jh220031

Supercomputer resources:
Tsubame3.0 @Tokyo Tech, BDEC @ Univ. Tokyo

Targeting exa-scale systems: performance portability and scalable data analysis

Representative:	**Y. Asahi (JAEA)**	Code development
Deputy Representative:	S. Maeyama (Nagoya Univ.)	Plasma turbulence
Deputy Representative:	J. Bigot (MdlS, France)	Scalable data analysis
Collaborating researcher:	X. Garbet (CEA, France)	Global plasma turbulence
Collaborating researcher:	V. Grandgirard (CEA, France)	Large scale simulation
Collaborating researcher:	K. Obrejan (CEA, France)	Large scale simulation
Collaborating researcher:	T. Padioleau (MdlS, France)	Performance portability
Collaborating researcher:	K. Fujii (Kyoto Univ.)	Machine learning
Collaborating researcher:	T. Shimokawabe (Univ Tokyo,)	Deep learning
Collaborating researcher:	T.-H. Watanabe (Nagoya Univ.)	Local plasma turbulence
Collaborating researcher:	Y. Idomura (JAEA)	Large scale simulation
Collaborating researcher:	N. Onodera (JAEA)	Large scale simulation
Collaborating researcher:	Y. Hasegawa (JAEA)	Large scale simulation
Collaborating researcher:	T. Aoki (Tokyo Tech.)	Optimization on GPU

JHPCN 14th symposium, Shinagawa, Japan **Date: 7/July/2022**

Outline

Introduction

- Large scale computational fluid dynamics (CFD) simulation
- Performance portable implementation for exascale readiness
- Scalable data analysis method for exascale simulations

Performance portable implementation

- Testing of MPI + X with a kinetic plasma simulation code
- Preparation for exascale city wind flow simulations

Surrogate models for CFD simulations

- Rapid prediction of plume dispersion based with CNN + Transformer

In situ machine learning with loose coupling

- In-situ incremental PCA on large scale simulation data

Summary and future work

Preparation for exascale systems

Language

Directives/Higher level abstractions

+

OpenACC

OpenMP

Thrust

SUMMIT
NVIDIA V100

Frontier
AMD MI250X

Aurora
Intel Ponte Vecchio

Objective: compare stdpar with other frameworks

Language

VS

Directives/Higher level abstractions

OpenACC

OpenMP

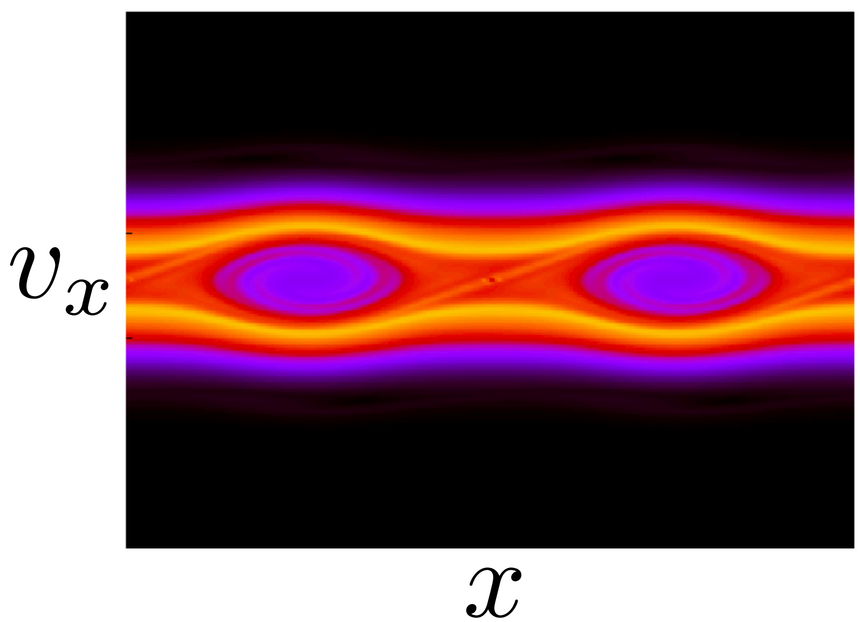
Thrust

Objectives

- Explore the GPU implementation with **C++ parallel algorithm**
- Maximize readability and productivity
- Evaluate performance portability across CPUs and GPUs
- Comparison with other frameworks

2D-2V Vlasov case: Mini-application for kinetic equation

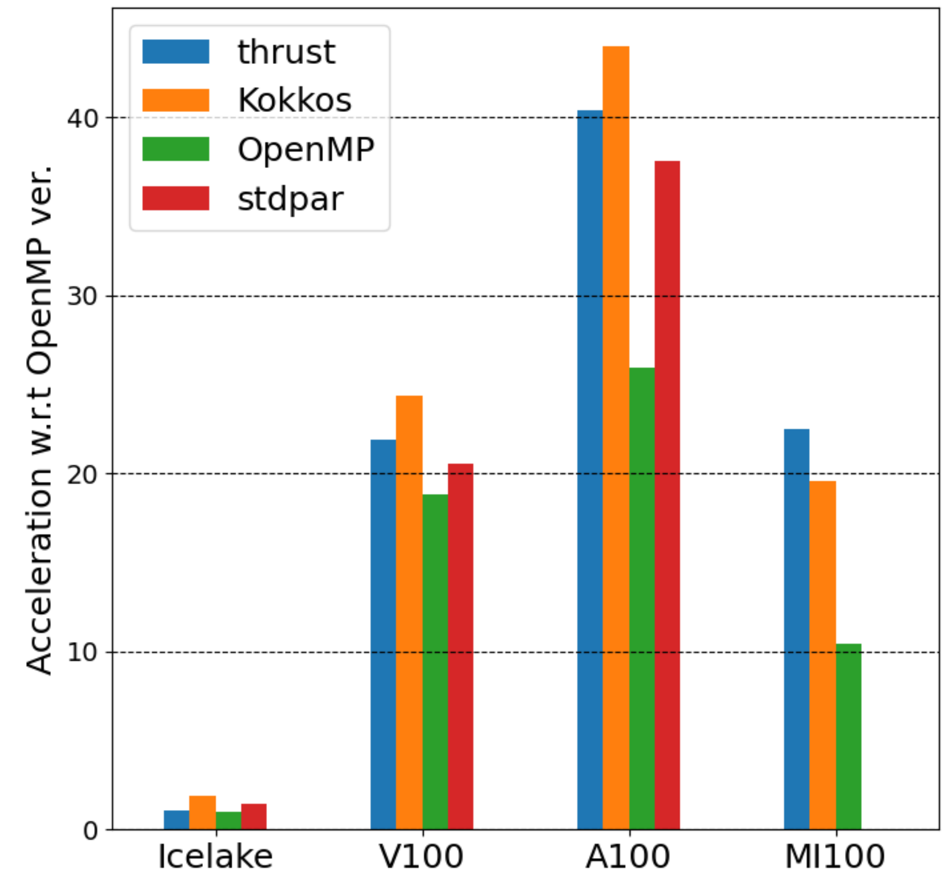
Problem size: 128^4
#Iterations: 128



4D advection with Strang splitting [1]

$$\frac{\partial f}{\partial t} + v_x \frac{\partial f}{\partial x} = 0 \text{ at } (y, v_x, v_y) \text{ fixed}$$
$$\frac{\partial f}{\partial t} + v_y \frac{\partial f}{\partial y} = 0 \text{ at } (x, v_x, v_y) \text{ fixed}$$
$$\frac{\partial f}{\partial t} + E_x \frac{\partial f}{\partial v_x} = 0 \text{ at } (x, y, v_y) \text{ fixed}$$
$$\frac{\partial f}{\partial t} + E_y \frac{\partial f}{\partial v_y} = 0 \text{ at } (x, y, v_x) \text{ fixed}$$

- **stdpar** version is quite competitive
- readability improved with **mdspan**



Velocity space integral (4D to 2D) appeared in Poisson equation

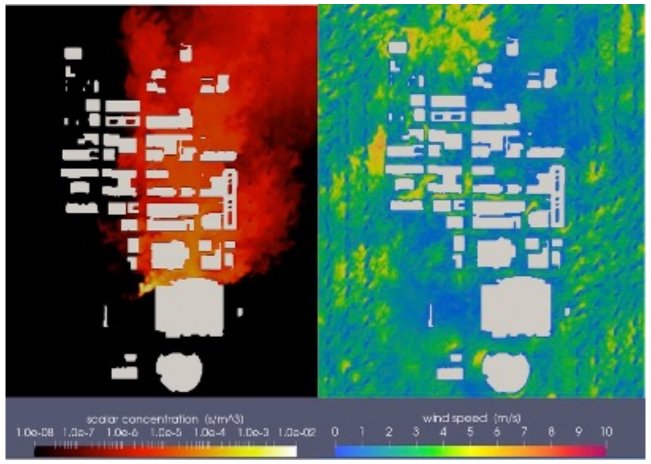
$$\rho(t, \mathbf{x}) = \int d\mathbf{v} f(t, \mathbf{x}, \mathbf{v})$$

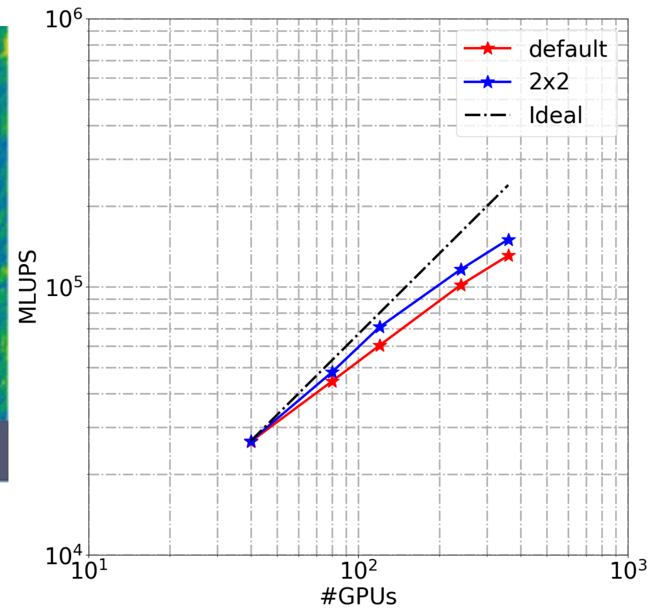
[1] G. Strang, et al, SIAM Journal on Numerical analysis (1968)
[2] Y. Asahi et al., OpenACC meeting, September, Japan
[3] Y. Asahi et al., waccpd (SC19), November, US
[4] https://github.com/yasahi-hpc/vlp4d

Y. Asahi et al, to be submitted

Exascale city wind flow simulation with CityLBM

Strong scaling up to 360 A100 GPUs

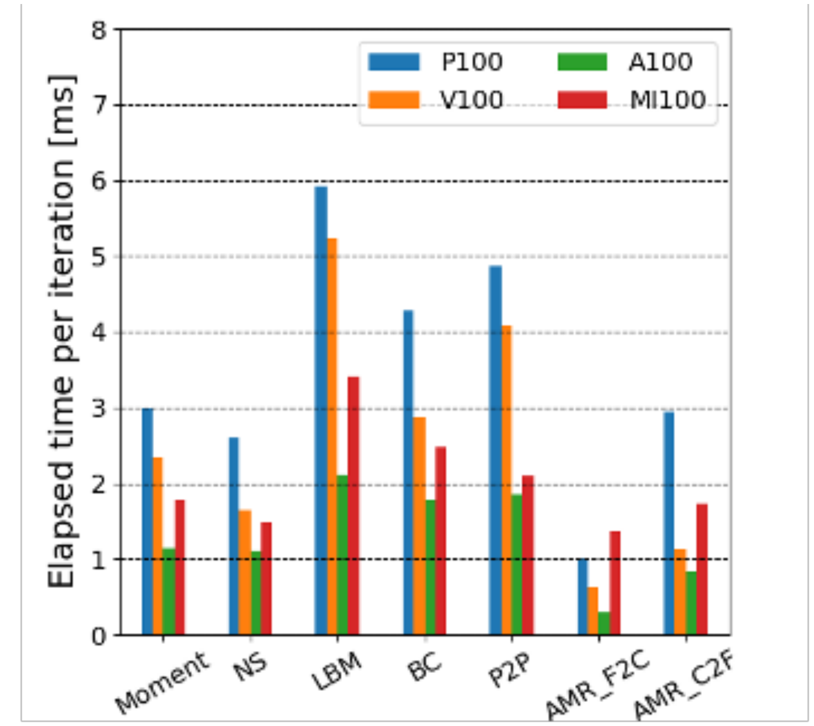




Oklahoma 5.8x5.8x0.8km with 1m grid

Good scalability with process mapping

AMD GPU readiness with HIP



Most of the computational kernels have been ported successfully

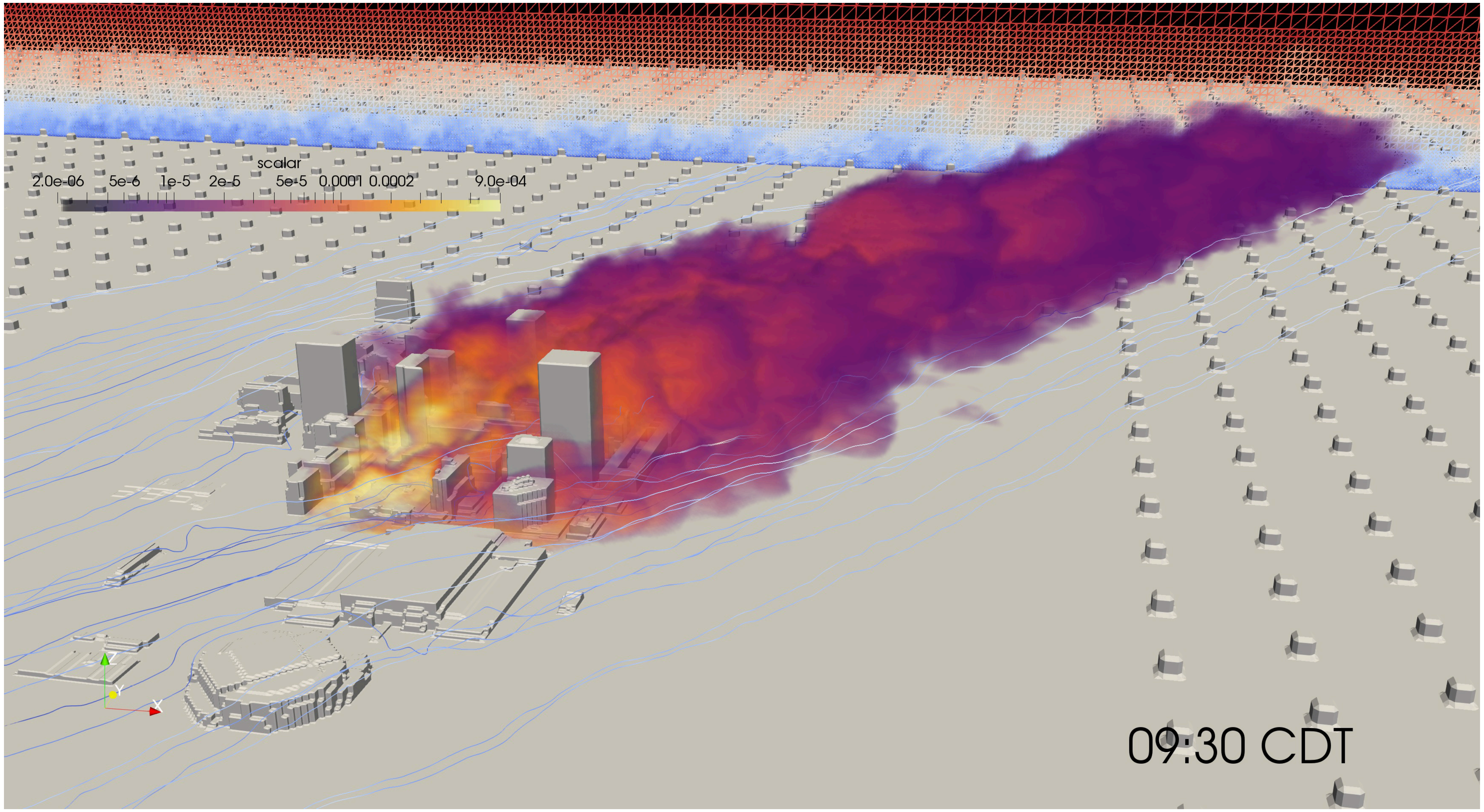
- Achieving a **real time high-resolution** simulations (1m) with **ensemble** data assimilation.

  → **Exascale** city wind flow simulations on Frontier

- Additional optimization needed for the AMR interpolation kernels (AMR_F2C, AMR_C2F) which involves irregular memory accesses

Frontier
AMD MI250X

Emergence plume dispersion prediction in an urban area



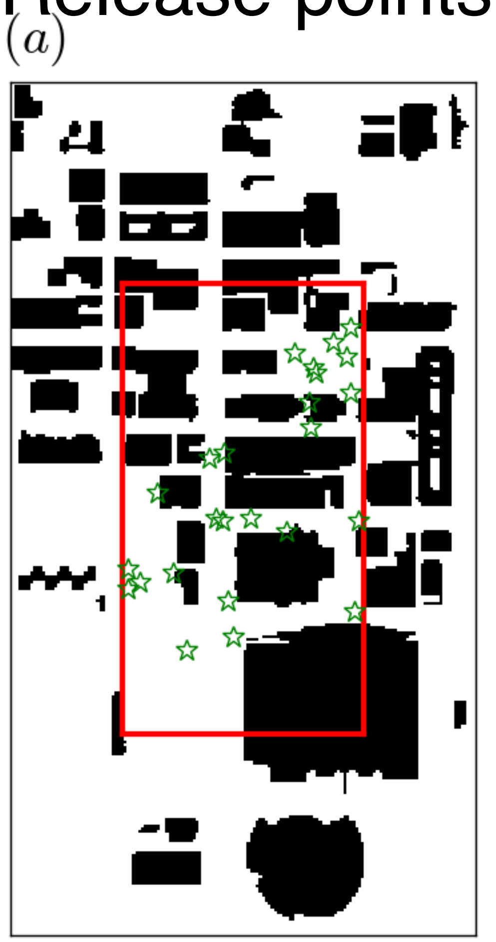
- Rapid prediction of the dispersion of harmful substances in an urban area
- "Real time" numerical simulation requires enormous computational costs [1]
- Deep learning based surrogate model for rapid and accurate prediction

[1] N. Onodera, et al, 2021

Simulation settings and dataset

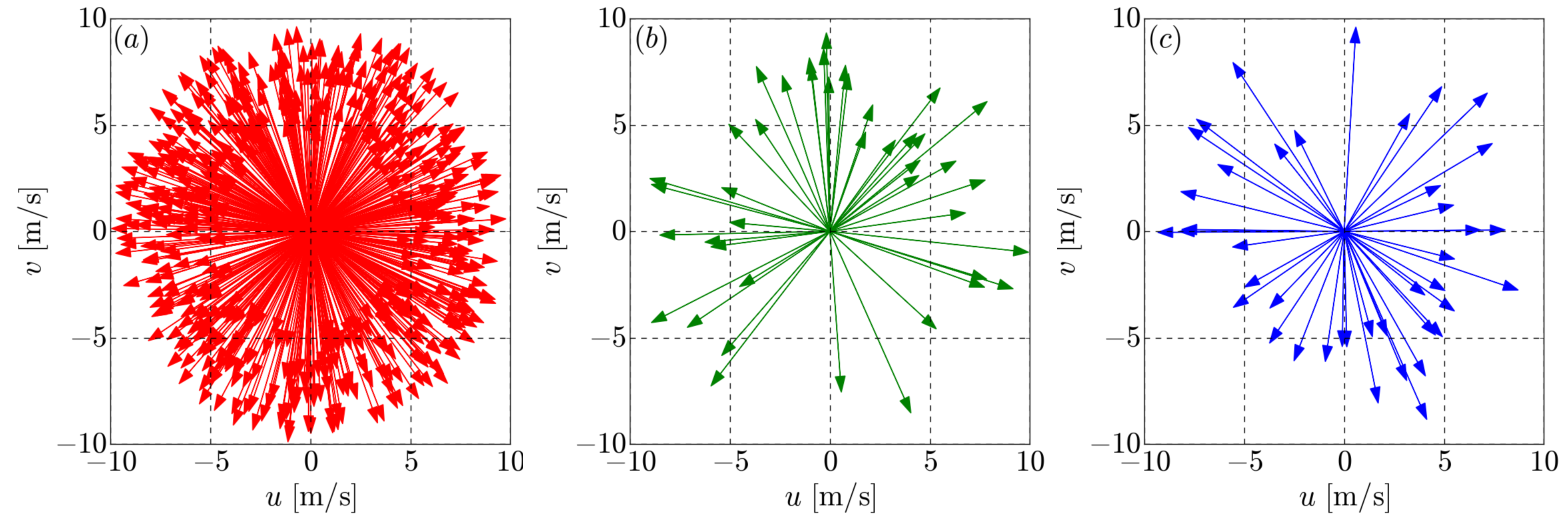
- Simulation for Oklahoma City: uniform flow condition (nudging at edge)
- Release points are set randomly

Release points

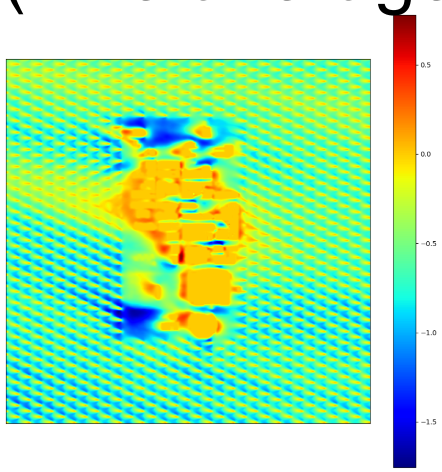


(Fixed) stations

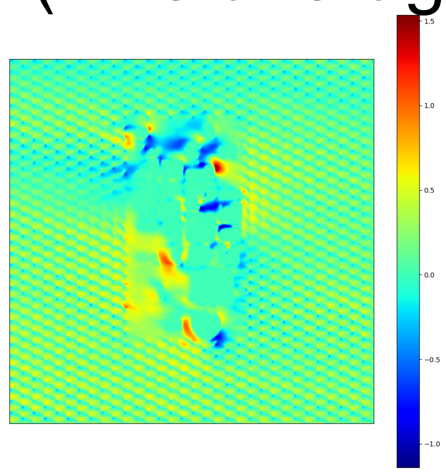
Train Val Test



u (time averaged)



v (time averaged)



- 650 cases with random flow directions
- Vertical profiles of u, v given by power law

$$\overline{u} = u_{\mathrm{ref}} \left(\frac{z - z_g}{z_{\mathrm{ref}}} \right)^{\alpha}$$

- 25 tracer particles released in each case

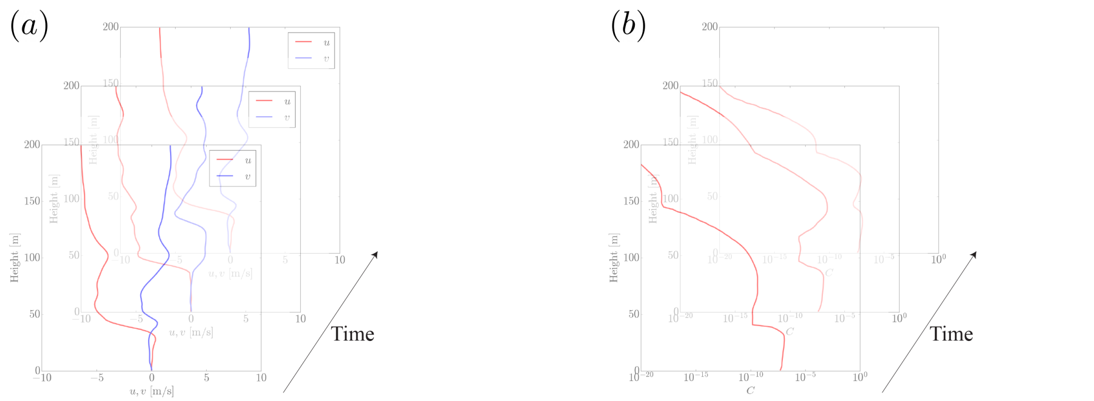
  3 distinguish time window

  650 (cases) x 25 (plumes) x 3 (time widows)
  = 37500

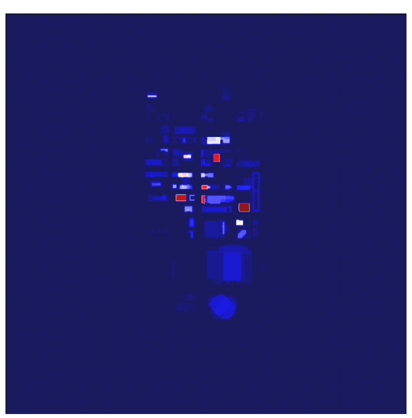
CityTransformer for dispersion prediction

Input

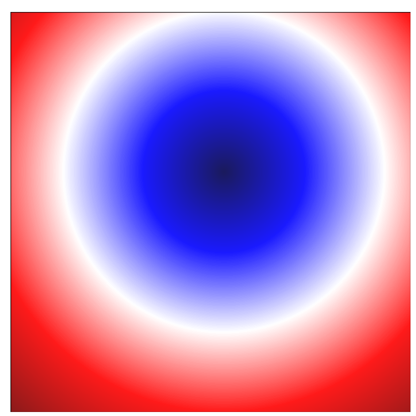
Monitoring time series data of flows and concentration



Building shapes

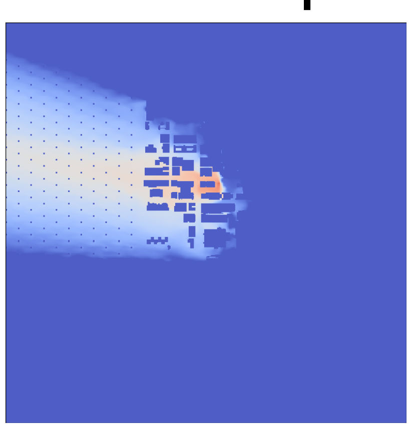


Release point

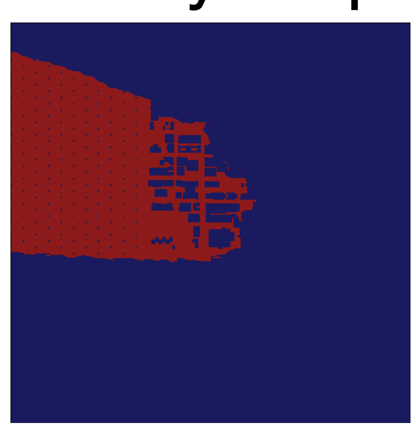


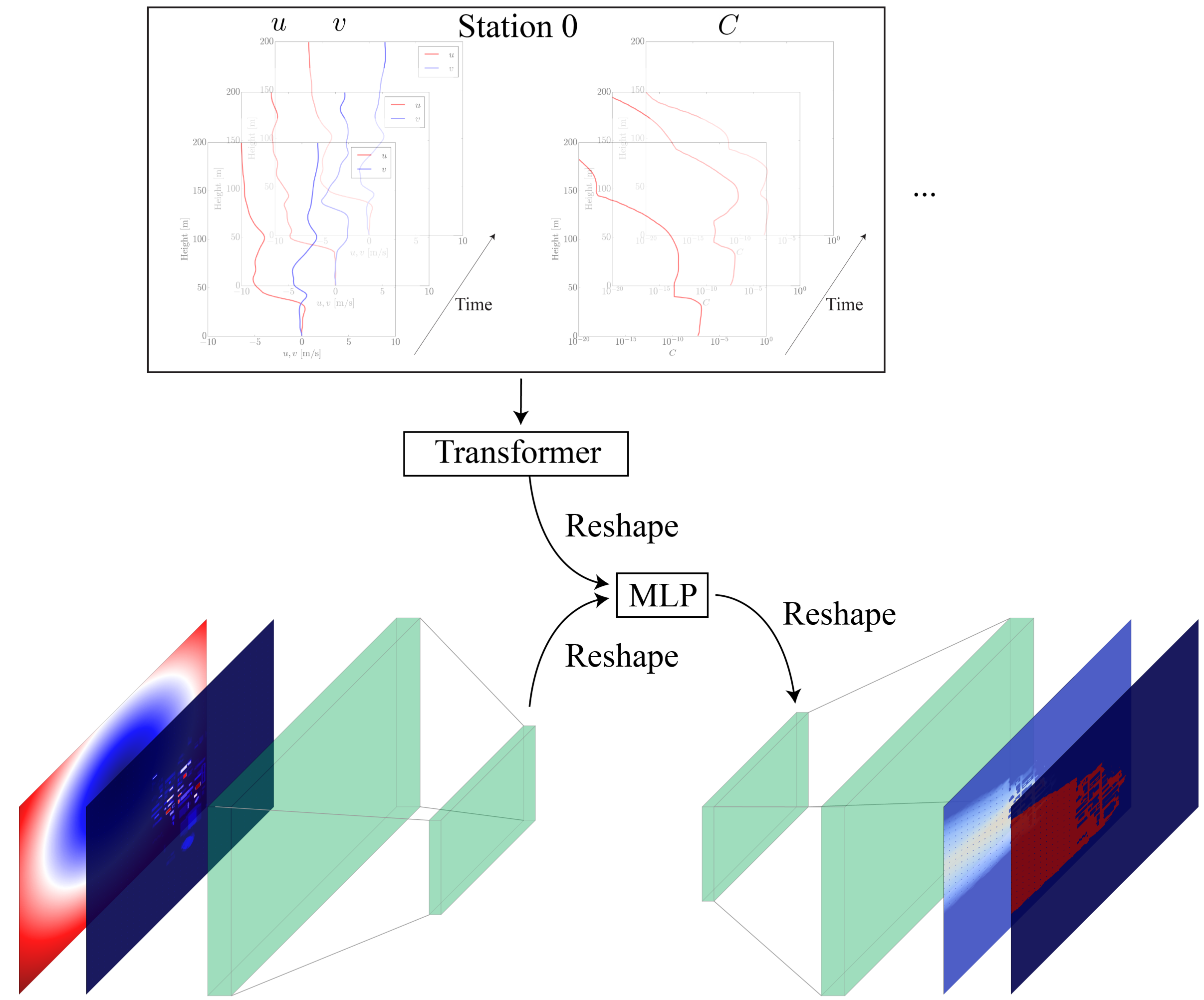
Output

Plume dispersion



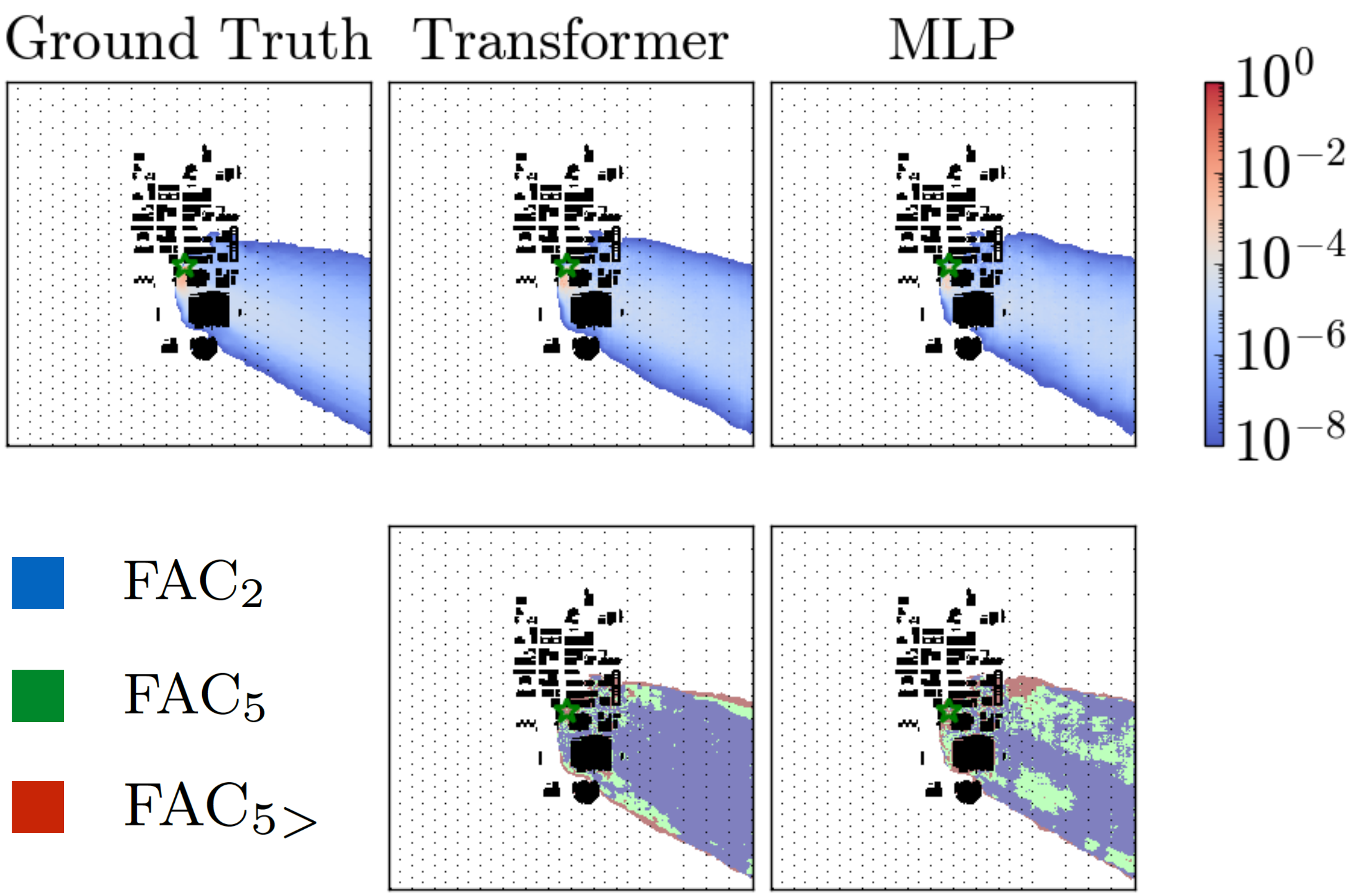
Binary map

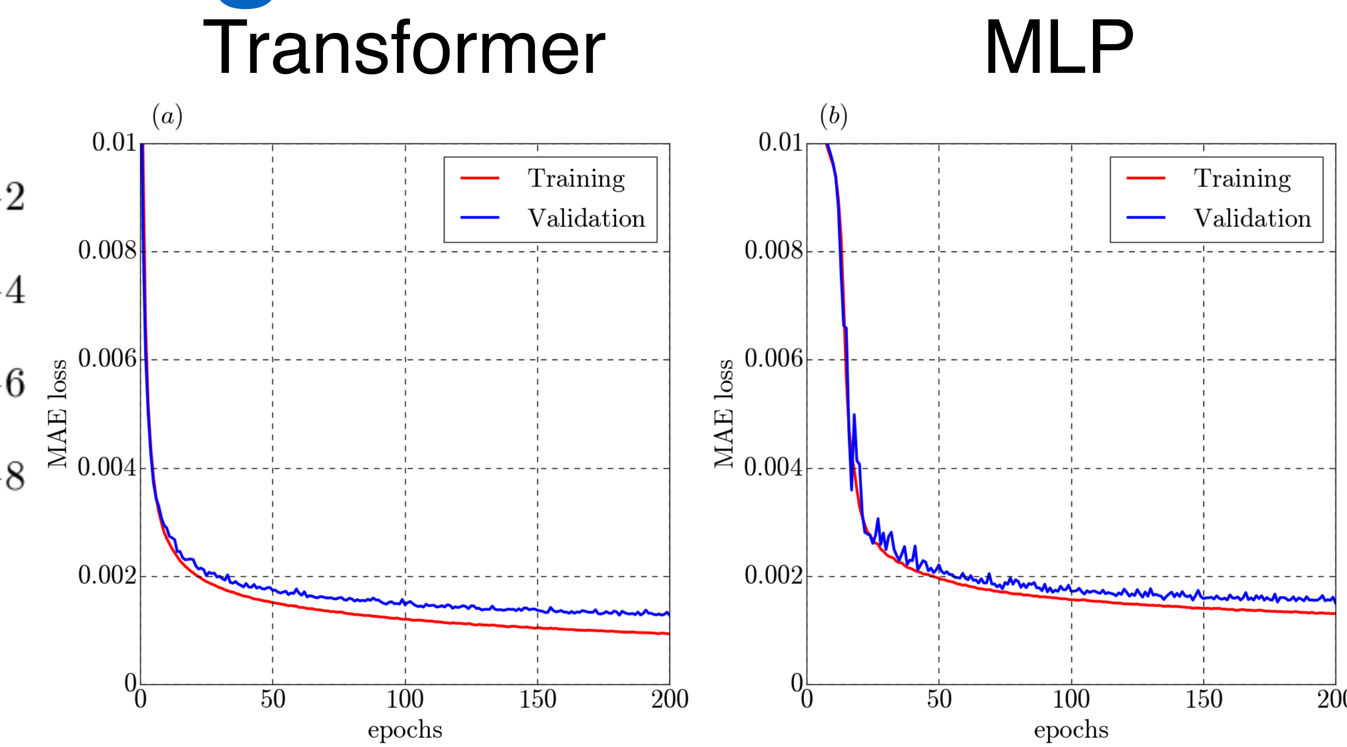




- **Plume concentration prediction** from building shapes and monitoring time series data
- Img2Img translation by CNN
- Encoding time series data by Transformer
- PyTorch and horovod (V100 x 4)

Transformer VS. MLP: using the time variation





- CNN/Transformer: Using the time **varying** monitoring data
- CNN/MLP: Using the time **averaged** monitoring data
- Both model gives reasonable prediction (mostly in the range of FAC2)
- Better performance of Transformer version: importance of **time variance** to improve the performance

Y. Asahi et al, under review

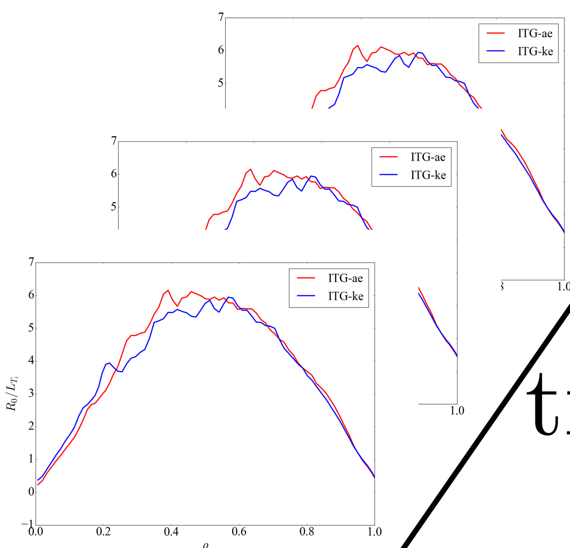
Analyzing 5D gyrokinetic simulation data

Conventional study

1D time series

Structures of radial profile

R_0/L_{ti}



time

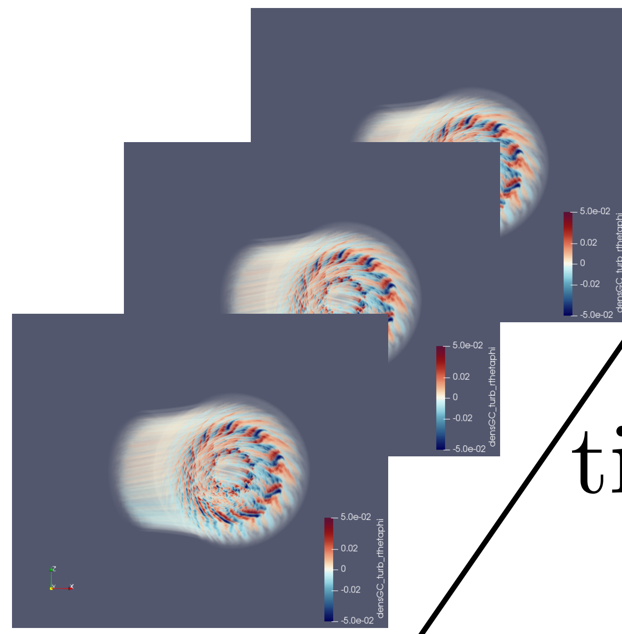
ρ

~kB

3D time series

Structures of Fluid moments

δn



time

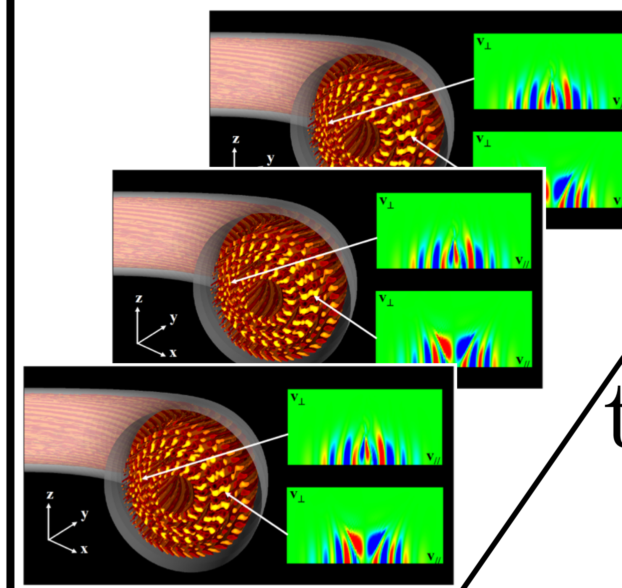
~10MB

This work

5D time series

Phase structure

δf



time

~10GB

High dimensional + huge data

Conventional Study: 3D structures (like convective cells), 1D structures (stair case, stiffness in temperature gradient)

This work [1]: Extracting phase space structure from the time series of 5D distribution function (pattern formation in phase space)

[1] Y. Asahi, et al., PoP, (2021)

Principal component analysis of distribution function

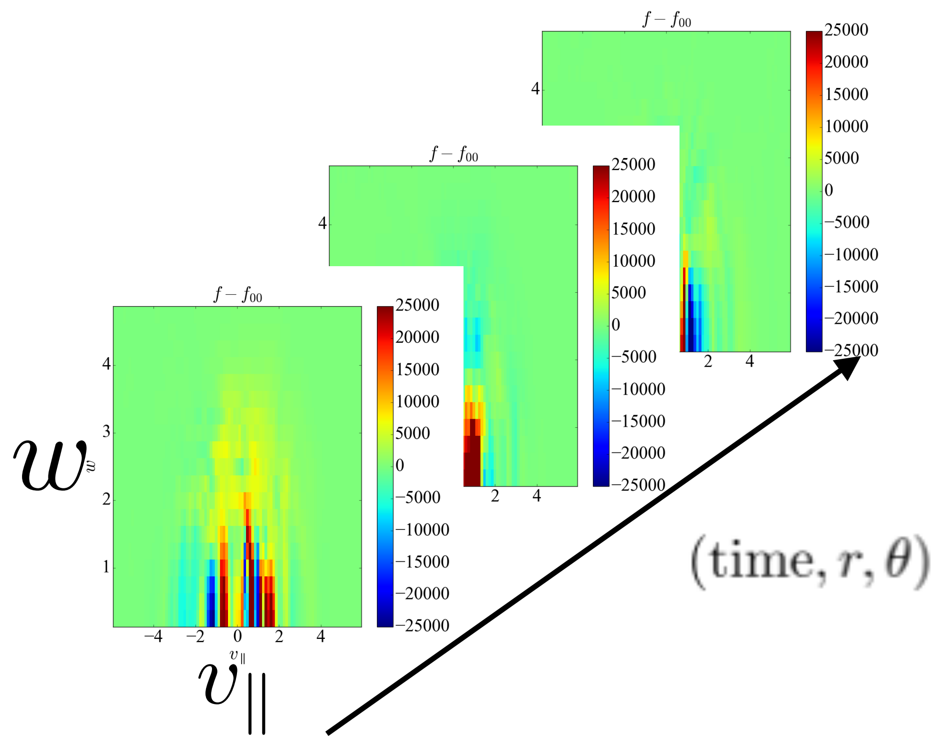
Analyzing 6D (3D space x 2D velocity x time) **Terabyte** data using Dask+Xarray

Input $(\mathrm{time}, r, \theta) \times (\varphi, v_{\|}, w)$

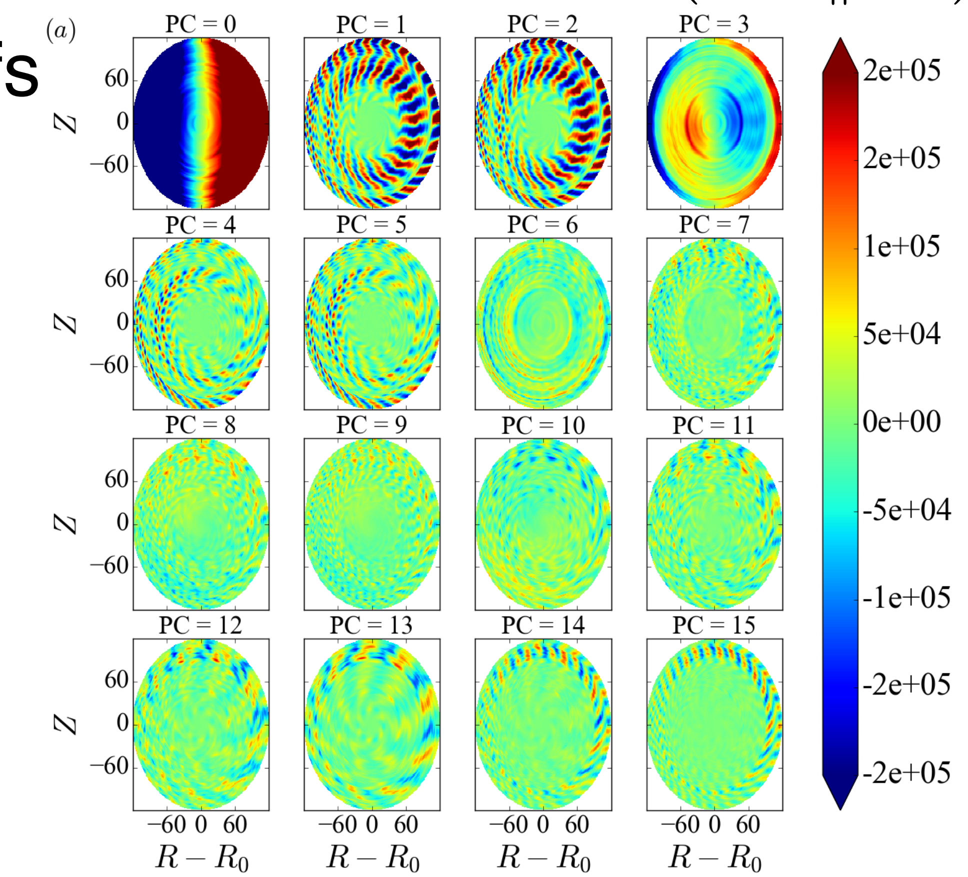
Samples Features

Output $(\mathrm{components}) \times (\varphi, v_{\|}, w)$

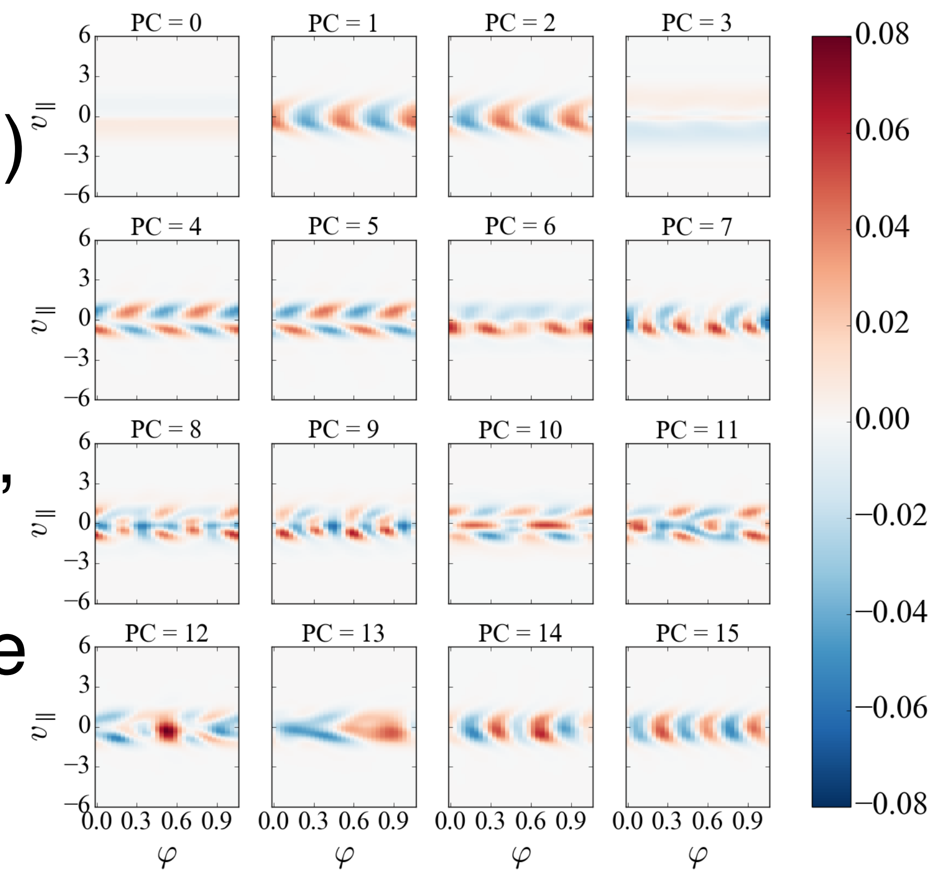
coefs



PCA

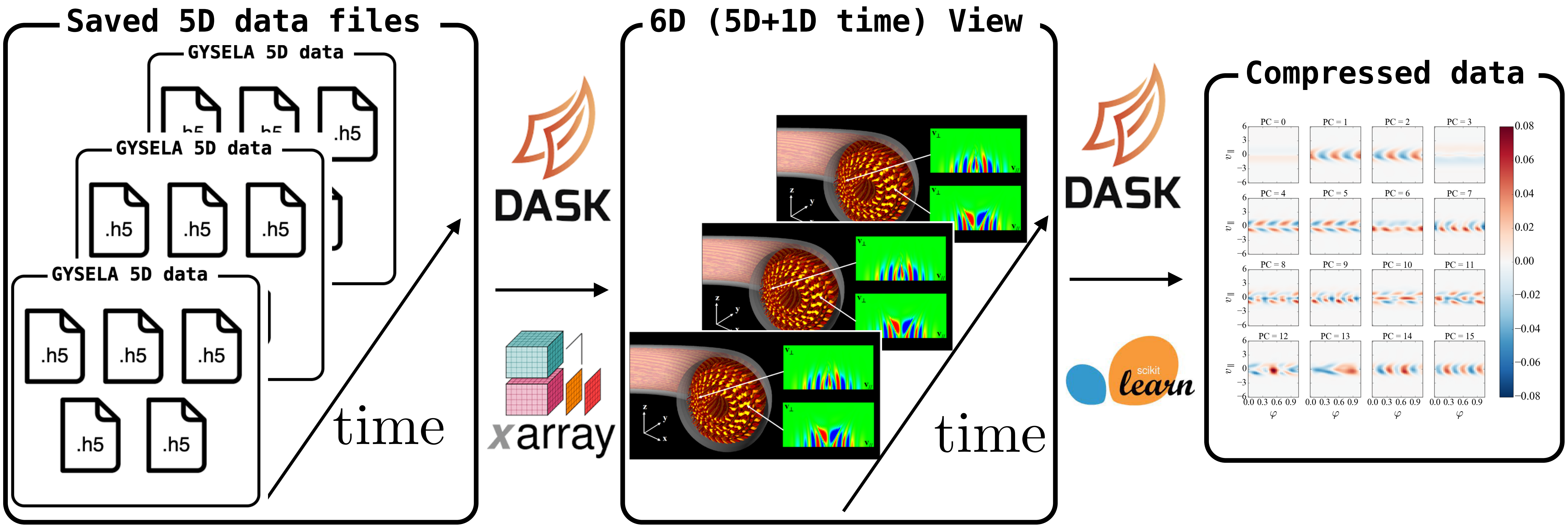


bases (w = 0)



- Easily manage **out-of-memory data** (> 1TB) without MPI parallelization
- **Interpretable PCs:** Magnetic geometry (PC 0), Ballooning modes (PC 1, 2)
- 3 order reduction (DOF: 10^12 to 10^9) of the data size

Incremental PCA (Conventional)



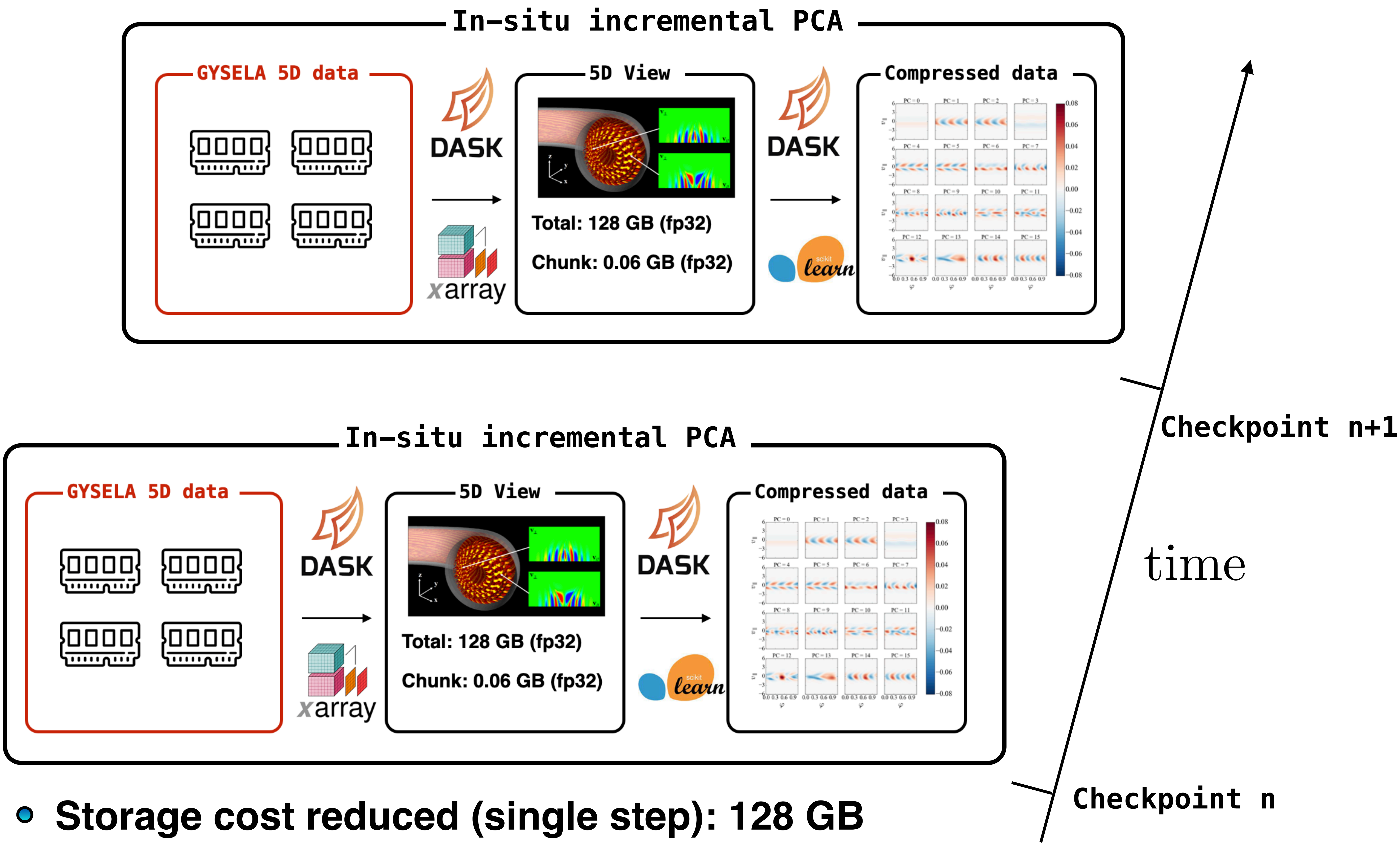
Overview:

- **All the 5D data kept on storage (10TB~100TB)**
- **5D series data managed as a Dask array (view)**
- **Incremental PCA (Incremental in time direction) on 5D series data**
- **Common basis on all the 5D data. Coefficients for each time step**

Issue:

- **enormous storage cost 128 GB x n steps ~ 10TB**

Incremental PCA using checkpoint data



- Storage cost reduced (single step): 128 GB
- Issue: To compute coefficients, another identical simulation needed

In-situ data analysis of Voice 1D+1V (w/o MPI) with PDI

Simulation

```
int iter = 0;
for (; iter < steps; ++iter) {
  // Computations

  // Evoke events by PDI
  PdiEvent("iteration")
    .with("iter", iter)
    .and_with("time", time)
    .and_with("f", f);
}
```

PDI [1] Interface

```
data:
  f_extents: { type: array,
               subtype: int64,
               size: 3 }
  f:
    type: array
    subtype: double
    size: [ '$f_extents[0]',
            '$f_extents[1]',
            '$f_extents[2]' ]

plugins:
  pycall:
    on_event: [iteration]
      - with: { time: $time,
                f: $f }
        exec:
          import insitu
          insitu.plot_f(f, time)
```

Python

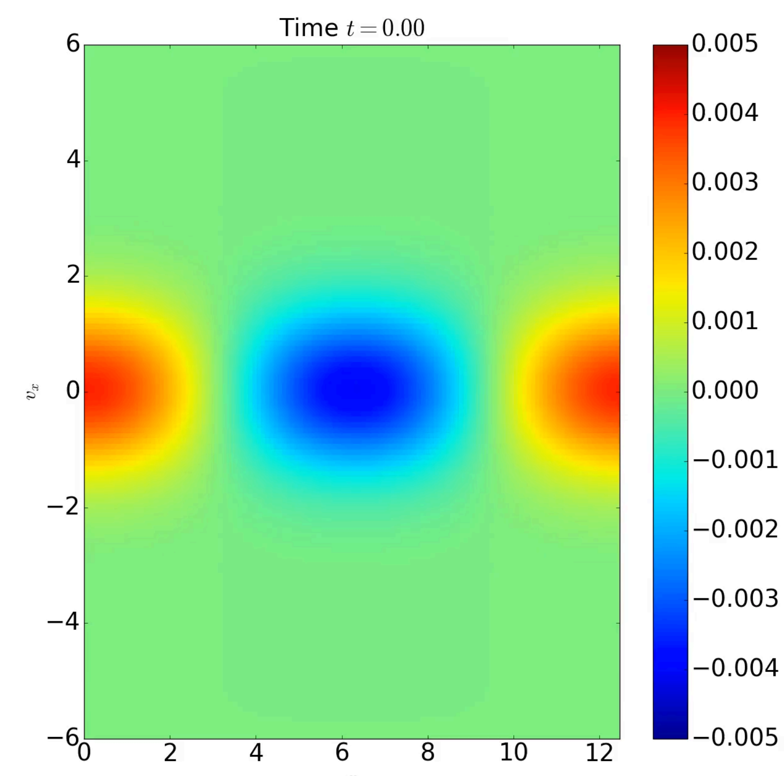
```
def plot_f(f, time):
    plot_(f, time, species=0)
```

Achieved

- In-situ visualization of f
- In-situ incremental PCA on f

On-going

- In-situ incremental PCA on GYSELA data with DEISA [2]



[1] https://pdi.dev/1.5/
[2] A. Gueroudji et al., HiPC, December 2021.

Summary

Performance portability for exascale simulation

- C++ parallel algorithm: Improved readability, performance and portability
- AMD GPU porting of CityLBM: preparation for exa simulations on Frontier

Surrogate models for CFD simulations

- DL-based surrogate model is fast while keeping the good accuracy

Scalable data analysis based on Dask

- Managing large scale data (> 10 TB) with Dask
- Preparation for in-situ incremental PCA on GYSELA data

Future works

- Performance evaluation for MPI + parallel algorithm (C++)
- In-situ machine learning with PDI + Dask (collaboration with J. Bigot)