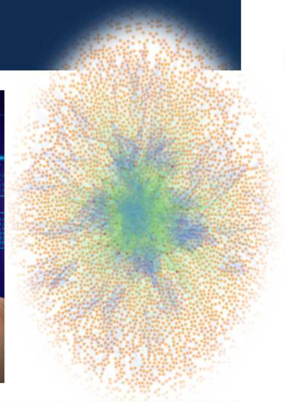
SAND2019-0456C



TuckerMPI: Optimised library for distributed data compression

Grey Ballard (WFU), Alicia Klinvex (NNL), Tammy Kolda, Gavin Baker, Tom Otahal, Prashant Rai, Drew Lewis, Ron Oldfield, ***Hemanth Kolla*** (SNL).

Sandia National Laboratories

Exceptional service in the national interest

ECP Annual Meeting: Lossy Data Compression Breakout

Jan 16th, 2019, Houston

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

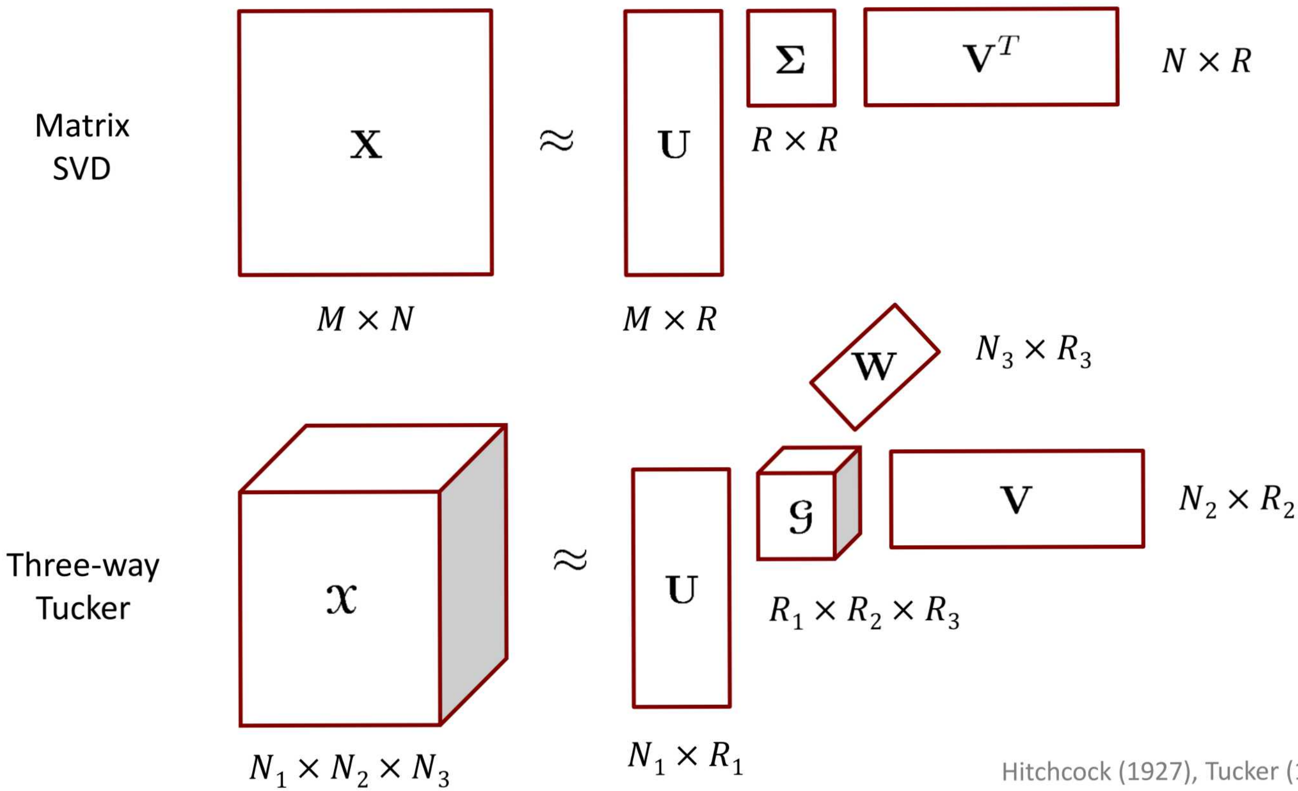
Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC.,a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

High level Objective

- Enable orders-of-magnitude (lossy) compression:
  - distributed data, *in-situ.*
  - scalable, efficient.
- Allow exploring the accuracy-compression-cost tradeoff.
- What are downstream implications (archival, analysis, sharing)?
- How do workflows change:
  - in-situ vs in-tandem vs offline.
  - Can we reconstruct partially?
  - Characterizing risks and mitigation factors.
- Augment the *in-situ* capability for SNL-ATDM apps.

Tucker Compression: Overview

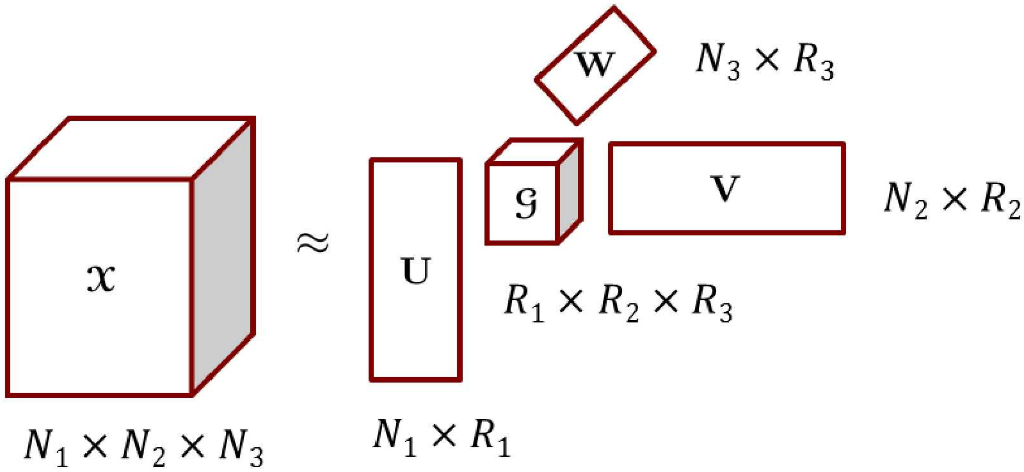
Extend Matrix SVD to multi-way arrays (tensors)



$$\text{CR} \approx \frac{N_1 \times N_2 \times N_3}{R_1 \times R_2 \times R_3}$$

Hitchcock (1927), Tucker (1966)

Tucker Compression - Overview



- For specified relative error, choose projection ranks R_1, R_2, R_3 such that:

$$\|\mathcal{X} - (\mathcal{G} \times_1 \mathbf{U} \times_2 \mathbf{V} \times_3 \mathbf{W})\| \leq \epsilon \|\mathcal{X}\|$$

- Find orthogonal matrices **U**, **V**, **W** that reduce the tensor size but retain its "mass".

$$\mathcal{G} = \mathcal{X} \times_1 \mathbf{U}' \times_2 \mathbf{V}' \times_3 \mathbf{W}' \quad \Longrightarrow \quad \|\mathcal{X}\|^2 - \|\mathcal{G}\|^2 \leq \epsilon^2 \|\mathcal{X}\|^2$$

- User can specify either ϵ or $R_1/R_2/R_3$.

Algorithm: ST-HOSVD

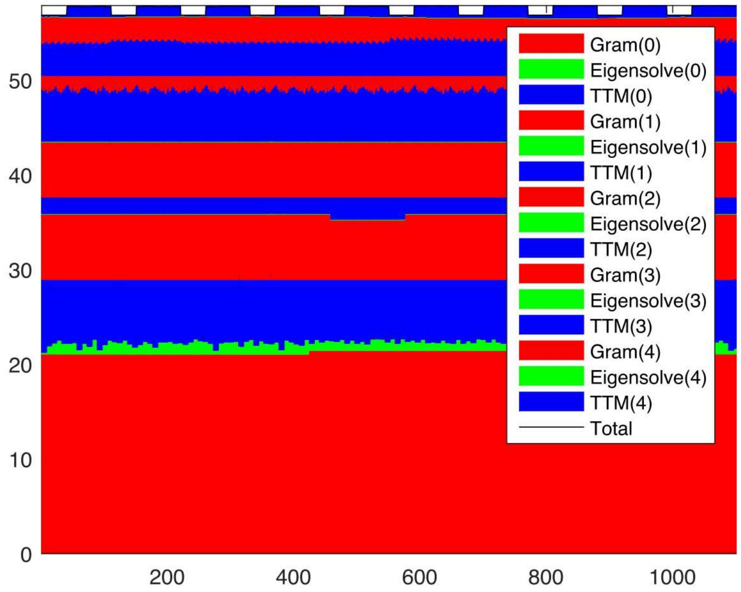
Sandia National Laboratories

1. Choose $\mathbf{U}$ with projection rank R_1 such that: $\|\mathbf{X}_{(1)}\|^2 - \|\mathbf{U}'\mathbf{X}_{(1)}\|^2 \leq \epsilon^2\|\mathcal{X}\|^2/3$
    a) Compute gram matrix: $\mathbf{X}_{(1)}\mathbf{X}_{(1)}'$
    b) Use eigendecomposition of N_1 x N_1 matrix to choose R_1
    c) Set $\mathbf{U}$ = R_1 leading eigenvectors of gram matrix
2. Shrink to size R_1 x N_2 x N_3: $\mathcal{Y} = \mathcal{X} \times_1 \mathbf{U}'$
3. Choose $\mathbf{V}$ with projection rank R_2 such that: $\|\mathbf{Y}_{(2)}\|^2 - \|\mathbf{V}'\mathbf{Y}_{(2)}\|^2 \leq \epsilon^2\|\mathcal{X}\|^2/3$
    a) Compute gram matrix: $\mathbf{Y}_{(2)}\mathbf{Y}_{(2)}'$
    b) Use eigendecomposition of N_2 x N_2 matrix to choose R_2
    c) Set $\mathbf{V}$ = R_2 leading eigenvectors of gram matrix
4. Shrink to size R_1 x R_2 x N_3: $\mathcal{Z} = \mathcal{Y} \times_2 \mathbf{V}'$
5. Choose $\mathbf{W}$ with projection rank R_3 such that: $\|\mathbf{Z}_{(3)}\|^2 - \|\mathbf{W}'\mathbf{Z}_{(3)}\|^2 \leq \epsilon^2\|\mathcal{X}\|^2/3$
    a) Compute gram matrix: $\mathbf{Z}_{(3)}\mathbf{Z}_{(3)}'$
    b) Use eigendecomposition of N_3 x N_3 matrix to choose R_3
    c) Set $\mathbf{W}$ = R_3 leading eigenvectors of gram matrix
6. Shrink to size R_1 x R_2 x R_3: $\mathcal{G} = \mathcal{Z} \times_3 \mathbf{W}'$

Vannieuwenhoven, Vandebril, Meerbergen (SISC 2012)

Scalable Parallel implementation

- Three main kernels (Gram, Evecs, TTM) have been parallelized.
- MPI implementation of Tucker available (open):
  - `git@gitlab.com:tensors/TuckerMPI.git`



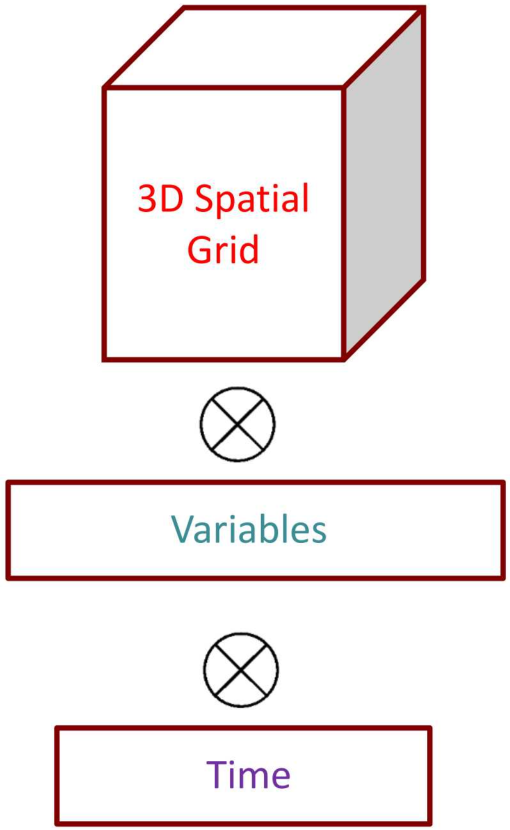
- 4.4TB -> 10GB (410X).
- Total of 55s; 1100 cores.
- Bulk of time is in first mode (GRAM computation).
- Fast BLAS for Eigensolve.
- Time for I/O is order of magnitude greater (~450s)

[1] W. Austin, G. Ballard and T.G. Kolda, IPDPS'16 (arXiv:1510.06689).
[2] Tammy Kolda, SC16 talk.

TuckerMPI: Usage

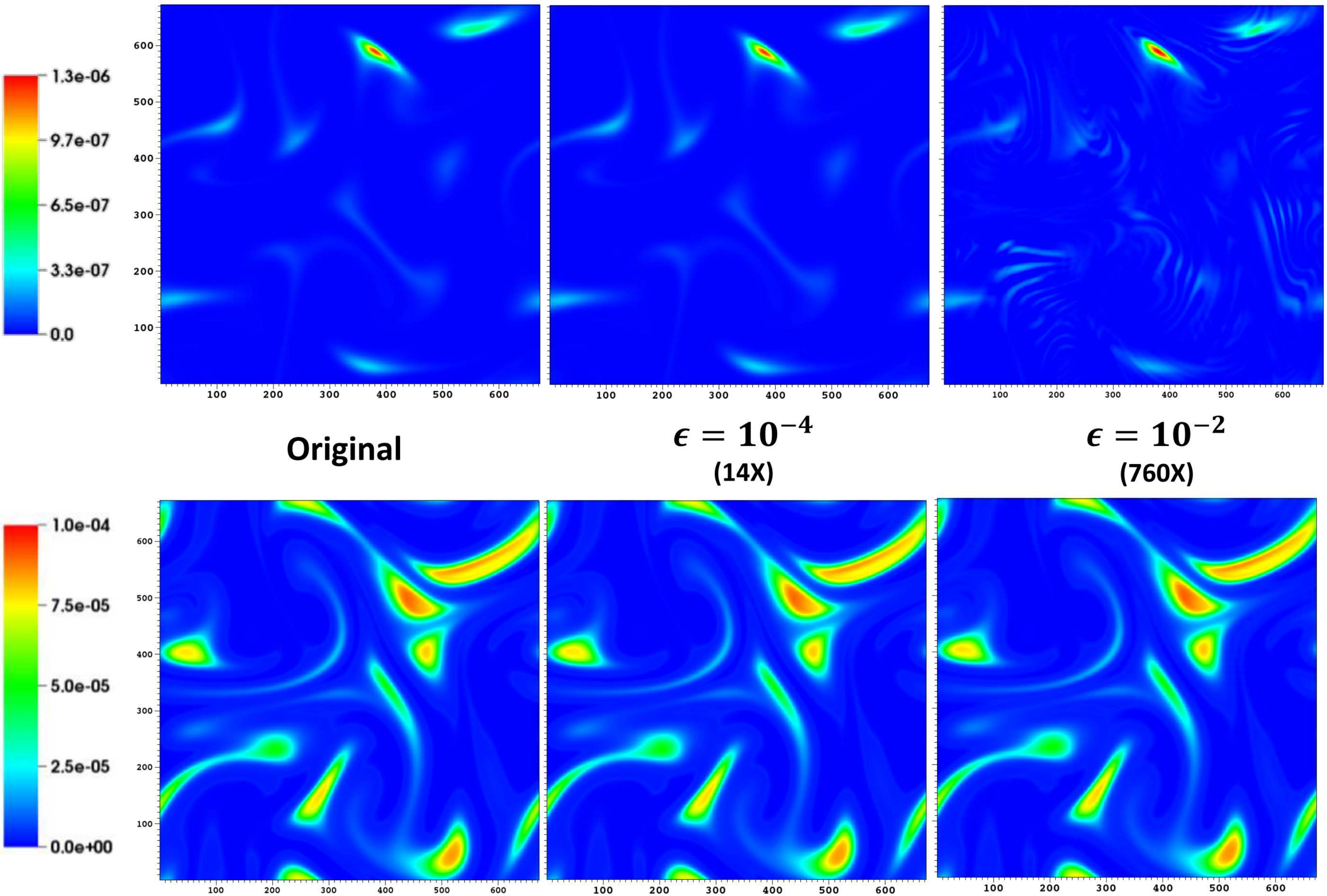
- Primary use case: structured mesh distributed data sets
  - Recently extended to handle multi-block capability.
  - Not currently usable for unstructured meshes, AMR meshes, particle data.
- Clean, portable C++ library with minimal dependencies (MPI, BLAS/LAPACK).
- Currently set up to work with parallel MPI-IO (read/write).
- Can be used offline, or *in-situ* (later slides).
- User need only specify:
  - Global dimensions of input data (tensor)
  - Desired global error threshold (ϵ), OR,
  - Desired ranks of truncation along each mode ($R_1/R_2/R_3$)

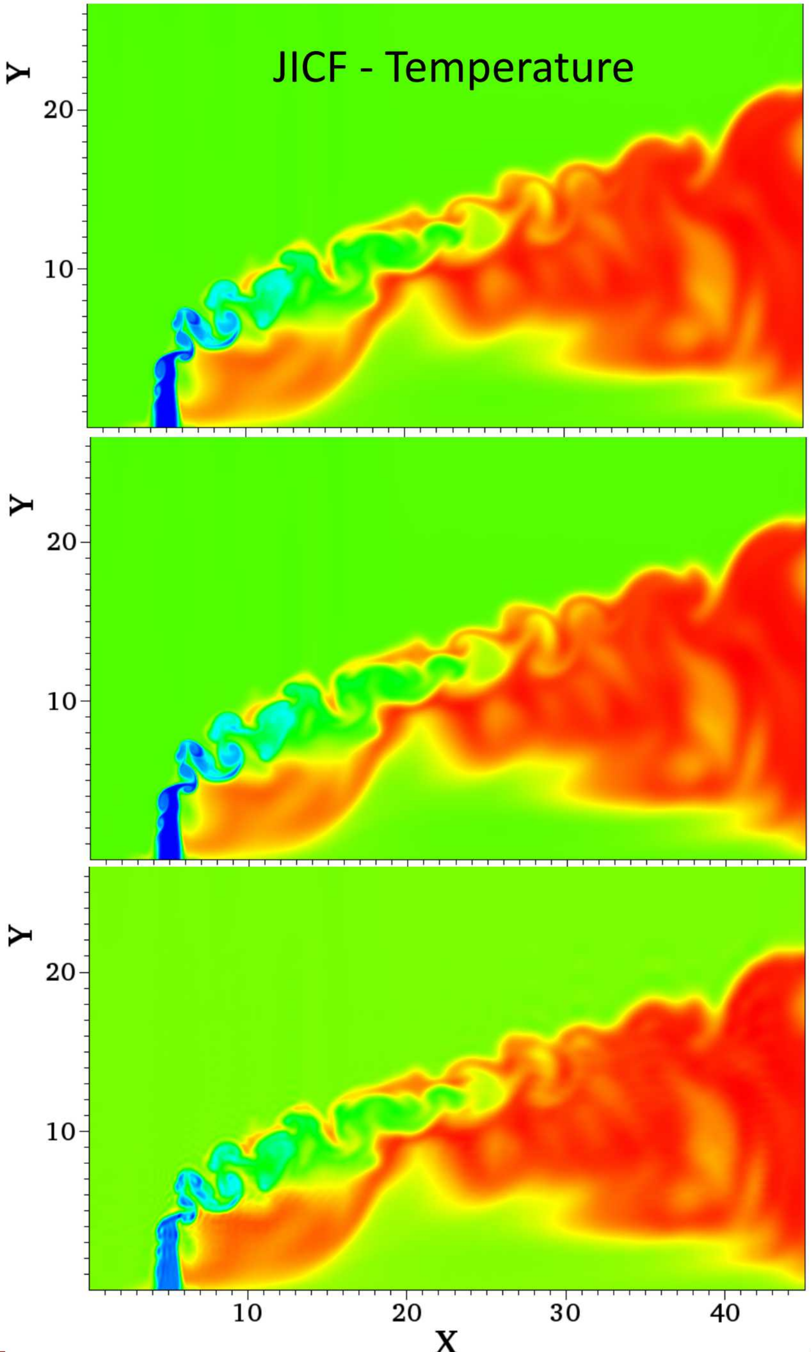
Sample Results: S3D datasets



	Original	$\epsilon = 10^{-4}$	$\epsilon = 10^{-2}$
HCCI	672 x 672 x 32 x 626	330 x 310 x 31 x 199 **(14 X)**	111 x 105 x 22 x 46 **(760 X)**
SP	500 x 500 x 500 x 11 x 400	95 x 129 x 125 x 7 x 125 **(410 X)**	30 x 38 x 35 x 6 x 11 **(20000 X)**
JICF	1500 x 2080 x 1500 x 18 x 10	424 x 387 x 261 x 18 x 10 **(110 X)**	90 x 61 x 48 x 13 x 6 **(40000 X)**

HCCI – OH concentration

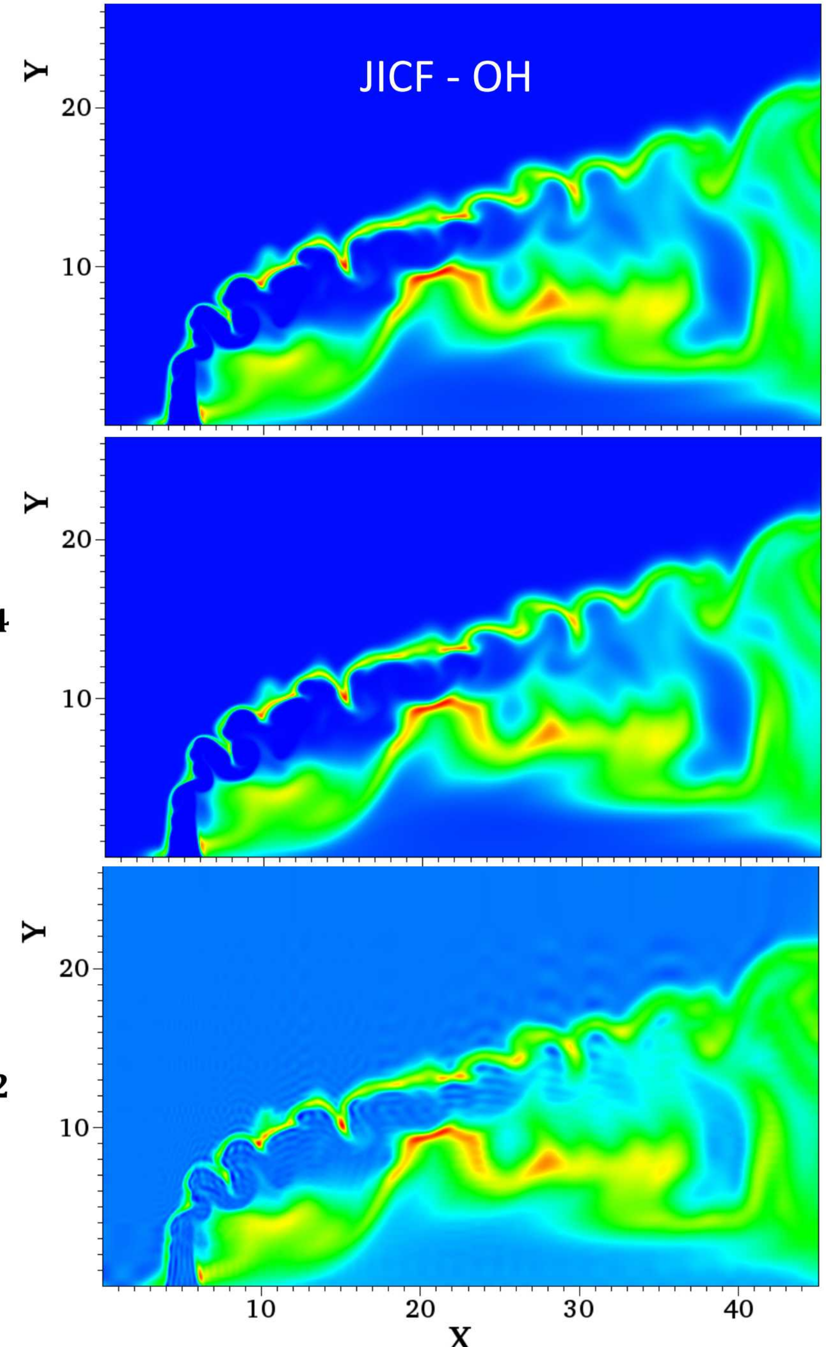




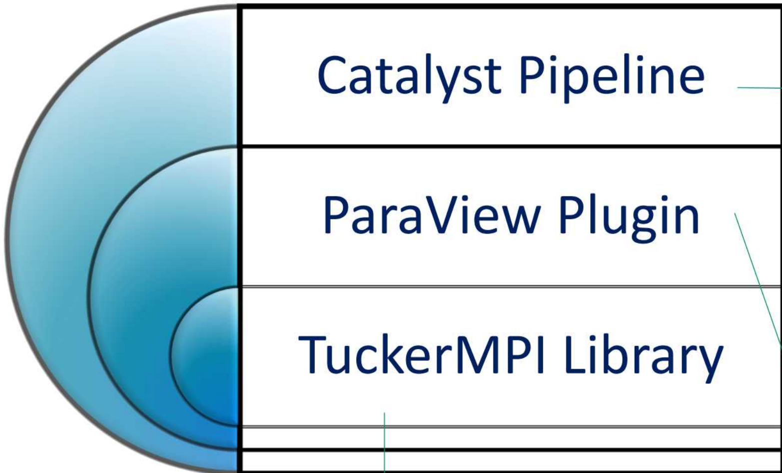
Original

$\epsilon = 10^{-4}$
(110X)

$\epsilon = 10^{-2}$
(40000X)



In-Situ capability with ParaView/Catalyst




```
def DoCoProcessing(datadescription):

    global coprocessor
    coprocessor.UpdateProducers(datadescription)

    LoadPlugin(‘/path/to/plugin/libSMTuckerMPICompression.so')

    vtk_test = coprocessor.CreateProducer(datadescription, 'input')

    ts = str(datadescription.GetTimeStep())
    SaveData('grid_compressed_' + ts + '.tucker', proxy=vtk_test)
```

- Dynamic library interface to TuckerMPI
- API for Tensor data structures, MPI domain decomposition maps/objects
- API for key kernels: tensor matricization, ST-HOSVD, Gram matrix computation.

- Plugin can be loaded into the ParaView GUI or Catalyst in-situ
- Plugin maps between ParaView and TuckerMPI data structures in parallel, MPI communicator re-mapping, and supports input types of vtkMultiBlockDataSet, vtkImageData, and vtkStructuredGrid
- Writer compresses vtkMultiBlockDataSet input by iterating over all blocks in parallel.

Ongoing Work

- In-situ reader (i.e. reconstruction) ParaView plugin.
- User-specified partial reconstruction.
- In-situ analysis on compressed data.
- Compress time snapshots in streaming manner.
- Compression of unstructured mesh data:
  - Key Idea: Unstructured mesh data are realizations of multi-D function
  - Compress by seeking low rank approximations of multi-D functions.

Thank you!