Open Standard for Particle-Mesh Data

openPMD

HDF5 User Group Meeting

Columbus, OH
August 17th, 2023

Axel Huebl

Lawrence Berkeley National Laboratory

On behalf of the openPMD Community incl. content from
Franz Poeschel (CASUS), Lipeng Wan (GSU),
Norbert Podhorszki (ORNL), Junmin Gu (LBNL),
Maxence Thévenet (DESY), Erik Schnetter (PITP), Remi Lehe (LBNL)

NNSA National Nuclear Security Administration

CAMPA Consortium for Advanced Modeling of Particle Accelerators

ATAP ACCELERATOR TECHNOLOGY & APPLIED PHYSICS DIVISION

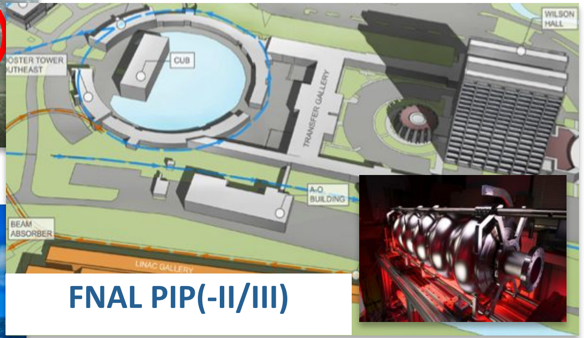
BERKELEY LAB Lawrence Berkeley National Laboratory

U.S. DEPARTMENT OF ENERGY Office of Science

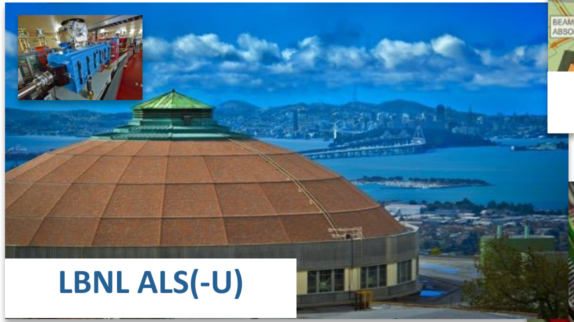
My Background: Particle Accelerator & Laser-Plasma Physics

Cyclotron Berkeley (CA), 1939

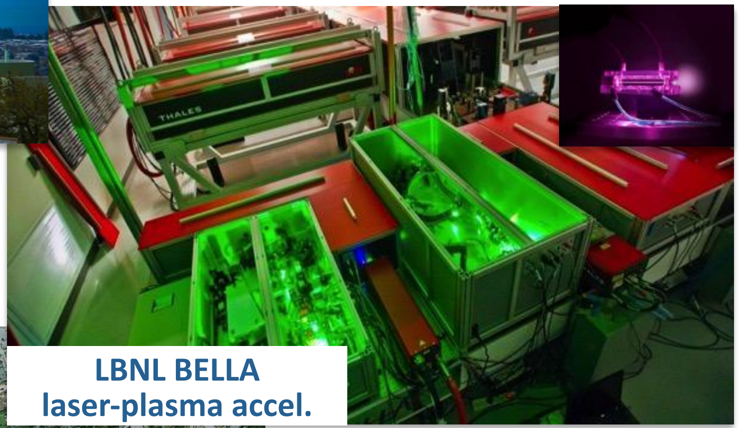
CERN (HL-)LHC



FNAL PIP(-II/III)



LBNL ALS(-U)



LBNL BELLA
laser-plasma accel.

EU XFEL
free electron laser

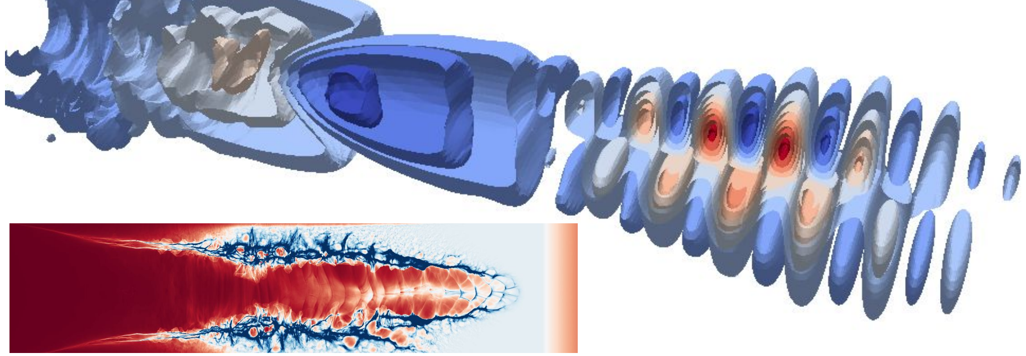
Selected Science Cases *A Need for Data Exchange*

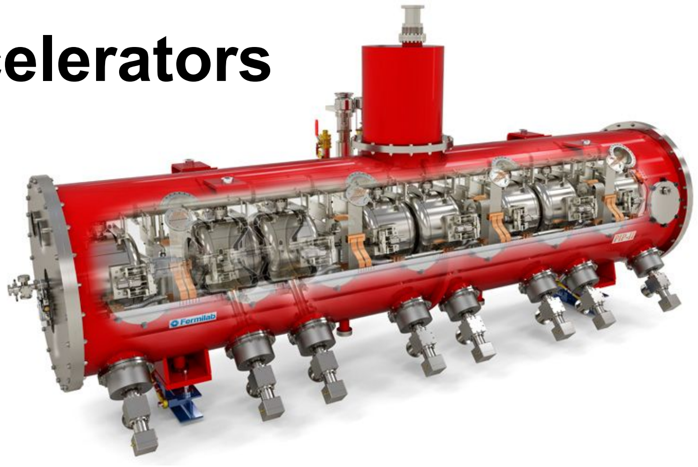
Laser-Plasma Physics

Particle Accelerators

Machine Learning

Simulation, Experiment

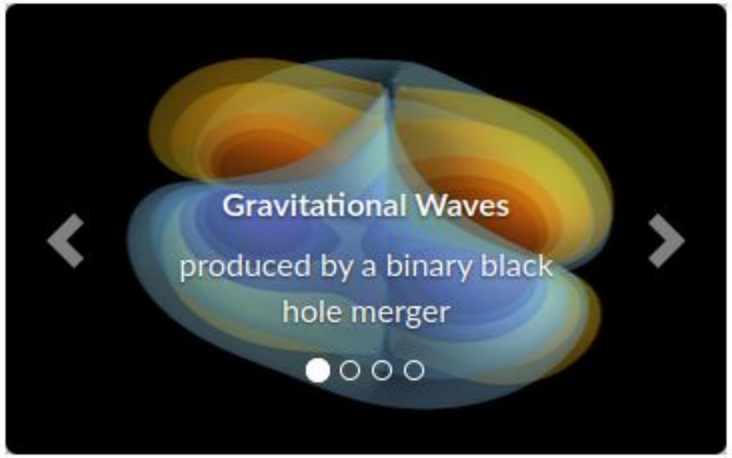




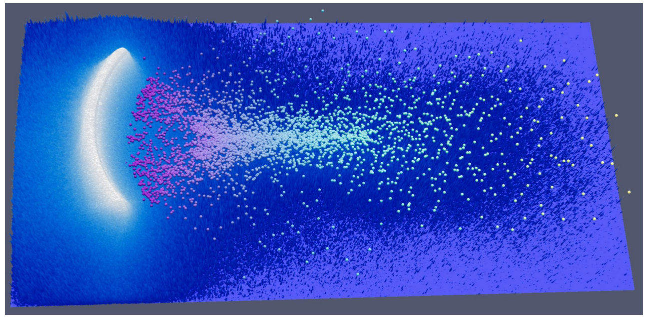
X-Ray Light Sources, FELs, Laser Pulses

General Relativity

The Einstein Toolkit



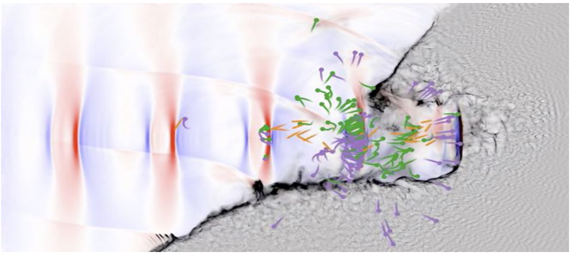
Fusion Energy Science



Astro-physical plasmas

Inertial Confinement Fusion
Laboratory Astrophysics
High-Energy Density Physics

High-Field Physics



Life Sciences
Particle-Based Image Analysis

Outline

- **openPMD Concepts**
  - design principles of our meta-data standard
  - components & modularity
  - example: sparse data for mesh-refinement

- **Open Ecosystem**
  - projects, libraries, tooling, integrations
  - open community

- **Selected R&D Highlights**
  - fast reads: data layouts & in situ statistics
  - from files to data streams

Design Principles

Open Standard for Particle-Mesh Data

- **high-level** description
- **minimal**: users can add more
- **human** readable & **machine** actionable

- file format **agnostic, portable**
- **scalable** from desktop to supercomputer
- versioned, **forward-updatable:**

  start “strict“, relax later

h5py reader:
<400 lines
of code

- __init__.py
- field_reader.py
- params_reader.py
- particle_reader.py
- utilities.py

openPMD - A Self-Describing, FAIR Standard

Findable: Standardized metadata to identify the data producer

```
string    /author             attr    = "franz"
string    /software           attr    = "PIConGPU"
string    /softwareVersion    attr    = "0.5.0-dev"
```

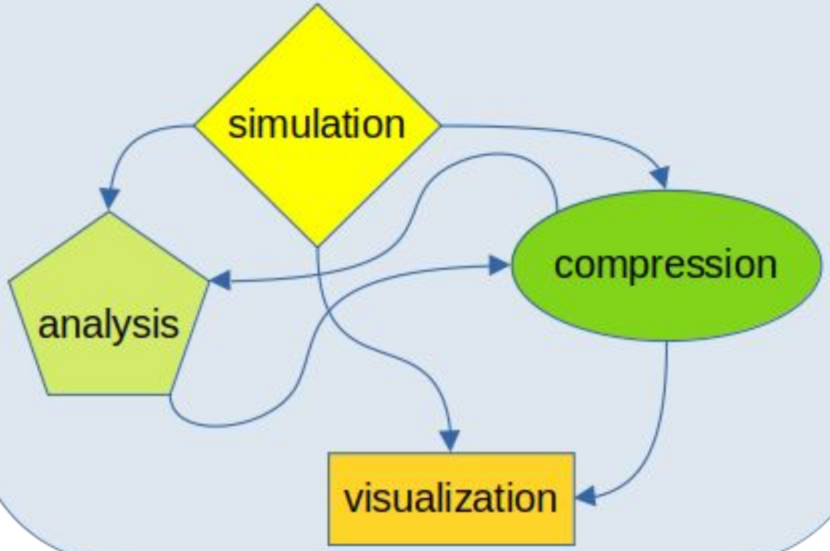
Accessible: Open standard, implementable in various formats

*currently implemented,
but not limited to

"The FAIR Guiding Principles for scientific data management and stewardship" (Mark D. Wilkinson et al.)

openPMD - A Self-Describing, FAIR Standard

Interoperable:
Data exchange spans applications, platforms and teams



Reusable:
Rich and standardized description for physical quantities

Name	Value
axisLabels	[b'z' b'y' b'x']
dataOrder	b'C'
fieldSmoothing	b'none'
geometry	b'cartesian'
gridGlobalOffset	[0. 0. 0.]
gridSpacing	[4.252342 1.0630856 4.252342]
gridUnitSI	4.1671151662e-08
position	[0. 0. 0.]
timeOffset	0.0
unitDimension	[-3. 0. 1. 1. 0. 0. 0.]
unitSI	15399437.98944343

"The FAIR Guiding Principles for scientific data management and stewardship" (Mark D. Wilkinson et al.)

openPMD Components & Modularity

openPMD

- **markup** / schema for arbitrary hierarchical data formats
- truly, scientifically **self-describing**
- basis for **open data workflows**

openPMD standard (1.0.0, 1.0.1, 1.1.0)
the underlying file markup and definition
A Huebl et al., DOI:10.5281/zenodo.33624

base standard
general description

extensions
domain specific

wavefronts, particle species, particle beams,weighted particles, PIC, MD, mesh-refinement, CCD images, ...

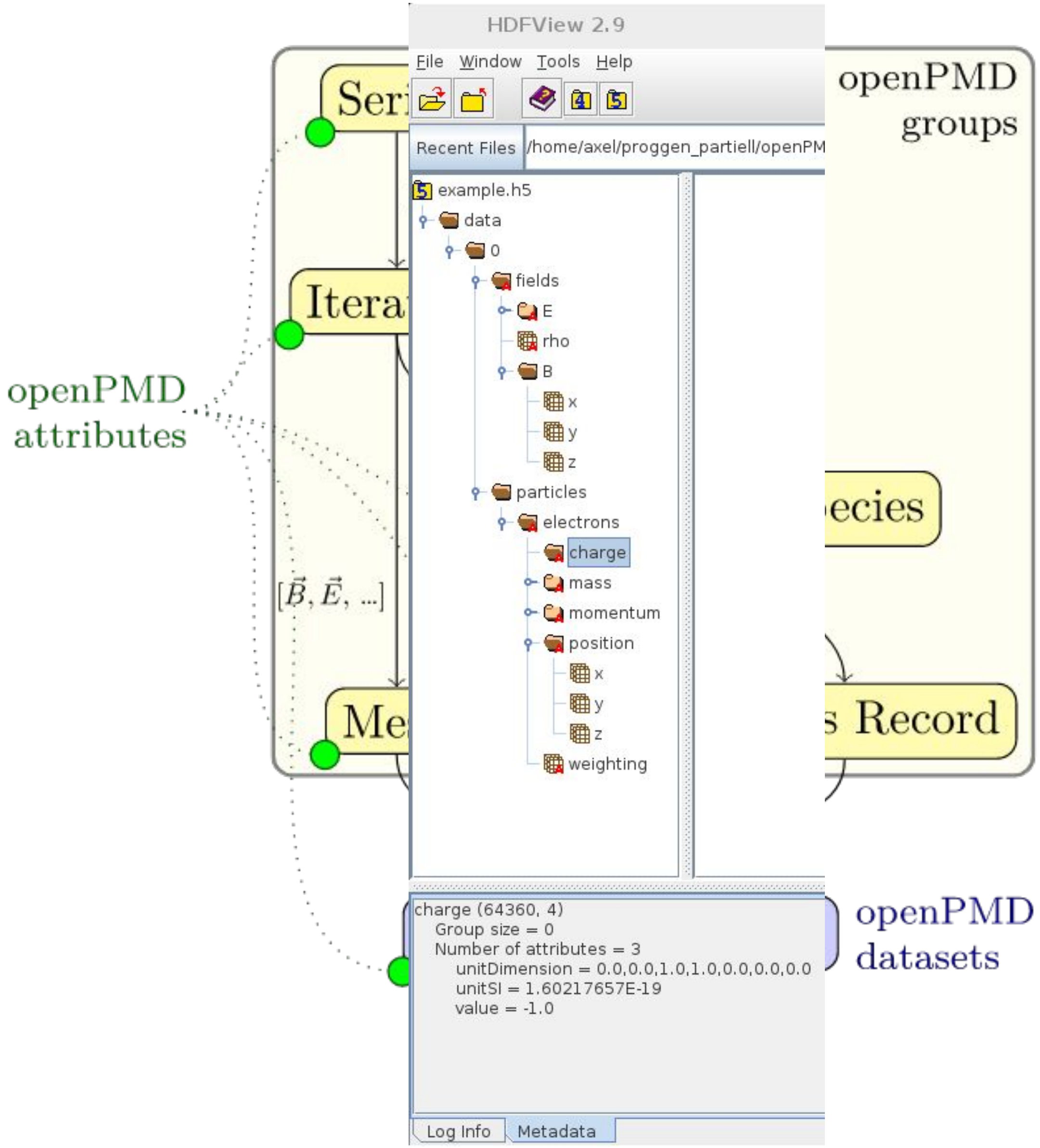
ADIOS HDF {JSON} [T]

openPMD-viewer
quick visualization
explore, e.g., in Jupyter

openPMD-api
reference library
file-format agnostics API

openPMD-updater
auto-update to new standard, verify
openPMD-validator

openPMD - Hierarchy for Data Series

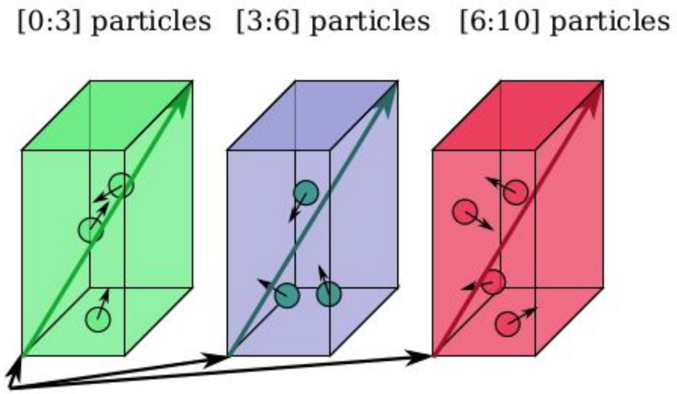


- **Structure** for series & snapshots/iterations encoded in either:
  - ***files*** new files per iteration
  - ***groups*** reuse files
  - ***variables*** reuse files & variables*

  *ADIOS2-only via *steps*

- Records for **physical observables**
  constants, mixed precision, complex numbers
- **Attributes:** unit conversion, description, relations, mesh geometry, authors, env. info, …

- domain-decomposition



openPMD: Block-Structured Mesh-Refinement (WarpX)

BLAST
BEAM PLASMA & ACCELERATOR SIMULATION TOOLKIT

WarpX Particle-in-Cell Code

implemented on AMReX
structured fields + many particles

AMReX

WarpX was first PIC code to win prestigious ACM Gordon Bell award, the “Oscars” of Supercomputing.

open source
ecp-warpx.github.io

>100TB per output
Multi-PB per sim

Multi-Node parallelization

- MPI: 3D domain decomposition
- dynamic load balancing

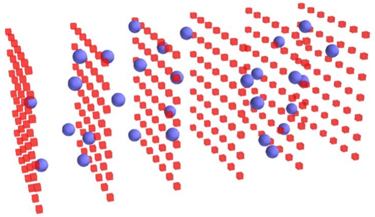
On-Node Parallelization

- GPU: CUDA, HIP and SYCL
- CPU: OpenMP

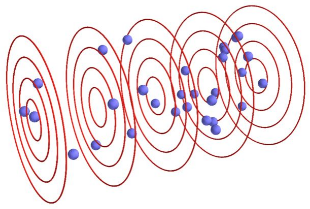
mac OS

Geometries

- 1D3V, 2D3V, 3D3V and RZ (quasi-cylindrical)



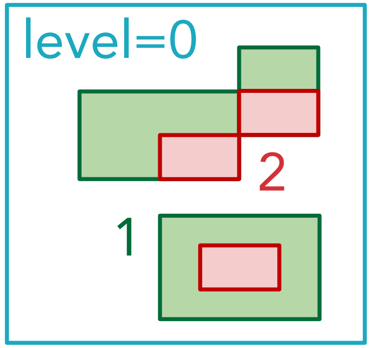
3D Cartesian grid



Cylindrical grid (schematic)

Block-Structured Mesh-Refinement in openPMD

- ADIOS2:
  - variable: <u>1 per level</u>
  - fill with partial blocks
- HDF5:
  - one variable per patch, <u>N per level</u>
  - overhead…? see: Elena's *sparsity* proposal
- alternative: AMReX HDF5
  1 variable per level, but loses HDF5 self-description (concatenated patches)

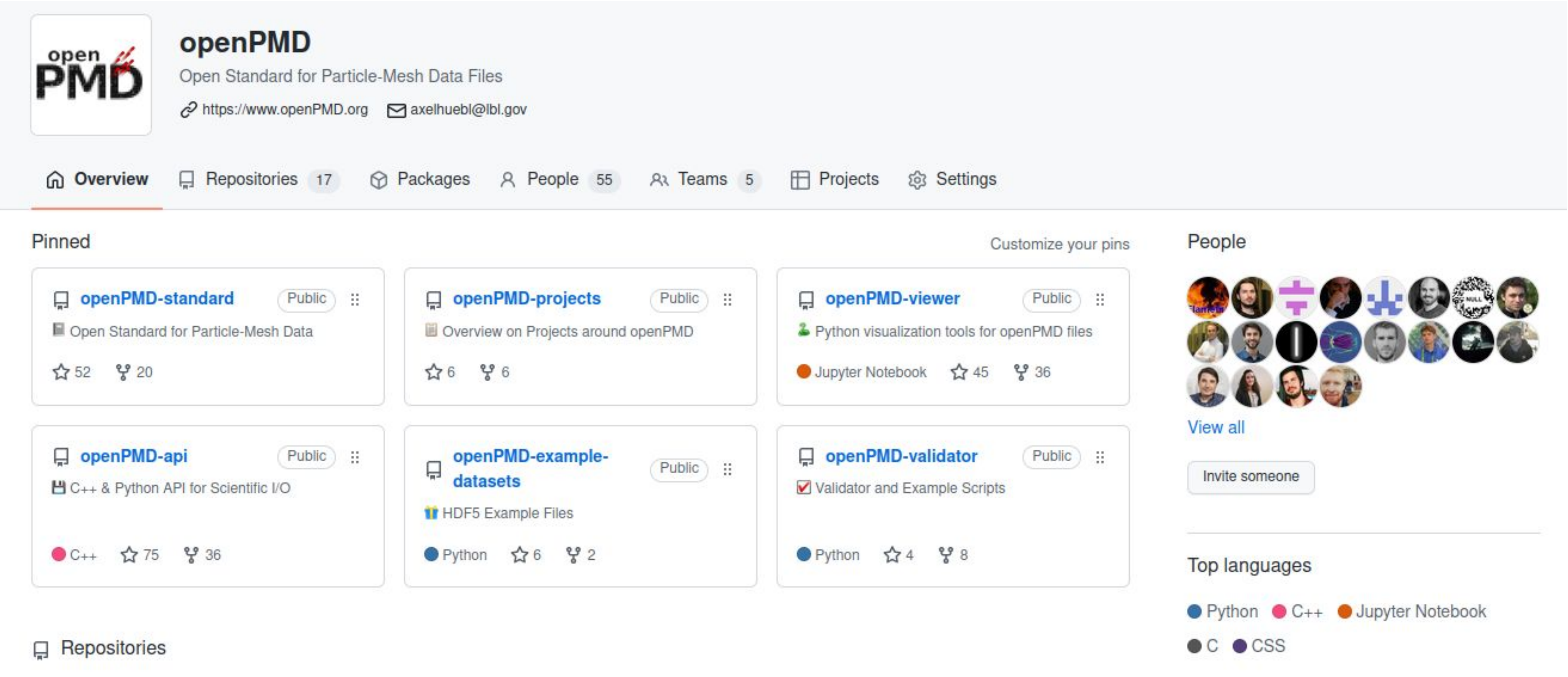


Ref.: github.com/openPMD/openPMD-standard/pull/252

Outline

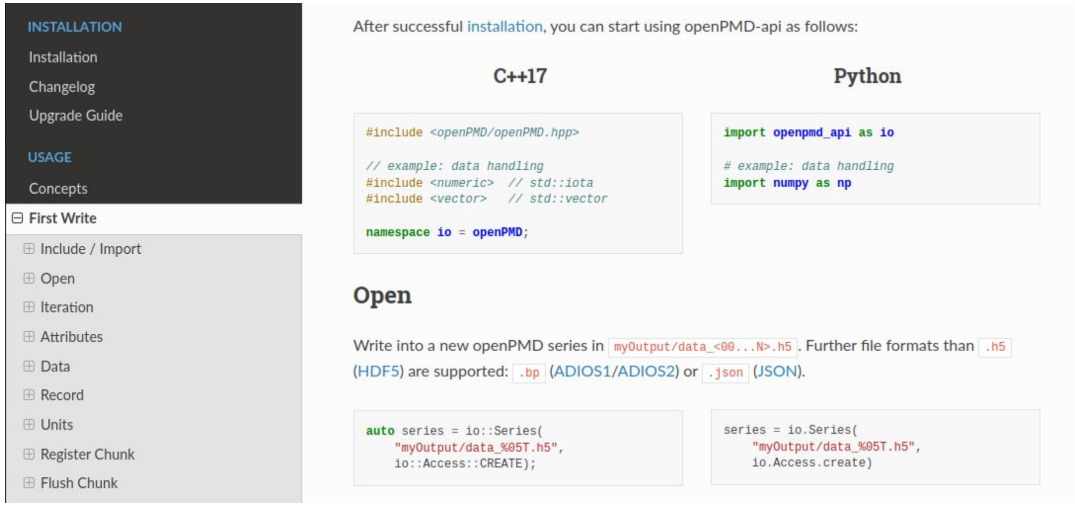
- **openPMD Concepts**
  - design principles of our meta-data standard
  - components & modularity
  - example: sparse data for mesh-refinement
- **Open Ecosystem**
  - projects, libraries, tooling, integrations
  - open community
- **Selected R&D Highlights**
  - fast reads: data layouts & in situ statistics
  - from files to data streams

First Steps *https://github.com/openPMD*

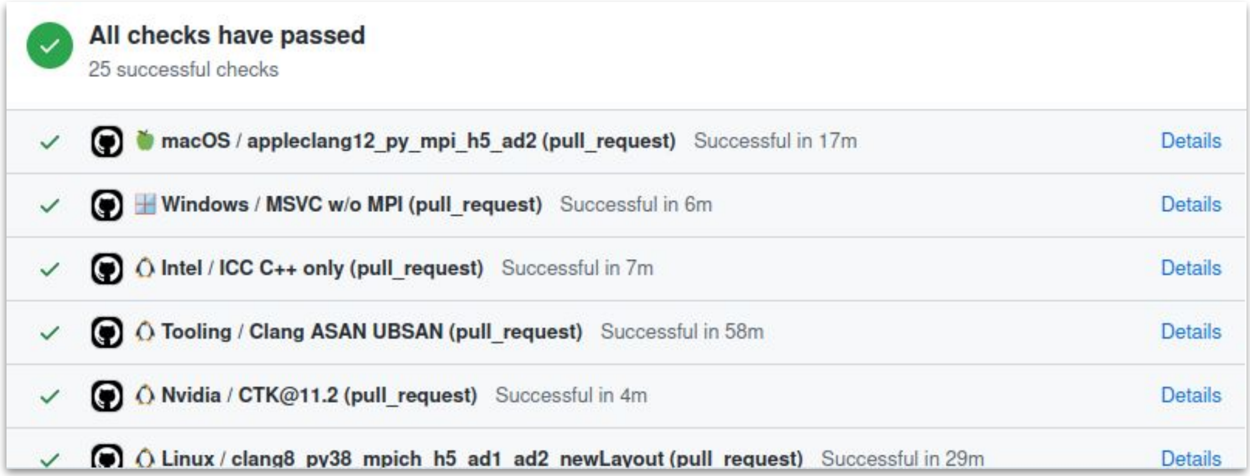


Reference Implementation in C++ & Bindings: Python, Julia

Online Documentation:
openpmd-api.readthedocs.io



Open-Source Development & Tests:
github.com/openPMD/openPMD-api



Rapid and easy installation on any platform:

python3 -m pip install openpmd-api

brew tap openpmd/openpmd
brew install openpmd-api

cmake -S . -B build
cmake --build build --target install

conda install -c conda-forge openpmd-api

spack install openpmd-api

module load openpmd-api

A Huebl, F Poeschel, F Koller, J Gu, et al.
"openPMD-api: C++ & Python API for Scientific I/O with openPMD" (2018) DOI:10.14278/rodare.27

openPMD is used in many Scientific Simulations

Simulation Codes

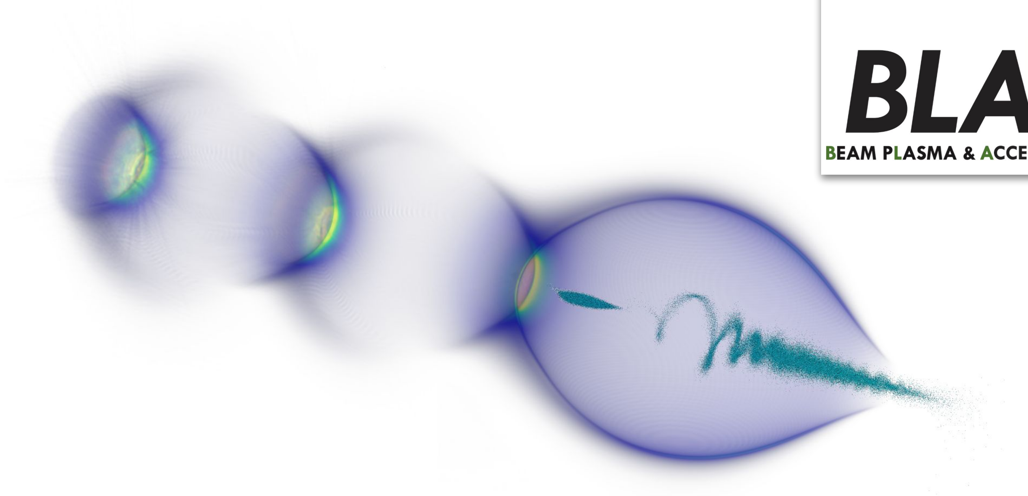
- WarpX
- PIConGPU
- HiPACE++
- ImpactX
- Synergia3
- Wake-T
- Warp
- FBPIC
- SimEx
- LUME
- CarpetX
- Osiris
- VPIC (prototype)
- ACE3P
- ParaTAXIS
- Bmad
- UPIC-Emma
- …

openPMD/openPMD-projects

Many use openPMD via HDF

scientific simulations

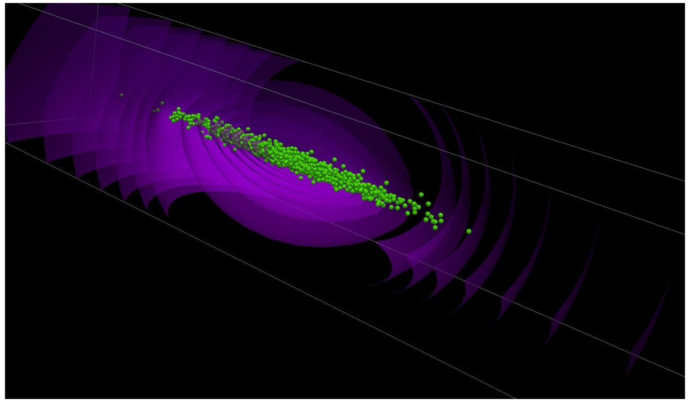
BLAST
BEAM PLASMA & ACCELERATOR SIMULATION TOOLKIT



Credit: M. Thévenet, A. Ferran Pousa

VisualPIC

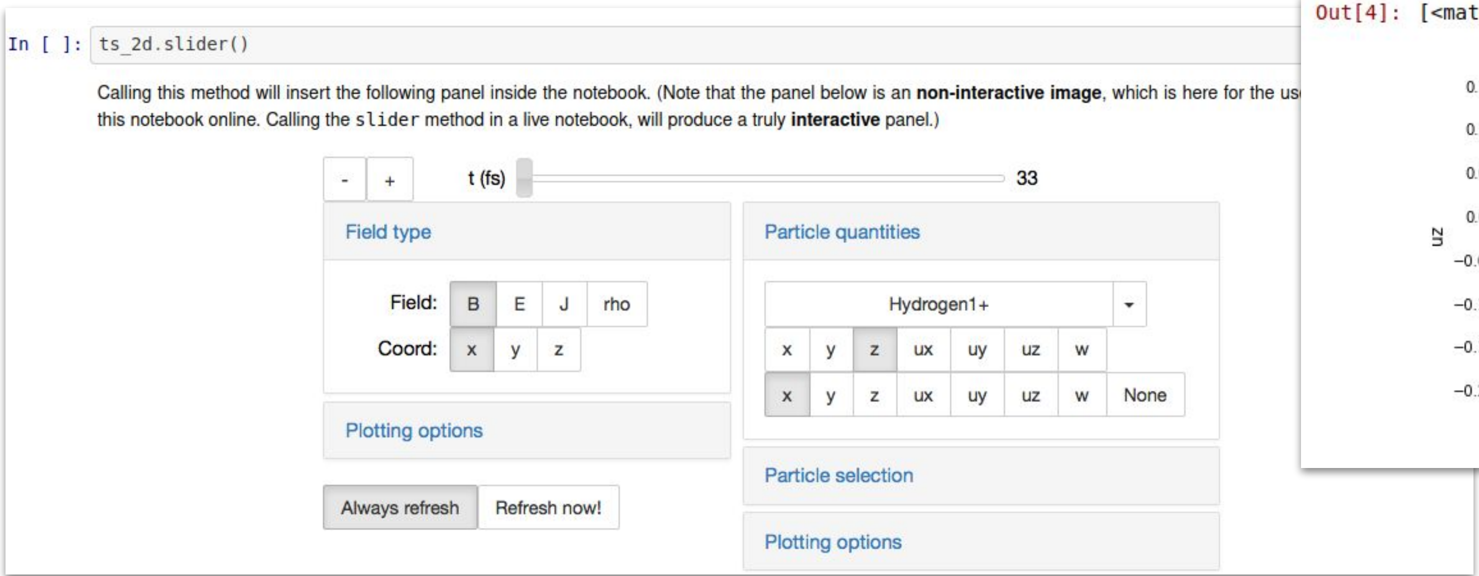
PIConGPU

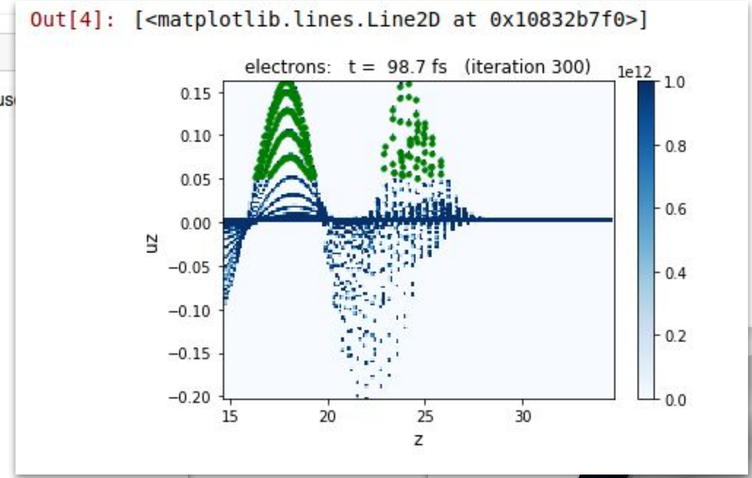


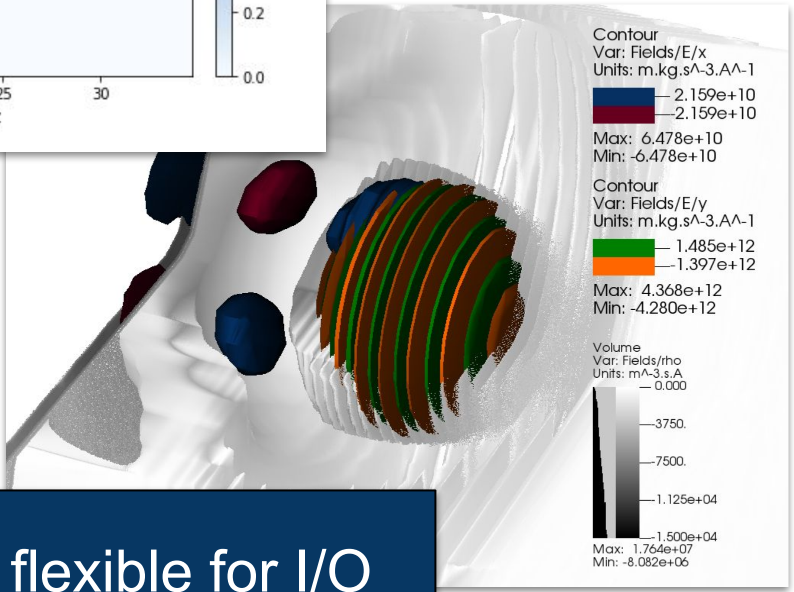
Batteries Included: Bridges into Data Science & Visualization

We integrate for scientific productivity

including data analytics frameworks & graphical user interfaces







Open standardization, i.e. openPMD, makes us flexible for I/O libraries, tooling & domain-science needs.

Community

www.openPMD.org

openPMD

HZDR
HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF

BERKELEY LAB
Lawrence Berkeley National Laboratory

DESY

OAK RIDGE National Laboratory

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

TECHNISCHE UNIVERSITÄT DRESDEN

IOQ JENA
Friedrich-Schiller-Universität

CBG
Max Planck Institute of Molecular Cell Biology and Genetics

UCLA

PERIMETER INSTITUTE

Lawrence Livermore National Laboratory

cea PARIS-SACLAY

SLAC NATIONAL ACCELERATOR LABORATORY

MAISON DE LA SIMULATION

CORNELL UNIVERSITY FOUNDED A.D. 1865

EUCALL

European XFEL

NVIDIA

Fermilab

The **openPMD standard** is co-authored by Axel Huebl, Rémi Lehe, Jean-Luc Vay, David P. Grote, Ivo F. Sbalzarini, Stephan Kuschel, David Sagan, Frédéric Pérez, Fabian Koller, Franz Poeschel, Carsten Fortmann-Grote, Ángel Ferran Pousa, Juncheng E, Maxence Thévenet, and Michael Bussmann.

The authors are thankful for the **community contributions** to libraries, software ecosystem, user support, review and integrations. Particularly, thank you to Yaser Afshar, Lígia Diana Amorim, James Amundson, Weiming An, Igor Andriyash, Ksenia Bastrakova, Jean Luca Bez, Richard Briggs, Heiko Burau, Jong Choi, Ray Donnelly, Dmitry Ganyushin, Marco Garten, Lixin Ge, Berk Geveci, Daniel Grassinger, Alexander Grund, Junmin Gu, Marc W. Guetg, Ulrik Günther, Sören Jalas, Manuel Kirchen, John Kirkham, Scott Klasky, Noah Klemm, Fabian Koller, Mathieu Lobet, Christopher Mayes, Ritiek Malhotra, Paweł Ordyna, Richard Pausch, Norbert Podhorszki, David Pugmire, Felix Schmitt, Erik Schnetter, Dominik Stańczak, Klaus Steiniger, Michael Sippel, Frank Tsung, Lipeng Wan, René Widera, and Erik Zenker!

Outline

- **openPMD Concepts**
  - design principles of our meta-data standard
  - components & modularity
  - example: sparse data for mesh-refinement

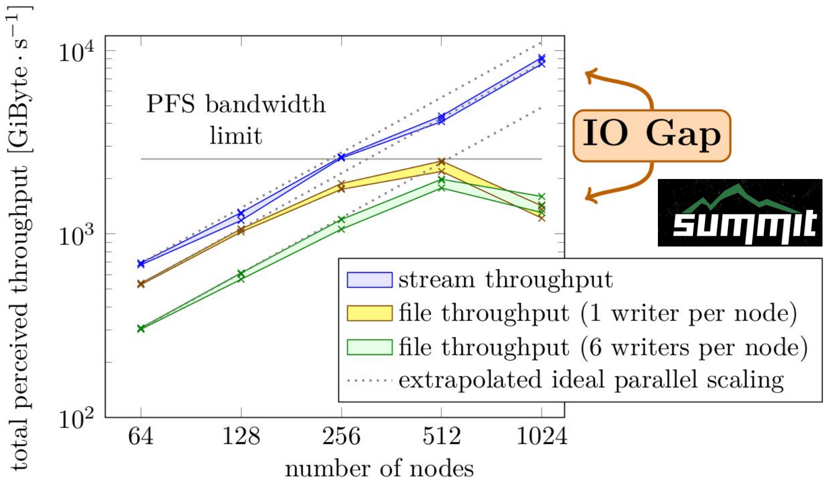
- **Open Ecosystem**
  - projects, libraries, tooling, integrations
  - open community

- **Selected R&D Highlights**
  - fast reads: data layouts & in situ statistics
  - from files to data streams

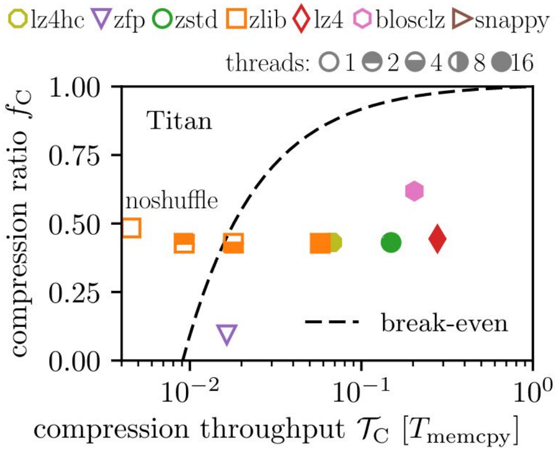
Performance, Data Layouts and Fileless I/O

Application Challenges

- R&D in: scalable techniques, data layouts, libraries
- scientific data analysis & sharing

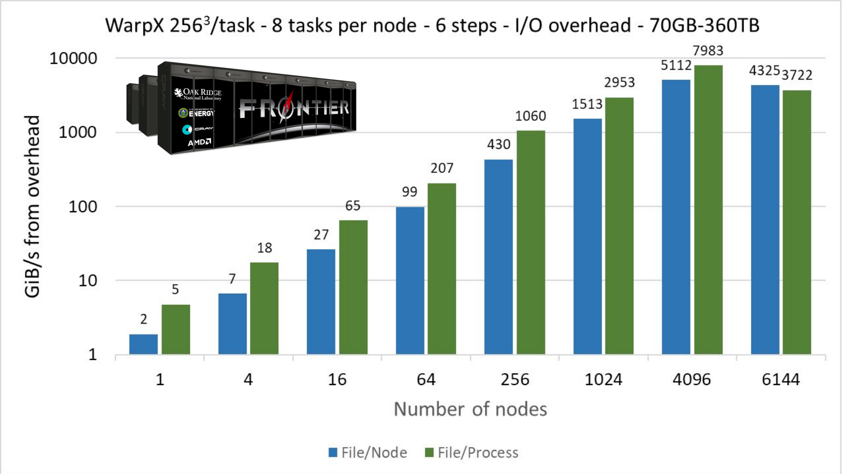


Streaming Data Pipelines:
DOI:10.1007/978-3-030-96498-6_6
by F Poeschel, A Huebl et al., SMC21 (2022)

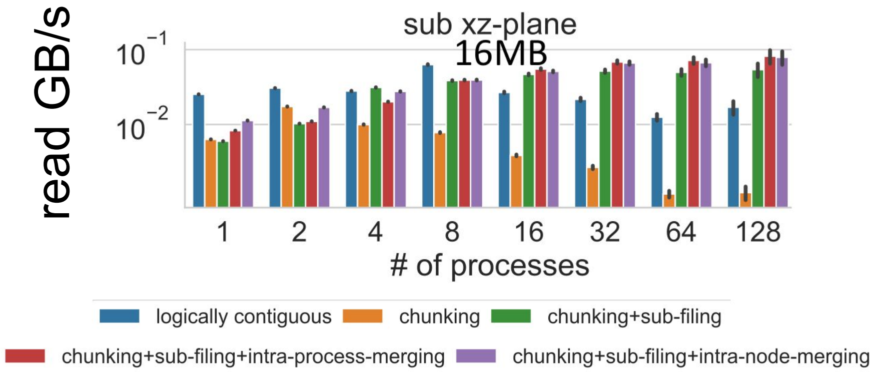


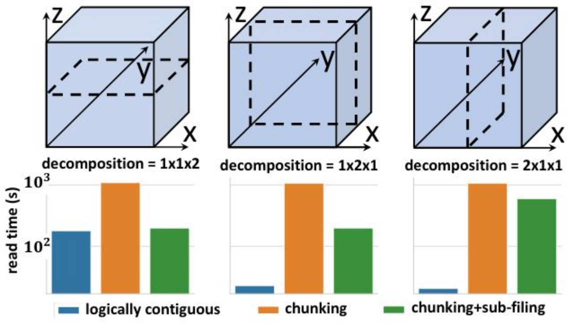
Fast Compressors Needed:
DOI:10.1007/978-3-319-67630-2_2
by A Huebl et al., ISC DRBSD-1 (2017)

ADIOS openPMD-api w/ WarpX



>5.5TB/s FS BW: two-tier lustre w/ high-performance storage & progressive files

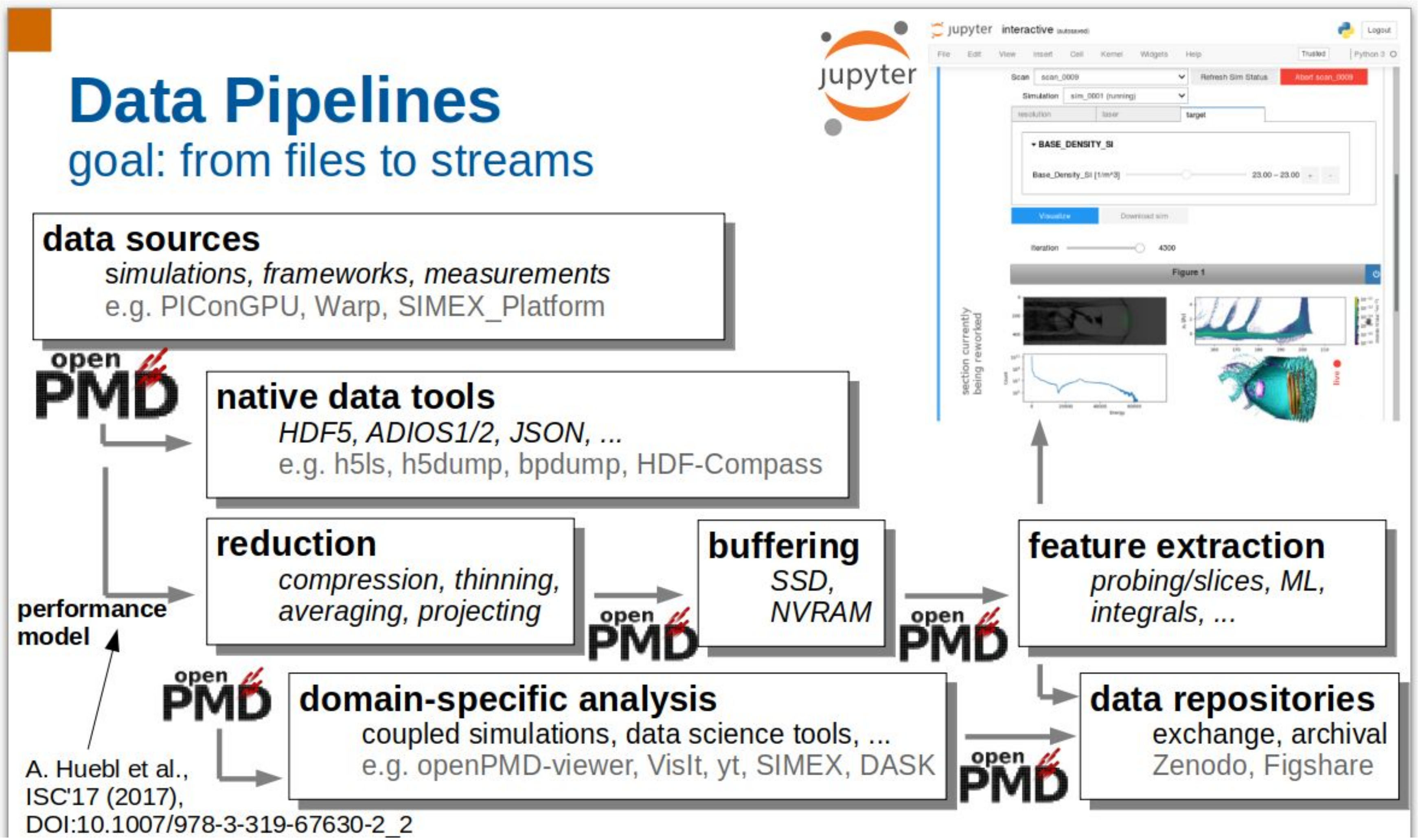


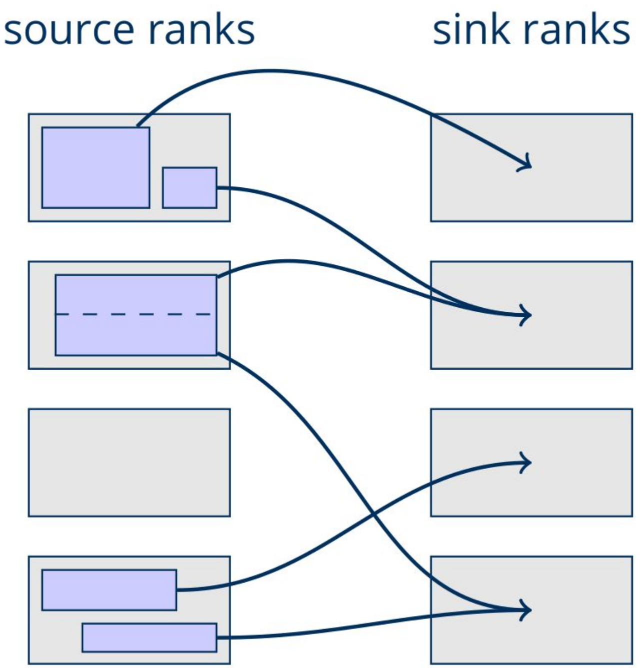


Impact of decomposition schemes when reading

Online Data Layout Reorganization:
DOI:10.1109/TPDS.2021.3100784
by L Wan, A Huebl et al., TPDS (2021)

Streaming: Avoid Files Altogether





Right: For each arrow in the left figure, one could change the mapping between computing nodes (ranks).

F. Poeschel, … A. Huebl. "Transitioning from file-based HPC workflows to streaming data pipelines with openPMD and ADIOS2," SMC21, DOI:10.1007/978-3-030-96498-6_6 (2022)

Summary & Outlook

openPMD is

- a **standardized schema** and
- a large **open-source ecosystem**
  - documentation, example data, validation, scripts, integrations, reference libraries

Our community

- **evolves** and practices **open standardization**
- integrates multiple **state-of-the art** computer science **formats & tools**
  - we are not afraid of PByte-scale workflows
  - from parallel, in-transport data processing to file-less scripting workflows using RMDA

A growing number of international contributors

- try the tools, interact with us on what your needs are, contribute/share data & code

openPMD   openpmd.slack.com   openpmd.org

Future Directions

- Integration with **experimental data acquisition** systems
- **Data Repositories** / Portals
  - AI/ML training, testing, validation, calibration
  - combine data sets
  - preservation, recasting and reinterpretation

presented by: Axel Huebl (LBNL)
📧 **axelhuebl@lbl.gov**

Acknowledgements

This project has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 654220. Supported by the CAMPA collaboration, a project of the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research and Office of High Energy Physics, Scientific Discovery through Advanced Computing (SciDAC) program. Previously supported by the Consortium for Advanced Modeling of Particles Accelerators (CAMPA), funded by the U.S. DOE Office of Science under Contract No. DE-AC02-05CH11231.

This work was partially funded by the Center of Advanced Systems Understanding (CASUS), which is financed by Germany's Federal Ministry of Education and Research (BMBF) and by the Saxon Ministry for Science, Culture and Tourism (SMWK) with tax funds on the basis of the budget approved by the Saxon State Parliament.

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725 and of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under Contract No. DE-AC02-05CH11231.

Backup Slides

Abstract 15'+5'

The Open Standard for Particle-Mesh Data (openPMD) is a metadata standard for tabular (particle/dataframe) and structured mesh data in science and engineering. We show the basic components of openPMD, its extensions to specific domains and applications from laser-plasma physics, particle accelerators, light sources, astrophysics to imaging.

openPMD is implemented on top of portable, hierarchical data formats, especially HDF5 and ADIOS/ADIOS2. An extensive community ecosystem enabled productive workflows for developers and users alike, spanning Exascale simulations, in-transit data processing, post-processing, 3D visualization, GPU-accelerated data analytics and AI/ML. We will present the organization of this community, benefits and experience from supporting multiple data format backends, and future directions.

[1] Axel Huebl, Remi Lehe, Jean-Luc Vay, David P. Grote, Ivo F. Sbalzarini, Stephan Kuschel, David Sagan, Christopher Mayes, Frederic Perez, Fabian Koller, and Michael Bussmann. "openPMD: A meta data standard for particle and mesh based data," DOI:10.5281/zenodo.591699 (2015)
[2] Homepage: https://www.openPMD.org
[3] GitHub Organization: https://github.com/openPMD
[4] Projects using openPMD: https://github.com/openPMD/openPMD-projects
[4] Reference API implementation: Axel Huebl, Franz Poeschel, Fabian Koller, and Junmin Gu. "openPMD-api 0.14.3: C++ & Python API for Scientific I/O with openPMD," DOI:10.14278/rodare.1234 (2021) https://openpmd-api.readthedocs.io
[5] Selected earlier presentations on openPMD: https://zenodo.org/search?page=1&size=20&q=openPMD&type=presentation
[6] Axel Huebl, Rene Widera, Felix Schmitt, Alexander Matthes, Norbert Podhorszki, Jong Youl Choi, Scott Klasky, and Michael Bussmann. "On the Scalability of Data Reduction Techniques in Current and Upcoming HPC Systems from an Application Perspective," ISC High Performance 2017: High Performance Computing, pp. 15-29, 2017. arXiv:1706.00522, DOI:10.1007/978-3-319-67630-2_2
[7] Franz Poeschel, Juncheng E, William F. Godoy, Norbert Podhorszki, Scott Klasky, Greg Eisenhauer, Philip E. Davis, Lipeng Wan, Ana Gainaru, Junmin Gu, Fabian Koller, Rene Widera, Michael Bussmann, and Axel Huebl. Transitioning from file-based HPC workflows to streaming data pipelines with openPMD and ADIOS2, Part of Driving Scientific and Engineering Discoveries Through the Integration of Experiment, Big Data, and Modeling and Simulation, SMC 2021, Communications in Computer and Information Science (CCIS), vol 1512, 2022. arXiv:2107.06108, DOI:10.1007/978-3-030-96498-6_6

Organizing Scientific Records

- **electric field** $\vec{E}(\vec{r})$

  / ... / **meshes** / **E** /

  x **y** **z**

- **temperature** $T(\vec{r})$

  / ... / **meshes** /

  T

- **electron position** $\vec{r}$

  / ... / **particles** / **electrons** / **position** /

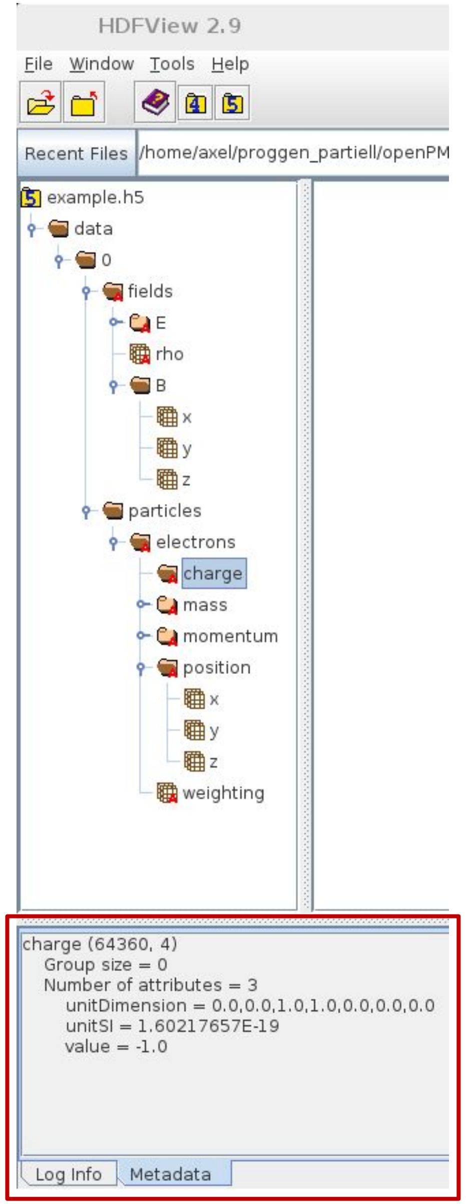
  x **y** **z**

group

data

attributes

extensions to the **base standard** define additional, compatible keywords



Unit System

- **unitDimension**

  automated description of physical dimension

  only powers of base dimensions

  length **L**, mass **M**, time **T**, electric current **I**, thermodynamic temperature **theta**, amount of substance **N**, luminous intensity **J**

  magnetic field: [B] = M / (I * T^2)
  → **(0., 1., -2., -1., 0., 0., 0.)**

- **unitSI** (recommended)

  relation to an *absolute* unit system

attributes

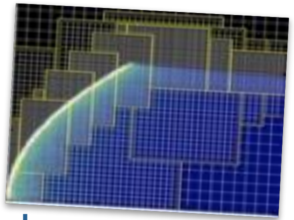
WarpX is a GPU-Accelerated PIC Code for Exascale

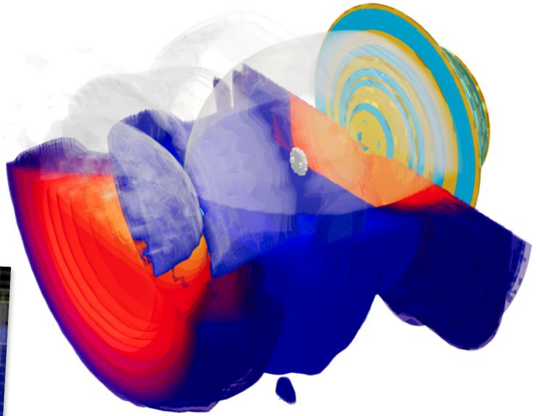
BLAST
BEAM PLASMA & ACCELERATOR SIMULATION TOOLKIT

Particle-in-Cell Loops

electrostatic & electromagnetic (fully kinetic)

WarpX was first PIC code to win prestigious ACM Gordon Bell award, the “Oscars” of Supercomputing.





Advanced algorithms

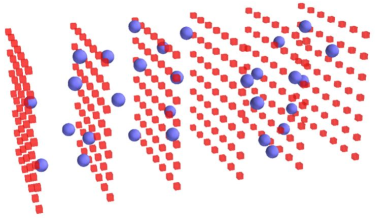
boosted frame, spectral solvers, Galilean frame, embedded boundaries + CAD, MR, ...

Multi-Physics Modules

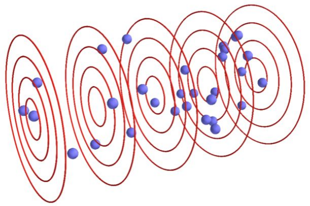
field ionization of atomic levels, Coulomb collisions, QED processes (e.g. pair creation), macroscopic materials

Geometries

- 1D3V, 2D3V, 3D3V and RZ (quasi-cylindrical)



3D Cartesian grid



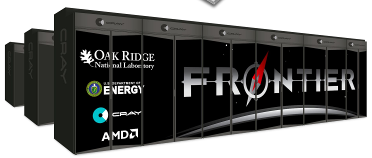
Cylindrical grid (schematic)

Multi-Node parallelization

- MPI: 3D domain decomposition
- dynamic load balancing

On-Node Parallelization

- GPU: CUDA, HIP and SYCL
- CPU: OpenMP



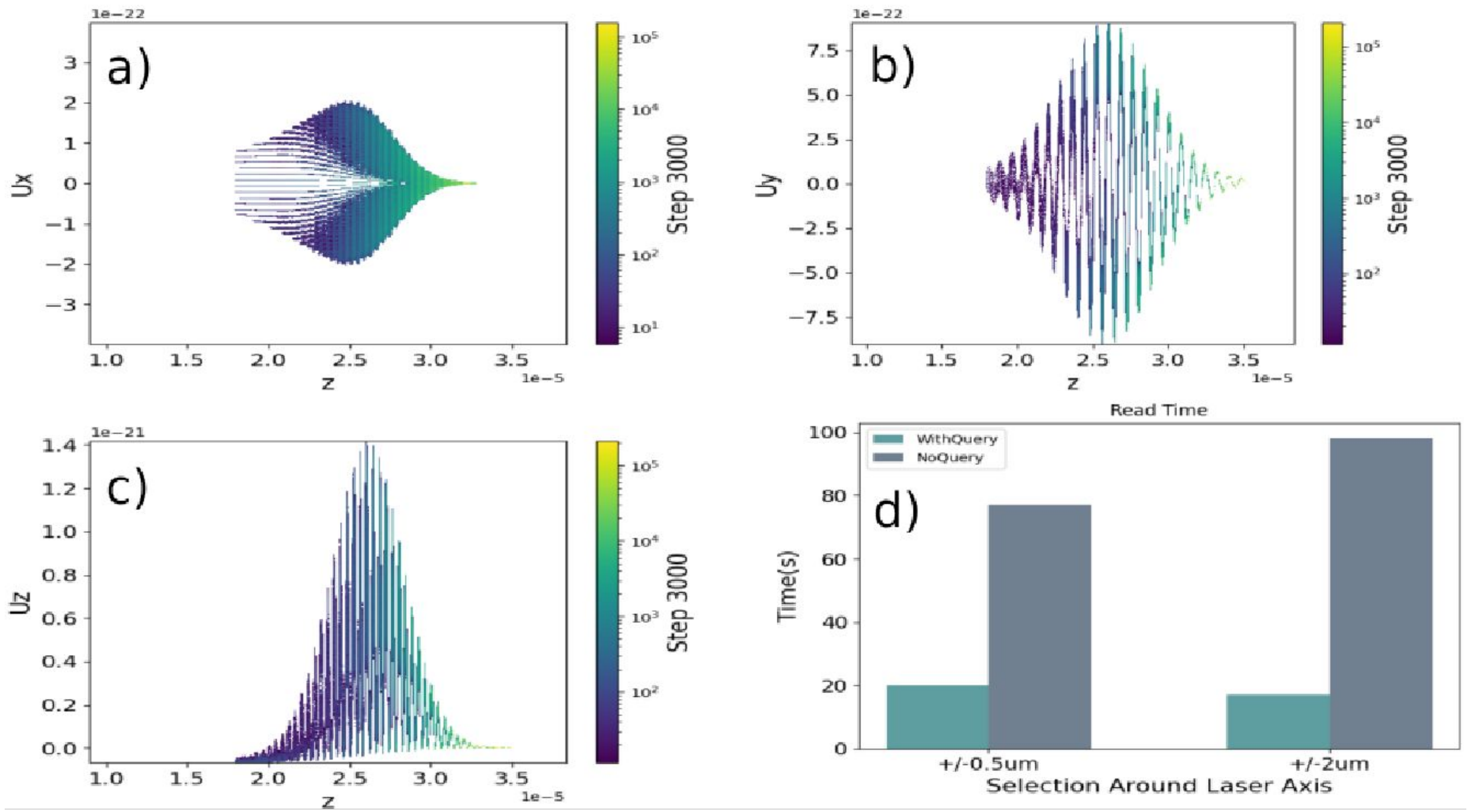
Scalable, Standardized I/O

- PICMI Python interface
- openPMD (HDF5 or ADIOS)
- in situ diagnostics

Open Source:
ecp-warpx.github.io

ECP EXASCALE COMPUTING PROJECT

In-Transport Data Processing: Data Statistics



Data analysis using region of interest filtering with ADIOS queries. a)-c) Phase space projections of plasma particles oscillating in a laser pulse, filtered close to the laser axis. d) Read time comparison between conventional reads and pre-filtered reads with queries.

J Gu, P Davis, G Eisenhauer, W Godoy, A Huebl, S Klasky, M Parashar, N Podhorszki, F Poeschel, J-L Vay, L Wan, R Wang, and K Wu, “Organizing Large Data Sets for Efficient Analyses on HPC Systems,” submitted (2021)

Integration into the BELLA Control System

BELLA GEECS

- **G**eneralized **E**quipment and **E**xperiment **C**ontrol **S**ystem
- control and data acquisition system

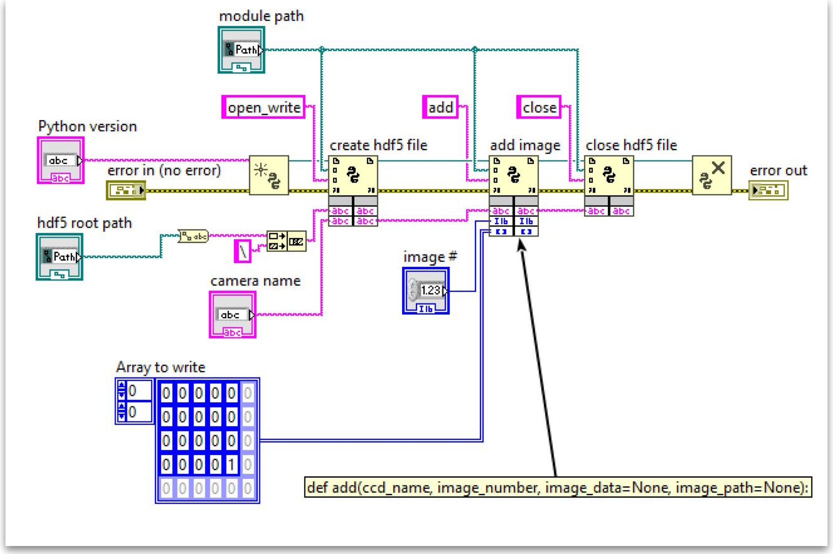
openPMD/openPMD-CCD

GEECS-BELLA

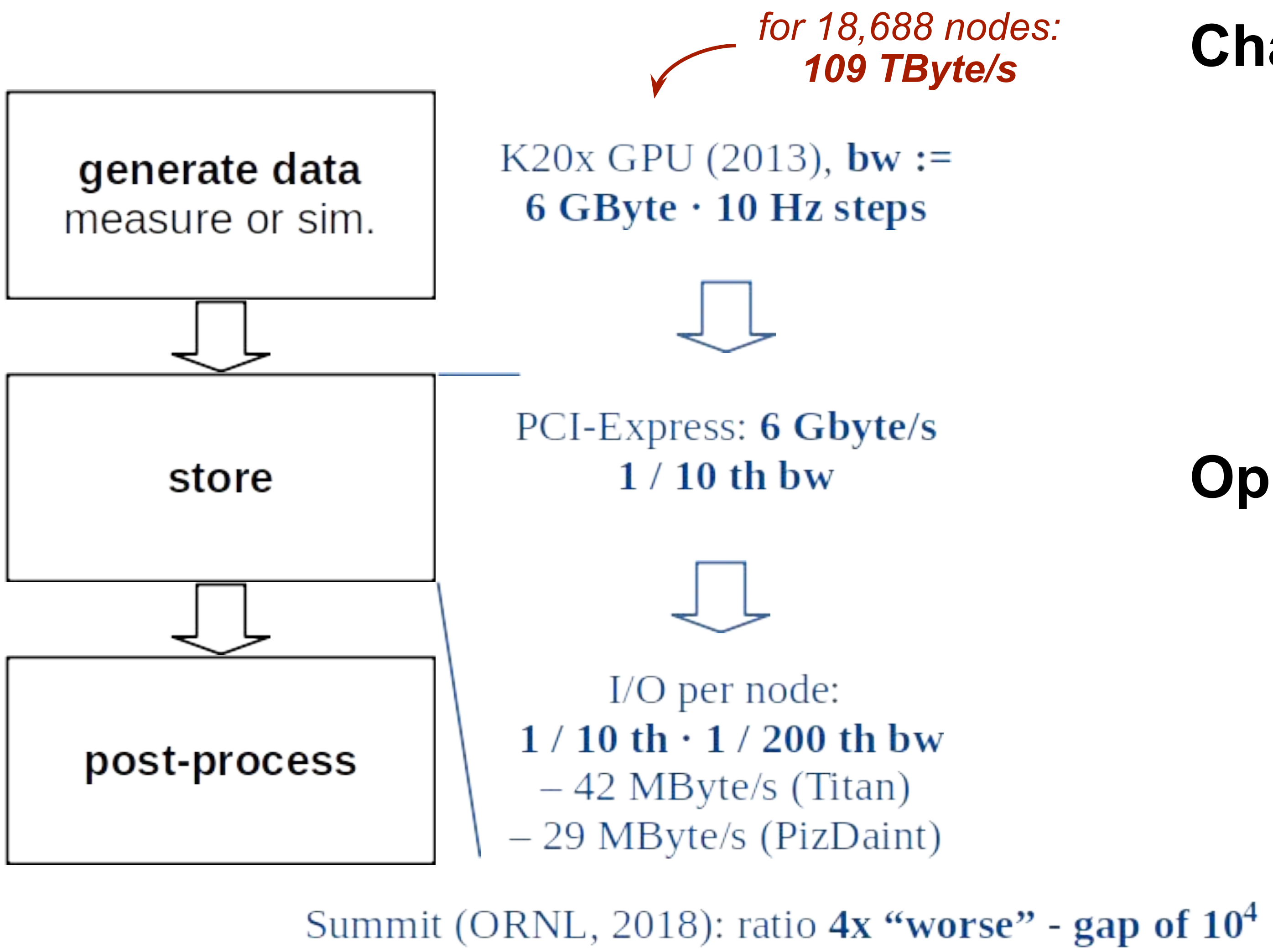
openPMD-CCD

- **Python** module for organizing CCD images with openPMD
- Optional integration with **LabView 2020+**

```python
from openpmd_ccd import CCD

scan = CCD("defaultCam_scan001.h5", overwrite=True,
           name="Go Pro", model="HERO8 Black", serial="12345678",
           operator="Axel Huebl <axelhuebl@lbl.gov>",
           # resolution=None, roi=None, exposure_time=None
)

scan.add(0, "tests/data/Scan005_SimCam_001.png")
scan.add(1, np.array([[1., 2.], [3., 4.]]))

# scan.recalibrate(...)

scan.close()
```



HPC Background: Our Data Processing Bottlenecks Look Alike



A Huebl et al., ISC 2017, DOI:10.1007/978-3-319-67630-2_2 (2017)

Challenges

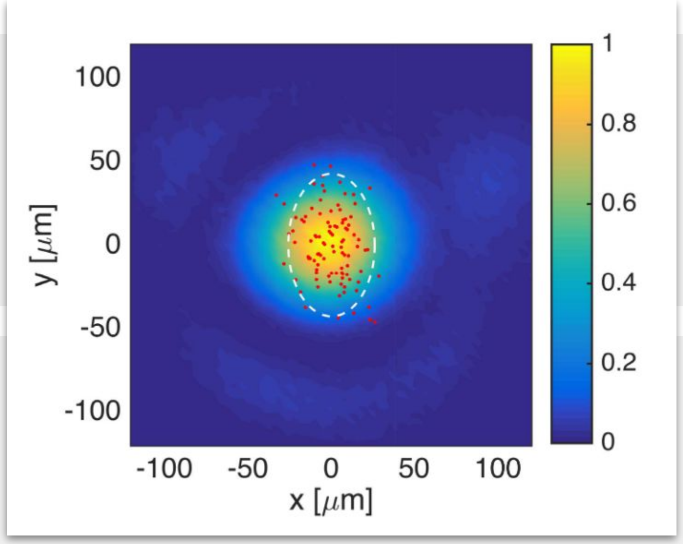
- **3 orders of magnitude gap** between producing devices and storage
- “store & analyze everything” is *unaffordable*

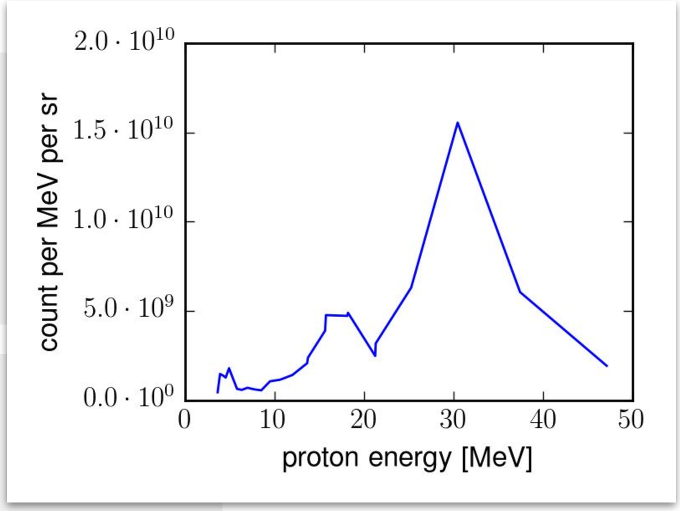
Opportunities

- analysis tasks have varying **fidelity** needs
- many common tasks can be done **in situ**
- manual steps: limit the sampling of raw data to **setup phase**

Data Processing & Reduction Examples

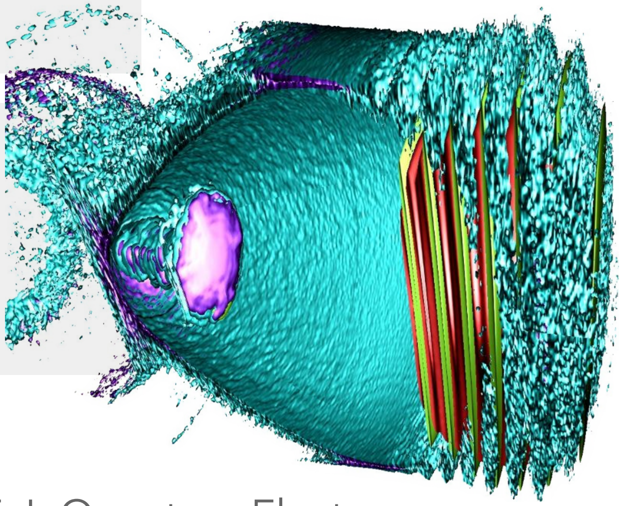
Binning of a **spectrogram**
Fitting of an **ellipsoid**





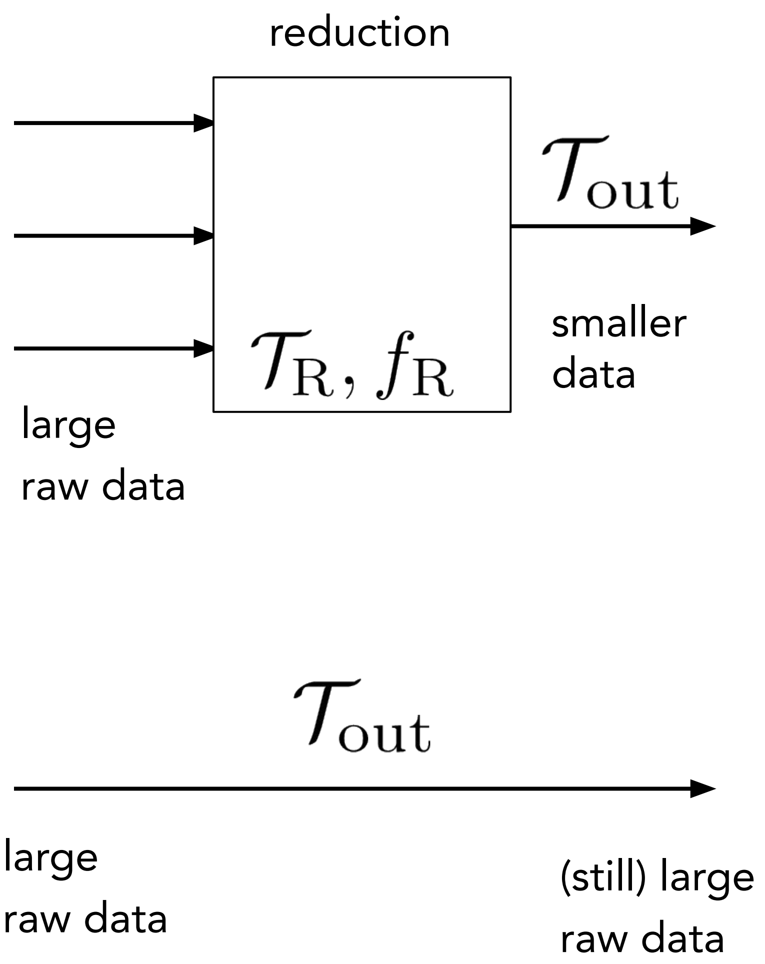
Compression (lossless/lossy)

Ray-casting 3D data,
training a neural network, etc.

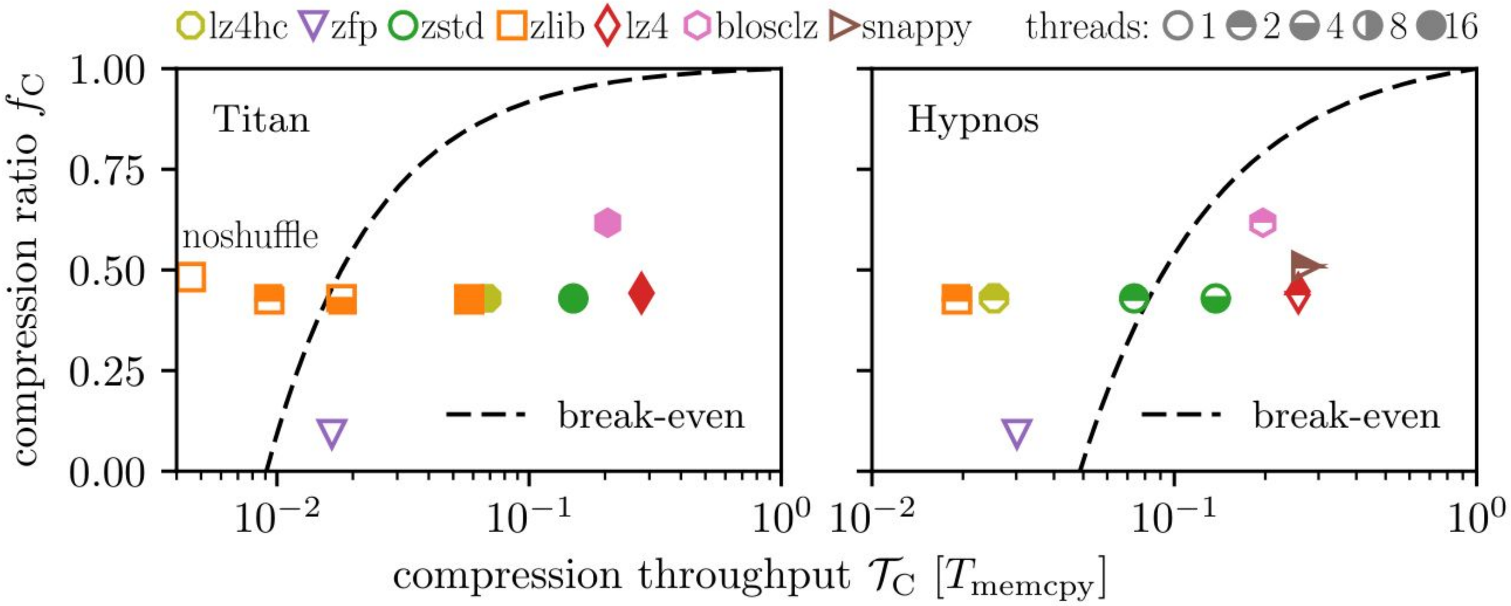


A Matthes, A Huebl et al., ISC 2017, DOI:10.14529/jsfi160403 (2017); A Huebl et al., ISC 2017, DOI:10.1007/978-3-319-67630-2_2 (2017); K Nakamura et al., IEEE J. Quantum Electron, DOI:10.1109/JQE.2017.2708601 (2019)

Avoid Backlog: Design Criteria for Data Reduction Pipelines



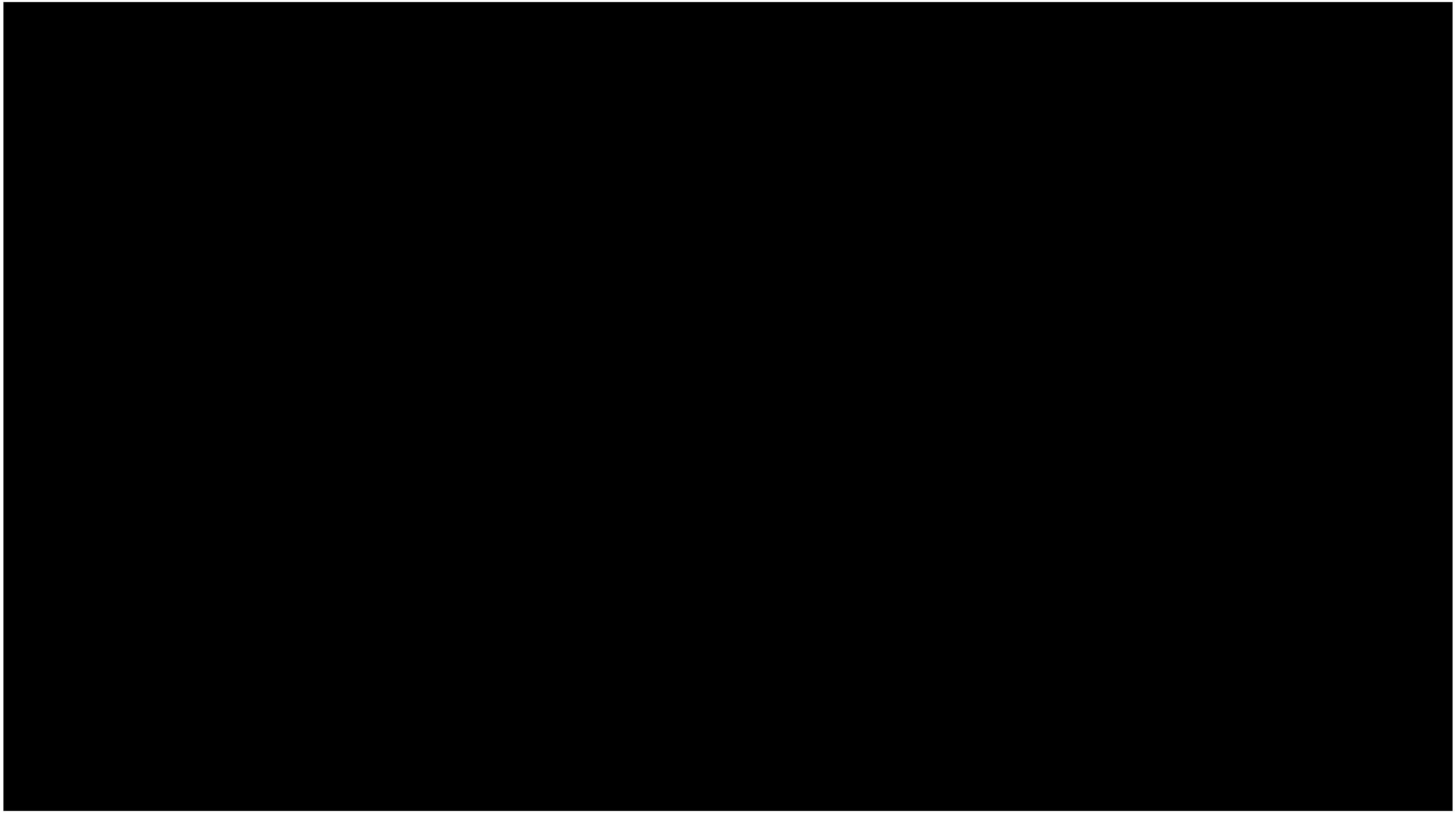
$$\frac{\mathcal{T}_{\mathrm{R}} \times (1 - f_{\mathrm{R}})}{1 - \mathcal{T}_{\mathrm{R}}} > \mathcal{T}_{\mathrm{out}}$$



Result: **trade compute for throughput**, >100 GByte perceived application throughput and **280 GByte/s** peak parallel filesystem throughput

A Huebl et al., "On the Scalability of Data Reduction Techniques in Current and Upcoming HPC Systems from an Application Perspective," ISC High Performance Workshops, DOI:10.1007/978-3-319-67630-2_2 (2017)

DASK Example: Modern Access to Parallel Processing



openPMD - Extension Example

Wavefronts, Particle Species, Particle Beams, Weighted Particles, PIC, MD, Mesh-Refinement, CCD images, ...

Convention for Specifying Particle Species

openPMD extension name: `SpeciesType`

Introduction

This convention is for standardizing the names of particle species, e.g. in particle physics.

Additional Record Attribute

The following additional attribute for openPMD `mesh records` and `particle groups` is defined in this extension:

- `speciesType`
  - type: *(string)*
  - scope: *optional*
  - description: particle species in this record. If there are multiple species to be specified, they can be specified using a semicolon separated list.
  - allowed values:
    - *see the lists below* and additionally
    - `other` if none of the ones below applies, user are free to append a free text after a colon, e.g. `other:neutralino` or `other:cherry`
  - examples:
    - `electron` (e.g. on an electron `particle record` or an electron density `mesh record`)
    - `electron;proton;#12C` (e.g. on a `mesh record` for a plasma's local charge density)
    - `other:apple;other:orange` (for a `record` mixing apples & oranges)

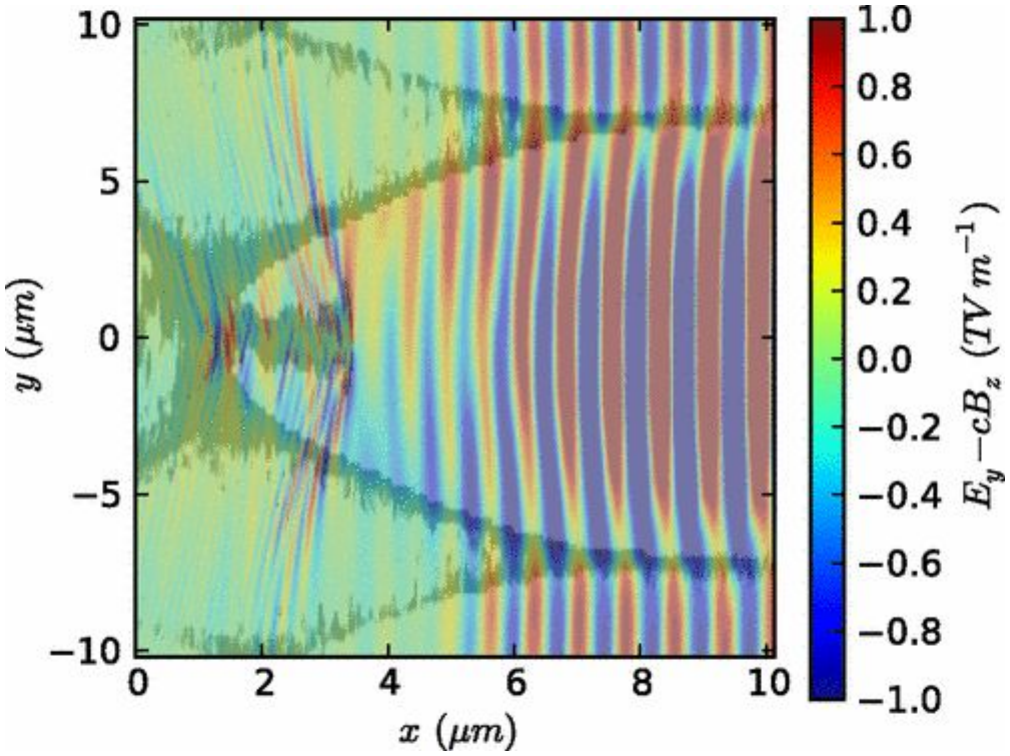
This attribute can be used with any `record` (including `mesh records`).

Elementary Particles

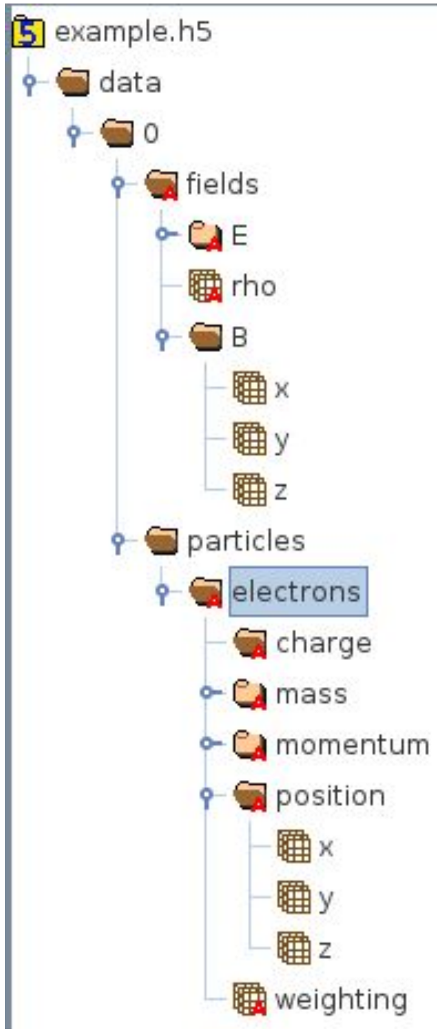
Namings for fundamental fermions and their anti-matter particles.

Quarks:

- `up` , `anti-up`



openPMD-api: Reference Implementation



Flagship implementation: **openPMD-api**

- **API** in C++ and Python (upcoming: Julia)
- Flexibly store to / read from interchangeable **backends**:
  - ADIOS1/2
  - HDF5
  - JSON (serial only)
- Applies **best practices** & **performance** tuning

openPMD/openPMD-api

Example users:

- Simulations: WarpX, PIConGPU, Wake-T, GPos
- Frameworks: SimEx, LUME, OASYS1-PaNOSC
- Visualization & analysis: ParaView, DASK, VisualPIC, APTools

A Huebl, F Poeschel, F Koller, J Gu and contributors (2018).
openPMD-api: C++ & Python API for Scientific I/O with openPMD. DOI:10.14278/rodare.27

Ecosystem

openPMD/openPMD-projects

openPMD standard (1.0.0, 1.0.1, 1.1.0)
the underlying file markup and definition
A Huebl et al., DOI:10.5281/zenodo.33624

base standard
general description

extensions
domain specific
wavefronts, particle species, particle beams,weighted particles, PIC, MD, mesh-refinement, CCD images, ...

native data tools
HDF5, ADIOS1/2, NetCDF, ...
e.g. h5ls, h5repack, h5dump, bpdump

writers & converters
simulations, frameworks, measurements
e.g. PIConGPU, Warp, SIMEX_Platform

HDF Compass
HDF5 & ADIOS file explorer
open and explore file trees

readers
coupled simulations, post-processing frameworks, ...
e.g. SIMEX_Platform, VisIt, yt-project, openPMD-viewer

openPMD-updater
update to new standard
edit in- or new file

openPMD-api
I/O library abstraction
file format agnostic

data repositories
exchange and long-time archival
e.g. Zenodo, RODARE (HZDR)

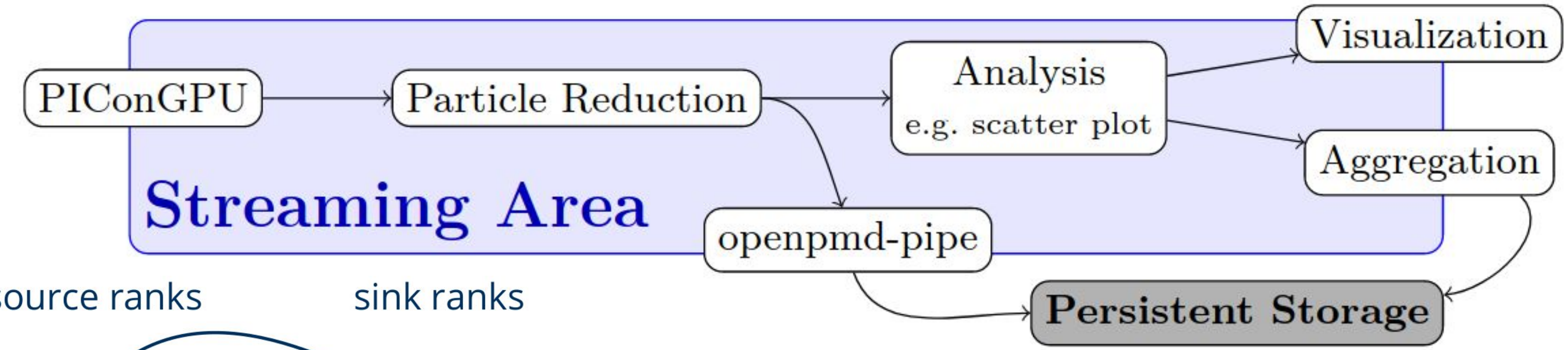
A Rising Data Creation vs. Storage Gap

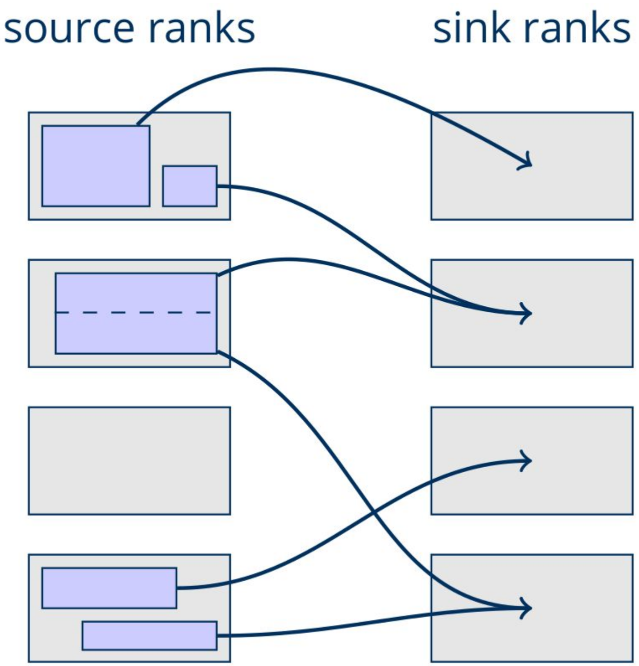
system	compute performance $[\mathrm{PFlop} \cdot \mathrm{s}^{-1}]$	parallel FS bandwidth $[\mathrm{TiByte} \cdot \mathrm{s}^{-1}]$	FS capacity [PiByte]	example storage requirements [PiByte]
Titan	27	1	27	5.3
Summit	200	2.5	250	21.1
Frontier	> 1500	5 - 10	500 - 1000	80 - 100

Table 1: System performance: OLCF Titan to Frontier. The last column shows the storage size needed by a full-scale simulation that dumps all GPU memory in the system 50 times.

F. Poeschel, … A. Huebl. “Transitioning from file-based HPC workflows to streaming data pipelines with openPMD and ADIOS2,” accepted in SMC21, https://arxiv.org/abs/2107.06108 (2021)

Streaming: Avoid Files Altogether





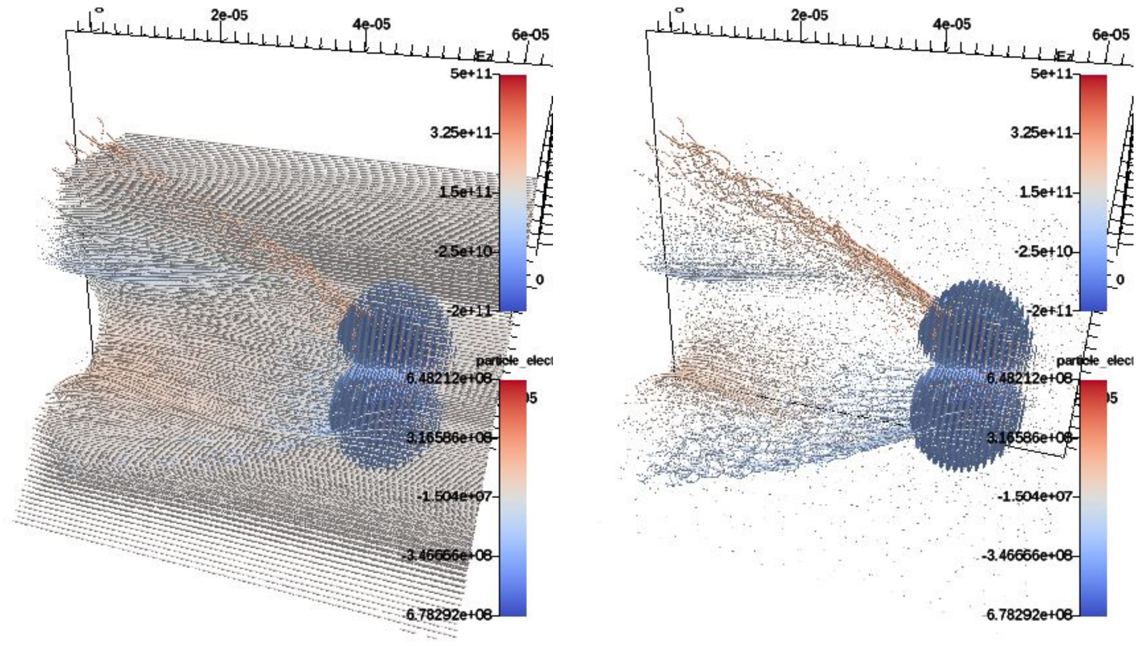
Left: For each arrow in the above figure, one could furthermore change the mapping between computing nodes (ranks).

F. Poeschel, … A. Huebl. "Transitioning from file-based HPC workflows to streaming data pipelines with openPMD and ADIOS2," accepted in SMC21, https://arxiv.org/abs/2107.06108 (2021)

Novel Visualization Techniques

Particle *Adaptive Sampling*

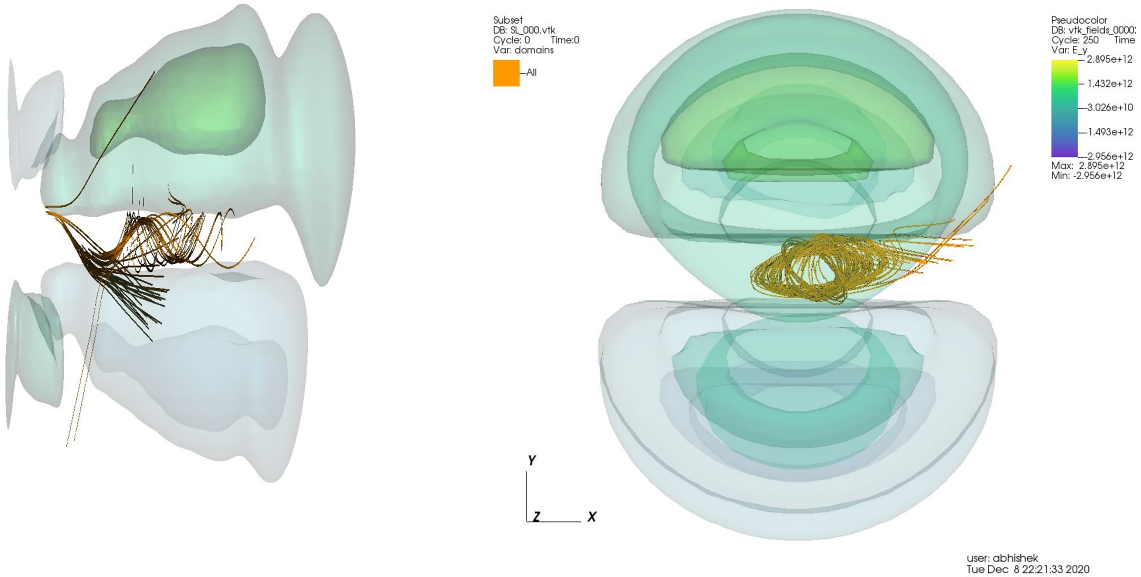
- **emphasis** on "uncommon" properties
- inverse sampling to incidence of a property



A. Biswas et al., "In Situ Data-Driven Adaptive Sampling for Large-scale Simulation Data Summarization," ISAV18 @SC18 (2018)

Physics-Informed Flow Tracelines

- traditional flow vis. depends only on *local field values*
- plasma particles:
  - **inert**: track *relativistic momentum* on a traceline
  - **Lorentz-Force**: 6 fields (electromag.), leap-frog
- chance to **significantly reduce particle I/O** in real-life workflows through savings on **temporal fidelity**

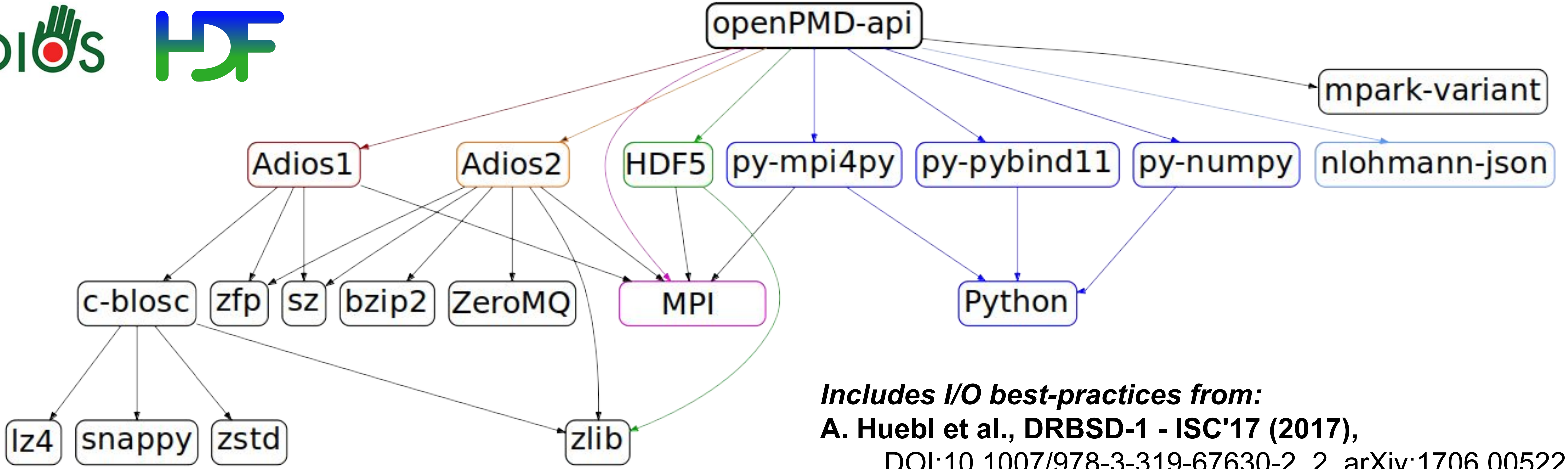


ALPINE: Ascent Biswas, Larsen, Lo

VTK-m Yenpure, Childs, Pugmire

openPMD-api: I/O Middleware Library

ADIOS HDF



Includes I/O best-practices from:

A. Huebl et al., DRBSD-1 - ISC'17 (2017),
DOI:10.1007/978-3-319-67630-2_2, arXiv:1706.00522

F. Poeschel, A. Huebl et al., SMC21 (2021)
arXiv:2107.06108

Available via:

Spack

module load

CONDA-FORGE

python Package Index

Homebrew

CMake **Superbuild**

Data Visualization & Analysis: Parallel, User-Facing Tooling

Continuously Improving User-Facing Data Workflows

- **VisIt** integration: mainlining of an existing openPMD plugin (HDF5)
  - including quasi-cylindrical geometry (azimuthal decomposition)

Lobet, Huebl (LBNL), Pugmire (ORNL)

- **ParaView 5.9+** integration for openPMD (all formats, e.g. ADIOS2)
  - productivity: parallel plugin is fully written in Python (<500 LOC) ParaView, openPMD-api, ADIOS2/HDF5 are all “lifting” in C/C++
  - next: azimuthal geometry, MR

Huebl (LBNL), Geveci (Kitware)

- **DASK**/RAPIDS
  - basis to parallelize Python analysis pipelines at NERSC & OLCF
  - potential candidate for further research with data staging (ADIOS)

Huebl (LBNL), Ganyushin (ORNL), Kirkham (Nvidia)

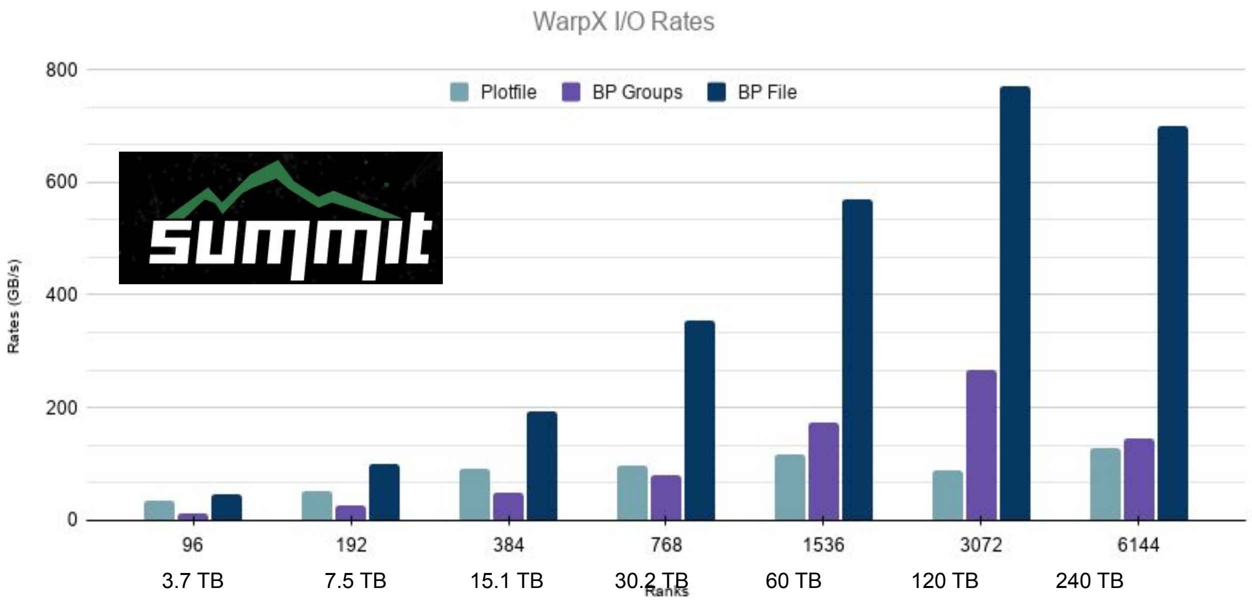
Open standardization, i.e. openPMD, makes us flexible for I/O libraries, tooling & domain-science needs.

Parallel I/O: integration of ADIOS2 through openPMD-api

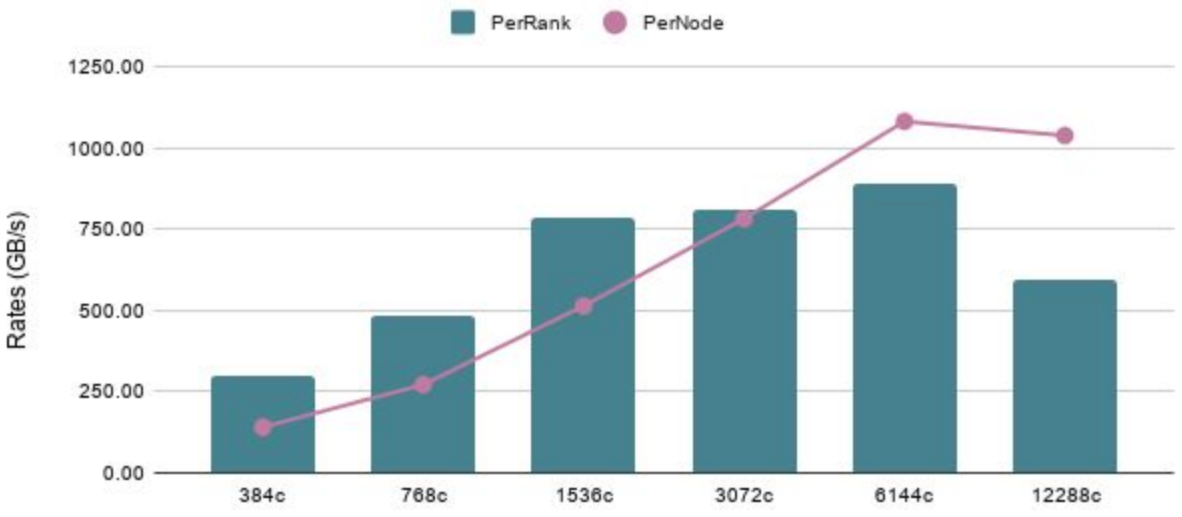
Reducing I/O Risks in AMR Particle-in-Cell

baseline: subfiling & chunking for fast writes

- aggregation of blocks, appending of steps
- representation of **sparsity in structured MR** meshes in **ADIOS blocks**



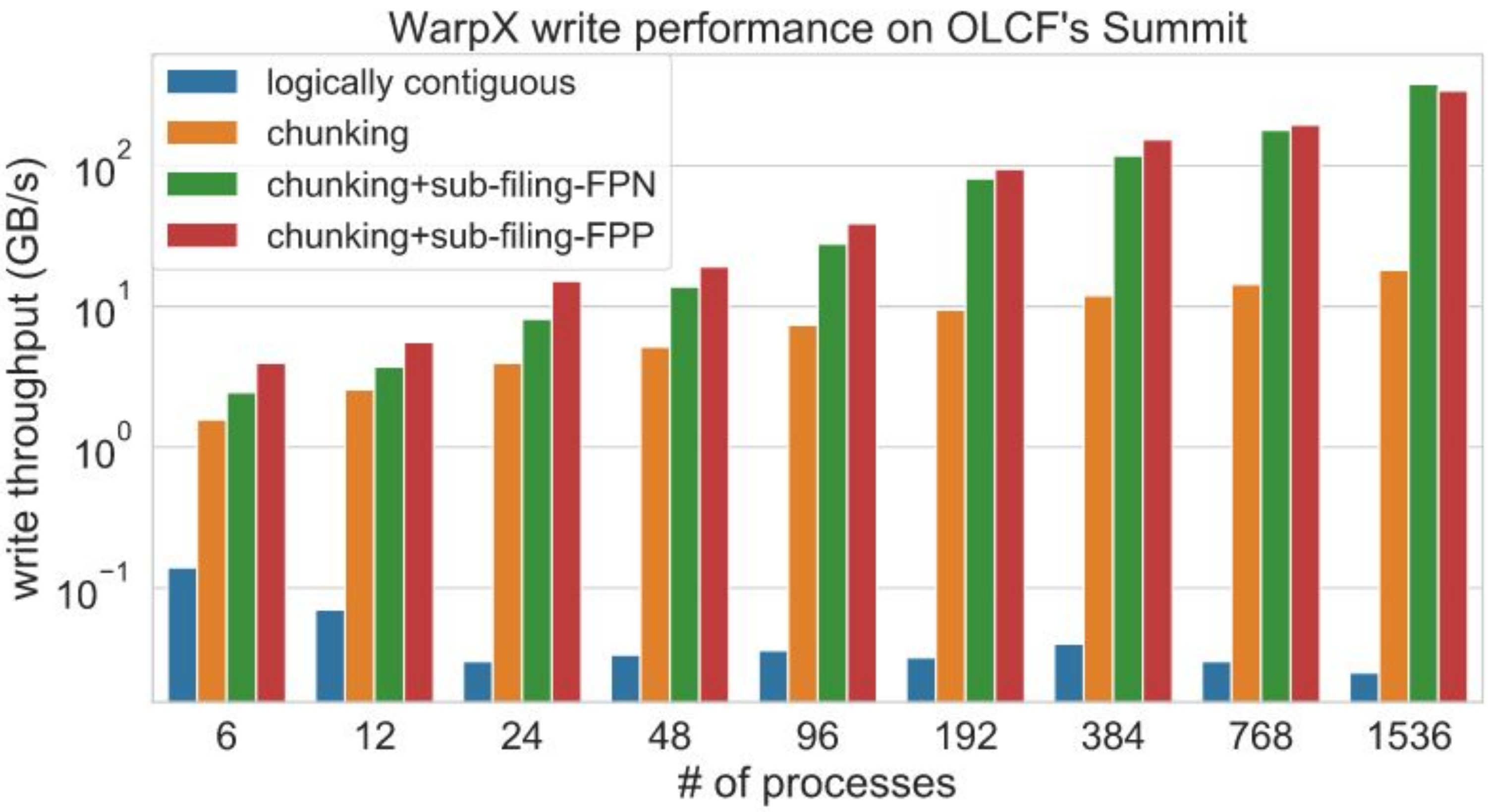
Write: plotfiles → ADIOS BP per rank & step → append to files



Aggregation: reduce number of files 6x

J Gu, P Davis, G Eisenhauer, W Godoy, A Huebl, S Klasky, M Parashar, N Podhorszki, F Poeschel, J-L Vay, L Wan, R Wang, and K Wu, "Organizing Large Data Sets for Efficient Analyses on HPC Systems," submitted (2021)

Parallel I/O: Logically Contiguous, Chunking, Sub-Filing



WarpX write performance under weak scaling tests

L. Wan, A. Huebl, J. Gu, F. Poeschel, A. Gainaru, R. Wang, J. Chen, X. Liang, D. Ganyushin, T. Munson, I. Foster, J.-L. Vay, N. Podhorszki, S. Klasky, “Data Layout Strategies for Parallel I/O: The Good, The Bad and The Ugly,” submitted (2021)

Data Layout & Merging Improve Start-2-End Performance

Fast Writes
- Do-able

Fast Reads
- Very hard

Bridging both
- algorithm development
- library *implementations*
- integration in **scalable** analysis frameworks

Looking Ahead
- in-situ and in-transit pipelines
- smoothly transition domain scientists **from post-hoc to in-situ** scripts

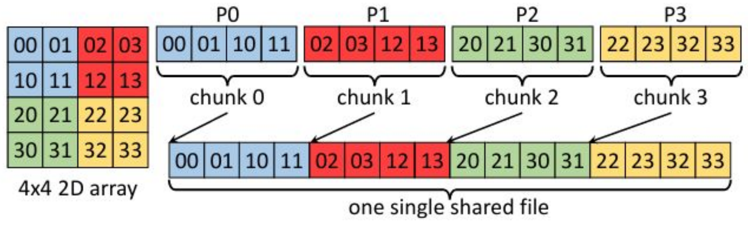


Fig. 2: Chunked Data layout: 1 chunk per process

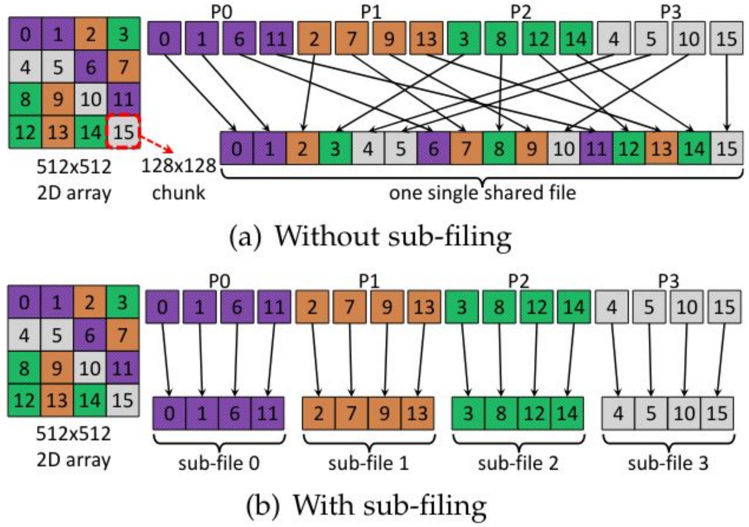


Fig. 3: Data layout: 4 chunks per process, 4 processors

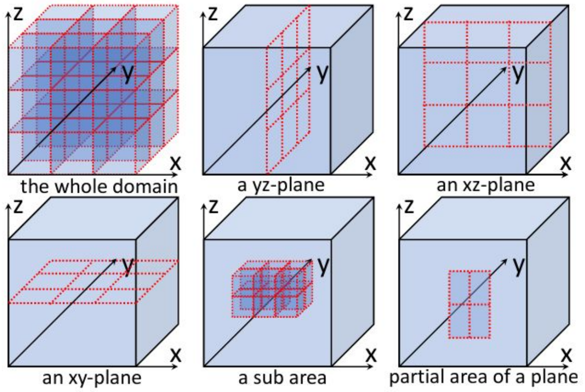


Fig. 5: Six common read patterns of a 3D mesh variable

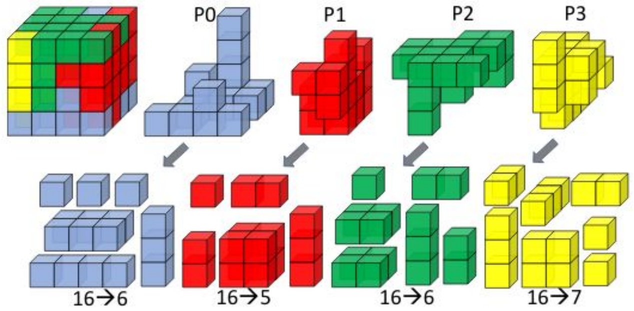


Fig. 8: Merging small blocks into bigger cuboid blocks

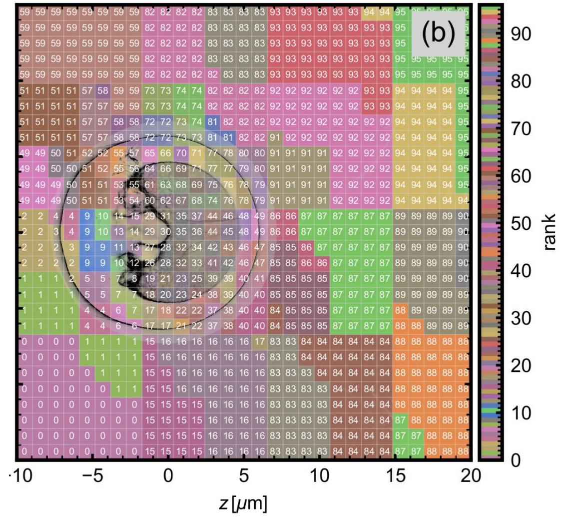
L. Wan, A. Huebl, J. Gu, F. Poeschel, A. Gainaru, R. Wang, J. Chen, X. Liang, D. Ganyushin, T. Munson, I. Foster, J.-L. Vay, N. Podhorszki, S. Klasky, "Data Layout Strategies for Parallel I/O: The Good, The Bad and The Ugly," submitted (2021)

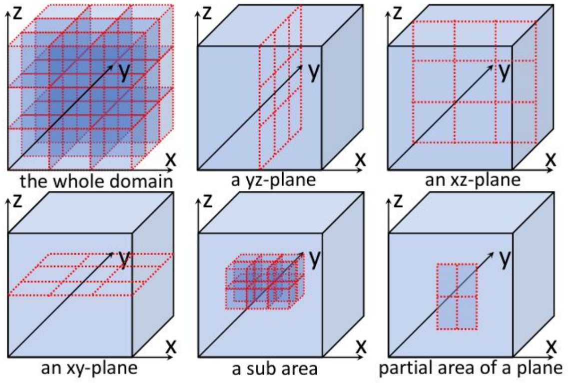
Block Meta-Data & Real-World Parallel Read Performance

No one-fits-all data layout → bridging the gap & parallel reads

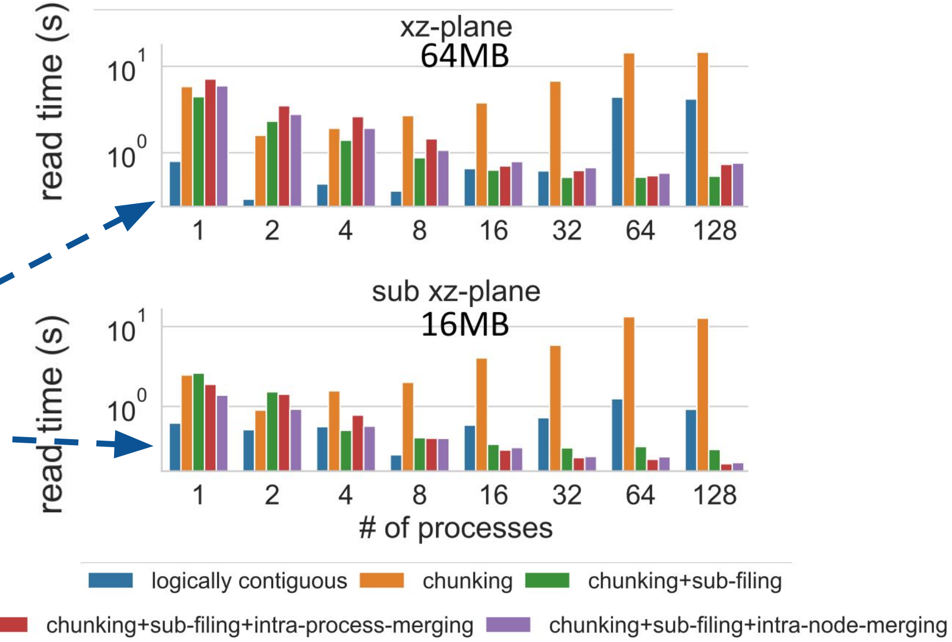
for example:

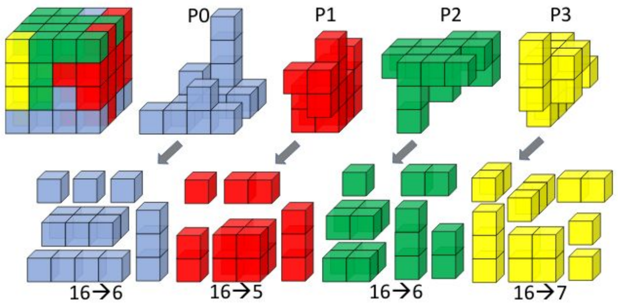
Z-order space filling curve



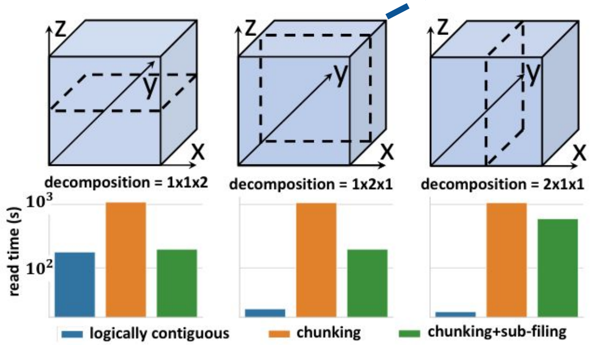


Six common read patterns of a 3D mesh variable





Merging small blocks into bigger cuboid blocks



Impact of decomposition schemes when reading

- ongoing: tuning MR fields +
  - lossy compressors: ZFP/MGARD
  - parallel post-processing; reordering
- orthogonal: async I/O (latency hiding)

L. Wan, A. Huebl, J. Gu, F. Poeschel, A. Gainaru, R. Wang, J. Chen, X. Liang, D. Ganyushin, T. Munson, I. Foster, J.-L. Vay, N. Podhorszki, S. Klasky, “Data Layout Strategies for Parallel I/O: The Good, The Bad and The Ugly,” submitted (2021)

Streaming: Avoid Files Altogether

Long-Term Strategy

- in-situ and in-transit pipelines
- smoothly transition domain scientists **from post-hoc to in-situ** scripts

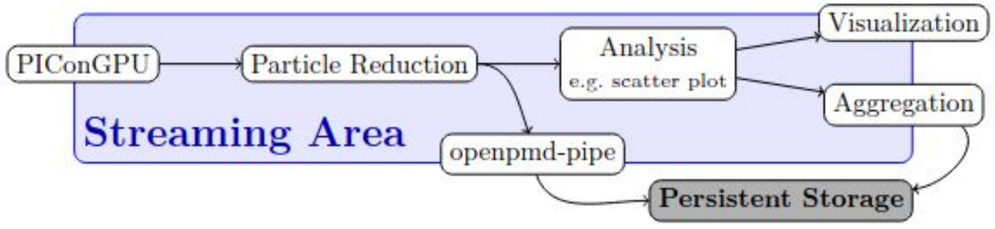


Fig. 2: A complex, loosely-coupled workflow: PIConGPU is the data producer, a domain-specific particle reduction can conserve relevant ensemble properties, the analysis step might filter and bin, and aggregation might create a temporal integration from high-frequency data. At various sections of the workflow, visualization or data dumps might be generated from subscribers.

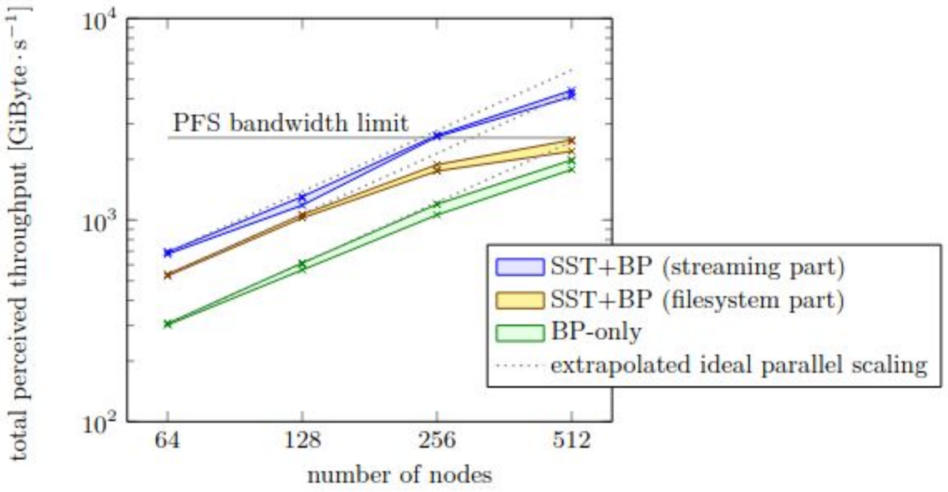


Fig. 5: Perceived total throughput. The file-based outputs (BP-only as well as SST+BP) are limited by the PFS bandwidth. At 512 nodes, the methods reach 4.15, 2.32, and 1.86 $\mathrm{TiByte \cdot s^{-1}}$ on average, respectively.

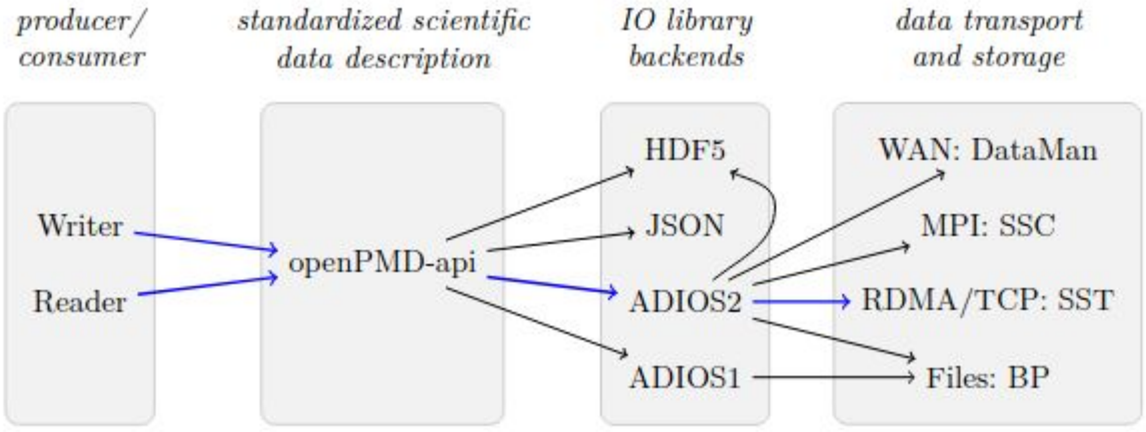


Fig. 3: IO software stack built by openPMD and ADIOS2

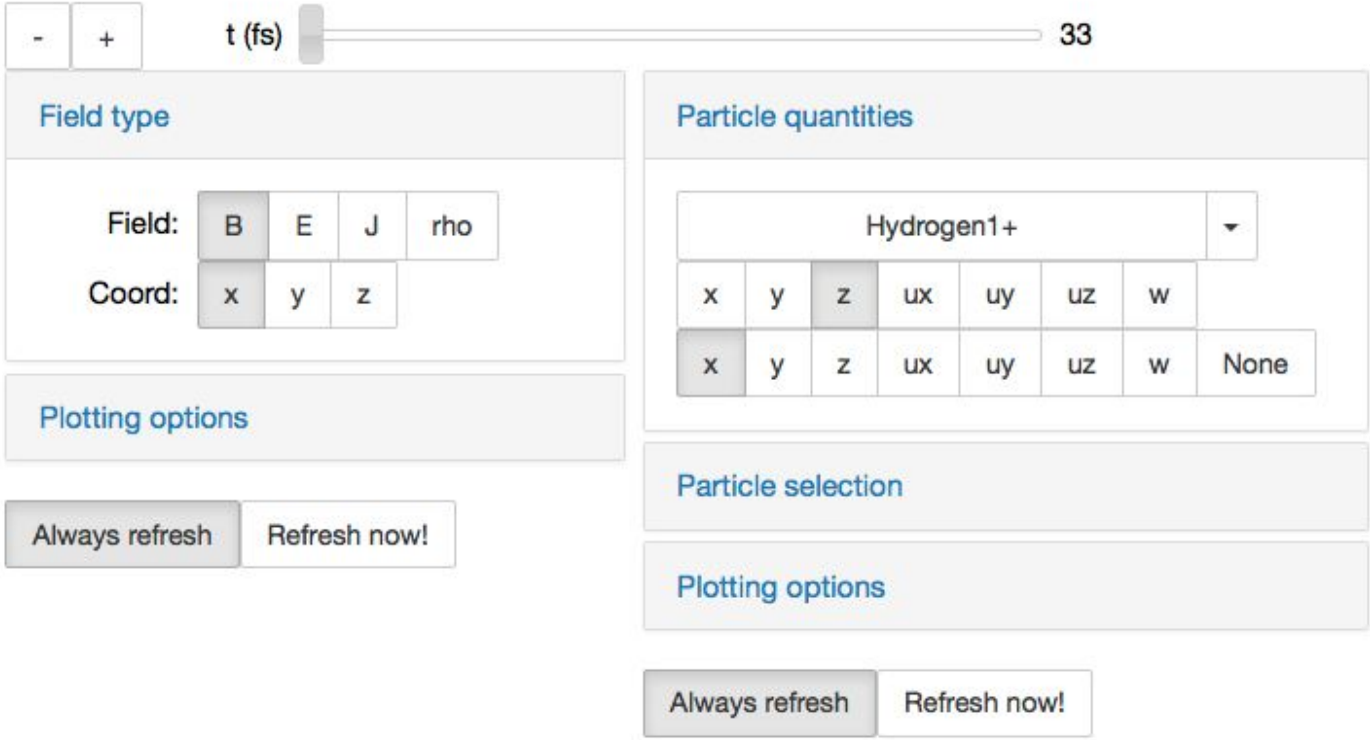
F. Poeschel, … A. Huebl. "Transitioning from file-based HPC workflows to streaming data pipelines with openPMD and ADIOS2," accepted in SMC21, https://arxiv.org/abs/2107.06108 (2021)

openPMD-viewer

maintainer: R Lehe (LBNL)

```
In [ ]: ts_2d.slider()
```

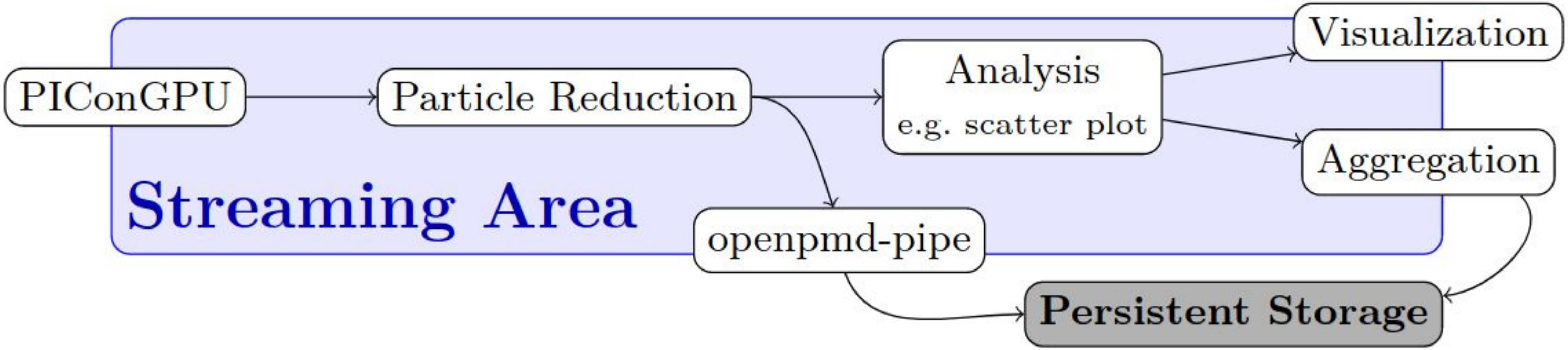
Calling this method will insert the following panel inside the notebook. (Note that the panel below is an **non-interactive image**, which is here for the users that are viewing this notebook online. Calling the `slider` method in a live notebook, will produce a truly **interactive** panel.)



Check out the tutorial notebooks: github.com/openPMD/openPMD-viewer

Vision: Loosely coupled data processing pipeline

Loose coupling: Cooperate between independent applications, exchanging data
Streaming I/O between application bypasses PFS bottleneck:



provide a uniform, scientific I/O communication layer between coupled applications and data processing stages: *from creation to archival & reuse*

F. Poeschel, … A. Huebl. "Transitioning from file-based HPC workflows to streaming data pipelines with openPMD and ADIOS2," accepted in SMC21, https://arxiv.org/abs/2107.06108 (2021)