Tools for Rapid and Efficient I/O for Exascale

Safaa Diab, Anna Mack, Patrick Vogler – HLRS
Arun Kumar Dwivedi - UHOH

GEFÖRDERT VOM
Bundesministerium für Forschung, Technologie und Raumfahrt

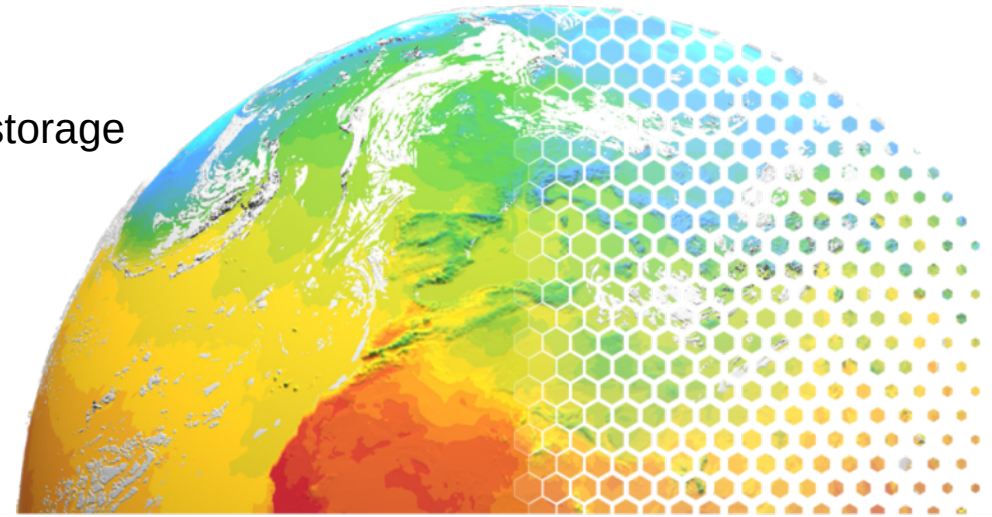
TOPIO: The Project

With increasing computing power, improvements in weather forecasting are leading to a dramatic increase in the amount of data generated per simulation.

- Current resolution of seasonal weather forecasts is insufficient to accurately represent the interactions between the ocean, land surface and atmosphere.
- Push for global weather modeling with a resolution of 1.5 km. High-resolution modeling method results in datasets that could exceed 1 Petabyte in size.

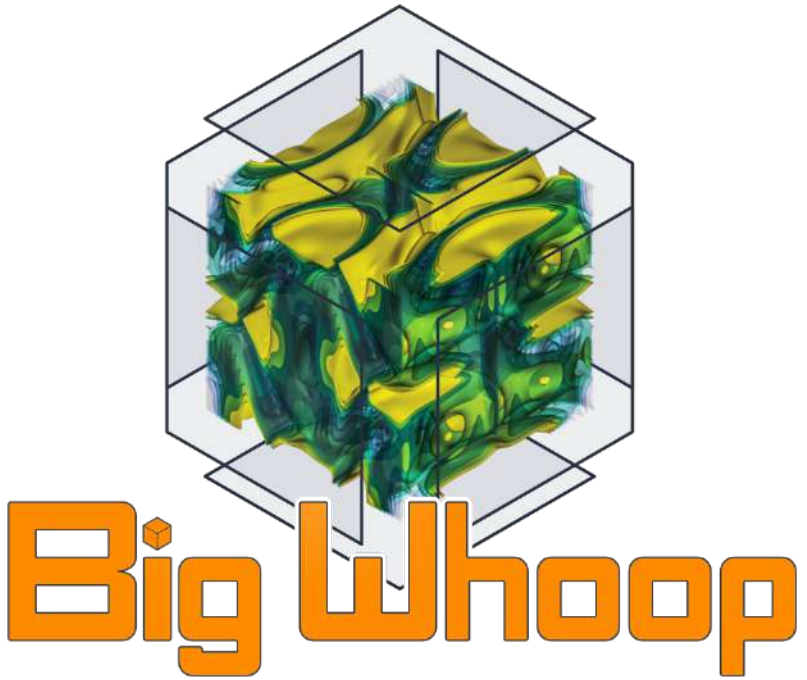
Improving the use of computational resources in Earth system simulation applications (e.g. MPAS) requires a redesign of the I/O concept.

- Apply data compression techniques to reduce the memory usage and storage foot print of the simulation.
- Use I/O auto-tuning to balance compression overhead and boost I/O speed.



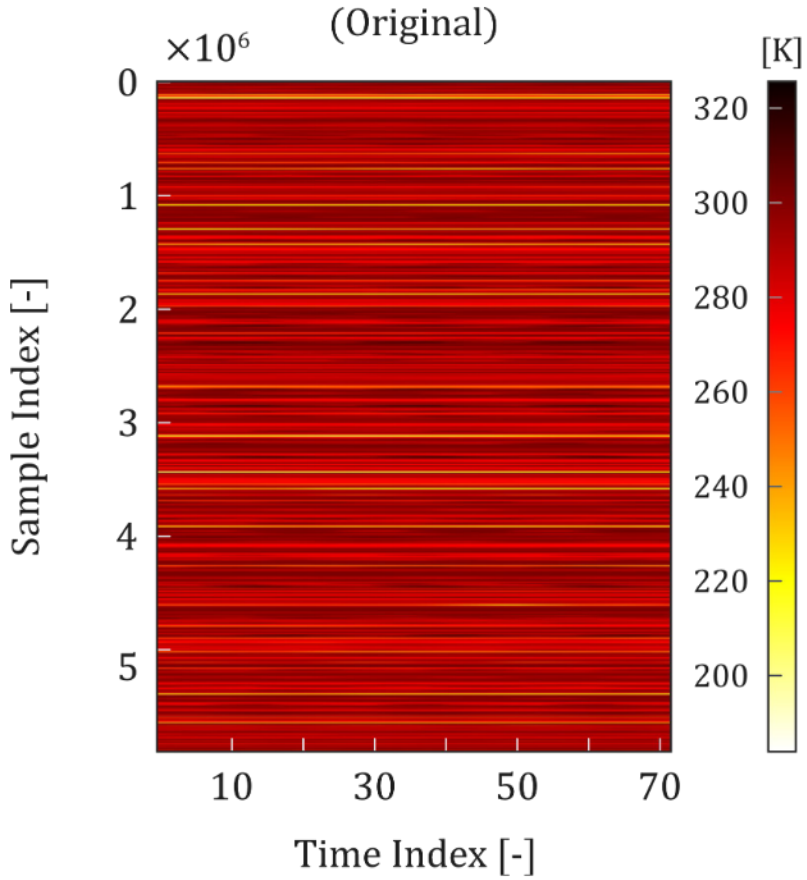
: The BigWhoop Compression Library

- Derived from JPEG 2000 for 1–4D scientific datasets with native support for single and double-precision IEEE 754 floats.
- Provides an optimally truncated, quality- and resolution scalable code-stream.
- Offers a clean, lightweight core API in C with bindings for Python and Fortran.
- Leverages OpenMP for multi-threaded compression and decompression.
- Includes plugins for HDF5 and ADIOS2, allowing direct drop-in replacement within existing high-throughput data pipelines.
- Currently developing integration with the popular hdf5plugin package to simplify installation and usage within Python workflows.
- For source code got to: https://github.com/ptvogler/BigWhoop



Compression Comparison: Test Case

- Spatially limited (partial-block) snapshot of near-surface air temperature at 2 m above ground level, extracted from a global weather-simulation:
  - Modell: MPAS-7.3.1
  - Resolution: 10 km
  - Encoding: IEEE-754 32-bit floating point (float32)
  - Dimensionen: nCells (5898242) x Zeit (72)
  - File Size: 1613 MiB
  - File Container: NetCDF-4
- Compression was carried out using HDF5 library.
- Dataset was processed ‘as is’ using the HDF5 filter interface.
- Results provided for lossy codecs represent max. usable reduction.

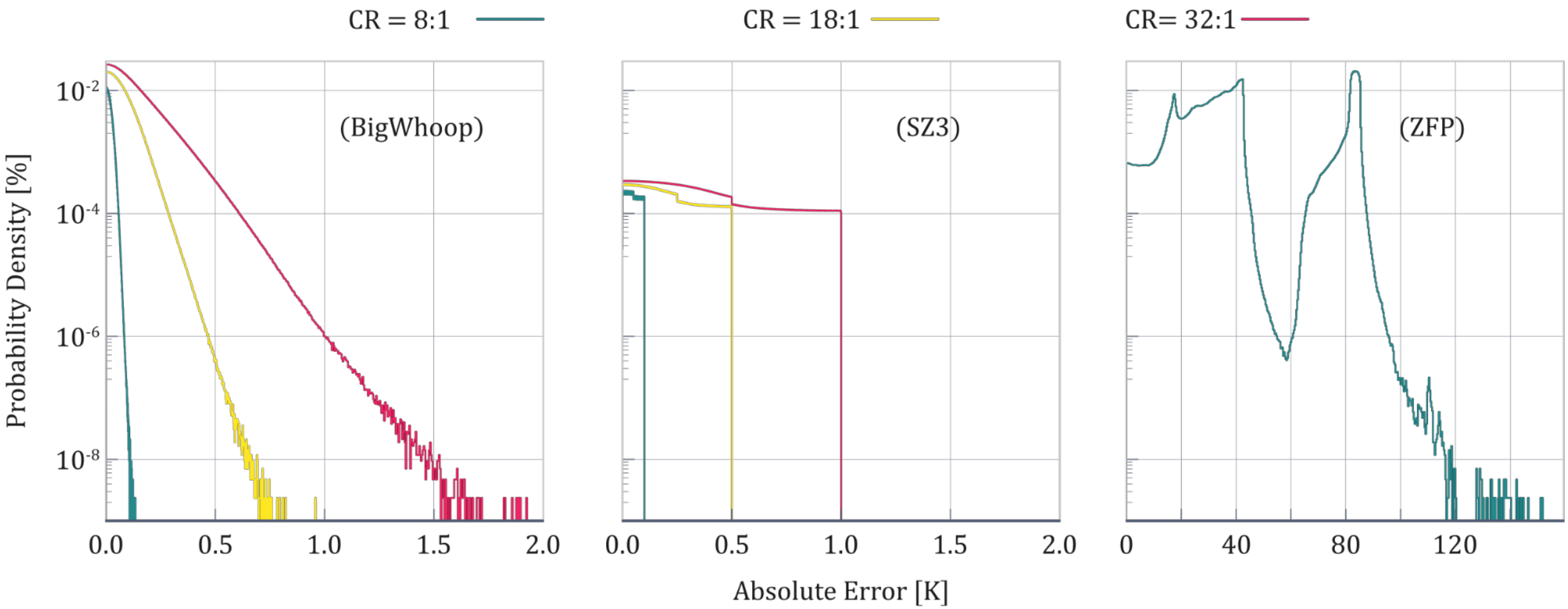


Compression Comparison: Performance Comparison

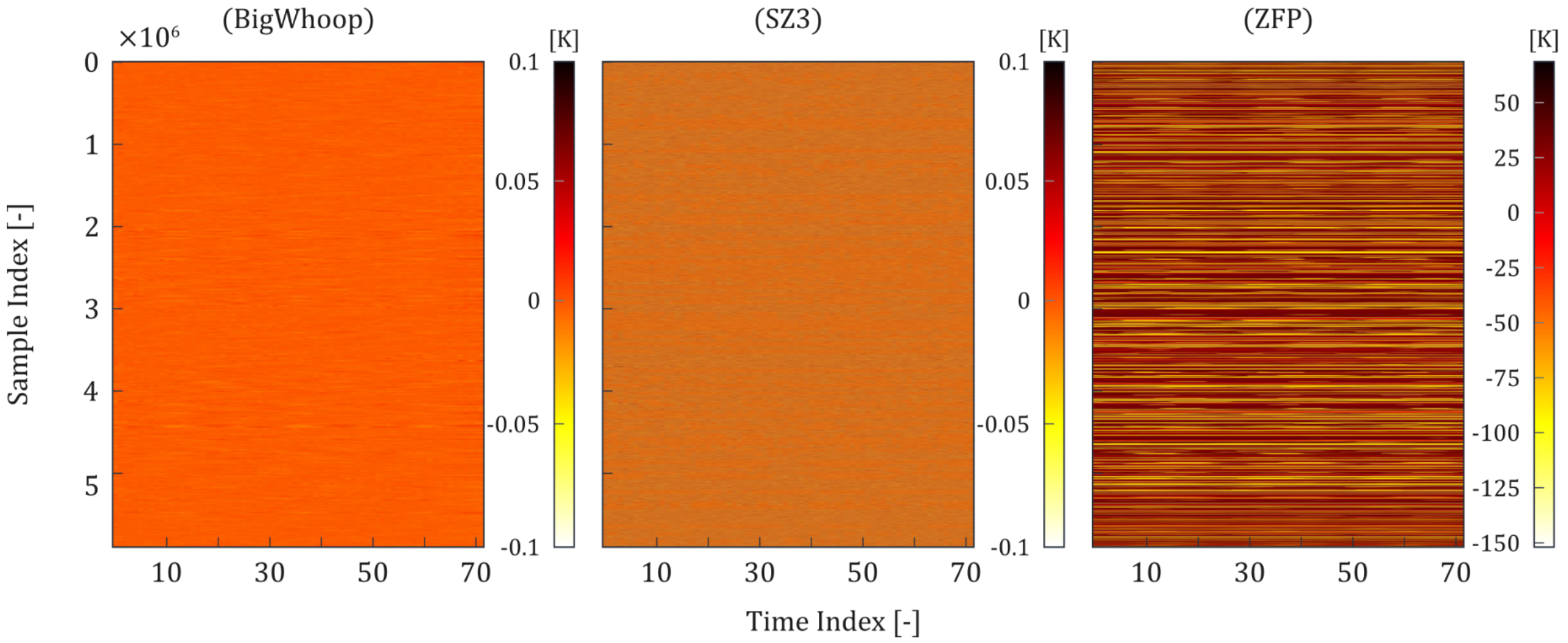
		Compression Time [s]	Compression Ratio [-]	Size [MiB]	PSNR[a] [dB]
BLOSC	blosclz	12.7	1.7	936	∞
	lz4	11.5	1.7	939	∞
	lz4hc	48.9	1.8	896	∞
	snappy	12.1	1.7	936	∞
	zlib	39.2	1.9	876	∞
	zstd	17.5	1.8	885	∞
BLOSC2	blosclz	16.0	1.8	903	∞
	lz4	13.5	1.8	887	∞
	lz4hc	61.8	1.9	866	∞
	zlib	47.1	1.9	854	∞
	zstd	20.8	1.9	861	∞
SZ		490.1	2.9	550	∞
SZ3		89.5	30.7	53	56.9
BigWhoop		89.3	32.1	50	53.2
ZFP		30.0	2	810	101.8

[a] By definition, the PSNR for lossless compressors is always infinite.

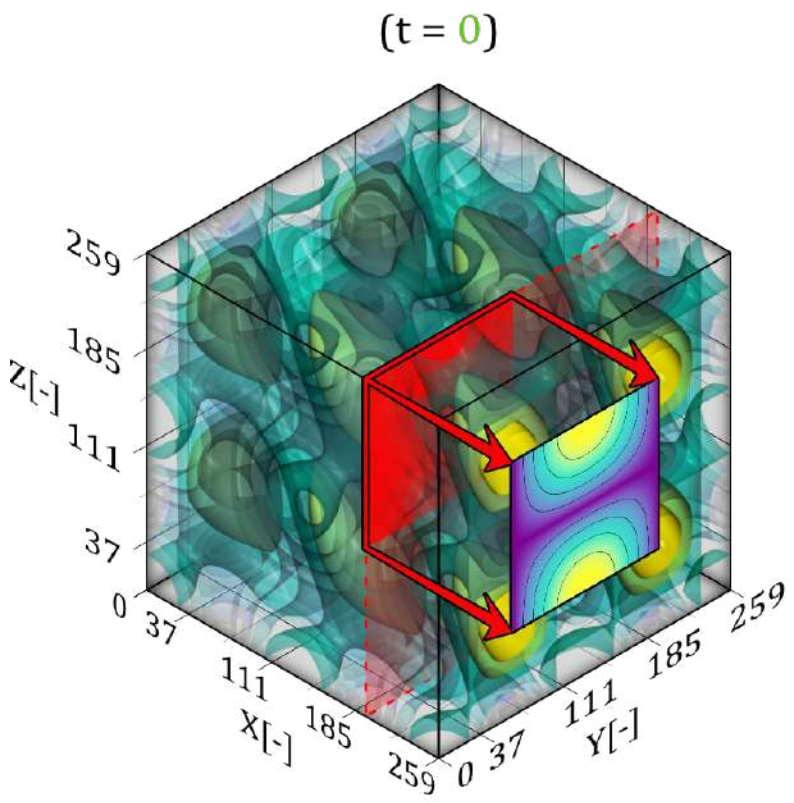
Compression Comparison: Performance Comparison

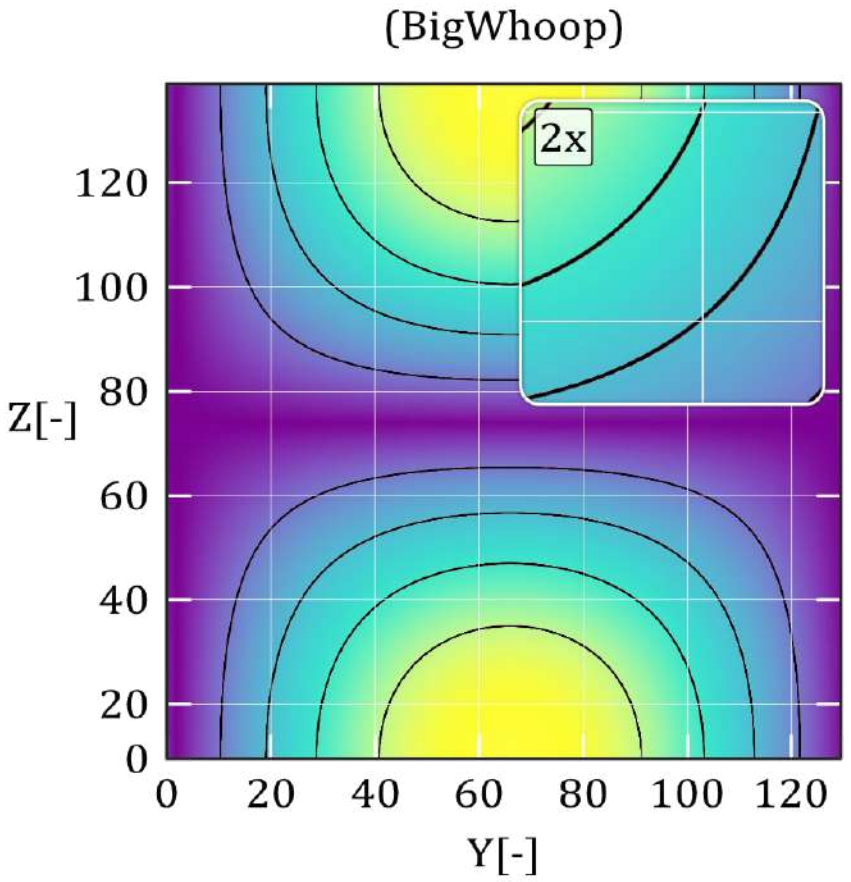


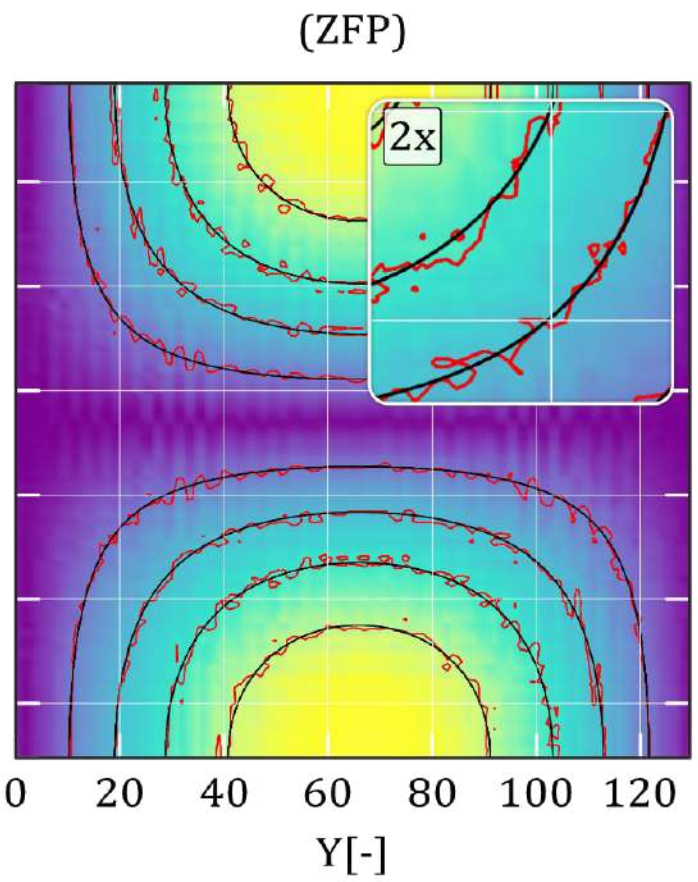
Compression Comparison: Performance Comparison



Compression Comparison: Artifacts







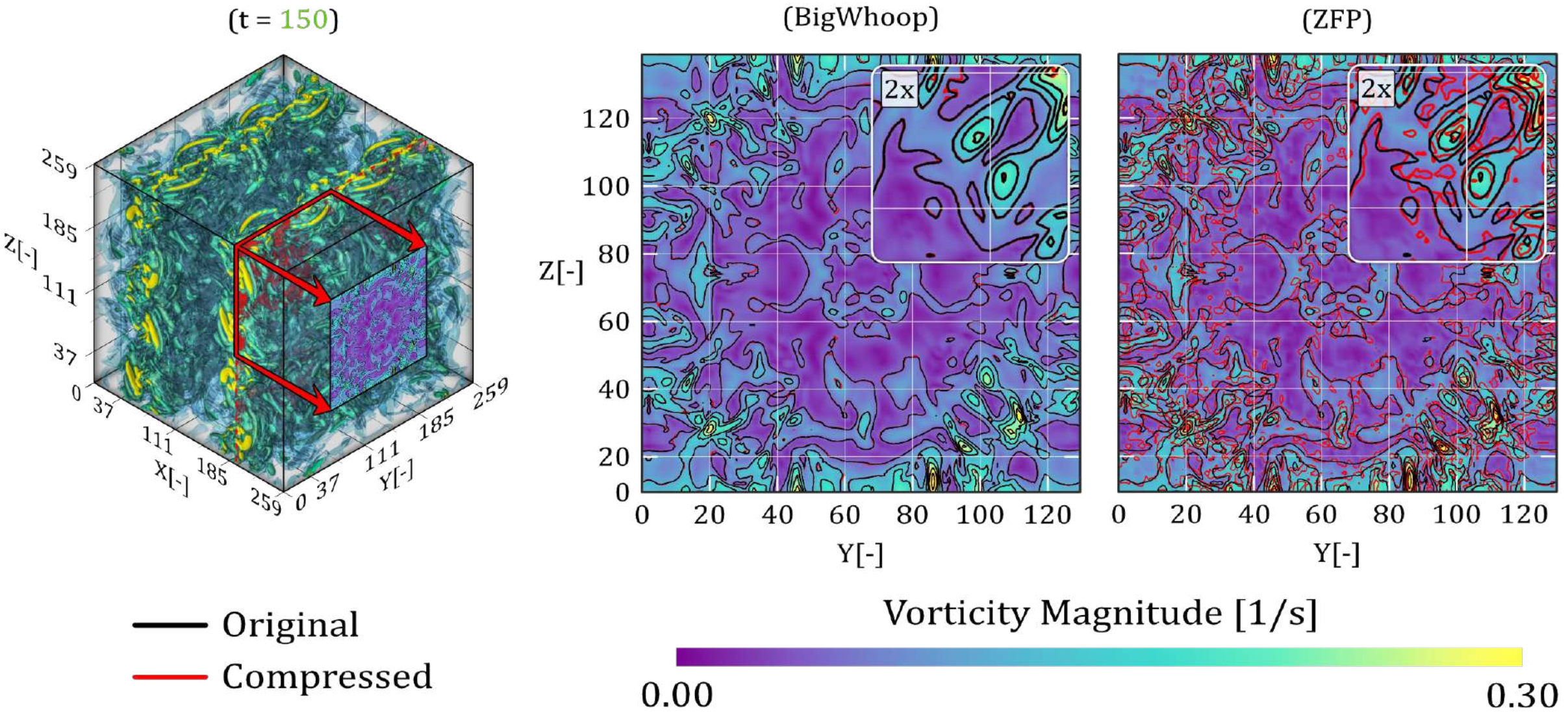
Original

Compressed

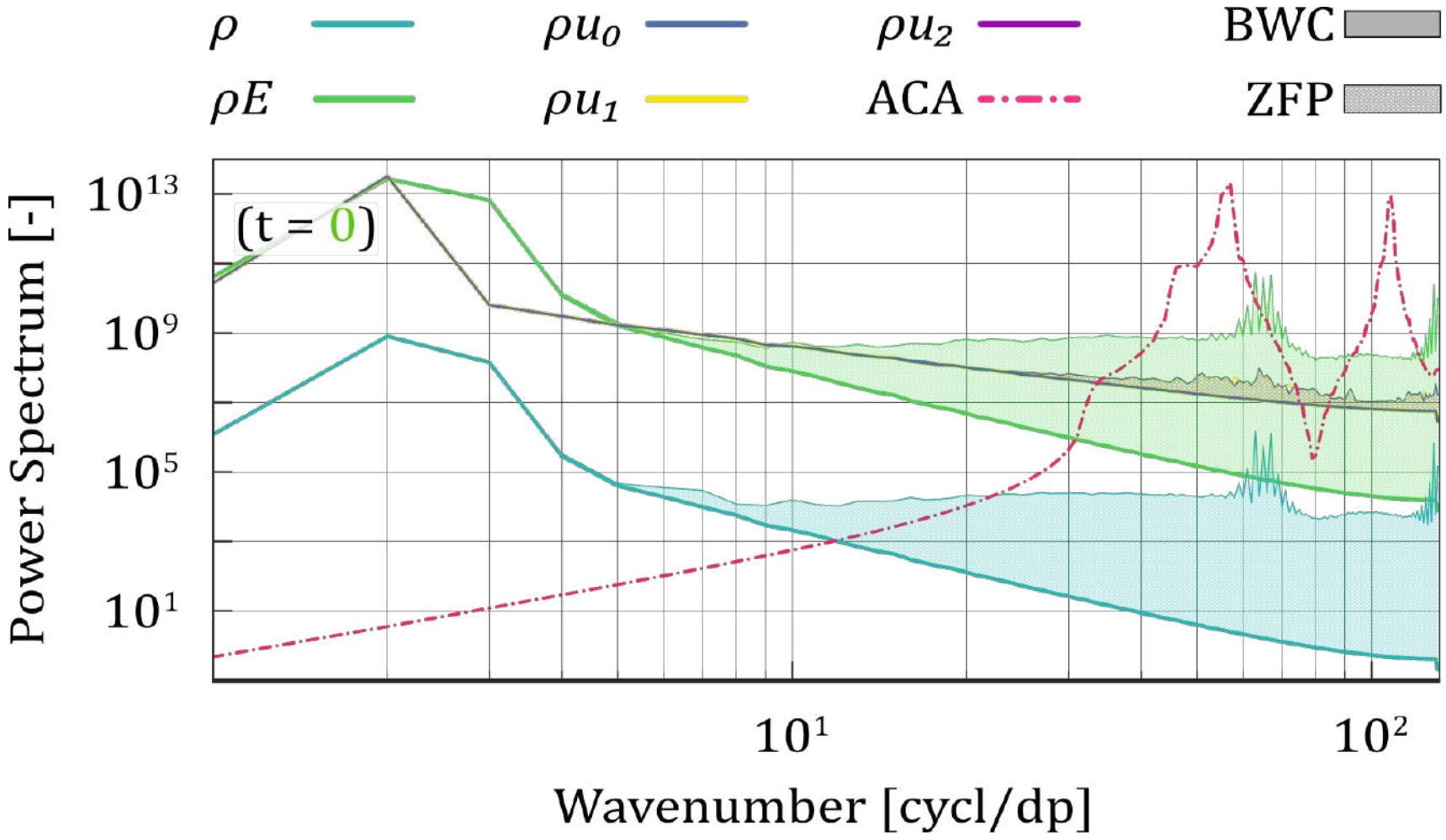
Vorticity Magnitude [1/s]

0.00 0.025

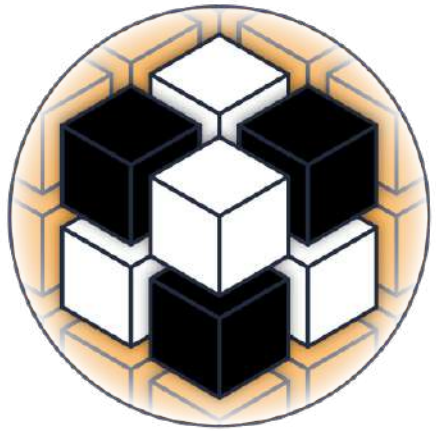
Compression Comparison: Artifacts



Compression Comparison: Artifacts



Artificial Compression Artefact

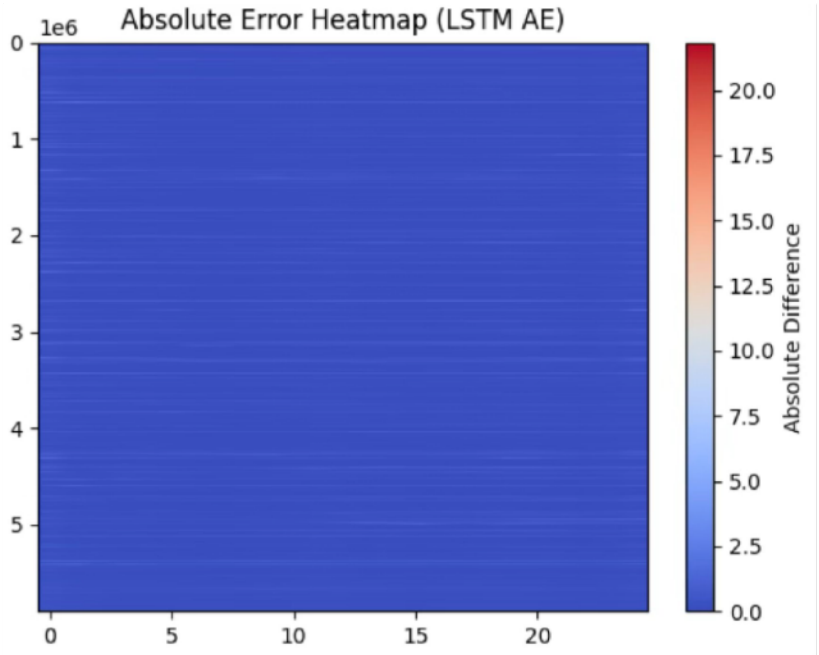


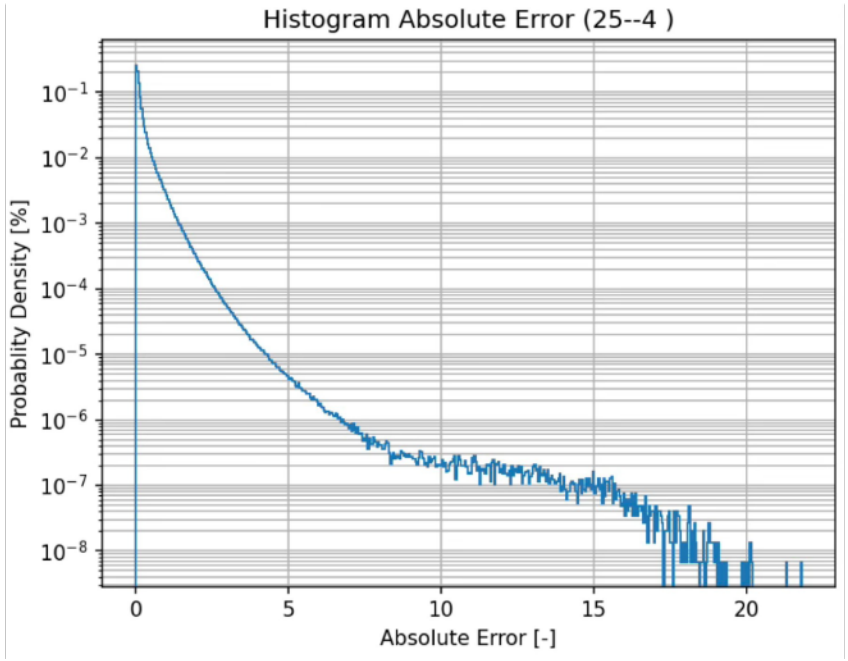
4x4x4

TOPIO: Compression with LTSM Autoencoders

- Autoencoders models learn to compress data into a lower-dimensional „latent“ representation, and then reconstruct the original data from this compressed form. Long-Short-Term-Memory (LSTM) autoencoders capture long-term dependencies in sequential data such as time series.
- We trained the LSTM autoencoder to compress the surface data of 10km grid generated for 25 time-steps.

# Epochs	5
Training time	4394.58 s
Compress time	279.51 s
Decompress time	214.1 s
Compression Ratio	6.25
pSNR	51.57 dB





Compression: Conclusion

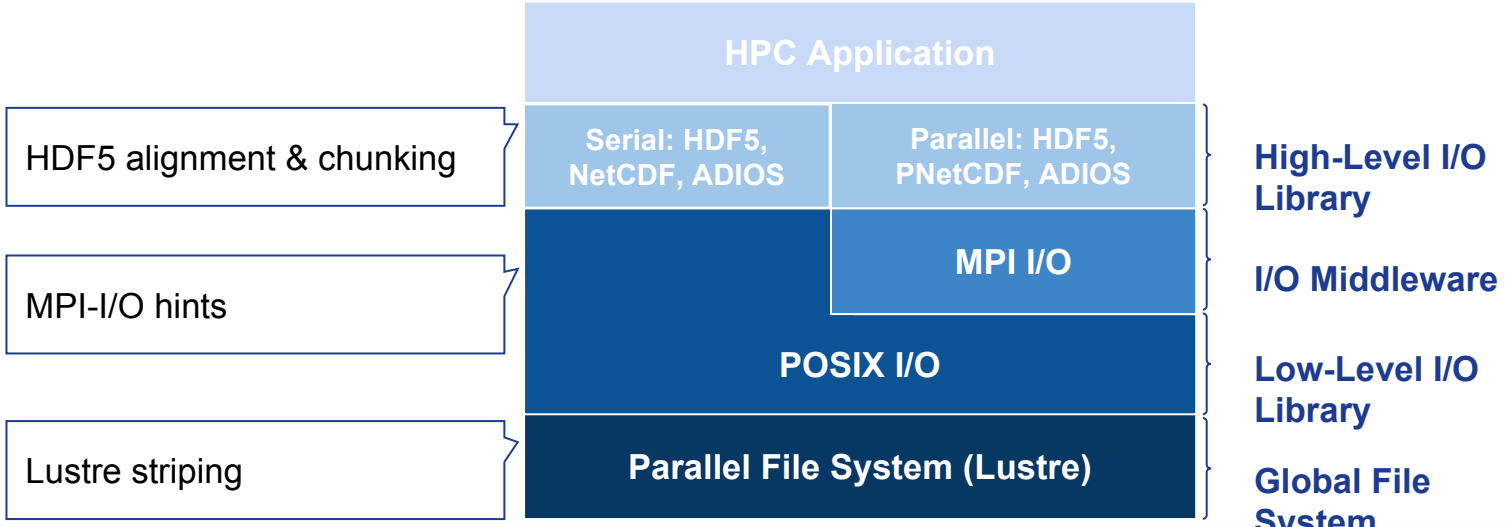
- Compression can and should be used to effectively reduce the storage footprint.
  - A conservative compression estimate suggests the dataset would shrink from roughly 207.7 GB to 20.8 GB for the 10 km case, and from 674.4 GB to 67.4 GB for the 3 km test case.
- Data compression approach needs to fit the use case:
  - 1-dimensional arrays → E.g. Blosc2 lz4
  - Higher-dimensional arrays → BigWhoop, SZ3
- Sort sample arrays before processing to flatten the power spectrum, which can enhance
- compression efficiency and overall data quality.
- Concatenation of multiple time-steps can aid in compression performance.
- Consider pre-compression truncation to reduce data size and enhance overall
- compression performance.

I/O Tuning on HPC Systems

- Multi-layer complexity: performing I/O involves various software and hardware layers, each offering tunable parameters, with no universal optimal configuration.

- Manually tuning is impractical: exploring the large parameters space is time and resource consuming, often requiring source-code modification

- Therefore, we leverage **IOFlex** a framework to efficiently optimize and auto-tune I/O hints for HPC applications



IOFlex: A Framework for Optimizing and Auto-Tuning I/O hints in HPC Applications

- IOFlex consists of two main components: an I/O wrapper and an I/O tuning library

1. **IOFlex I/O wrapper library**
   - Adjusts I/O configurations by intercepting MPI-IO calls and setting the hints seamlessly
   - Instruments applications via LD_PRELOAD at runtime or by linking at compile time
   - Supports multiple MPI implementations and libraries such as HDF5 and PNetCDF.
   - I/O configurations set through IOFlex override previous settings

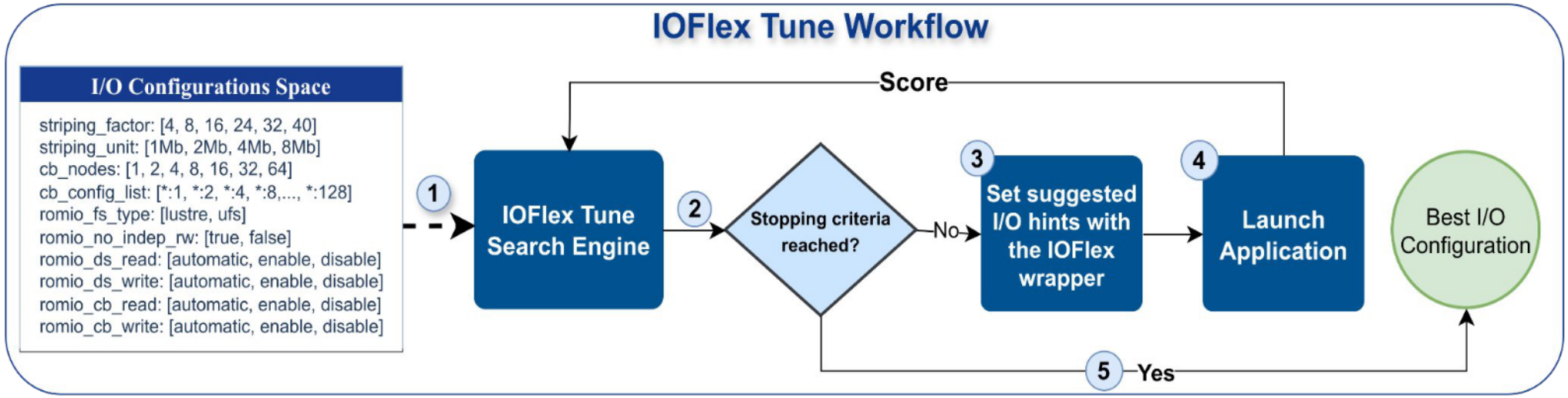
ROMIO I/O hints
striping_unit striping_factor romio_filesystem_type romio_ds_read romio_ds_write romio_cb_read romio_cb_write cb_nodes cb_config_list romio_no_indep_rw direct_io
Cray-MPICH I/O hints
cray_cb_nodes_multiplier cray_cb_write_lock_mode

IOFlex: A Framework for Optimizing and Auto-Tuning I/O hints in HPC Applications

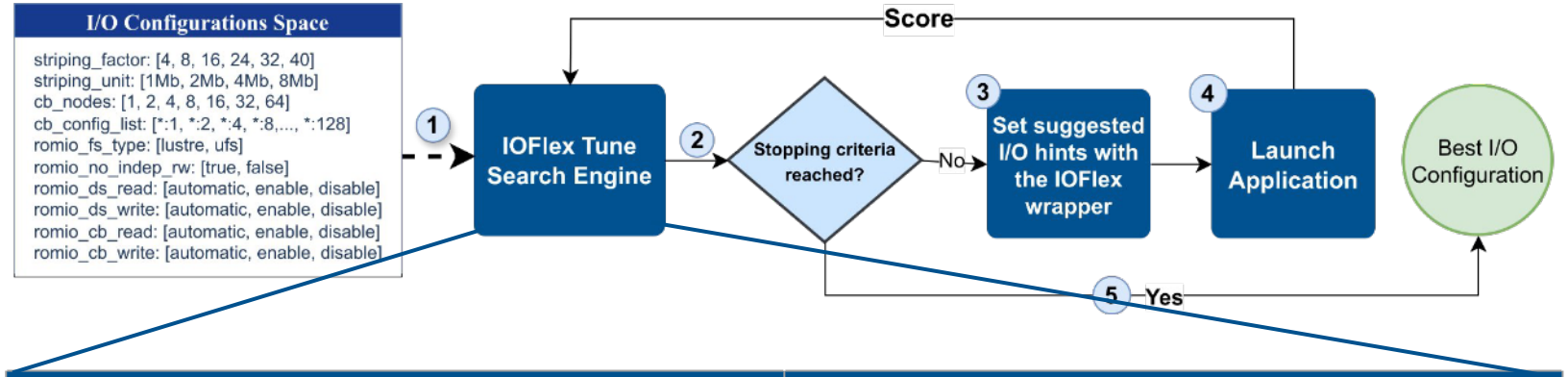
H L R S

2. IOFlex-Tune library:

- Efficiently explores the parameter space to identify near-optimal I/O configurations using advanced optimization libraries
- The objective score, either maximizing bandwidth (collected with Darshan) or minimizing runtime
- The stopping criteria is user-defined as number of trials



IOFlex-Tune Supported Search Libraries



Supported backend library	Search Algorithm
Optuna hyperparameter optimization framework that adapts state-of-the-art search algorithms	Gaussian Process Grid Search Nondominated Sorting Genetic Algorithm II Parzen-Tree Estimator
Nevergrad gradient-free optimization framework developed by Facebook	NgOpt meta-optimizer Two Points Differential Evolution Portfolio Discrete One Plus One

Auto Tuning MPAS a 3 km resolution grid on HAWK with IOFlex

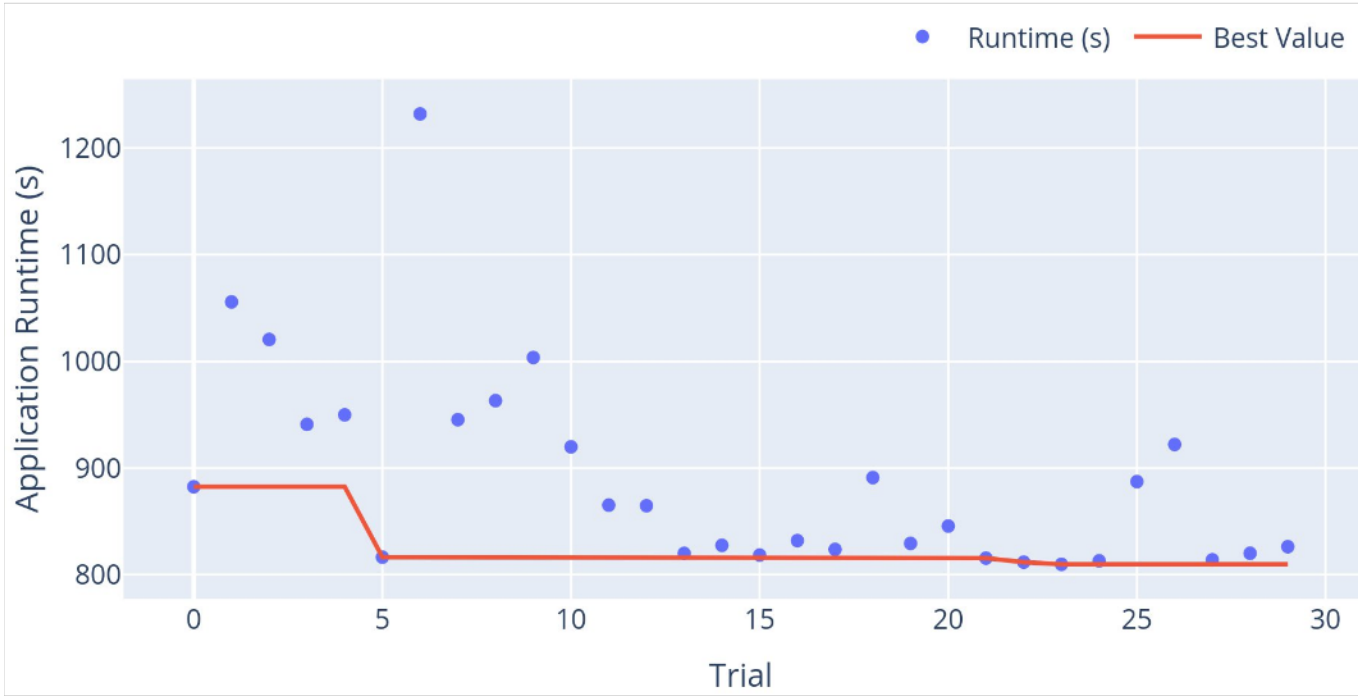
- Hawk Supercomputer: 128 nodes, 32 MPI processes per node, 4 OMP threads per process
- Input size ~756.10GB; Total output size 674.4GB; Default runtime: ~1090s
- The default Lustre striping is the progressive file layout, which is not optimal and requires tuning

parameters sorted by impact on performance in descending order

romio_ds_read	automatic
romio_cb_read	automatic
romio_fs_type	UFS
romio_no_indep_rw	False
cb_config_list	*:1
striping_factor	32
striping_unit	4Mb
romio_ds_write	disable
romio_cb_write	disable

Configurations Tuned with IOFlex Run 30 trials ~8hrs

Aligns with manual tuning results



Auto Tuning MPAS a 3 km resolution grid on HAWK with IOFlex

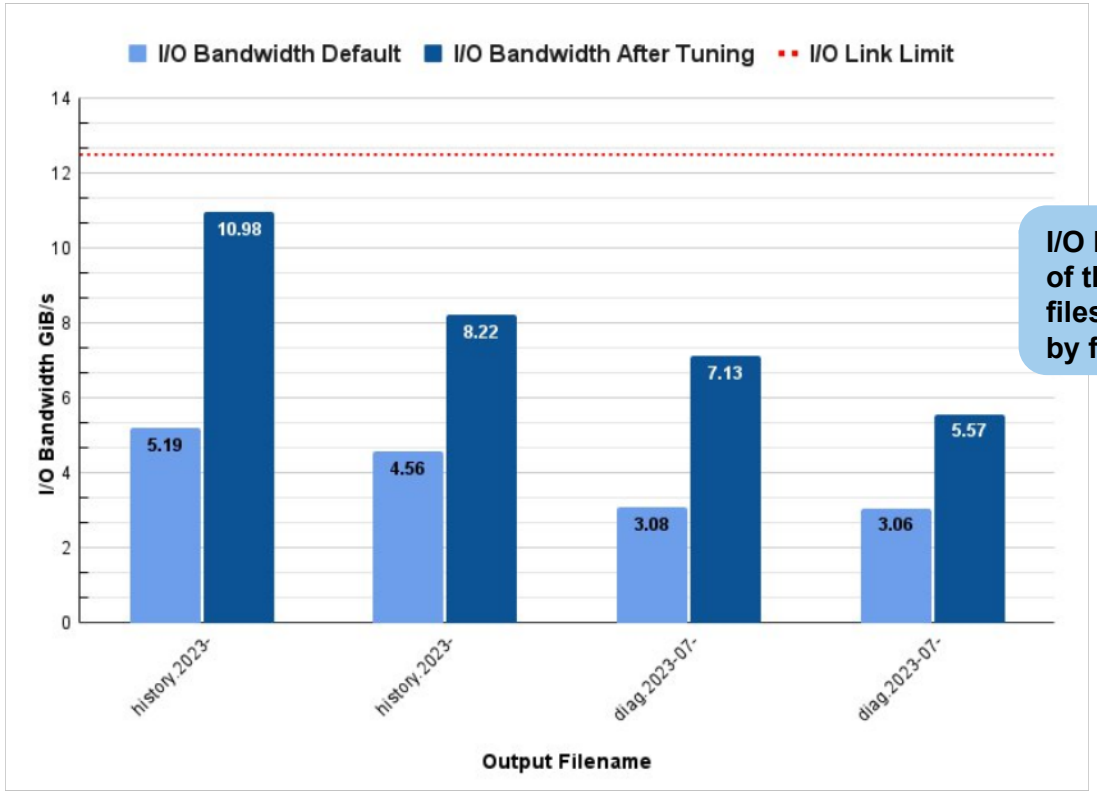
- Hawk Supercomputer: 128 nodes, 32 MPI processes per node, 4 OMP threads per process
- Input size ~756.10GB; Total output size 674.4GB; Default runtime: ~1090s
- The default Lustre striping is the progressive file layout, which is not optimal and requires tuning

parameters sorted by impact on performance in descending order

romio_ds_read	automatic
romio_cb_read	automatic
romio_fs_type	UFS
romio_no_indep_rw	False
cb_config_list	*:1
striping_factor	32
striping_unit	4Mb
romio_ds_write	disable
romio_cb_write	disable

Configurations Tuned with IOFlex Run 30 trials ~8hrs

Aligns with manual tuning results

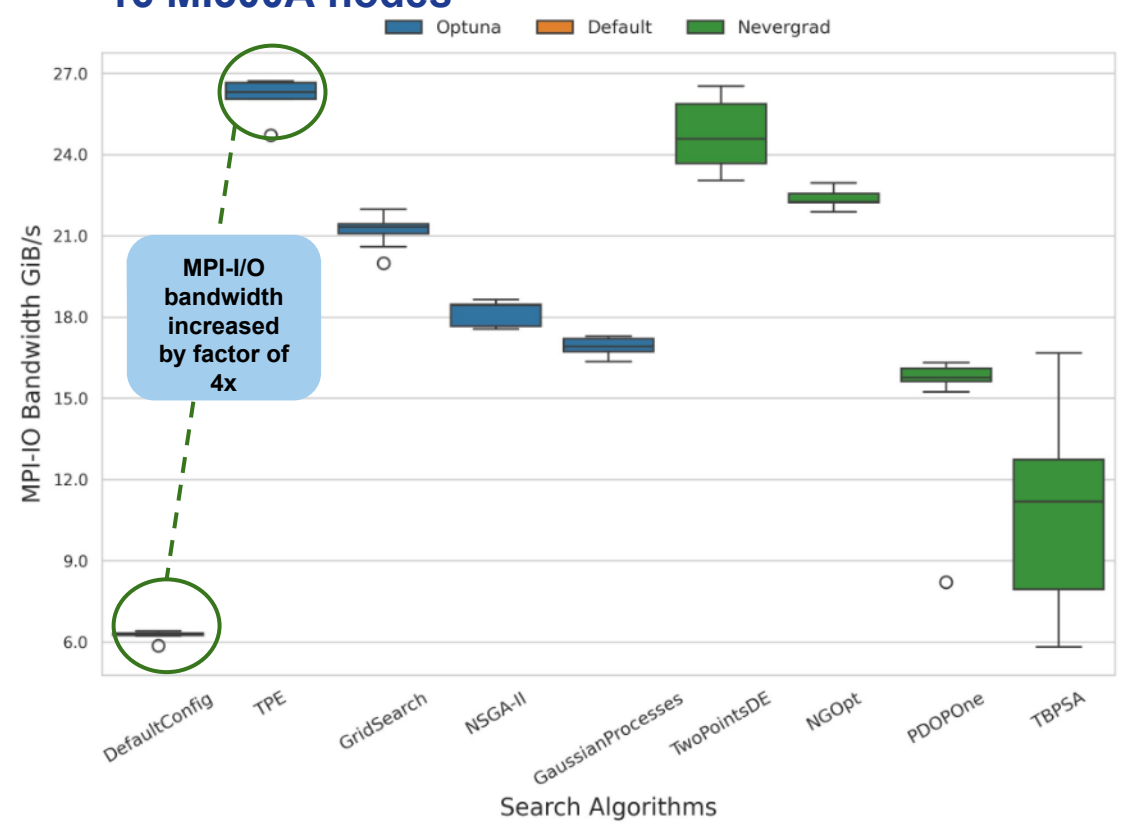


I/O Bandwidth of the output files increased by factor of 2

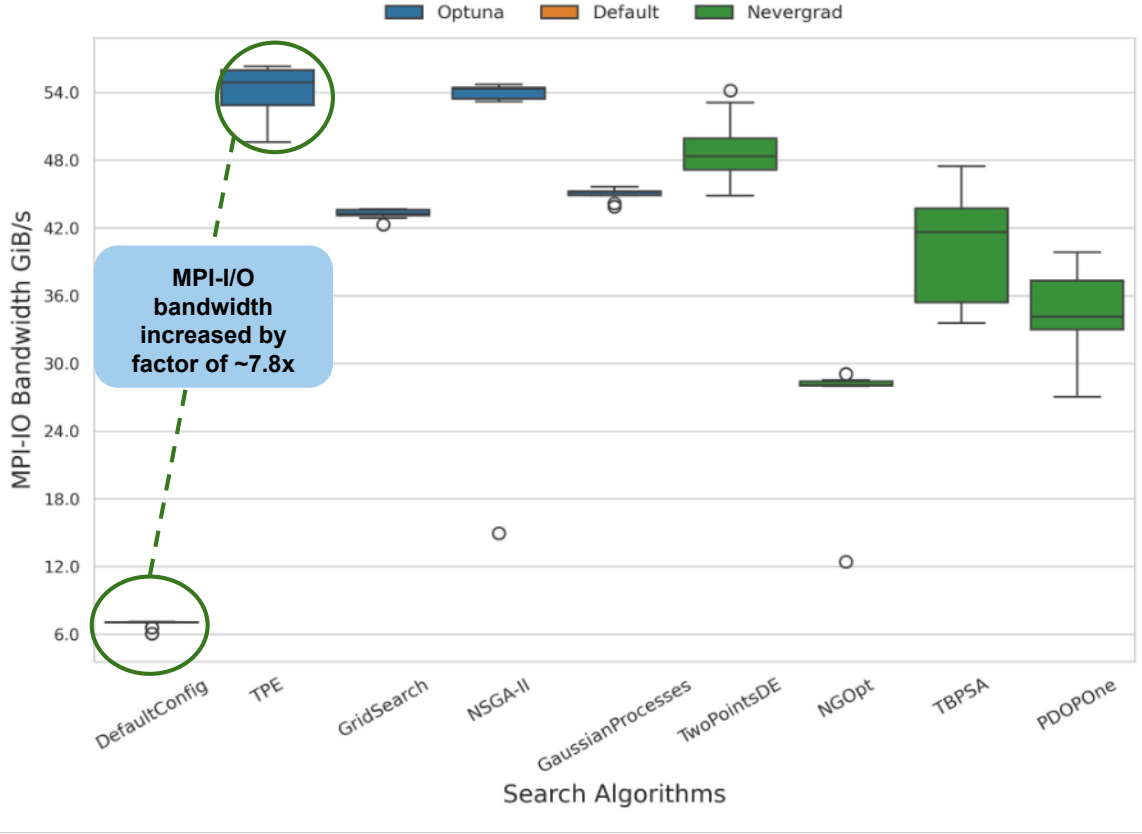
Auto Tuning MPAS a 10 km resolution grid on Hunter with IOFlex

- Input size: 68.05 GB; total output size: 60.69 GB
- We run MPAS with the best I/O configuration identified by each search algorithm multiple runs
- Boxplots show performance variability due to system noise
- Optuna's Parzen-tree estimator detects the best I/O configuration

16 MI300A nodes

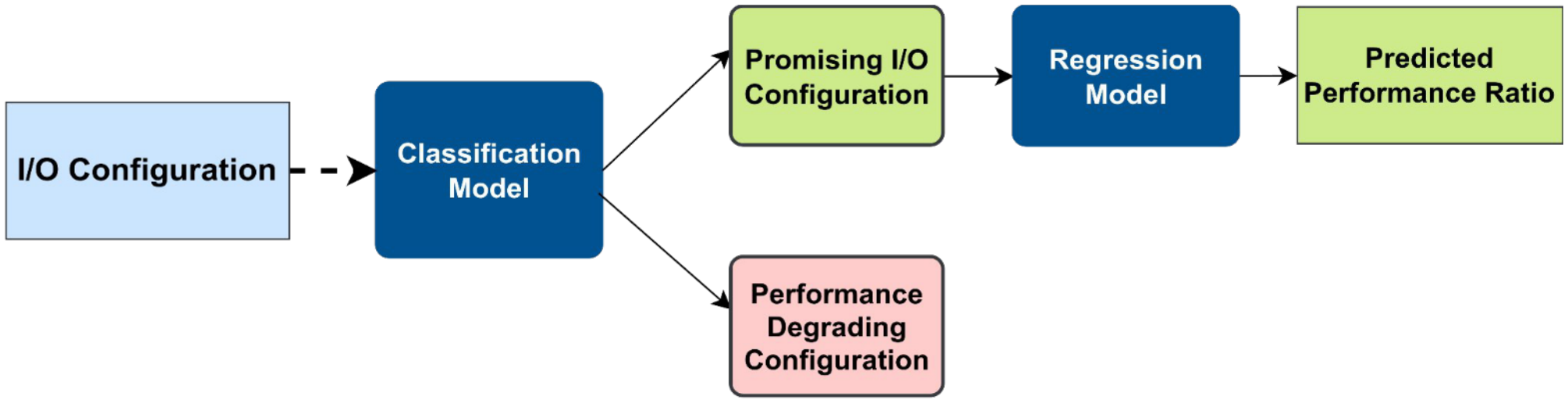


32 MI300A nodes



Predicting Performance of I/O Configurations

- IOFlex predict aims to estimate the performance impact of a given I/O configuration
- The model outputs a performance improvement ratio, defined as run_bandwidth / dflt_bandwidth (absolute value)
- We use the data generated from previous I/O tuning runs to train and test the prediction models



Evaluation of Classification & Regression Models

- We evaluate both classification and regression models to predict the performance impact of different I/O configurations

- **Data Preparation:** 675 samples, 18 features; one-hot encoding applied
- **Models:**
  - **Classification:** RandomForest Classifier, XGboost Classifier, and Logistic Regression
  - **Regression:** XGBoost Regressor, Random Forest, and Decision Tree Regressor
- **Validation:**
  - Used Stratified K-Fold (k=10) cross-validation to ensure balanced class distribution across folds
  - average(score) ± std(score) for 10-folds

- The best results were obtained by the XGboost models

Classification Metrics	XGboost Classifier	Regression Metrics	XGboost Regressor
Accuracy	0.933 ± 0.027	**MAE**	0.2692 ± 0.070
Precision	0.944 ± 0.056	**RMSE**	0.3720 ± 0.105
Recall	0.872 ± 0.078	**R^2**	0.8884 ± 0.079
F1-score	0.903 ± 0.043	**SMAPE**	12.4250 ± 2.788

Conclusion and Wrap-Up

- **TOPIO I/O**:
  - Designed the IOFlex framework to optimize I/O configurations efficiently
  - Demonstrated the importance of our technique by applying it to MPAS
  - Acheived good I/O perfromance after auto-tuning
  - Improve the accuracy of our prediction models and embed them in the tuning phase.

https://github.com/safaad/ioflex

- **TOPIO Compression:**
  - Evaluated multiple lossless and lossy compression techniques for MPAS model output data
  - Identified the effectiveness of Lossy compression methods such as BigWhoop (inhouse developed) and SZ3 for upper air data, given its large volume and multi-dimensional structure
  - Concluded that lossless compression can be utilized for the surface layer data due to its smaller size
  - Explored the potential of applying machine learning techniques in compression

https://github.com/ptvogler/BigWhoop.git

Thank you

Questions