ASWF /* ACADEMY SOFTWARE FOUNDATION

Ken Museth, TSC Chairperson

February 18, 2026

Technical Steering Committee

- Voting Members:
  - Ken Museth, chair (NVIDIA)
  - Andre Pradhana (NVIDIA)
  - Jeff Lait (SideFX)
  - Dan Bailey (ILM)
  - Richard Jones (ILM)
  - Nick Avramoussis (Weta)
  - Gregory Hurst (UT)
  - Jonathan Swartz, fVDB maintainer (NVIDIA)
- Meets **every second Wednesday** at 11:00am US Pacific Time (calendar at https://lists.aswf.io/g/openvdb-dev/calendar)
  - Agendas and notes at https://github.com/AcademySoftwareFoundation/openvdb/tree/master/tsc/meetings

ASWF /* ACADEMY SOFTWARE FOUNDATION

Release Plan

- Two minor releases per year
- One major release per year

Version 12.0.1 - April 3, 2025

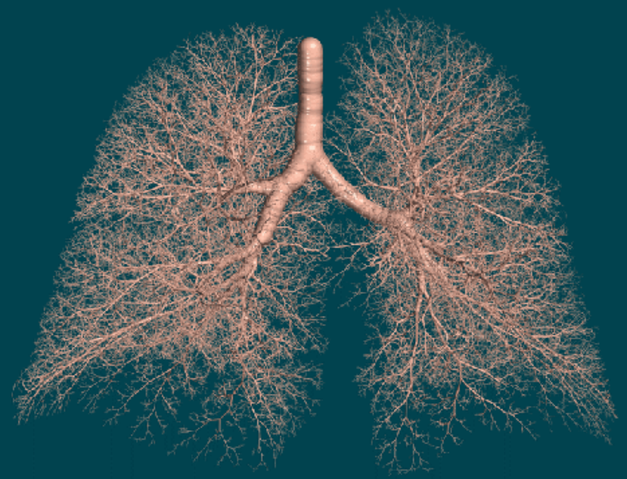
- Support for multiple GPUs to DeviceBuffer
- UnifiedBuffer class that wraps CUDA unified memory
- Example of Multi-GPU sparse 3D convolution
- CUDA utility functions for device queries

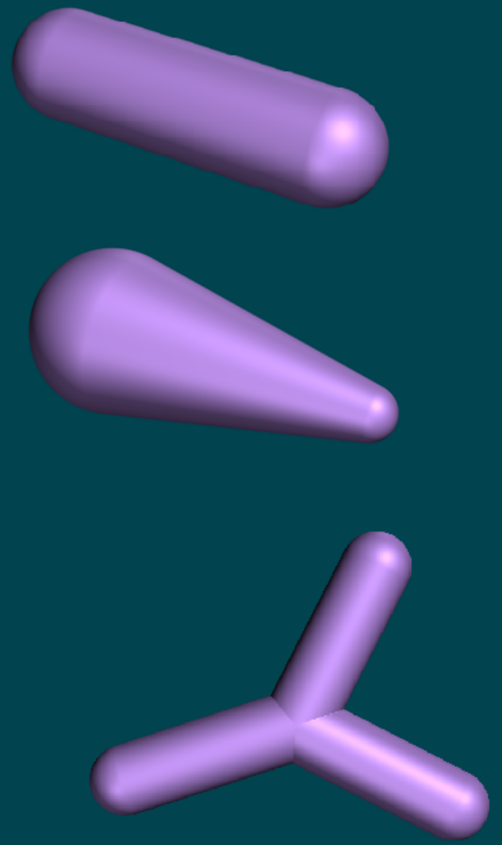
Version 12.1, August, 2025

- SDFs from tapered tubes (think 3D wire-frames)
- Anisotropic surfacing (elliptic particle footprints)
- Support for latest CLANG (requested by Apple)
- Nanovdb::VoxelBlockManager for streaming compute
- Moved AX to newer versions of LLVM

New in OpenVDB 12.1

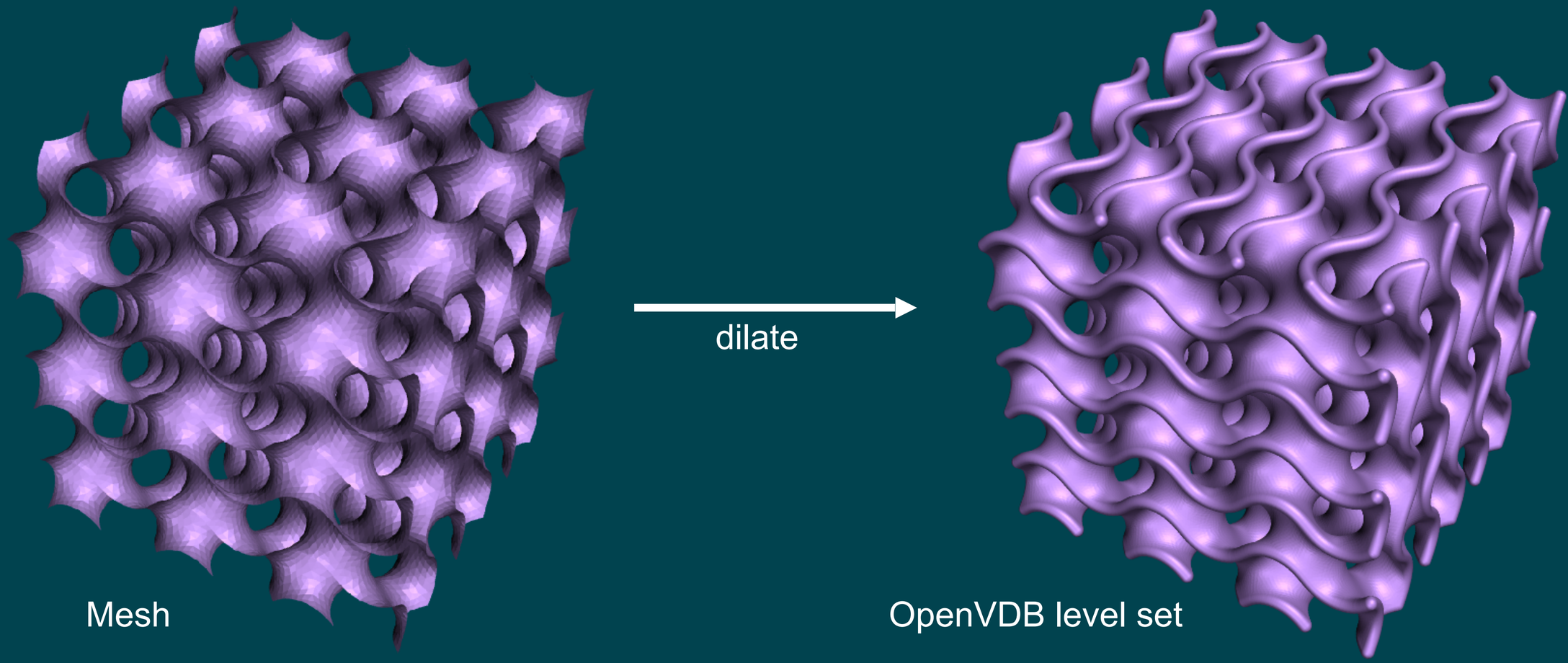
- tools::createLevelSetCapsule
- tools::createLevelSetTaperedCapsule
- tools::createLevelSetTubeComplex
- tools::createLevelSetDilatedMesh





createLevelSetDilatedMesh

Vertices and triangles of TPMS lattice parsed from ply file, then dilated:

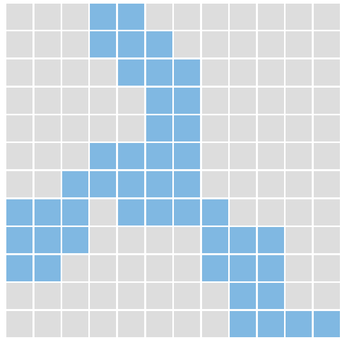


Version 13.0.0 - November 3, 2025

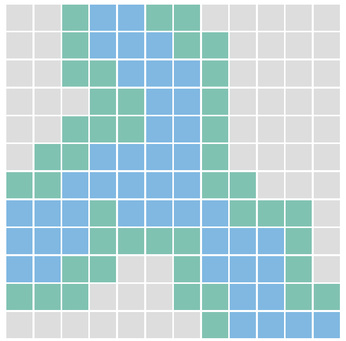
- In-memory half support (requested by Autodesk)
- Removed read support older than VDB version 1.0
- NanoVDB is no longer limited to static applications
- Dilate, merge, coarsen, refine, prune, inject

New GPU-accelerated topological operators

Input Grid

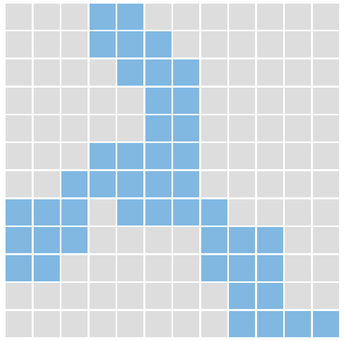


Dilated Grid

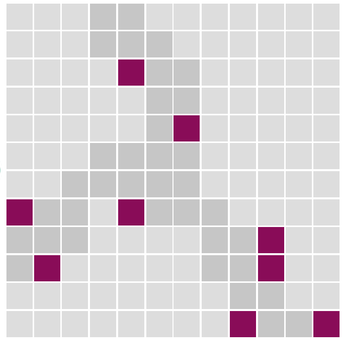


Perf: ~6ms/100M voxels

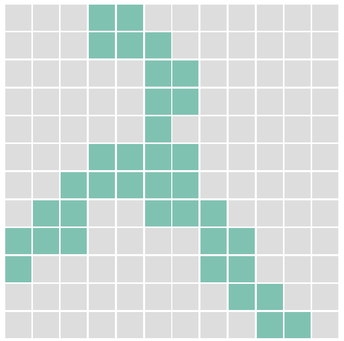
Input Grid



Prune Mask

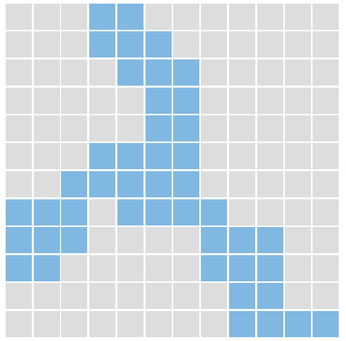


Pruned Grid

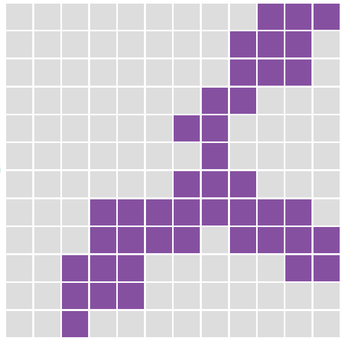


Perf: ~3ms/100M voxels

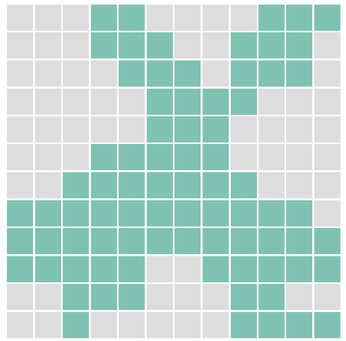
Primary Grid



Secondary Grid



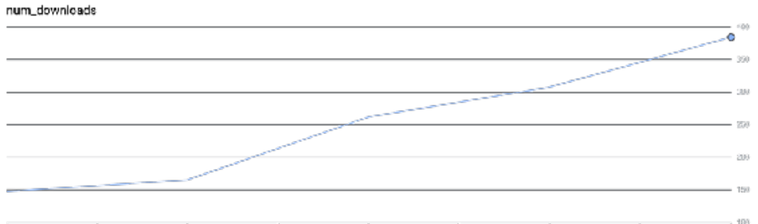
Merged Grid



Perf: ~4ms/100M voxels

fVDB

- Public release of version 0.3.0, October 24, 2025
- Low-level **SDK** (build on NanoVDB) for spatial intelligence, e.g. sparse 3D convolution and 3D attention
- Includes **batteries**: Gaussian splat training and segmentation
- More than 10 companies have already adopted fVDB
- Stats: 400 merged PRs, 8 new community contributors,30k CI jobs run (109k minutes),1300 downloads of source package



GitHub.com/openvdb

README.md

The OpenVDB GitHub Organization

Welcome to OpenVDB's GitHub organization, which hosts a collection of reposi
Software Foundation's OpenVDB project. While these projects are currently not
they are governed by the same Technical Steering Committee and adhere to the

Repositories

fvdb-core

The `fvdb-core` repository houses the core library and PyTorch extension for fV
differentiable, sparse volumetric operators built on top of NanoVDB and enables
data structure to build powerful and scalable spatial intelligence applications. T
NVIDIA.

fvdb-examples

The `fvdb-examples` repository contains examples of how to use the fVDB librar
pipelines. These are provided as reference for how to implement interesting or v
of fVDB including examples of pipelines that perform panoptic segmentation an
currently maintained by NVIDIA.

fvdb-reality-capture

The `fvdb-reality-capture` repository contains examples of how to use the fVD
centered on fVDB's 3D Gaussian splatting methods. This repository houses utili
train and render 3D Gaussian splatting models using fVDB as well as interesting
Gaussian scene representation such as meshing. This repository is currently ma

nanovdb-editor

The `nanovdb-editor` repository contains a library, python bindings and a stand
view and edit NanoVDB. This repository is currently maintained by NVIDIA.

openvdb-maya

openvdb.GitHub.io/fvdb-core

Welcome to fVDB!

fVDB is a Python library developed and maintained by NVIDIA, containing a collection o
structures and algorithms for building high-performance and large-domain spatial applic
NanoVDB on the GPU in PyTorch. Applications of fVDB include 3D deep learning, comp
graphics/vision, robotics, and scientific computing.

fVDB aims to be production ready with a focus on robustness, usability, and extensibilit
designed to be easily integrated into existing pipelines and workflows, and to support a
of use cases and applications. To this end, fVDB has a minimal set of dependencies and i
source under the Apache 2.0 license as part of the The Academy Software Foundation's

Miris + fVDB

OpenVDB SIGGRAPH Course
August 10, 2025

Presented by Joe Nordling, Applied Researcher

ƒVDB

Performance Improvements

Original Scene

Without-ƒVDB Training

With-ƒVDB Training

ƒVDB

Segmentation

Original Scene

Without-ƒVDB Segmentation

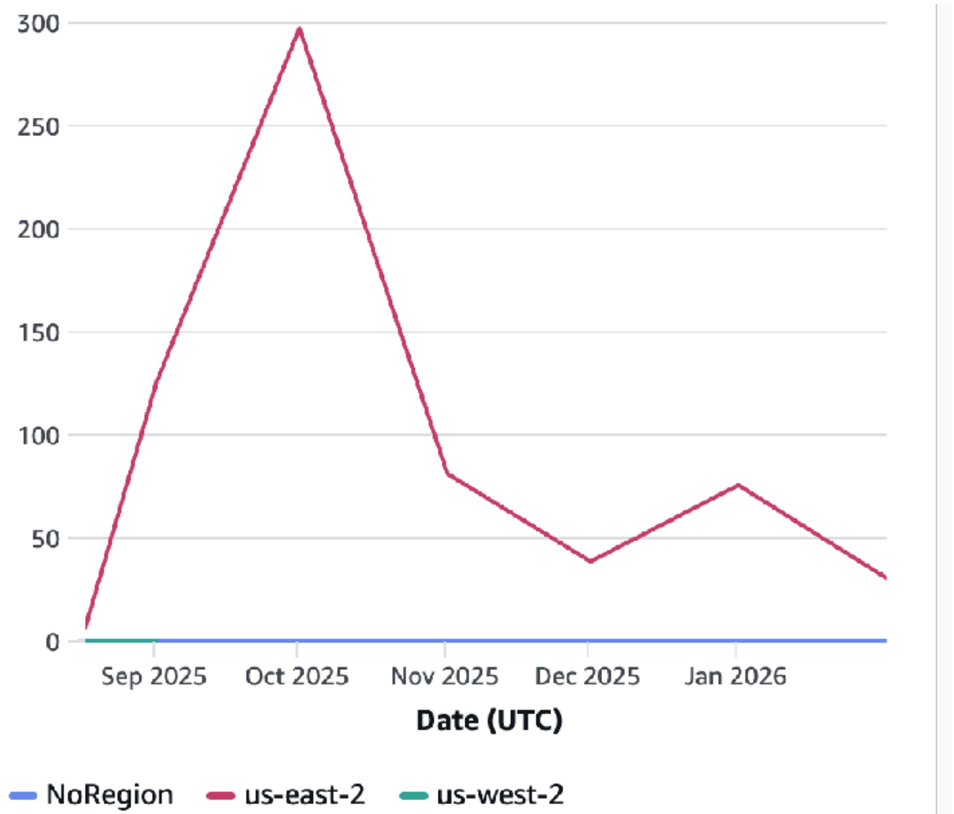
With-ƒVDB Segmentation


fVDB Reality Capture
177 images from an $800 drone


1.2 Billion Gaussian Splats

CI for fVDB

- Moved CI to AWS' EC2 instances
- From fractional L40s to multiple GB200
- $650 total cost over 7 months
- One misconfigured month (whoops)
- Typically $30-$75/month
- NVIDIA is paying



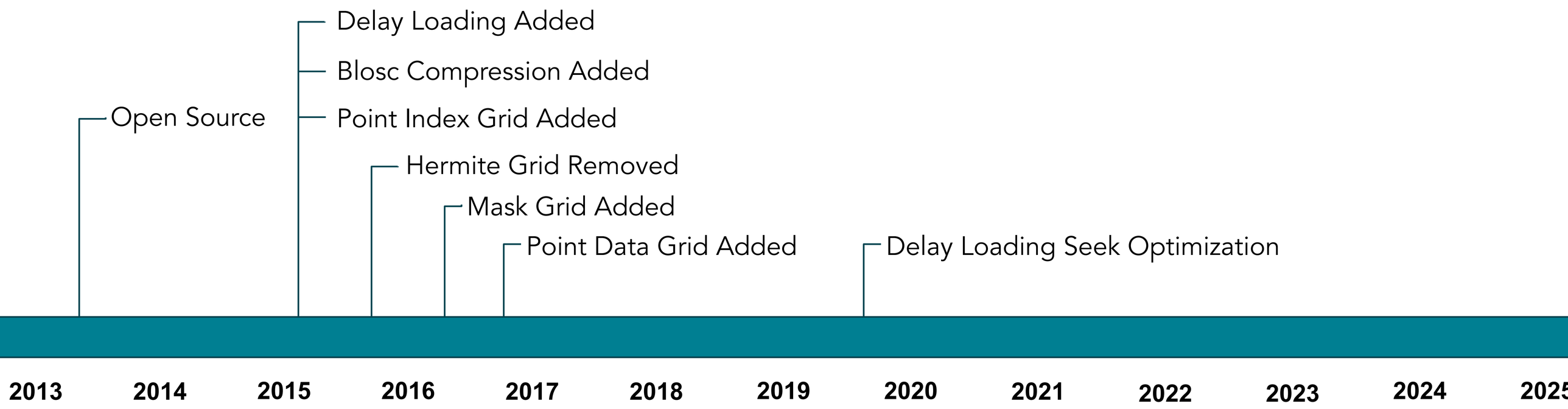
CI for OpenVDB

- Small on every commit
- Larger on nightly builds
- All on a weekly basis

CI for OpenVDB

- Small on every commit
- ~~Larger on nightly builds~~
- All on a weekly basis

History of VDB File Format



File Format is very stable and has not changed substantially in 10+ years!

Future: new file format (1)

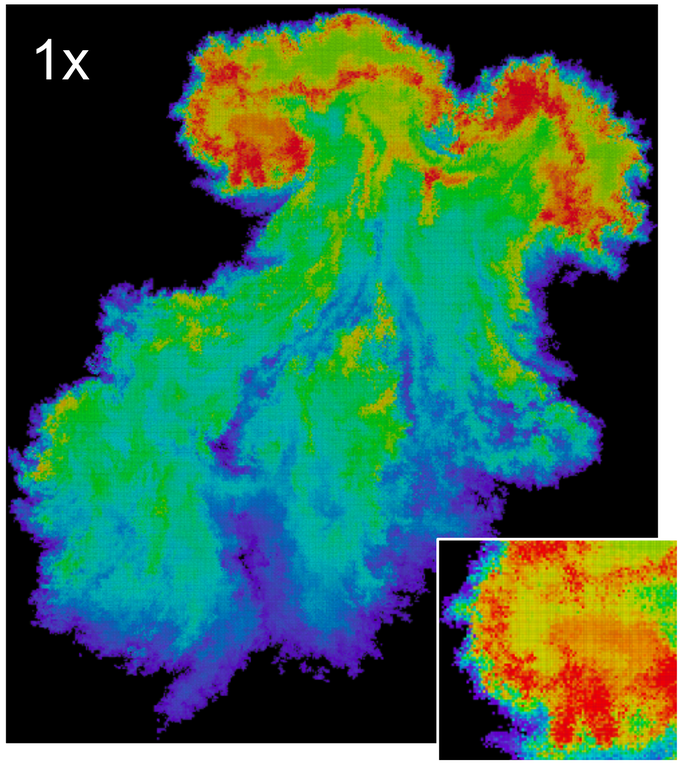
- Preserve backwards compatibility
- Enable read-support before write-support (opt-in)
- Separate topology and values (cf NanoVDB)
- Support lossy compression (cf NeuralVDB)
- Improve lossless compression with octree transcoding

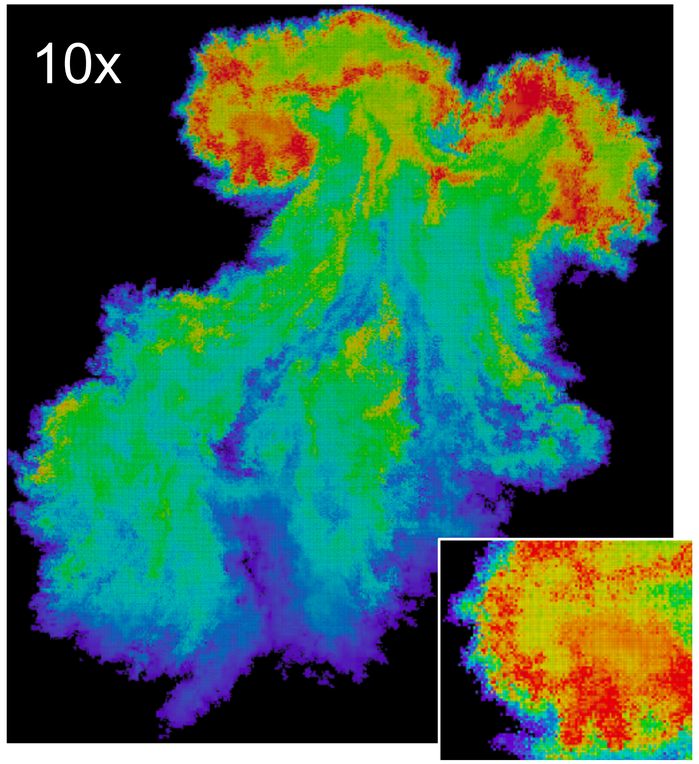
Future: new file format (2)

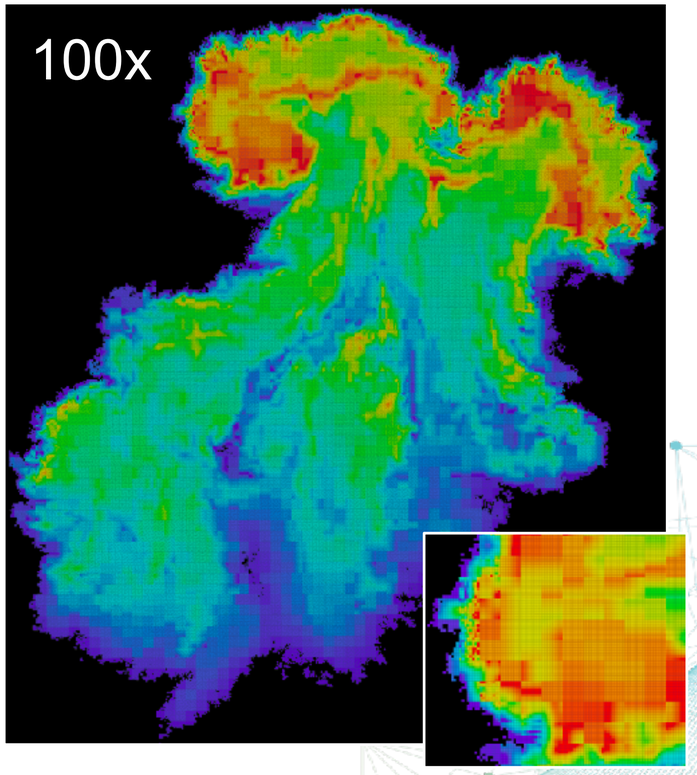
- Replace Delayed Loading with Selective Loading
  - Eliminates Boost dependency
- Modular I/O Codec Architecture
  - Continue to Read Legacy Format, No Grid ABI or API breakages
- Improve Lossless Codecs
  - Faster by overlapping I/O with Compute
- Add new Lossy Codecs
  - E.g. Discrete Cosine Transforms (DCT)

Future: new file format (2)

- Discrete Cosine Transform (DCT) is the algorithm behind JPEG/MPEG
- DCT for **images** performs lossy compression on 8x8 **pixel** data
- DCT for **volumes** performs lossy compression on 8x8x8 **voxel** data







ASWF /* ACADEMY SOFTWARE FOUNDATION

OpenVDB