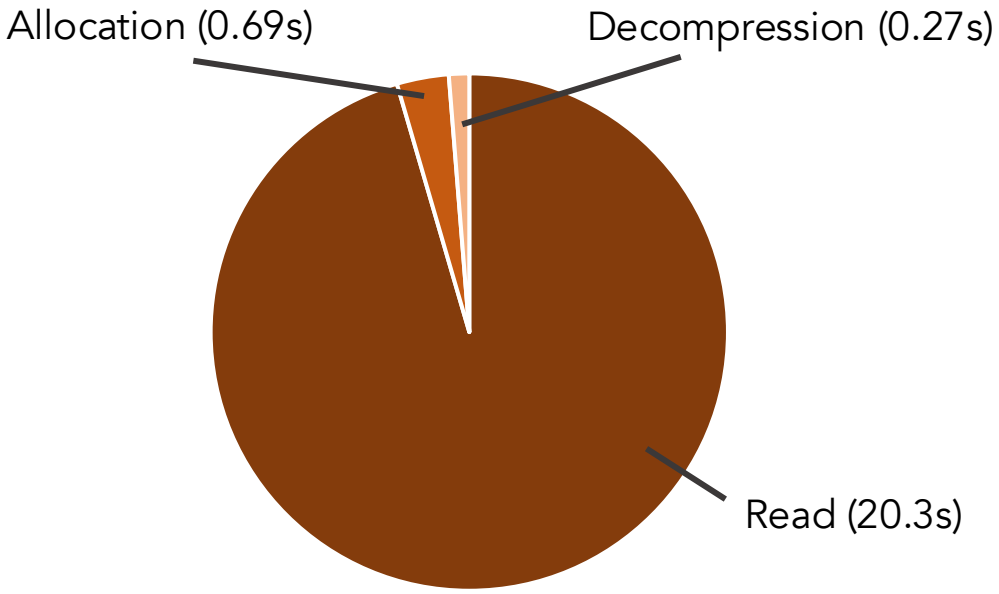


I/O-Bound vs Compute-Bound

Cold (~50 MB/s)

Shared Server

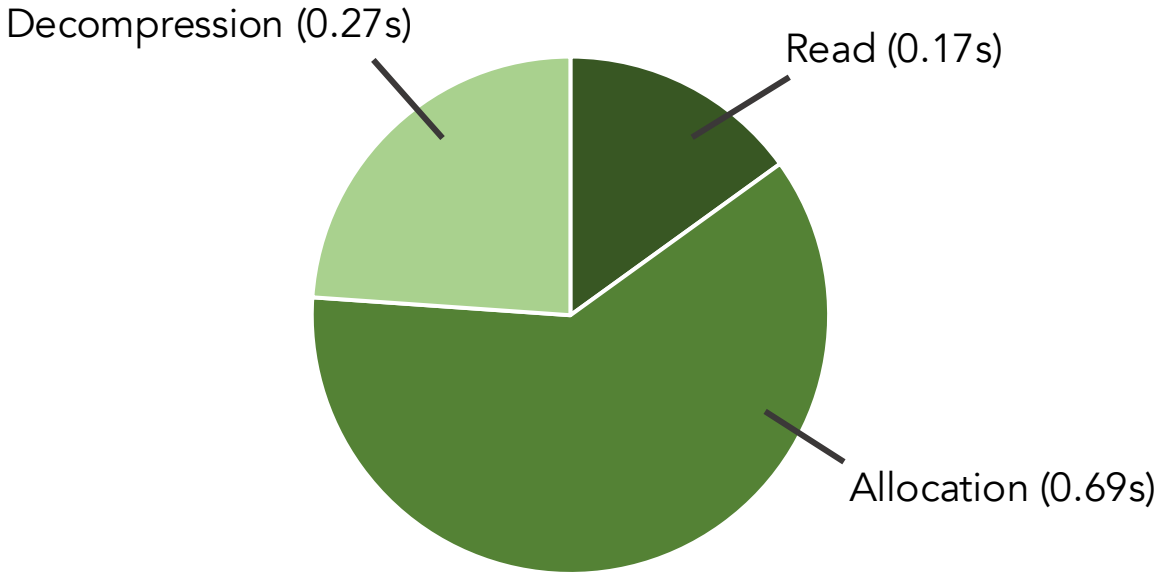
NFS / Isilon



Hot (~5 GB/s)

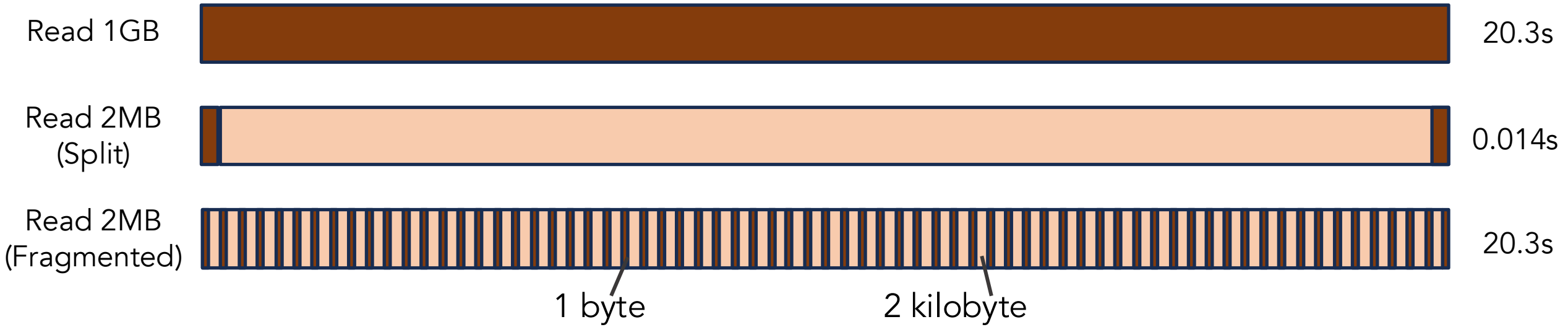
Local Workstation

NVMe

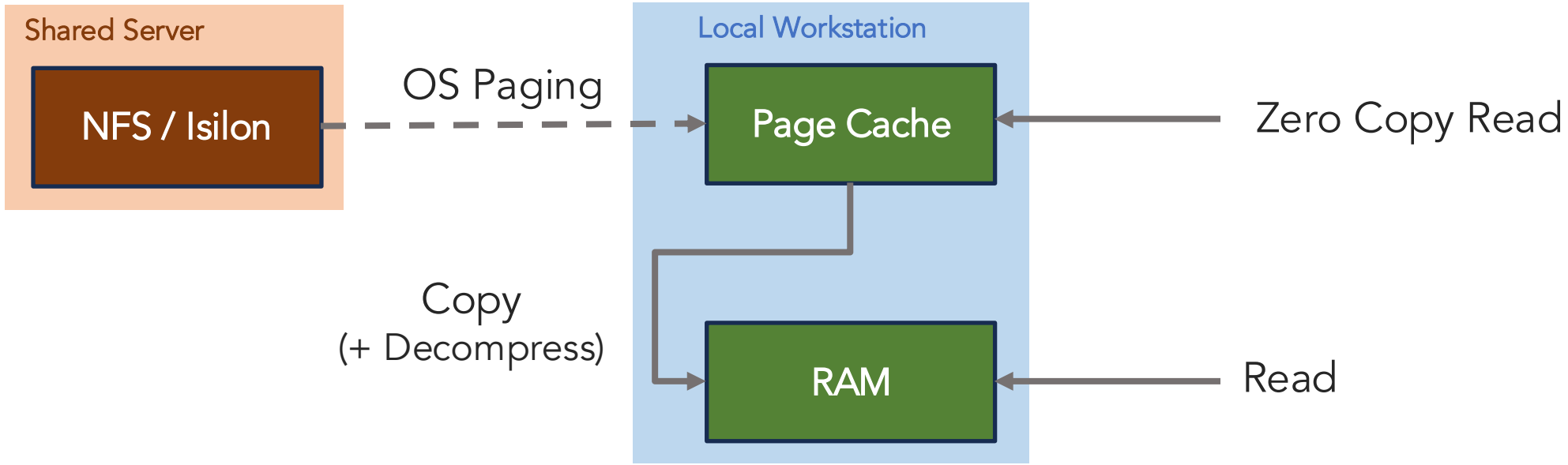


OS Paging and Prefetching

- Typical Size of an OS page is 4KB
- Retrieving a byte loads the whole page into the page cache
- Seeking does not trigger page faults
- Sequential reading is optimal for OS readahead



Memory Mapping



History of Open Source Projects

- Consider decisions made by other open-source projects:

- What trade offs did they make and why?

OpenEXR

- Started in 1990s, first open-source VFX project in 2003
- Lots of studio storage is EXR data
- Choice to write by scanline or by tile
- Codecs used to manage disk space cost:
  - RLE (lossless) – 2000
  - ZIP (lossless) – 2000
  - PIZ (lossless) – 2001
  - PIZ24 (perceptually lossless) – 2004
  - B44 + B44A (lossy) – 2006
  - DWAA + DWAB (lossy) – 2014
  - HTJ2K (lossless) - 2025 (lossy planned for 2026)

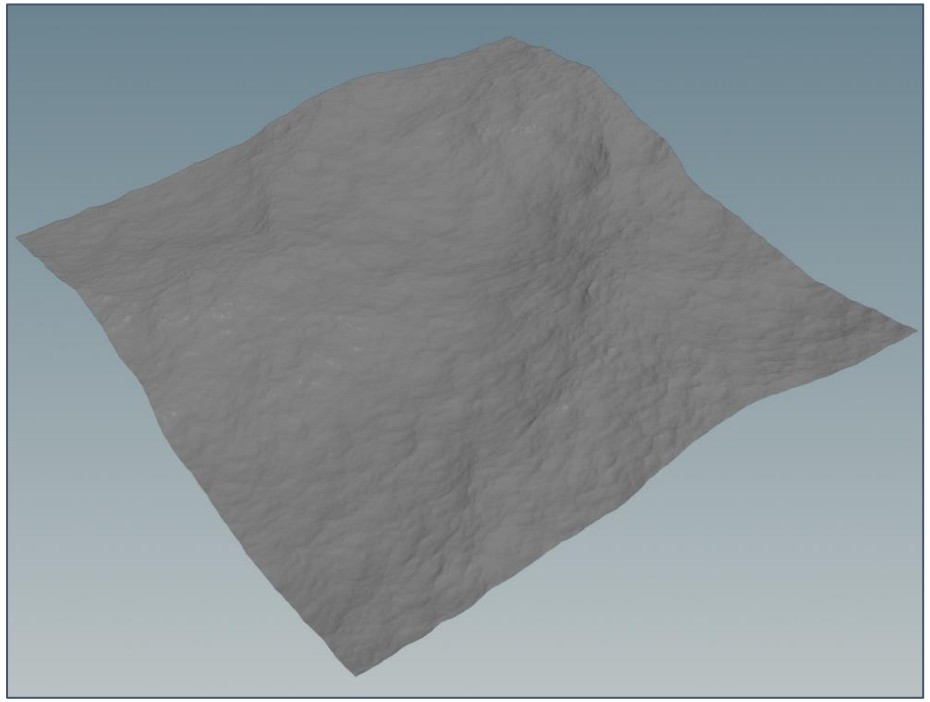
Alembic

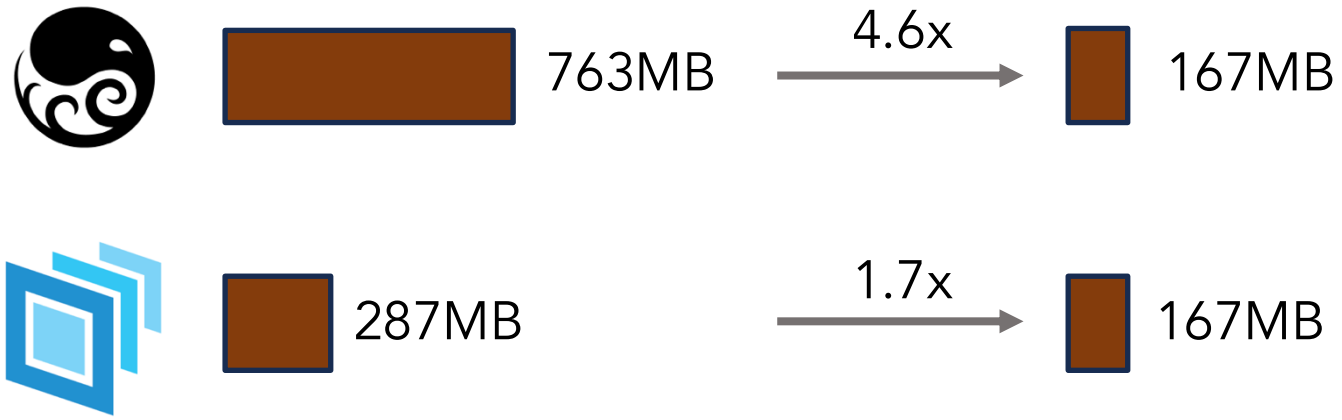
- Initial implementation was HDF5, but performance was slow
- Introduced Ogawa to resolve read locking
- Very efficient for instancing and sequence archives
- Uses zero copy memory mapping
- No data compression

USD

- USDC is binary (c stands for “crate” not “compressed”)
- USDZ is an uncompressed zip archive
- Bulk float data => zero copy memory mapping
- Integer data and “float as integer” data => LZ4 compression
- Prefetching can be disabled for better random access perf

Re-compression Test



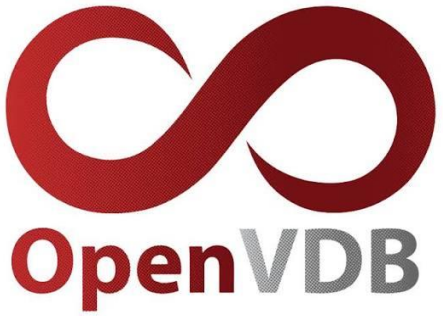


25 million polygons compressed with:
Byte Shuffle + LZMA

VDB (< 14)

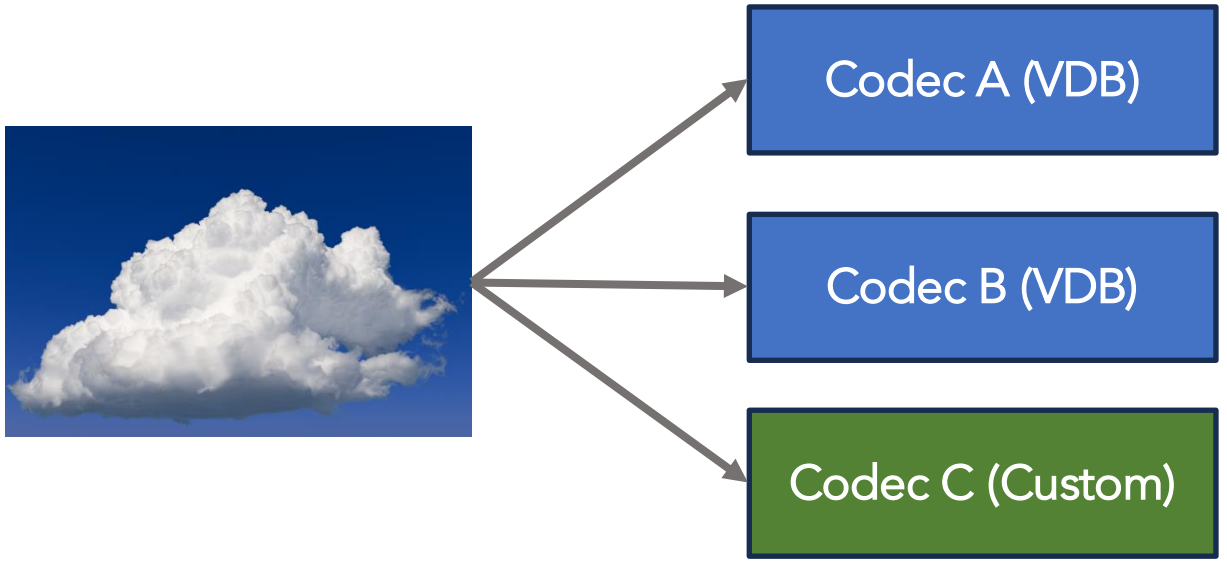
- Object-oriented I/O design (each node handles own I/O)
- Memory mapping, but used for delay loading *not* zero copy
- Slower performance due to use of atomics

- Blosc+LZ4 introduced with VDB 3.0 (2014)
  - Improved compression ratio (vs no compression)
  - Faster read+write performance (vs ZIP compression)



New VDB I/O Codec Framework

- Removed Tree I/O
- Removed Delay Loading + Memory Mapping
- VDBs can be written and read by multiple different codecs



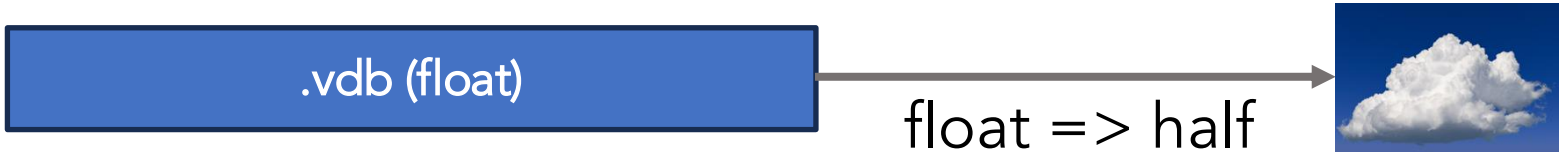
New VDB I/O Codec Framework

- Interleaving VDB grids for improved performance

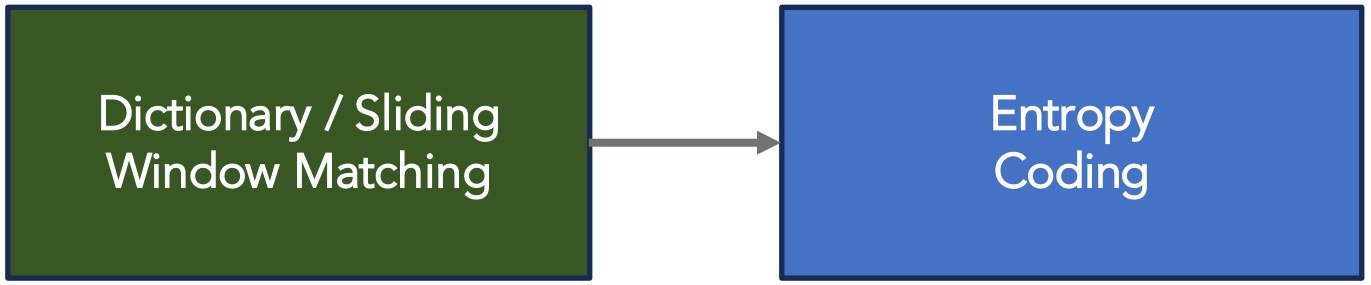
- Appending VDB grids to .vdb files

- Non-seekable streams

- On-the-fly grid conversion (including Half Grids)

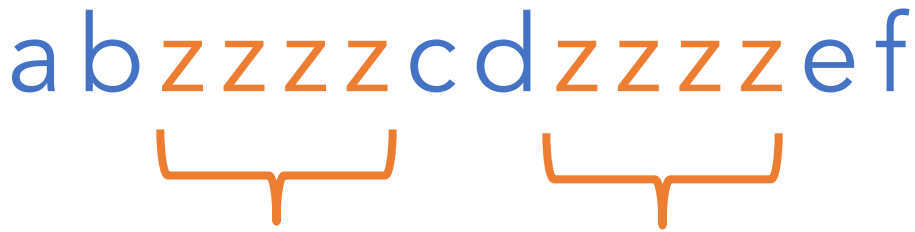


Lossless Compression



Dictionary / Sliding Window Matching – LZ77

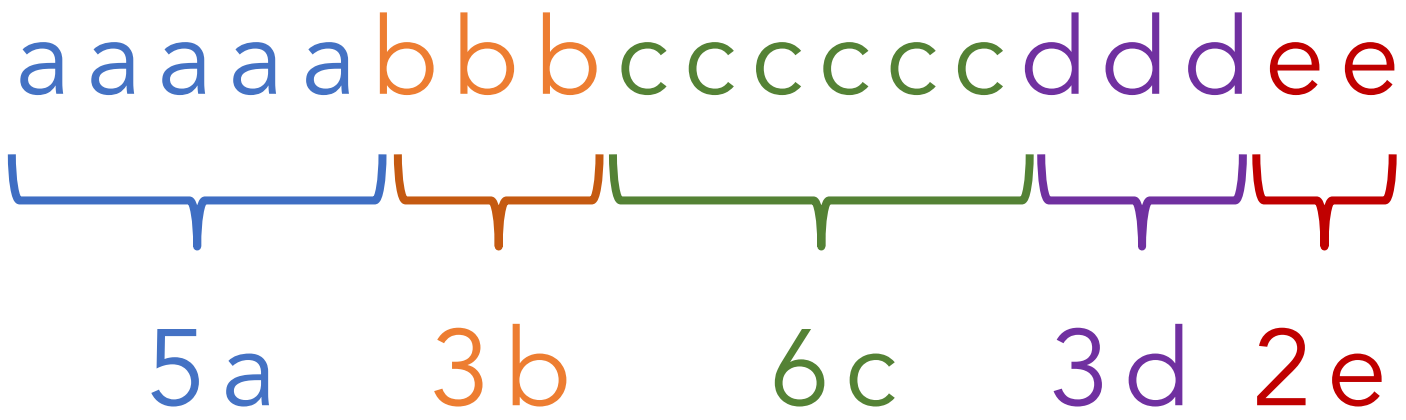
- Dominated by algorithms derived from LZ77 invented in 1977



(0,0,'a') (0,0,'b') (0,0,'z') (1,3,'c') (0,0,'d') (6,4,'e') (0,0,'f')

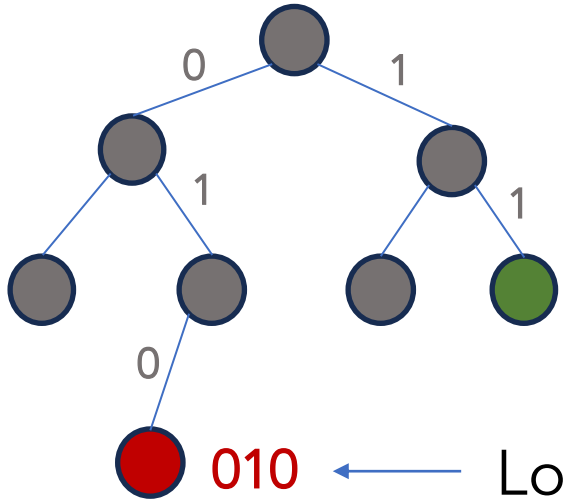
Entropy Coding – RLE

- Run Length Encoding used for analog television in 1967



Entropy Coding – Huffman

- Huffman Encoding invented back in 1952

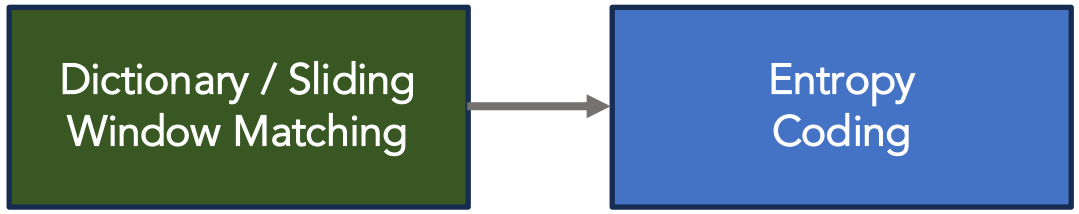


a a a a a b b b c c c c c c d d d e e

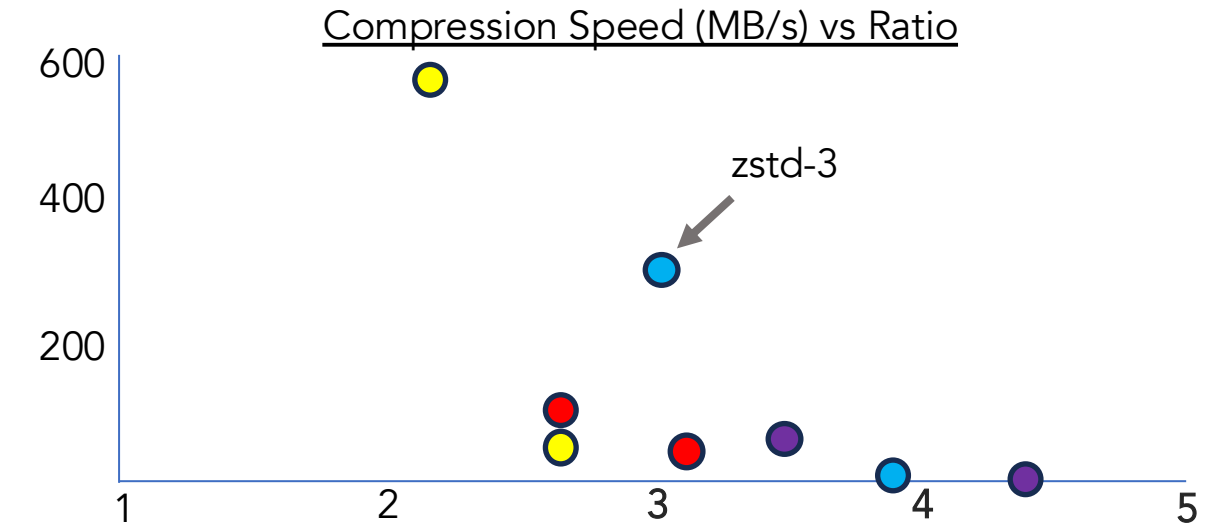
Shorter codes used to encode higher frequency data

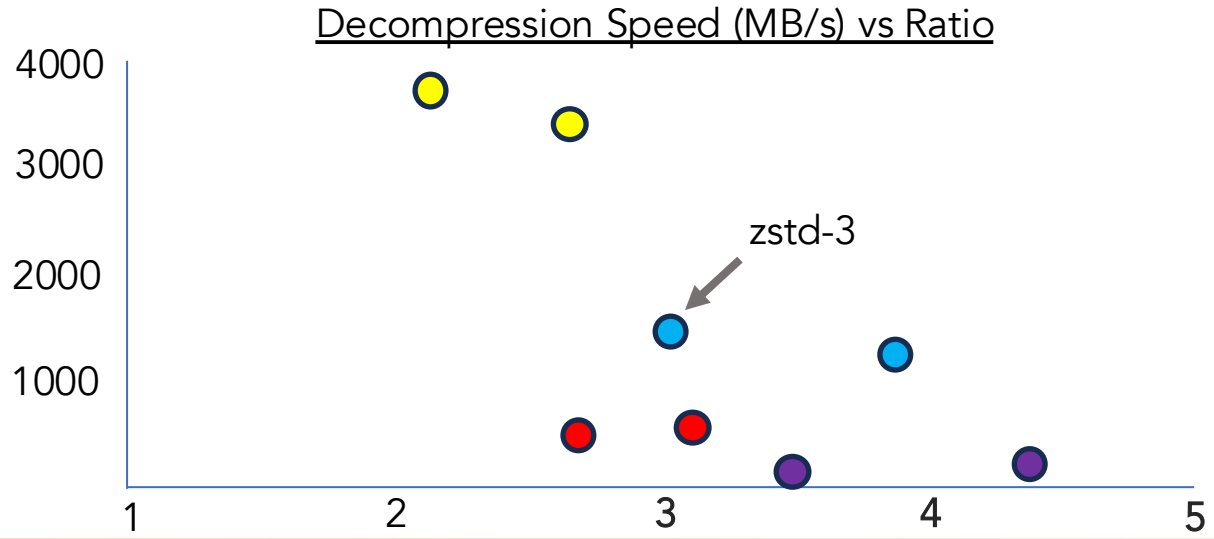
Longer codes used to encode lower frequency data

Lossless Compression Algorithms

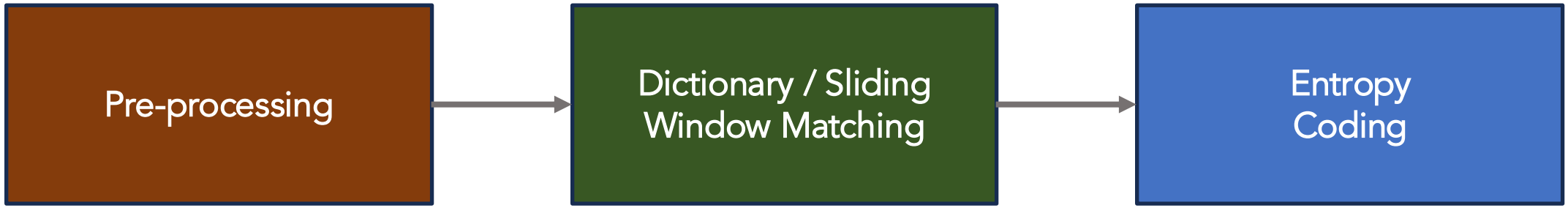


- ZIP/Deflate (1993) – LZ77 + Huffman
- LZMA (1998) – LZ77* + Range Coding
- LZ4 (2011) – LZ77
- ZStandard (2015) – LZ77 + tANS/Huffman





Lossless Compression



Pre-processing – Delta

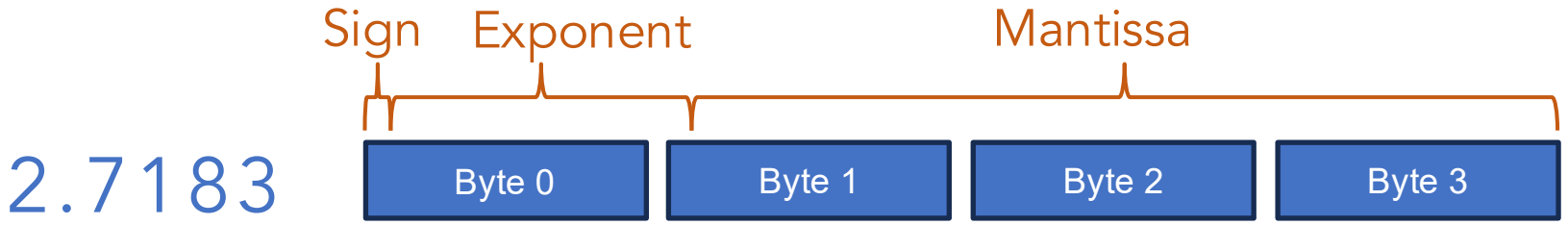
- A simple “delta” transformation improves compressibility

1 2 3 4 5 6 7 8 9

1 1 1 1 1 1 1 1 1

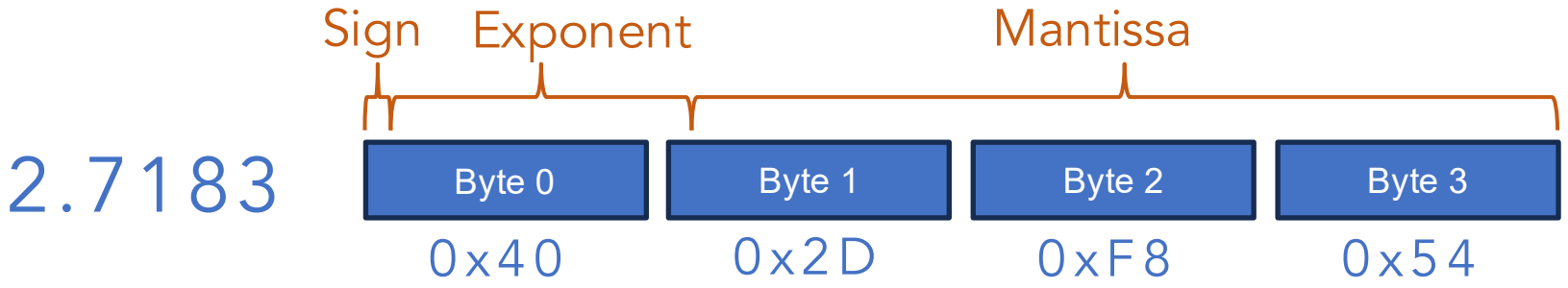
Pre-processing – Byte Shuffle

- Byte shuffling introduced in HDF5 for data science
- Re-implemented with SIMD as independent Blosc library



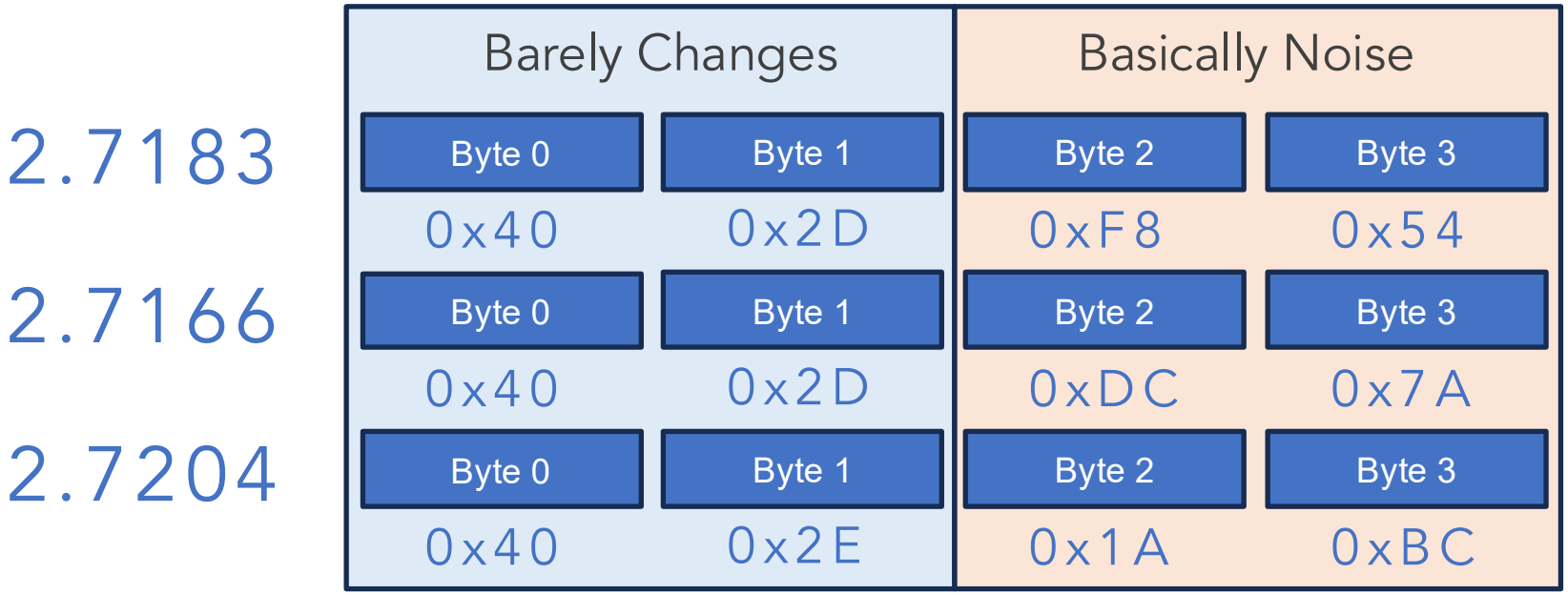
Pre-processing – Byte Shuffle

- Byte shuffling introduced in HDF5 for data science
- Re-implemented with SIMD as independent Blosc library

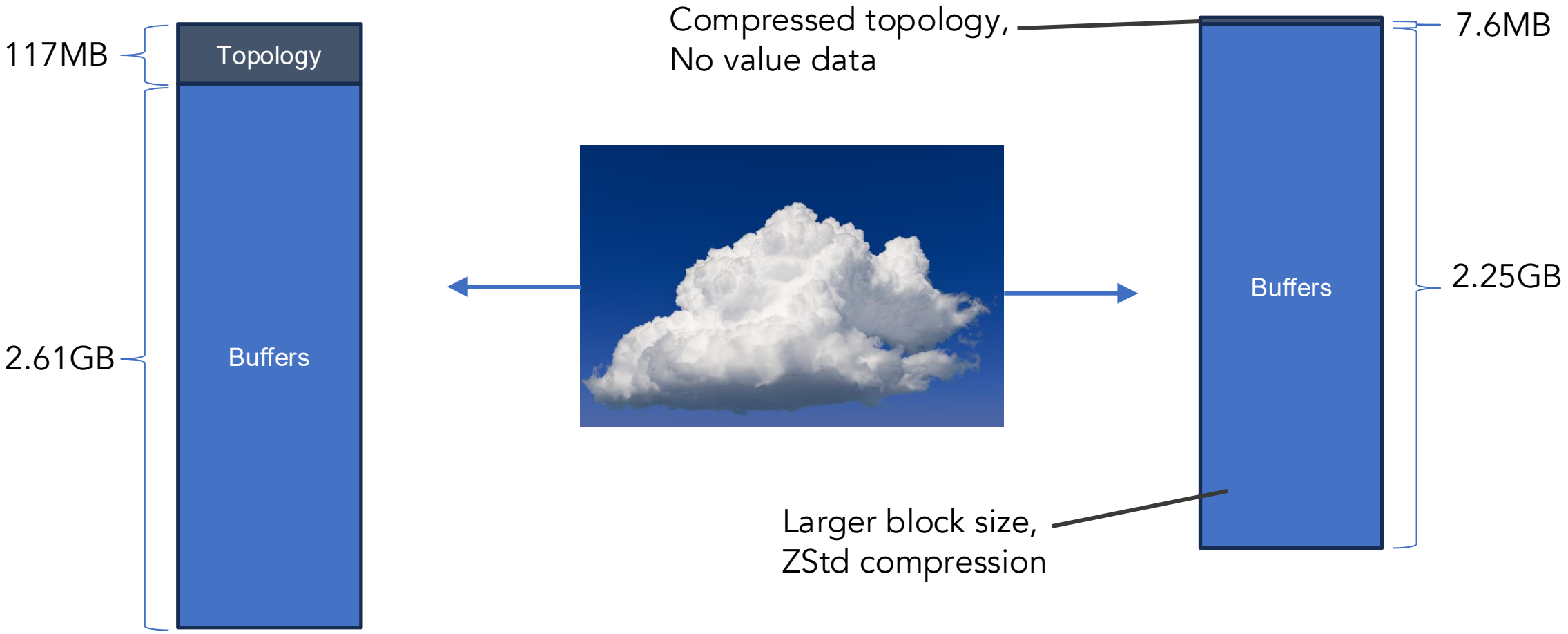


Pre-processing – Byte Shuffle

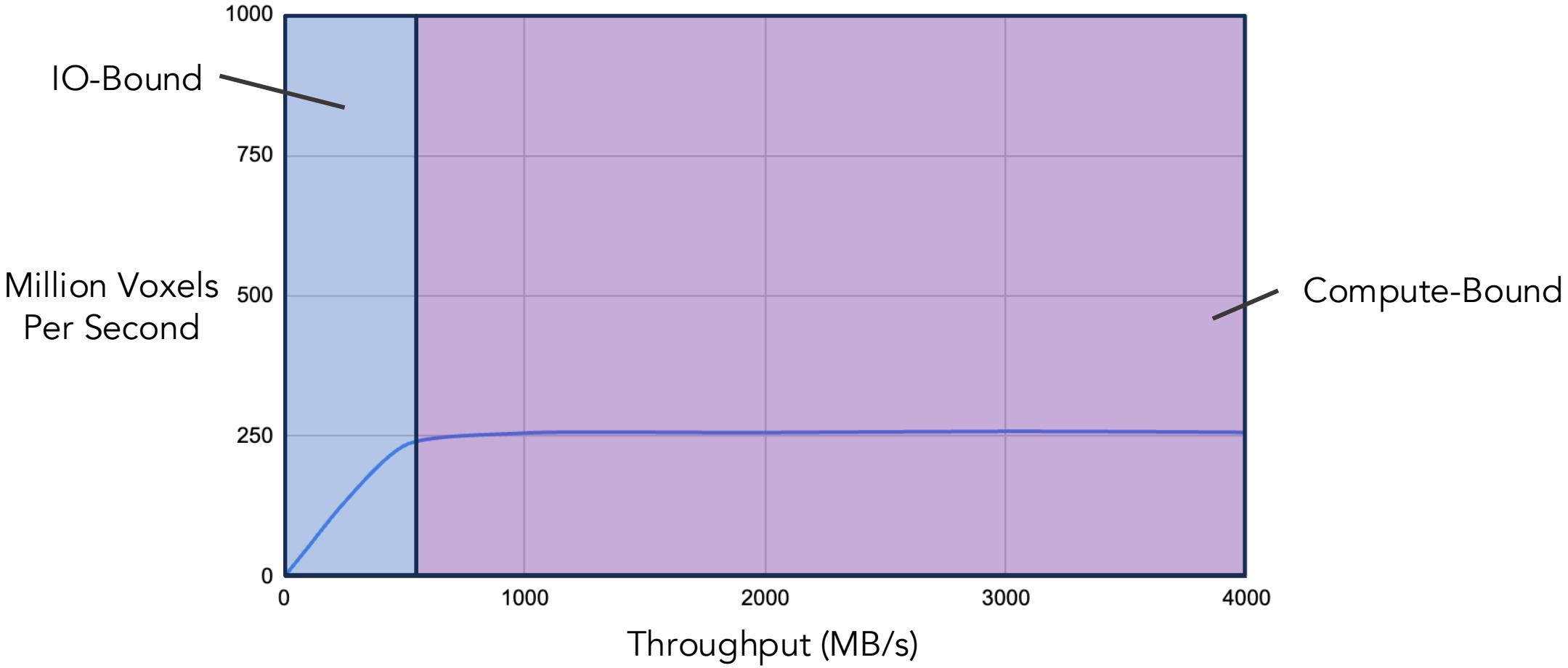
- Byte shuffling introduced in HDF5 for data science
- Re-implemented with SIMD as independent Blosc library



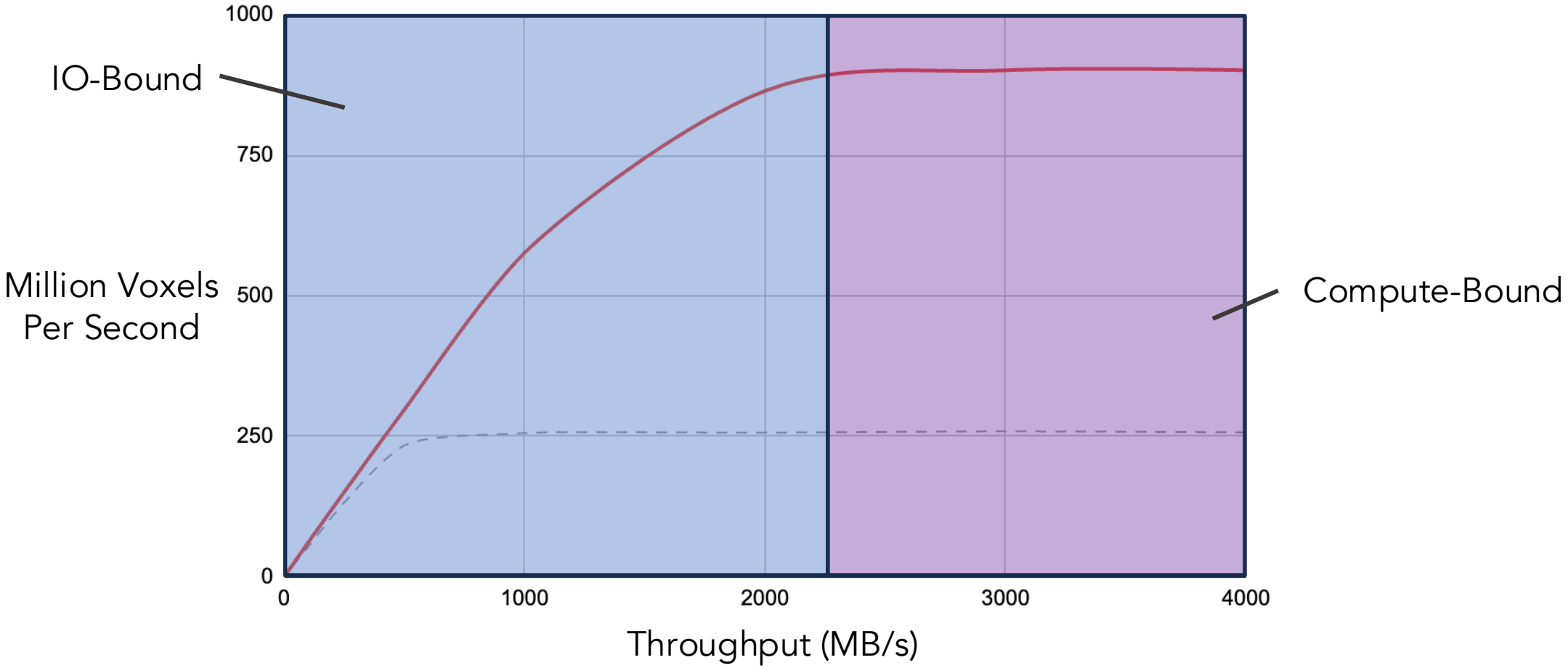
New VDB I/O Lossless Codec



Overlapping IO and Compute

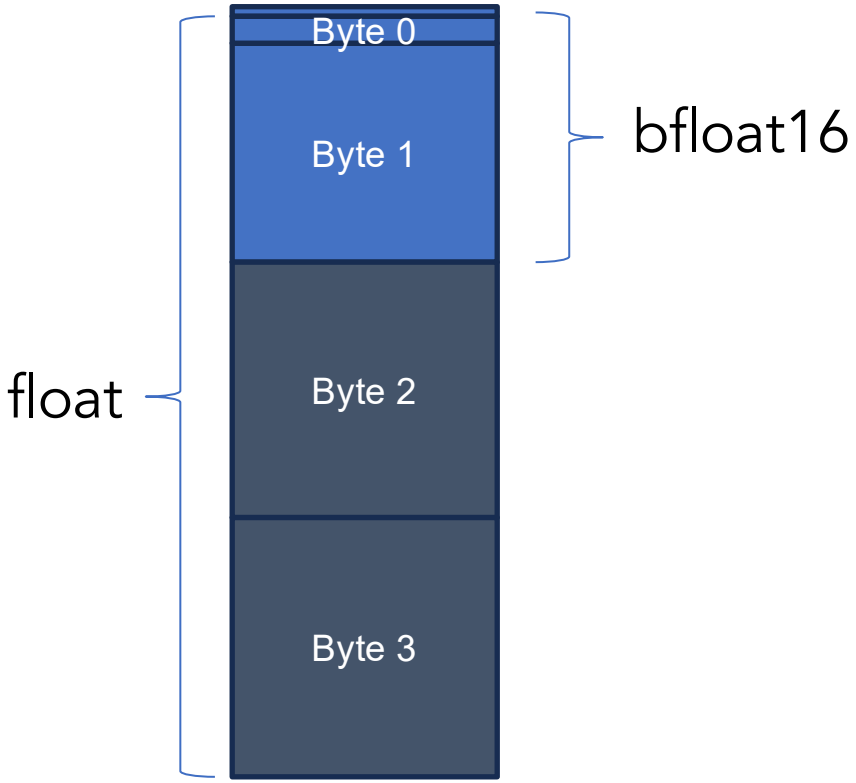


Overlapping IO and Compute

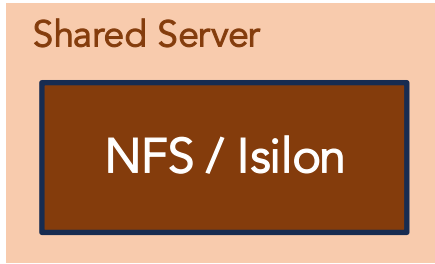


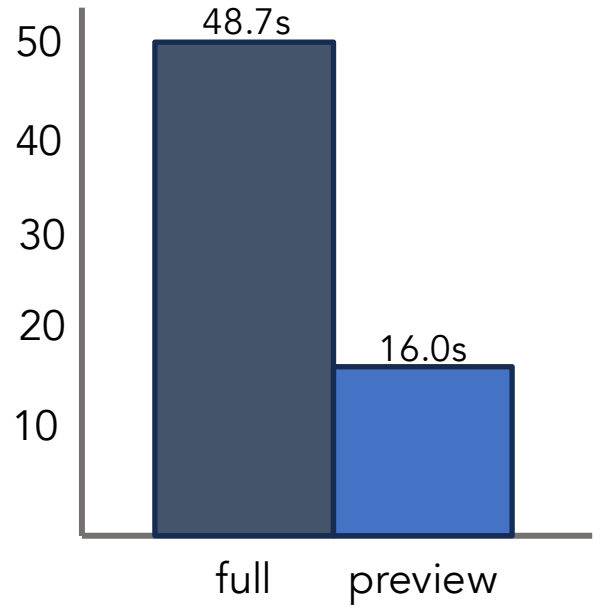
VDB I/O Lossless Preview

- Native byte shuffling for fast preview

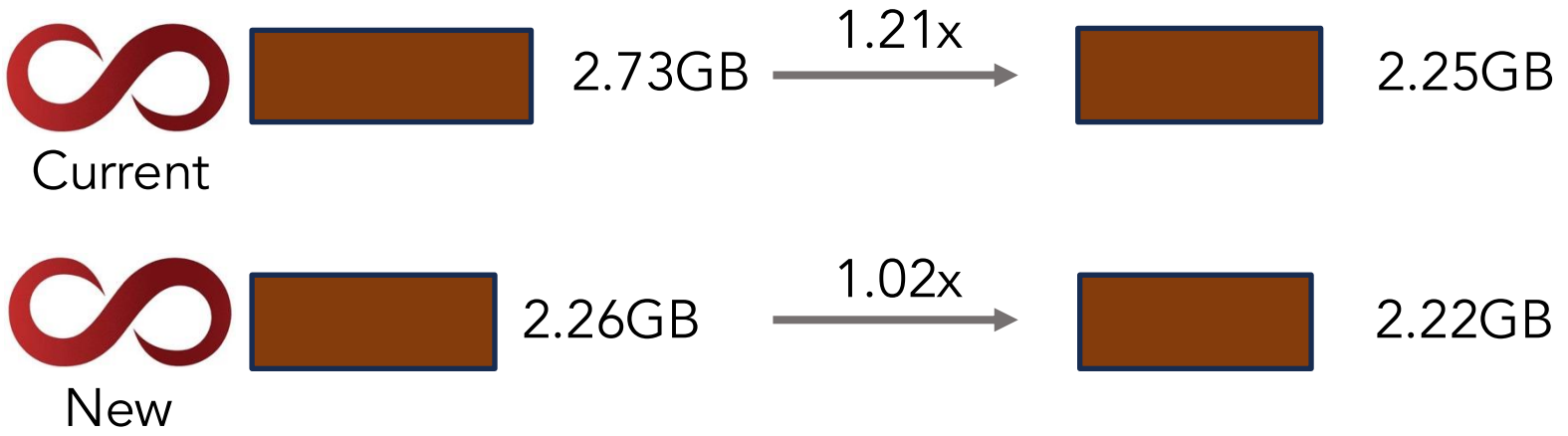


Cold (~50 MB/s)



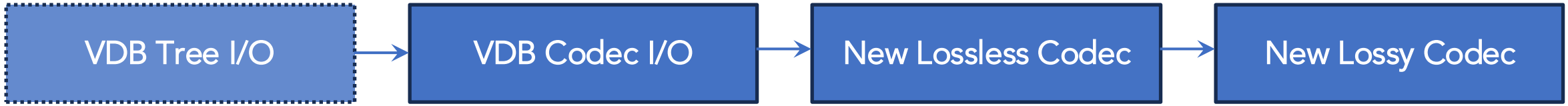


Re-compression Test



1.5 billion voxels re-compressed with LZMA

VDB I/O



Up to VDB 13.x

VDB 13.1 (upcoming)
No fallback in 14.0

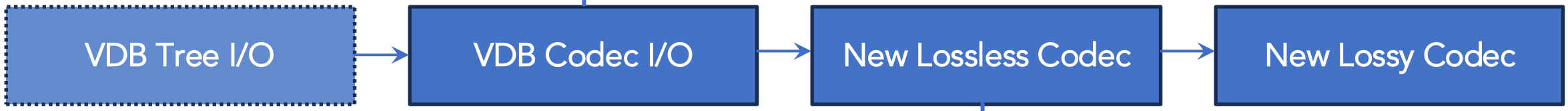
VDB 14.0 (planned)

VDB 14.x (planned)

VDB I/O

New backwards-compatible lossless codecs

- Can *write* VDBs that are readable from VDB 3.0+ (2014)
- Can *read* VDBs that were written in VDB 1.0+ (2012 / Hou 12.5



Up to VDB 13.x

VDB 13.1 (upcoming)
No fallback in 14.0

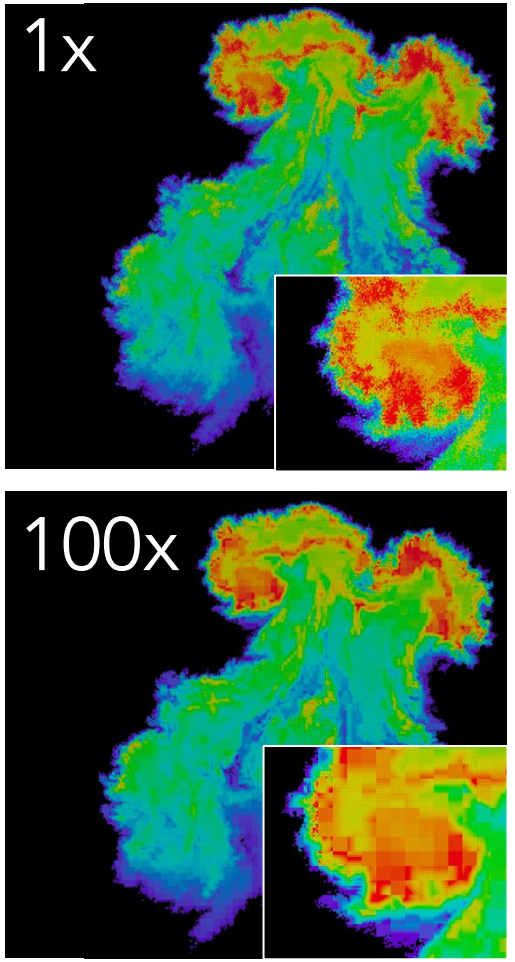
VDB 14.0 (planned)

VDB 14.x (planned)

- Can write VDBs that are only readable in VDB 14.0+
- Remain opt-in for the foreseeable future

Future Work

- Read new VDBs directly into NanoVDB
- Improved selective loading and Houdini tooling
- Lossy Compression Codec (DCT)
- Explore real-time, level sets, neural, etc



Conclusion

- ZStandard is best in class for fast lossless compression
- Floating point exponents are heavily compressible
- VFX I/O infrastructure is nearly always at the limit
- Write new VDB codecs instead of competing file formats

OpenVDB Birds of a Feather

/* ACADEMY SOFTWARE FOUNDATION

10am Monday 20th July in JW Marriot Ballroom 3

- SIMD Vectorisation
- NanoVDB Mutable Grids
- fVDB Updates
- Improvements to VDB Tool
- Q & A
- … and more