CASUS
CENTER FOR ADVANCED
SYSTEMS UNDERSTANDING

www.casus.science

Plasma-PEPSC Workshop
23 October 2024

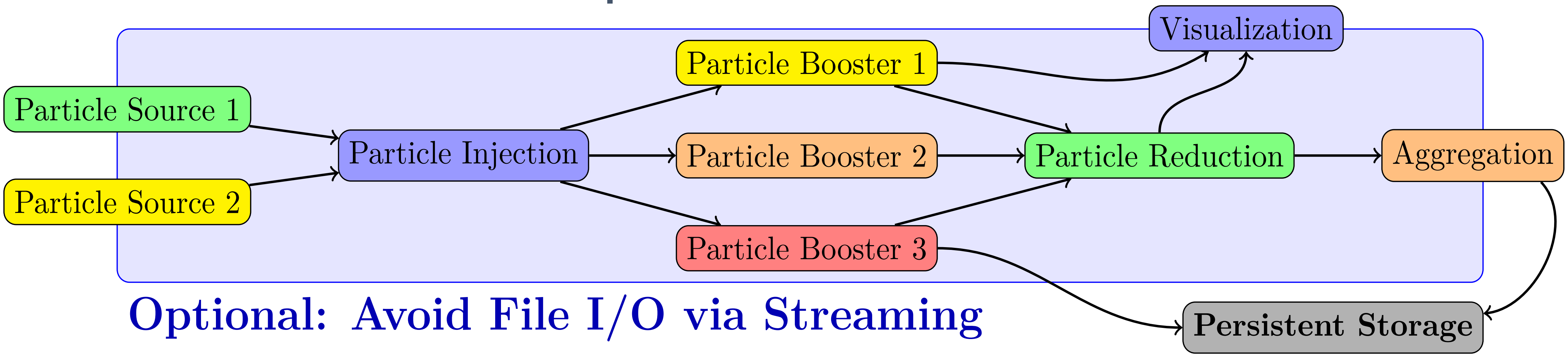
The Open Standard for Particle-Mesh Data

HZDR
HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF
TECHNISCHE UNIVERSITÄT DRESDEN
CBG Max Planck Institute of Molecular Cell Biology and Genetics
UFZ HELMHOLTZ Centre for Environmental Research
Uniwersytet Wrocławski

SPONSORED BY THE
Federal Ministry of Education and Research
STAATSMINISTERIUM FÜR WISSENSCHAFT KULTUR UND TOURISMUS
Freistaat SACHSEN

Heterogeneity through Standardized Data

Scientific workflows are complex:

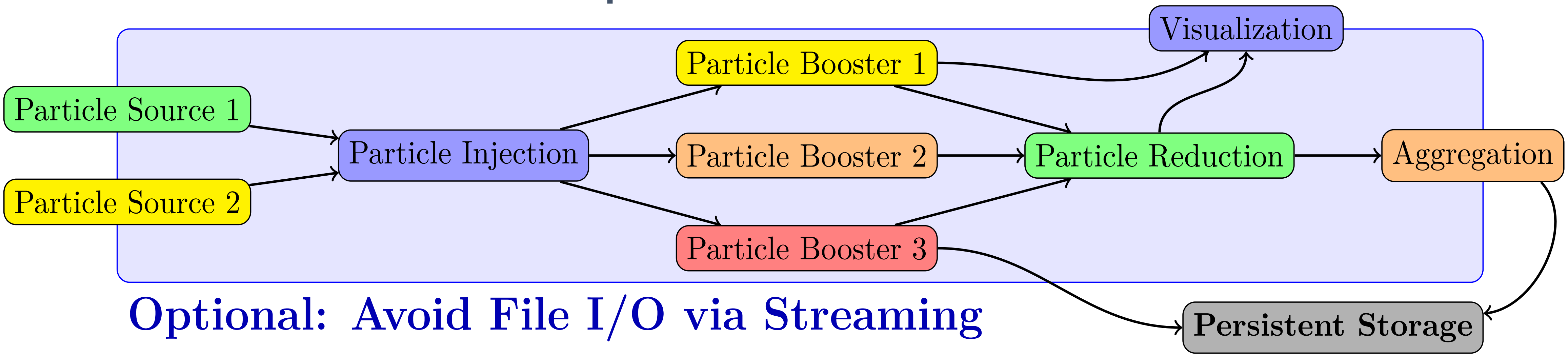


- need to span different **time** and **length scales**
- scientific modeling requires **multiple codes**, collaborating in a **data processing pipeline**
- **bridge heterogeneous models** by standardization of data

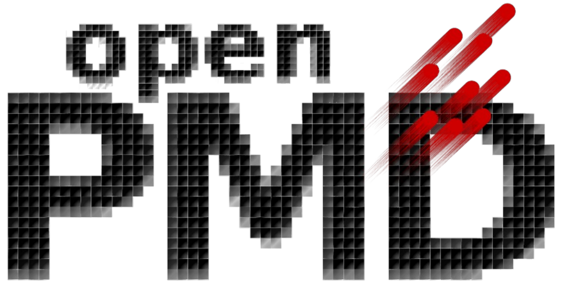
Axel Huebl et al. "openPMD: A meta data standard for particle and mesh based data". 2015. doi: 10.5281/zenodo.591699. url: https://openPMD.org
Franz Poeschel et al. "Transitioning from file-based HPC workflows to streaming data pipelines with openPMD and ADIOS2". 2021. doi:10.1007/978-3-030-96498-6_6

Heterogeneity through Standardized Data

Scientific workflows are complex:



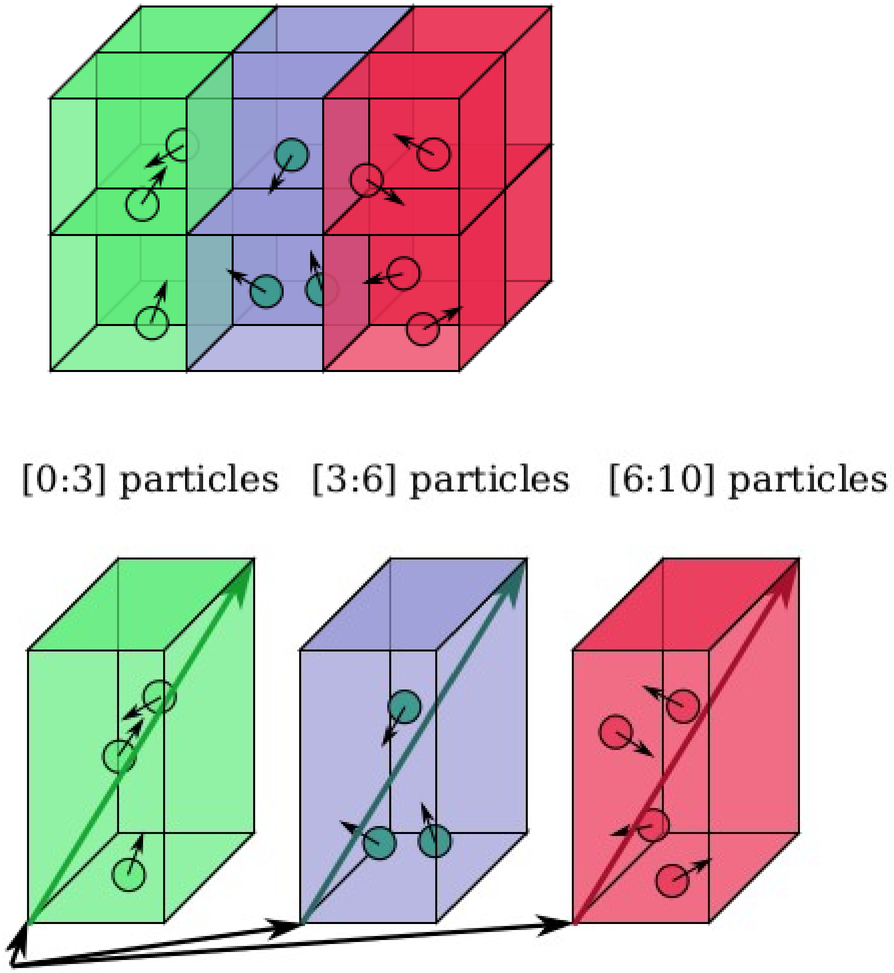
→ openPMD standard
for **p**article-**m**esh **d**ata
as communication layer



Axel Huebl et al. "openPMD: A meta data standard for particle and mesh based data". 2015. doi: 10.5281/zenodo.591699. url: https://openPMD.org

Franz Poeschel et al. "Transitioning from file-based HPC workflows to streaming data pipelines with openPMD and ADIOS2". 2021. doi:10.1007/978-3-030-96498-6_6

What is particle-mesh data?



Mesh

n-dimensional space,
divided into discrete cells

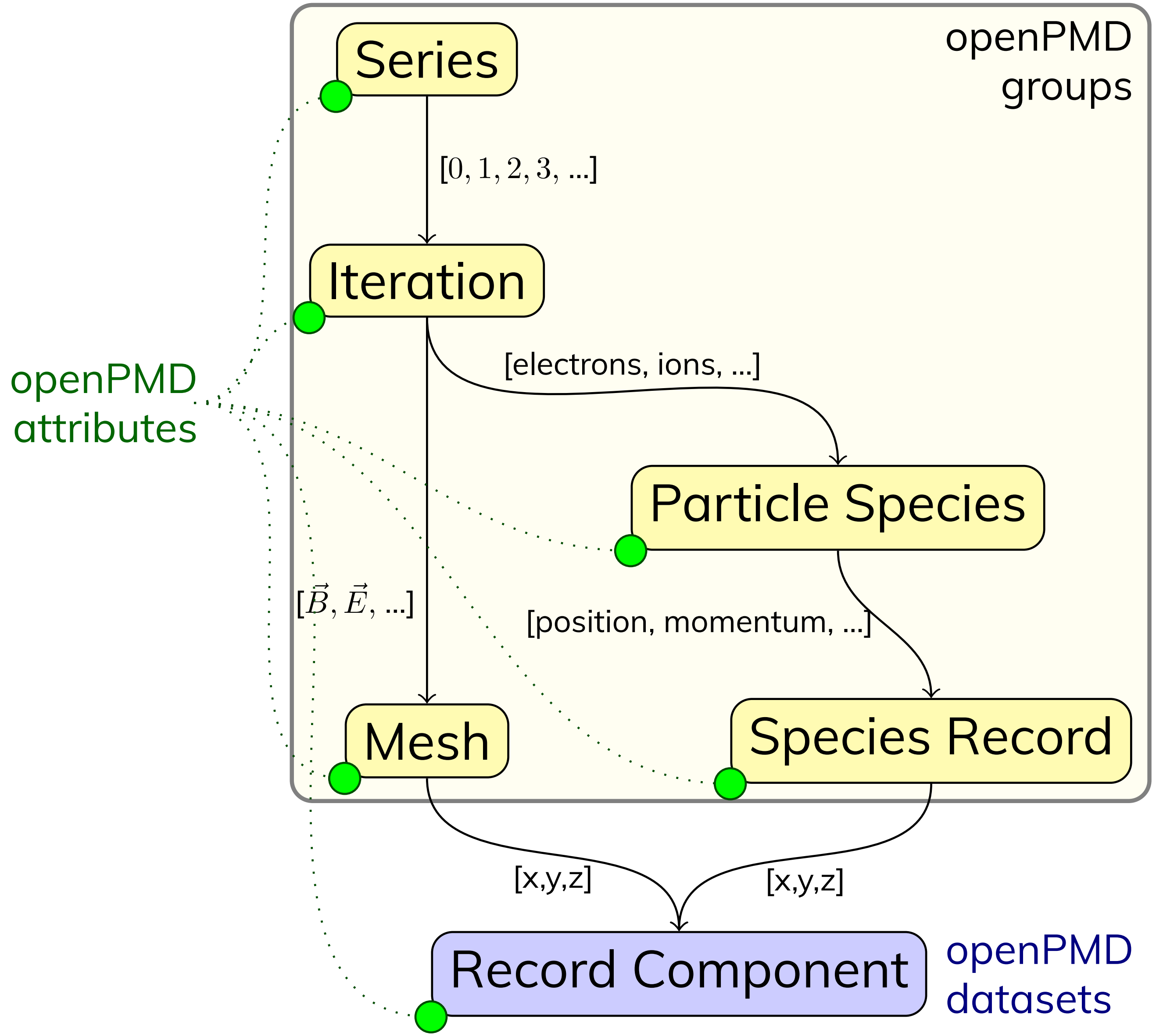
- e.g. temperature:
  store a scalar number per cell
- e.g. electrical fields:
  store a 3D vector per cell

Particles

A list of discrete objects,
located on the mesh

- for each particle: list its position
- optionally: list charge, weight, …

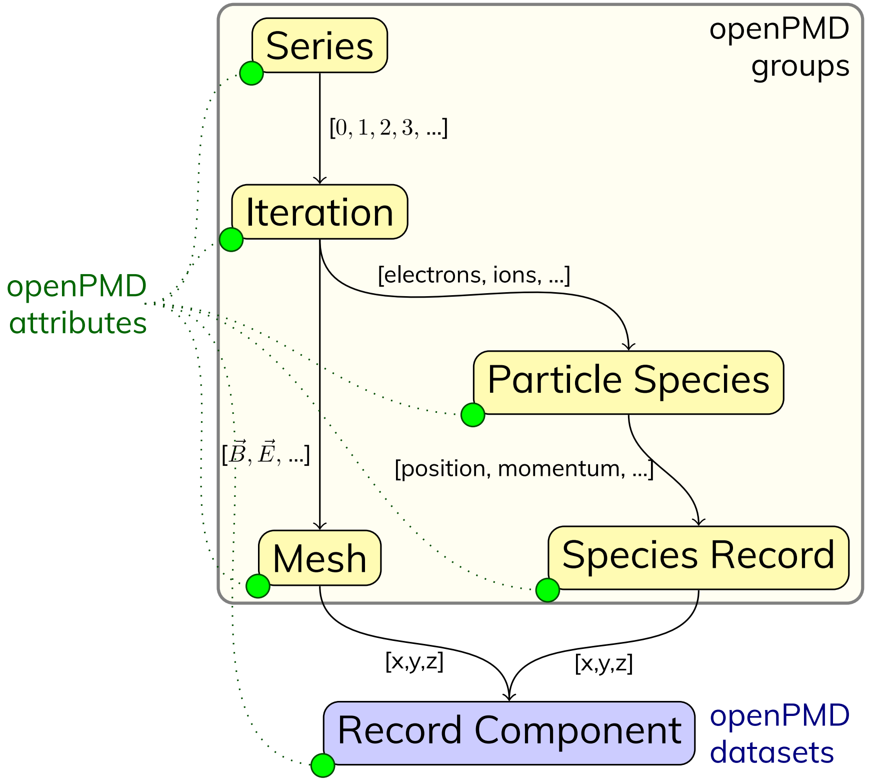
openPMD hierarchy

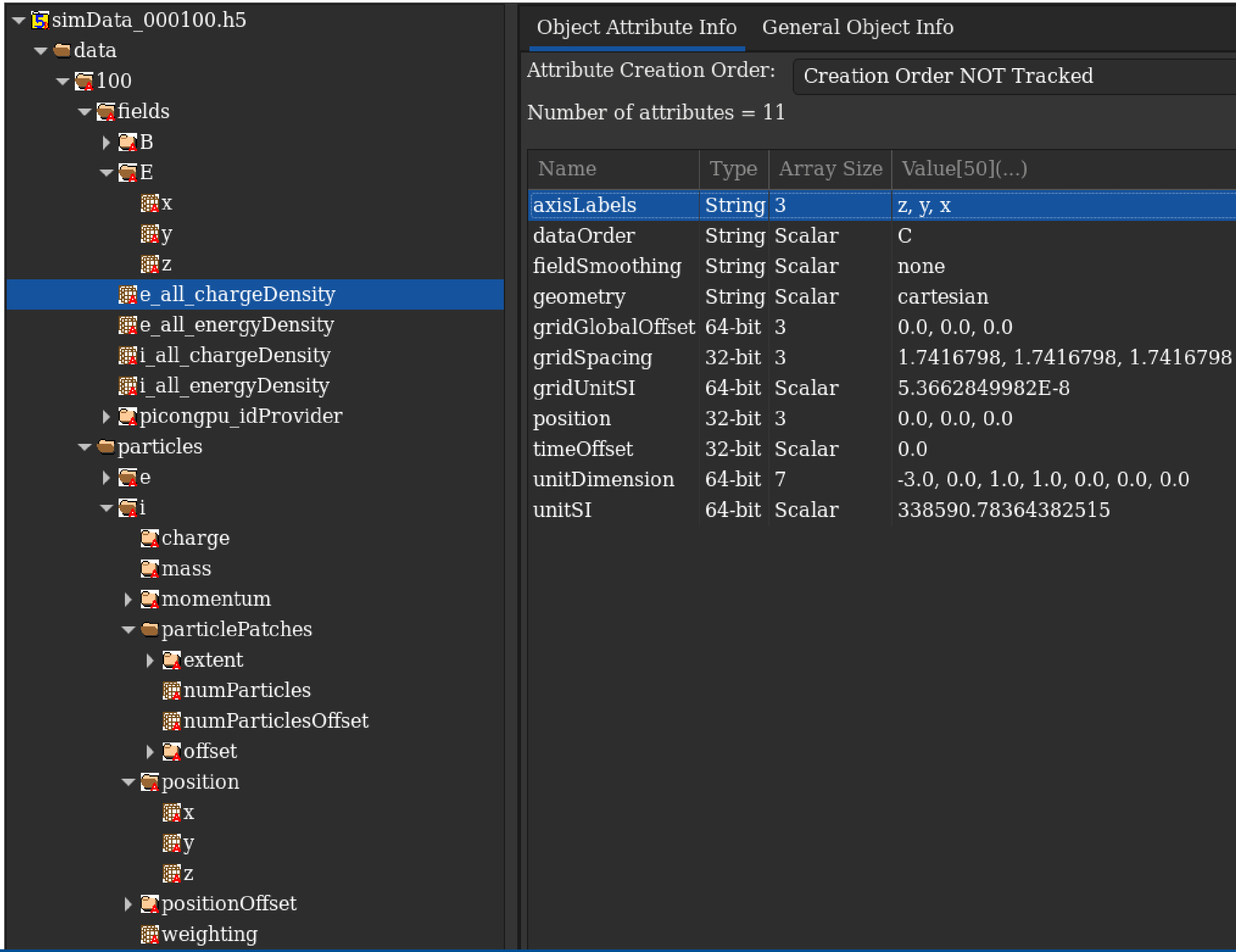


- **Structure** for series & snapshots encoded as either:
  - **files** (one file per iteration)
  - **groups** (reuse files)
  - **variables** (reuse files & variables in ADIOS2)

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

Example dataset: HDF5 backend





Sample data
created with PIConGPU

Example dataset: ADIOS2 backend

CASUS
CENTER FOR ADVANCED
SYSTEMS UNDERSTANDING

Hierarchical data organization

***n*-dim. datasets for heavyweight data**

```
float     /data/50/fields/E/x                        {128, 128, 128}
float     /data/50/fields/E/y                        {128, 128, 128}
float     /data/50/fields/E/z                        {128, 128, 128}
float     /data/50/particles/e/position/x            {50053105}
float     /data/50/particles/e/position/y            {50053105}
float     /data/50/particles/e/position/z            {50053105}
int32_t   /data/50/particles/e/positionOffset/x      {50053105}
int32_t   /data/50/particles/e/positionOffset/y      {50053105}
int32_t   /data/50/particles/e/positionOffset/z      {50053105}
```

Attributes for self-description

```
string    /data/50/fields/E/axisLabels          attr   = {"z", "y", "x"}
string    /data/50/fields/E/dataOrder           attr   = "C"
string    /data/50/fields/E/fieldSmoothing      attr   = "none"
string    /data/50/fields/E/geometry            attr   = "cartesian"
double    /data/50/fields/E/gridGlobalOffset    attr   = {0, 0, 0}
float     /data/50/fields/E/gridSpacing         attr   = {1.74168, 1.74168, 1.74168}
double    /data/50/fields/E/gridUnitSI          attr   = 5.36628e-08
float     /data/50/fields/E/timeOffset          attr   = 0
double    /data/50/fields/E/unitDimension       attr   = {1, 1, -3, -1, 0, 0, 0}
float     /data/50/fields/E/x/position          attr   = {0.5, 0, 0}
double    /data/50/fields/E/x/unitSI            attr   = 9.5224e+12
float     /data/50/fields/E/y/position          attr   = {0, 0.5, 0}
double    /data/50/fields/E/y/unitSI            attr   = 9.5224e+12
float     /data/50/fields/E/z/position          attr   = {0, 0, 0.5}
double    /data/50/fields/E/z/unitSI            attr   = 9.5224e+12
```

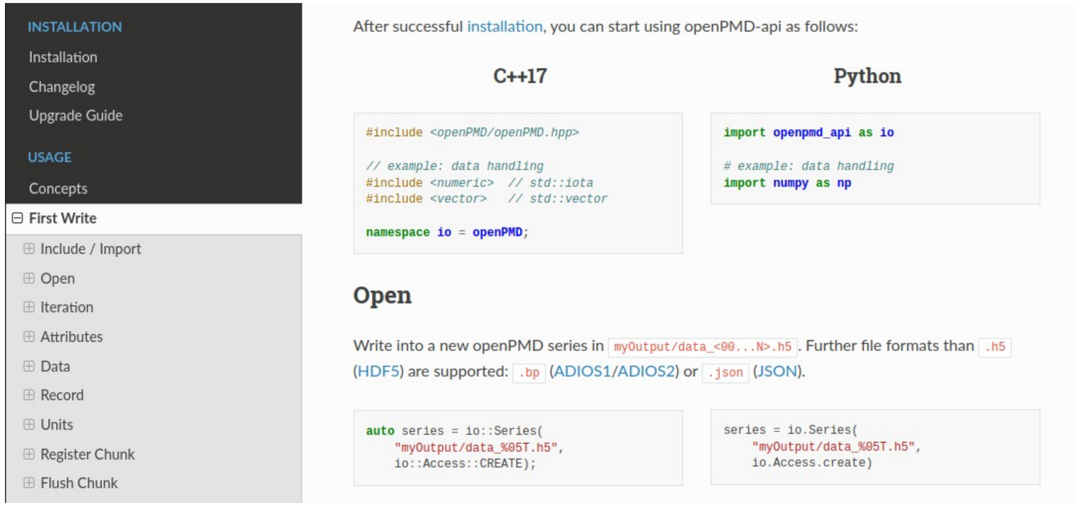
Example dataset: JSON/TOML backend


```
{
  "attributes": {
    "author": {
      "datatype": "STRING",
      "value": "franz"
    },
    "date": {
      "datatype": "STRING",
      "value": "2020-10-08 19:29:13 +0200"
    },
    "some more...": null
  },
  "data": {
    "0": {
      "attributes": {
        "cell_depth": {
          "datatype": "DOUBLE",
          "value": 4.252342224121094
        },
        "cell_height": {
          "datatype": "DOUBLE",
          "value": 1.0630855560302734
        },
        "cell_width": {
          "datatype": "DOUBLE",
          "value": 4.252342224121094
        },
        "many many more": null
      },
      "fields": {
        "B": {
            "attributes": {
                "axisLabels": {
                  "datatype": "VEC_STRING",
                  "value": [
                    "z",
                    "y",
                    "x"
                  ]
                }
            },
            "x": {
              "attributes": {
                "position": {
                  "datatype": "VEC_DOUBLE",
                  "value": [
                    0,
                    0.5,
                    0.5
                  ]
                },
                "unitSI": {
                  "datatype": "DOUBLE",
                  "value": 40903.82224060171
                }
              },
              "data": [
                [
                  [
                    "multidimensional dataset here"
                  ]
                ]
              ]
} } } } } }
```

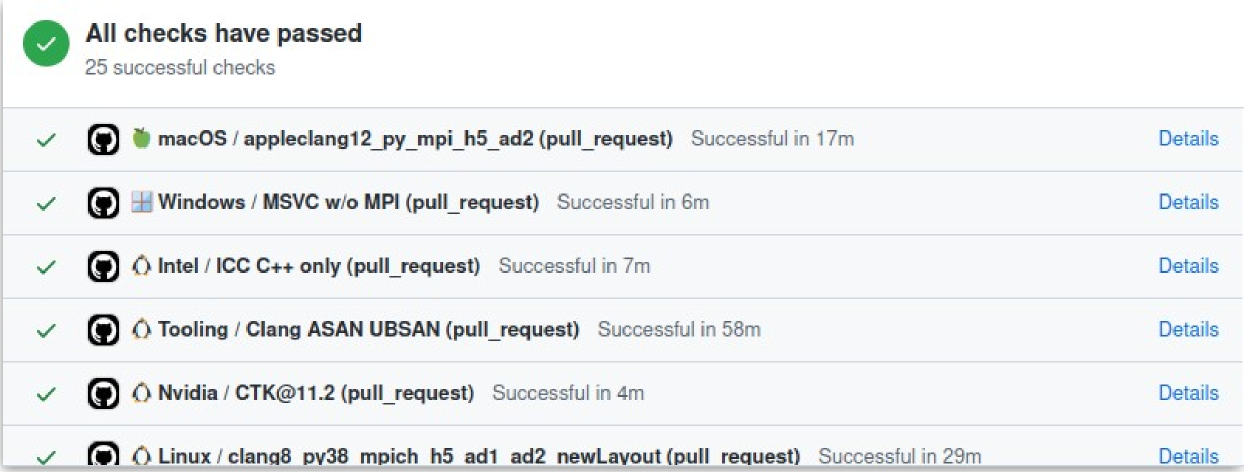

- Part of the package: No need to install 3rd-party dependencies
- Useful for debugging and prototyping
- Limited parallel support
- Courtesy to Nils Lohmann‘s JSON library for C++
- With recent release: Convert output to TOML
  Idea: openPMD formatted configuration files

Reference Implementation in C++ & Bindings: Python and 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

Hands-On:

openPMD-api: basic object model

Module environment at `/project/project_465001310/workshop_software/env.sh`
Materials at `https://github.com/alpaka-group/alpaka-workshop-slides/tree/oct2024_workshop`

Read the TODO comments inside `src/openPMDOutput.hpp`:

Unit System

unitDimension

automated description of physical dimension

only powers of base dimensions

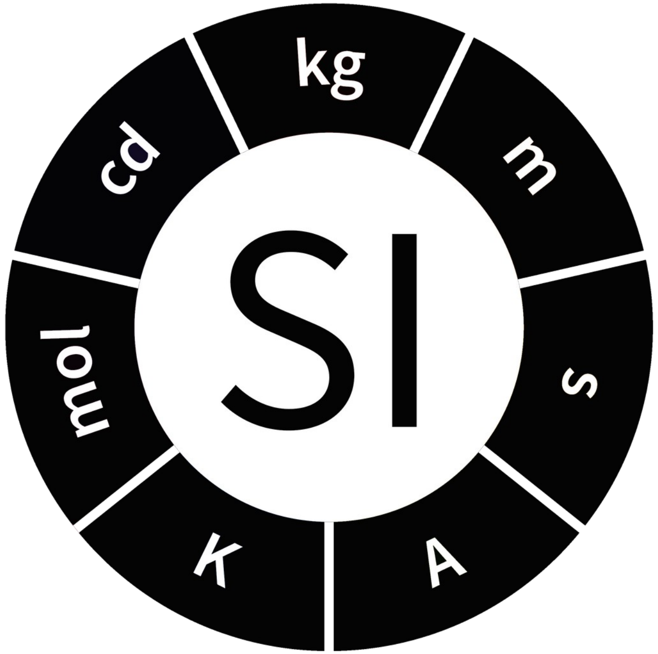
length **L**, mass **M**, time **T**, electric current **I**,
thermodynamic temperature **theta**,
amount of substance **N**, luminous intensity **J**

Magnetic field: [B] = M / (I * T²)

→ (0, 1, -2, -1, 0, 0, 0)

unitSI (recommended)

relation to an absolute unit system



Wikimedia Commons

openPMD – a 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

The HDF Group

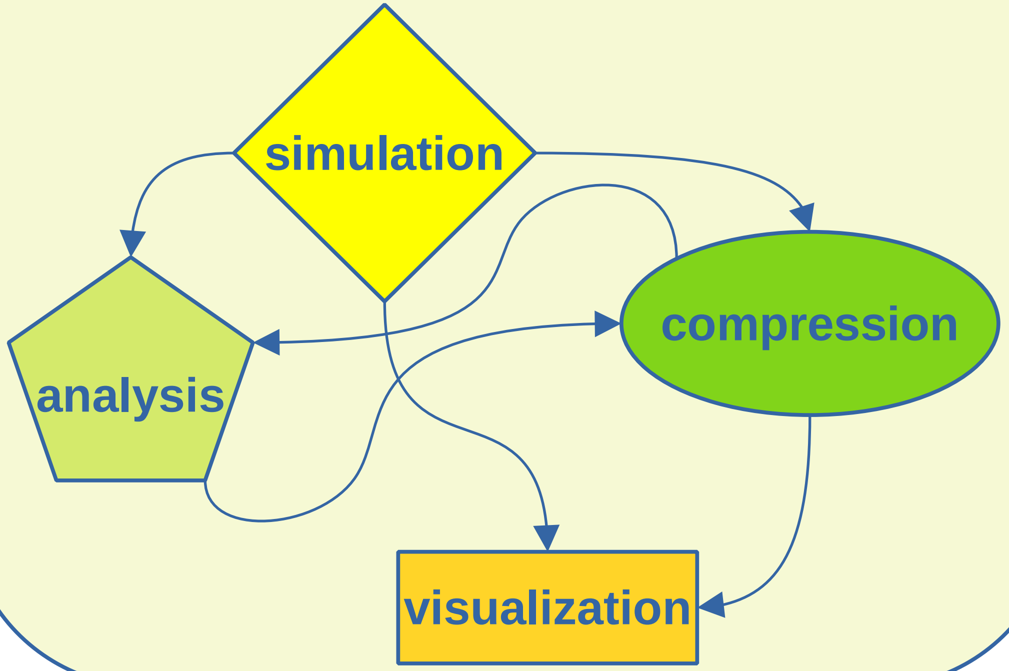
*currently implemented,
but not limited to

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

openPMD – a FAIR standard

Interoperable:

Data exchange spans applications, platforms and teams



Reusable:

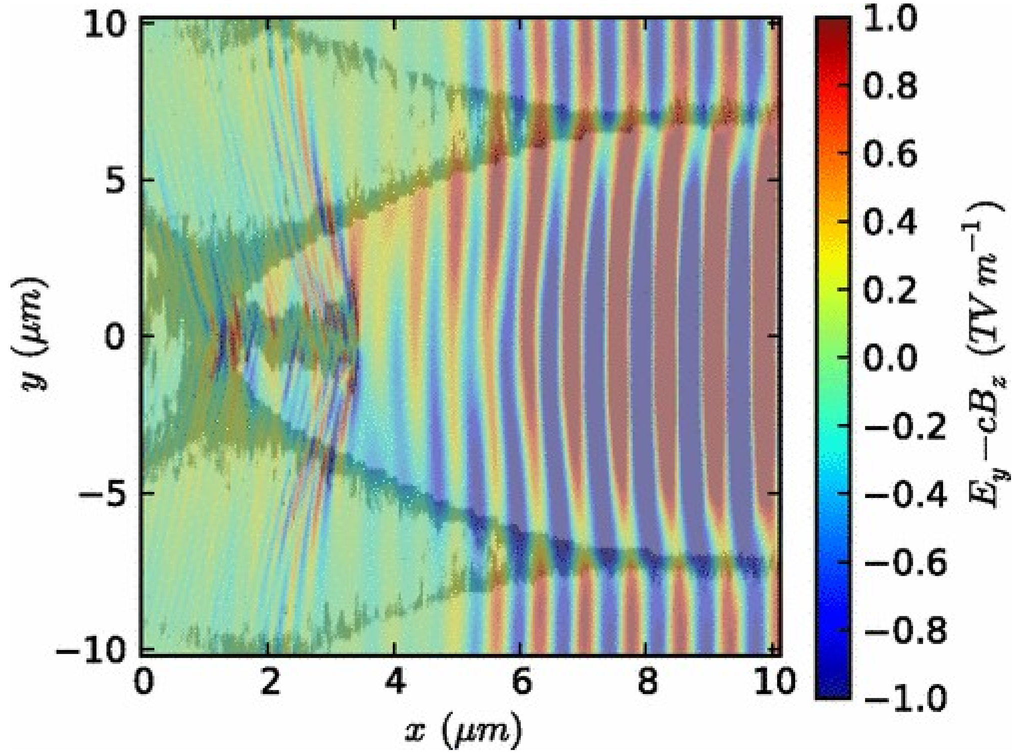
Rich and standardized description for physical quantities

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.)

Extensions: e.g. ED-PIC

Field image → field solver, smoothing



similar:

Emittance → particle push, field solver, shape

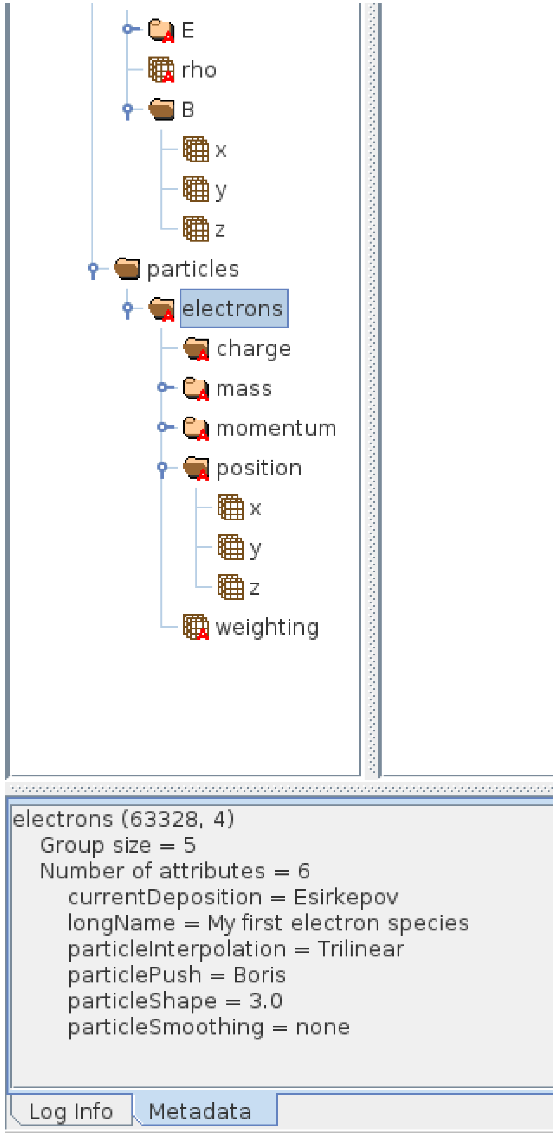


Image CC-BY 3.0: R. Lehe et al., RRSTAB 16, 021301 (2013),
DOI:10.1103/PhysRevSTAB.16.021301

CASUS CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

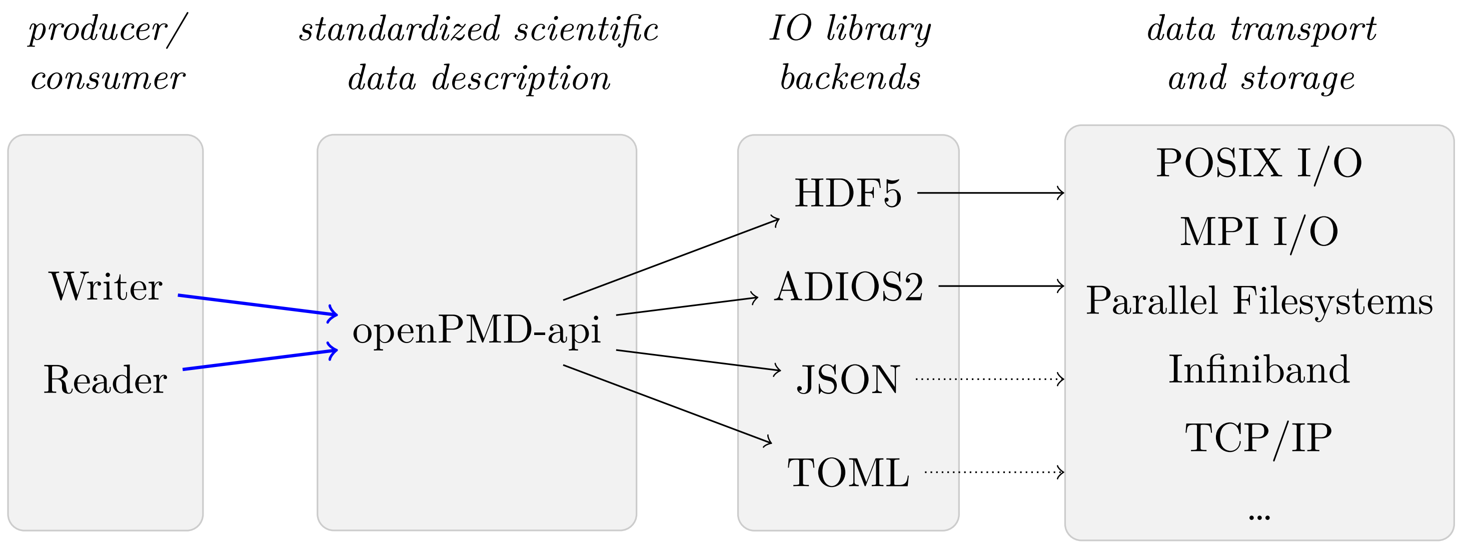
Hands-On:

openPMD-api: metadata

Module environment at `/project/project_465001310/workshop_software/env.sh`
Materials at `https://github.com/alpaka-group/alpaka-workshop-slides/tree/oct2024_workshop`

Read the TODO comments inside `src/openPMDOutput.hpp`:

openPMD-api – open stack for scientific I/O




```python
import openpmd_api as io

# pick and configure backend via JSON/TOML or inferred from filename extension
adios_config = """
  backend = "adios2"
  [[adios2.dataset.operators]]
  type = "blosc" # activate compression
"""
mode = io.Access.create
series = io.Series("simOutput.h5",   mode
                   """{"hdf5": {"vfd": {"type": "subfiling"}}}""")
series = io.Series("simOutput.bp5",  mode, adios_config)
series = io.Series("simOutput.sst",  mode, "@./or/load/config/from/file.json")
series = io.Series("simOutput.json", mode)
```

- MPI support at all levels
- Implemented in C++17
- Bindings in C++17, Python and (dev version only) Julia
- Specify backend at runtime:
  I/O library, transport, compression, streaming, aggregation, …

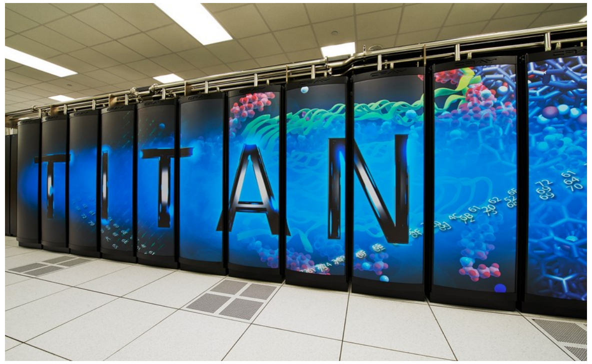
Hands-On:

openPMD-api: visualization, backend configuration

Module environment at `/project/project_465001310/workshop_software/env.sh`
Materials at `https://github.com/alpaka-group/alpaka-workshop-slides/tree/oct2024_workshop`
Launch Notebook at `https://www.lumi.csc.fi/`

Read the instructions inside `src/next_steps.md`:

I/O Performance lags behind Compute Performance



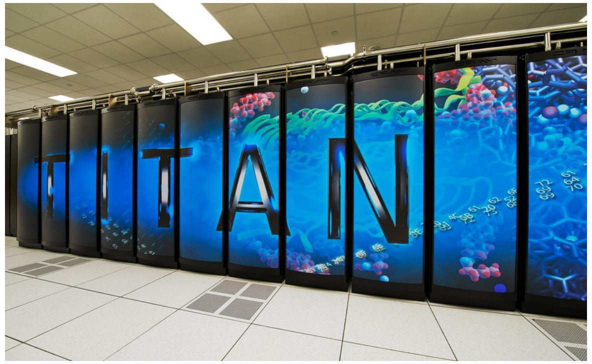
	Titan	Summit	Frontier	*Growth Factor*
Peak Performance:	27 Pflop/s	200 Pflop/s	1.6 Eflop/s	~60
FS Throughput:	1 TiByte/s	2.5 TiByte/s	5~10 TiByte/s	5~10
FS Capacity:	27 PiByte	250 PiByte	500~1000 PiByte	18~37

→ **parallel bandwidth** insufficient for HPC at full scale
→ **filesystem capacity** insufficient for HPC at full scale

Same trend in **experiments**?
→ Increasing **camera resolutions and data rates**

Franz Poeschel et al. "Transitioning from file-based HPC workflows to streaming data pipelines with openPMD and ADIOS2". 2022. doi: 10.1007/978-3-030-96498-6_6.

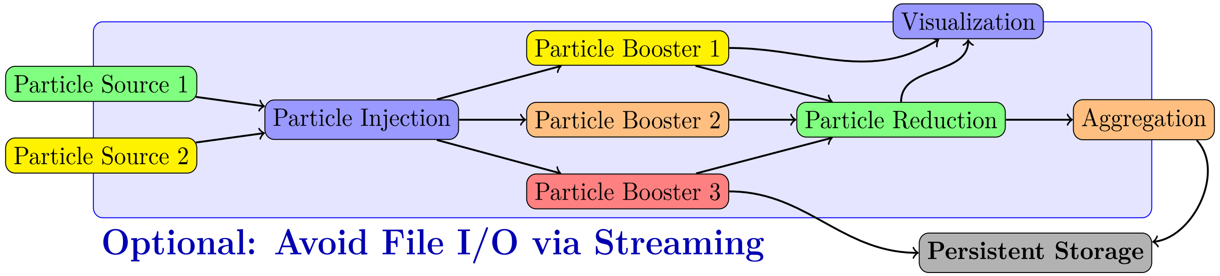
Compute Performance Outpaces Storage Performance



	Titan	Summit	Frontier	*Growth Factor*
Peak Performance:	27 Pflop/s	200 Pflop/s	1.6 Eflop/s	~60
FS Throughput:	1 TiByte/s	2.5 TiByte/s	5~10 TiByte/s	5~10
FS Capacity:	27 PiByte	250 PiByte	500~1000 PiByte	18~37

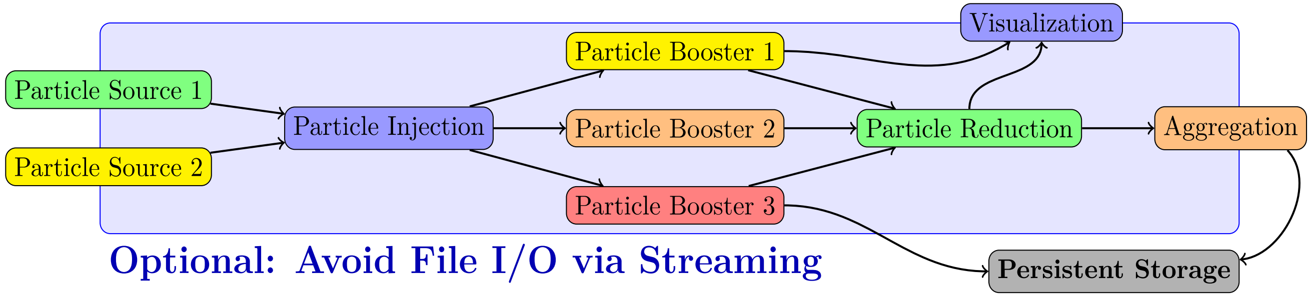
Why does this concern us?

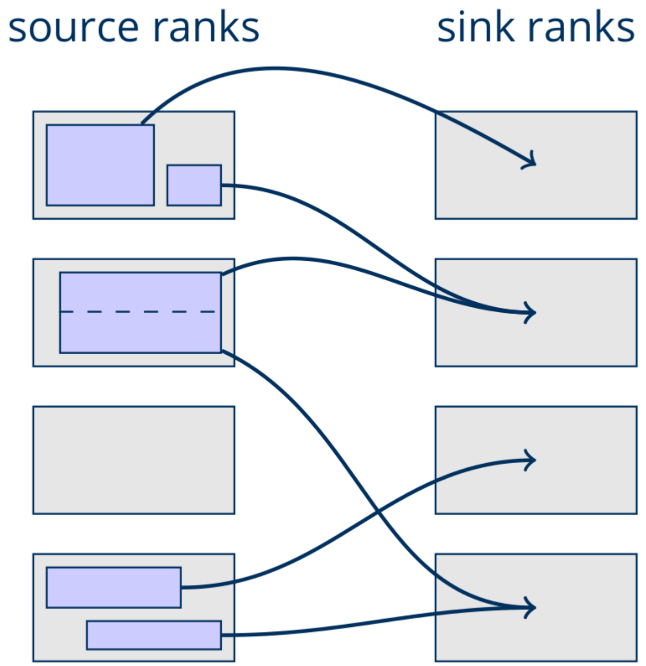
- Heterogeneous data processing pipelines traditionally have large I/O usage
- Scalable alternative: Streaming



Franz Poeschel et al. "Transitioning from file-based HPC workflows to streaming data pipelines with openPMD and ADIOS2". 2022. doi: 10.1007/978-3-030-96498-6_6.

Streaming: Don't touch the Filesystem at all



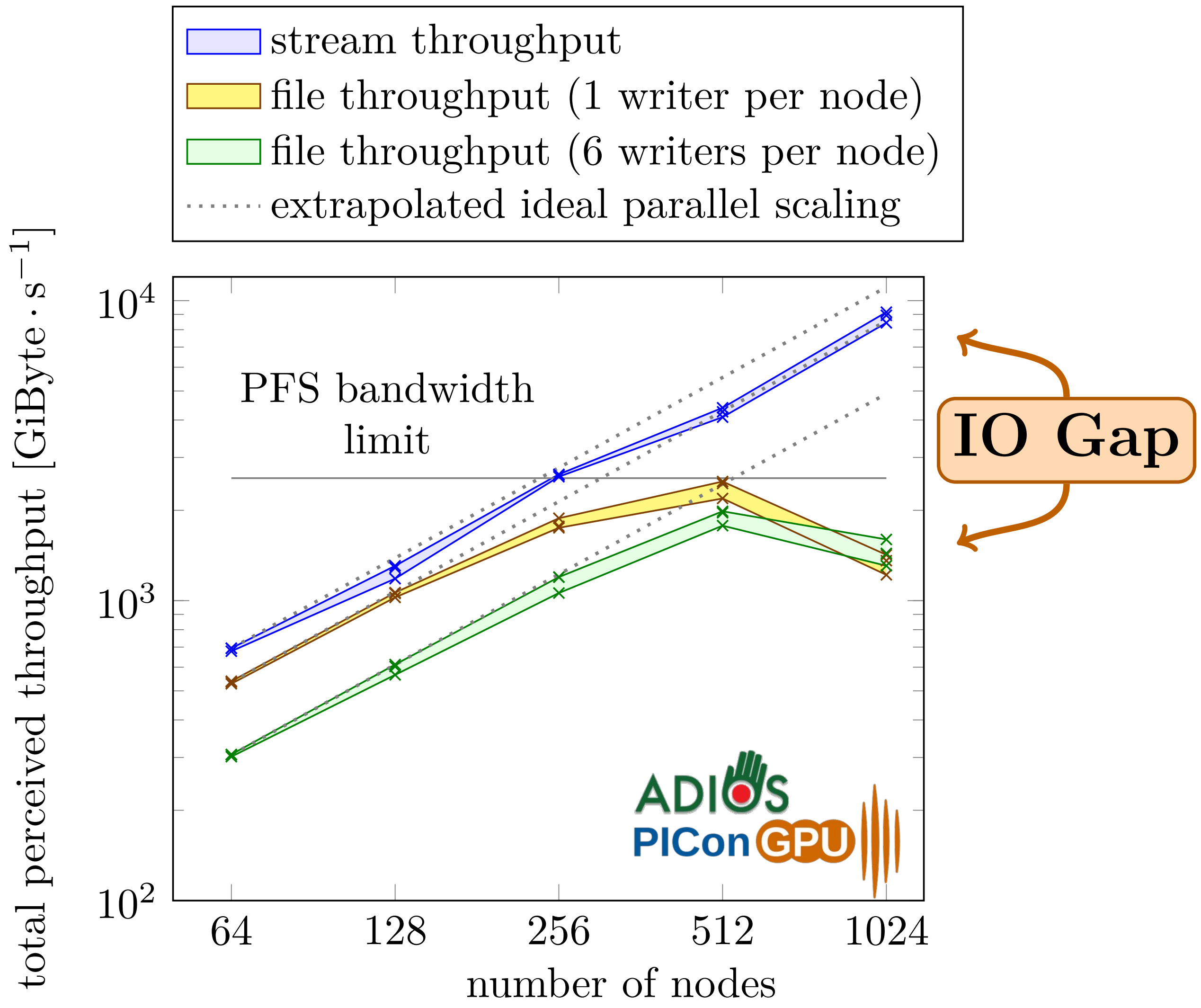


- Data processing pipelines and increasingly experiments setups have large I/O usage
- Scalable alternative: Streaming
  e.g. via Infiniband (on HPC systems)
  or wide area networks (in lab settings)

Challenge:
Compute a balanced, aligned, local mapping between two applications that remains useful in the problem domain

Franz Poeschel et al. "Transitioning from file-based HPC workflows to streaming data pipelines with openPMD and ADIOS2". 2022. doi: 10.1007/978-3-030-96498-6_6.

Break through Filesystem Bandwidth with Streaming

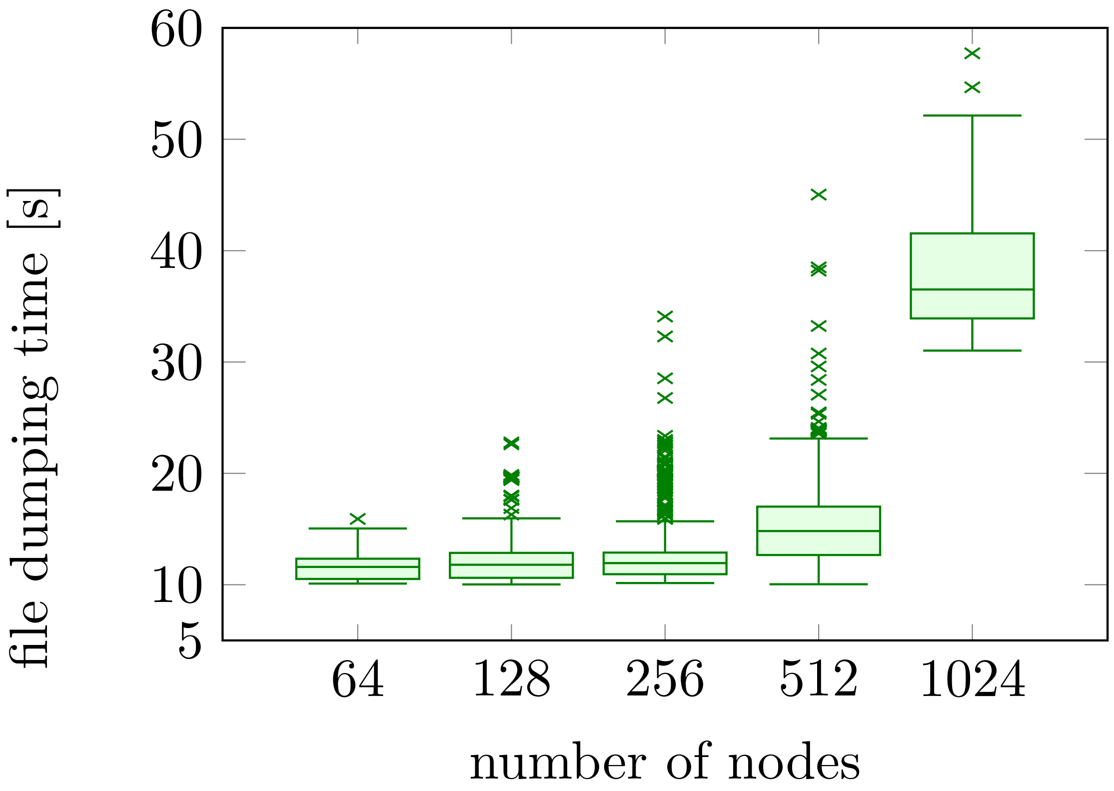


Memory-bound simulations reach the I/O system limits at a fraction of full scale

- Summit FS bandwidth (2.5TiByte/s) reached at 512 nodes (~11% of system size)
- Streaming workflows unaffected by filesystem bandwidth, use Infiniband hardware to scale beyond it

(benchmarks at 1024 nodes done after Summit system upgrade)

Summit: Performance fluctuations on single ranks



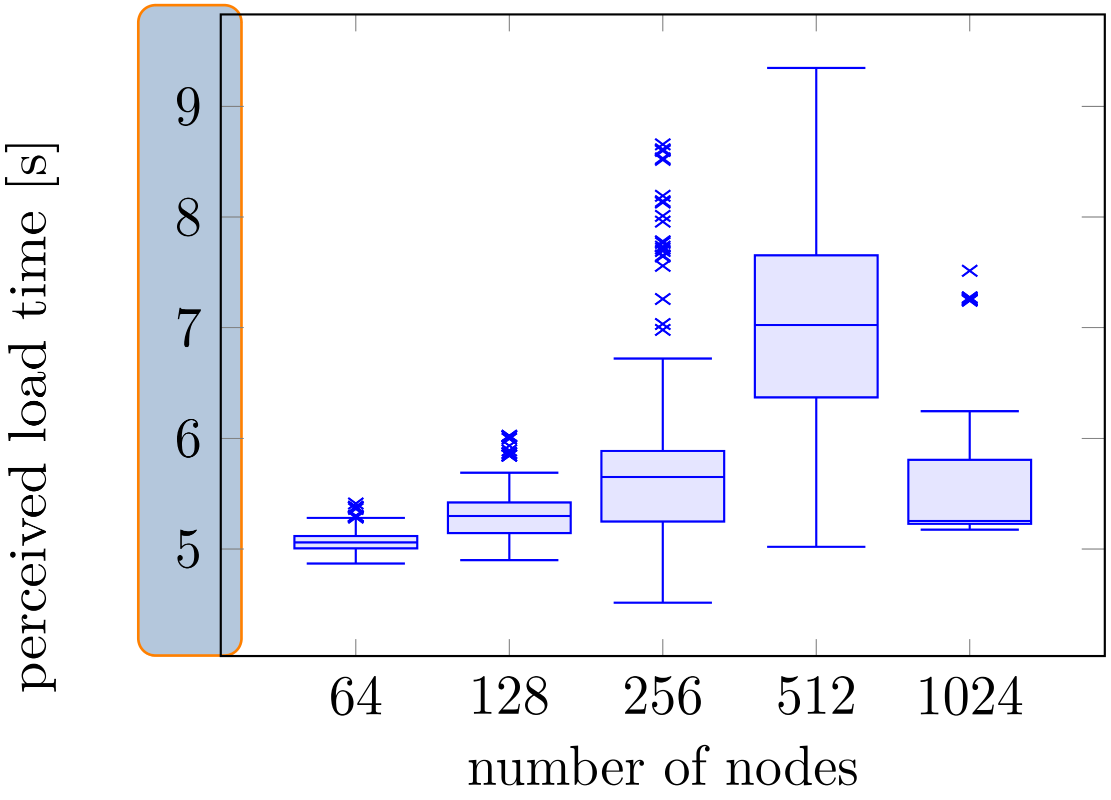
Same results, different display:

- Plot every single measurement
- Visualize reproducibility
- Box: 50% of measurement points
- Whiskers: "normal" measurements
- Others: outliers

Evaluation for file-only setup:

- Median time slightly raised at 512 nodes
  Scaling stops due to PFS limit after that
- Outliers increasing with scale
- Outliers fatal in parallel contexts

Summit: Well-reproducible performance of Infiniband Streaming



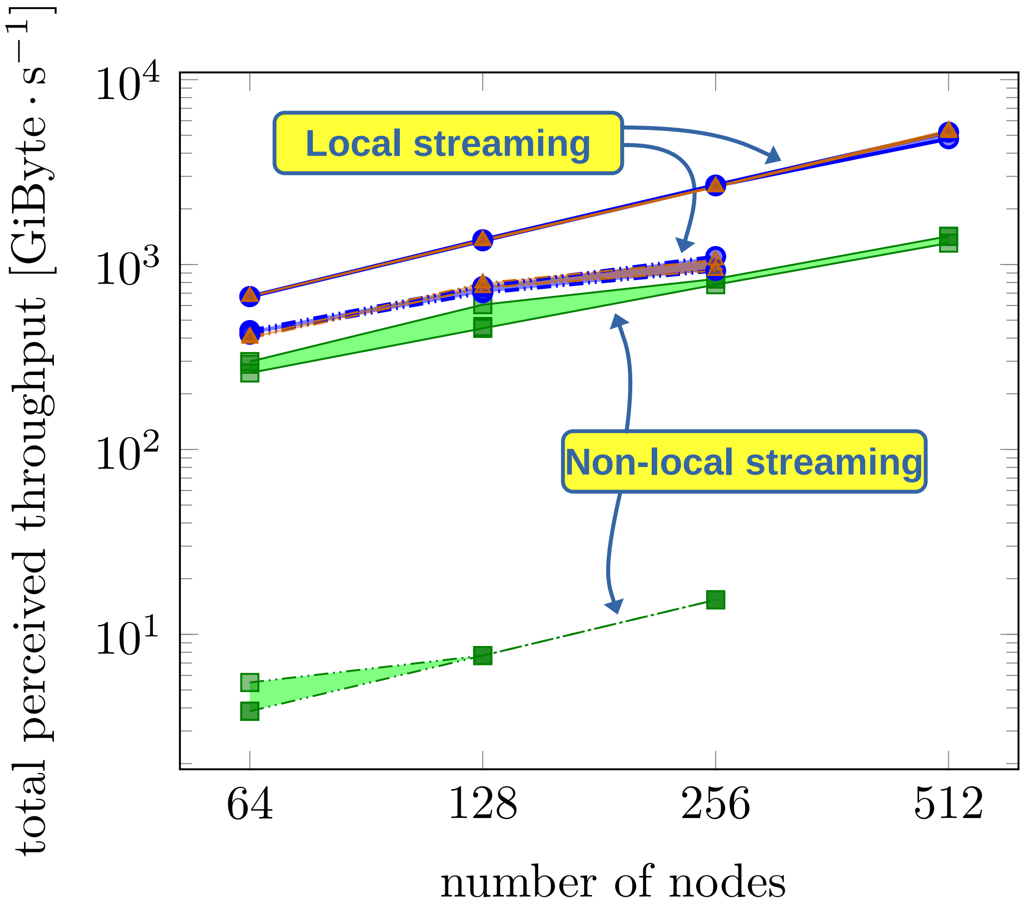
Same results, different display:

- Plot every single measurement
- Visualize reproducibility
- Box: 50% of measurement points
- Whiskers: “normal” measurements
- Others: outliers

stream+file setup (stream part):

- Overall times are lower
- Median between 5 and 7 seconds
- Outliers less dominating by far

For good throughput:
Local streaming patterns, Infiniband/RDMA



Local streaming:

Distribute data chunks only within a node

Non-local streaming:

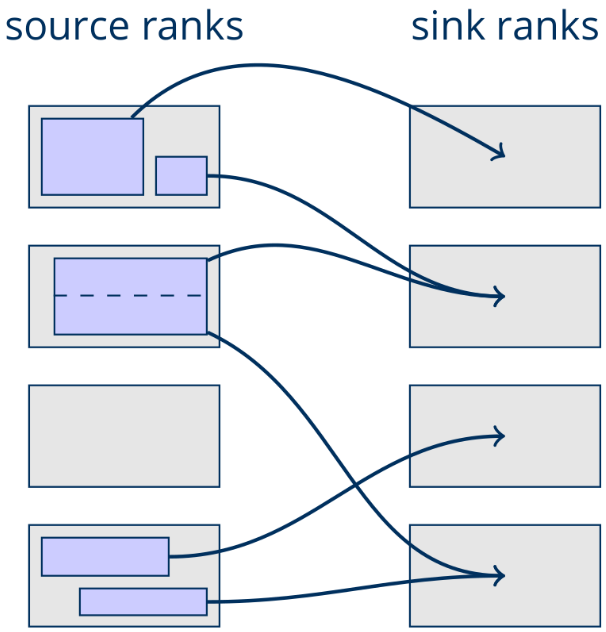
Distribute data chunks globally, optimize for balance and alignment

Straight lines:

Infiniband/RDMA

Dashed lines:

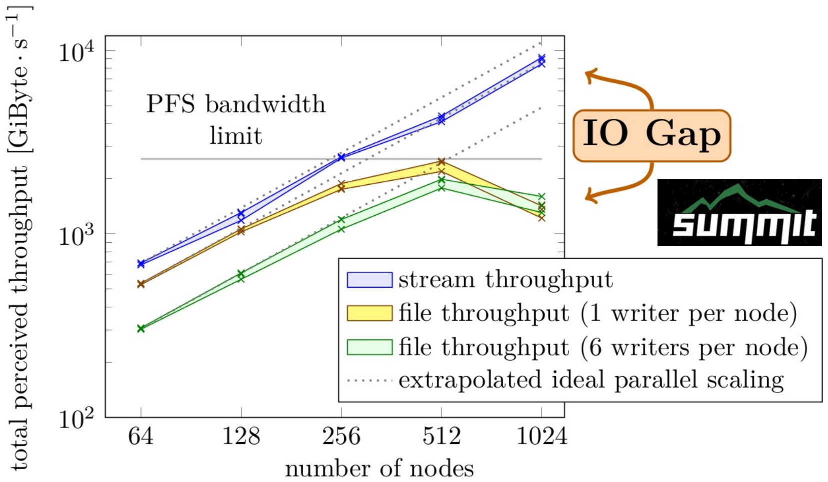
TCP/sockets



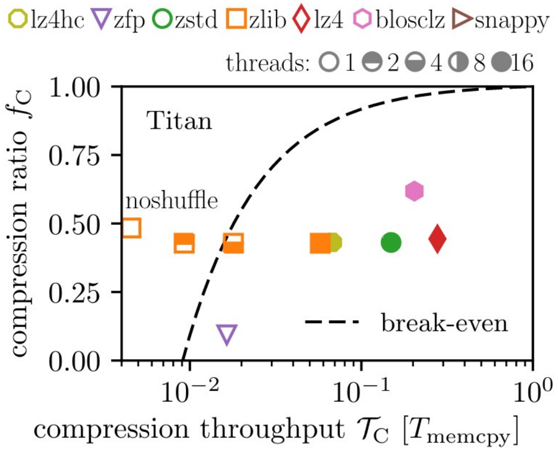
Setup: Couple PIConGPU with a scattering code (GAPD) exchange particle data only

J. C. E, L. Wang, S. Chen, Y. Y. Zhang and S. N. Luo. “GAPD: a GPU- accelerated atom-based polychromatic diffraction simulation code”. In: Journal of Synchrotron Radiation 25.2 (Mar. 2018), pp. 604–611.

Performance: Data Layouts and no-file I/O

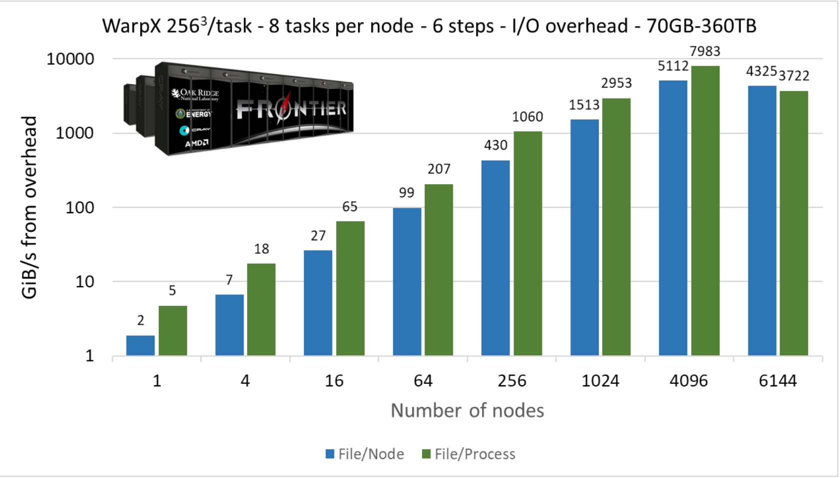


Streaming Data Pipelines:
DOI:10.1007/978-3-030-96498-6_6
by F Poeschel, A Huebl et al., SMC21 (2022)

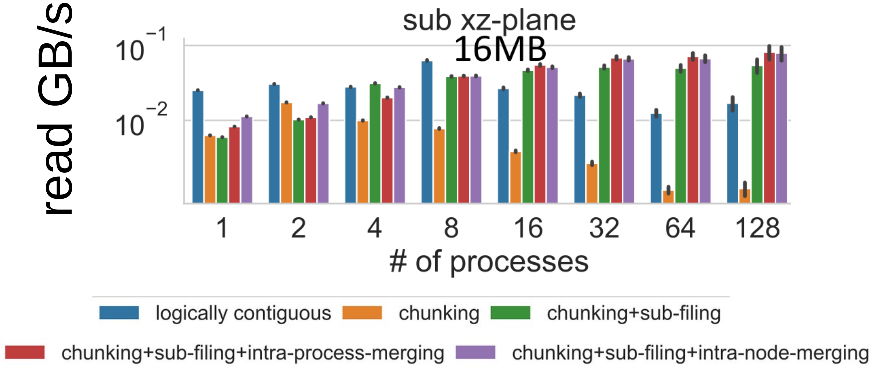


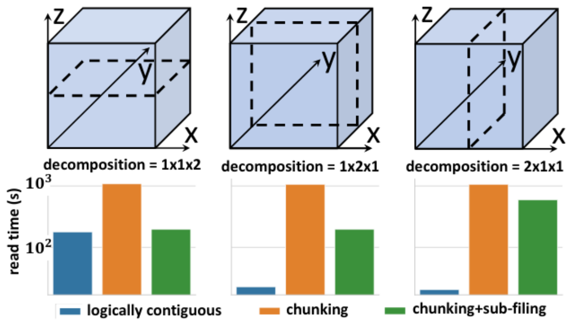
Fast Compressors Needed:
DOI:10.1007/978-3-319-67630-2_2
by A Huebl et al., ISC DRBSD-1 (2017)

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)

Hands-On:

openPMD-api: streaming I/O

Module environment at `/project/project_465001310/workshop_software/env.sh`
Materials at `https://github.com/alpaka-group/alpaka-workshop-slides/tree/oct2024_workshop`
Launch Notebook at `https://www.lumi.csc.fi/`

Read the instructions inside `src/visualize.py` (open as a Jupyter Notebook):

openPMD powered Projects and Users

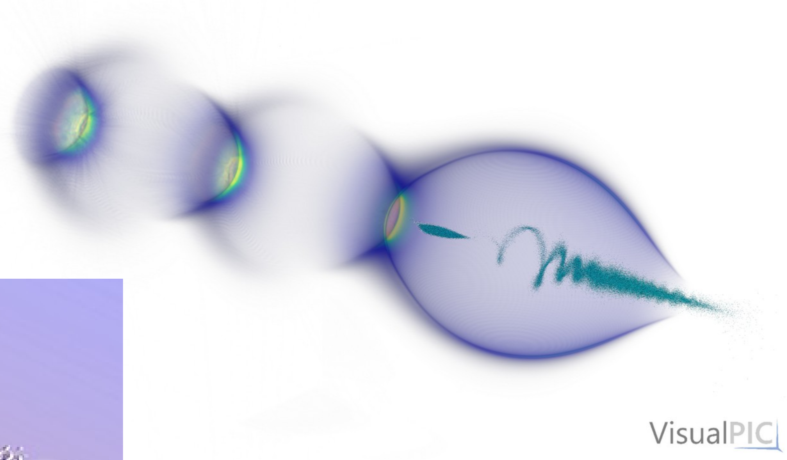
Documents:

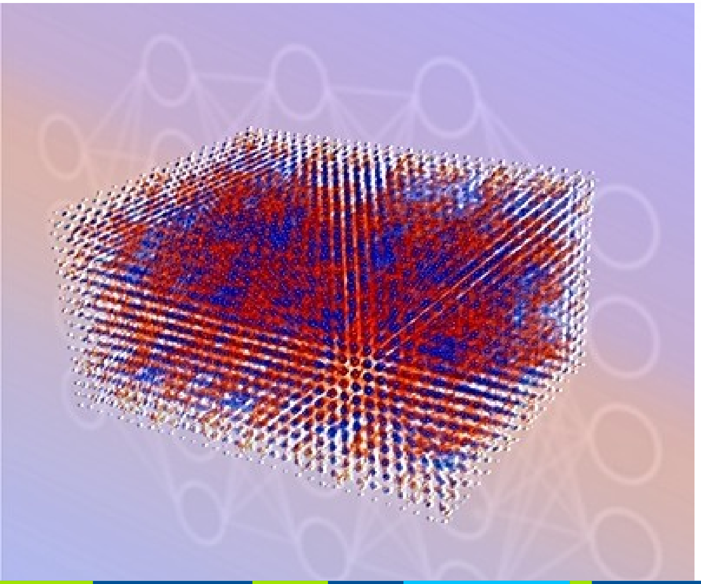
- **openPMD standard** (1.0.0, 1.0.1, 1.1.0)
  the underlying file markup and definition
  A Huebl et al., doi: 10.5281/zenodo.33624

Language Binding:

- **openPMD-api** (HZDR, CASUS, LBNL)
  reference API for openPMD data handling
  maintainers: A Huebl, J Gu, F Poeschel et al.

HiPACE++ → VisualPIC
Credit: M.Thévenet
& A. Ferran Pousa (DESY)





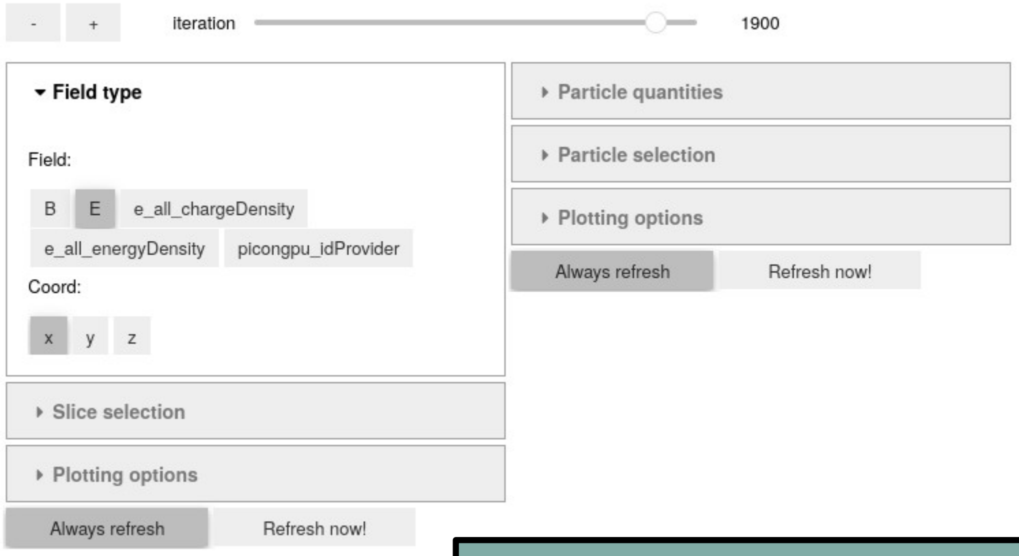
MALA → ParaView
Credit: A. Cangi (CASUS)

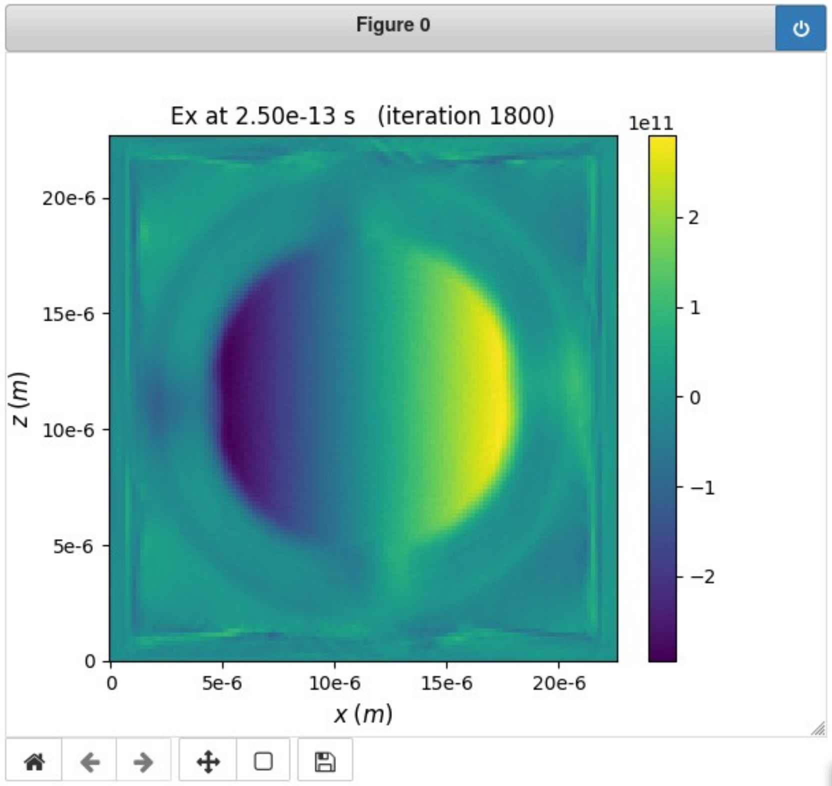
- **Wake-T** (DESY)
  fast particle-tracking code for plasma-based accelerators
  maintainer: A Ferran Pousa
- **HiPACE++** (DESY, LBNL)
  3D GPU-capable quasi-static PIC code for plasma accel.
  maintainers: M Thevenet, S Diederichs, A Huebl
- **Bmad** (Cornell)
  library for charged-particle dynamics simulations
  maintainers: D Sagan et al.
- **MALA** (CASUS, SNL)
  ML models that replace DFT calculations in materials science
  maintainers: Attila Cangi & Sivasankaran Rajamanickam
- and more…

see also: https://github.com/openPMD/openPMD-projects

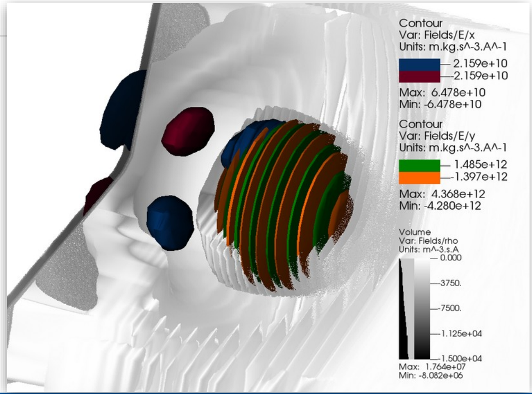
Analysis and Visualization

```
In [1]: import numpy as np
        %matplotlib notebook
        # or `%matplotlib inline` for non-interactive plots
        # or `%matplotlib widget` when using JupyterLab (github.com/matplotlib/jupyter-matplotlib)
        import matplotlib.pyplot as plt
        from openpmd_viewer import OpenPMDTimeSeries

In [2]: # Replace the string below, to point to your data
        ts = OpenPMDTimeSeries('/home/franzpoeschel/singularity_build/pic_run/openPMD')

In [3]: # Interactive GUI
        ts.slider()
```





openPMD/openPMD-viewer



Standardization of data

→ integration into modern scientific compute workflows

Hands-On:

openPMD-viewer:

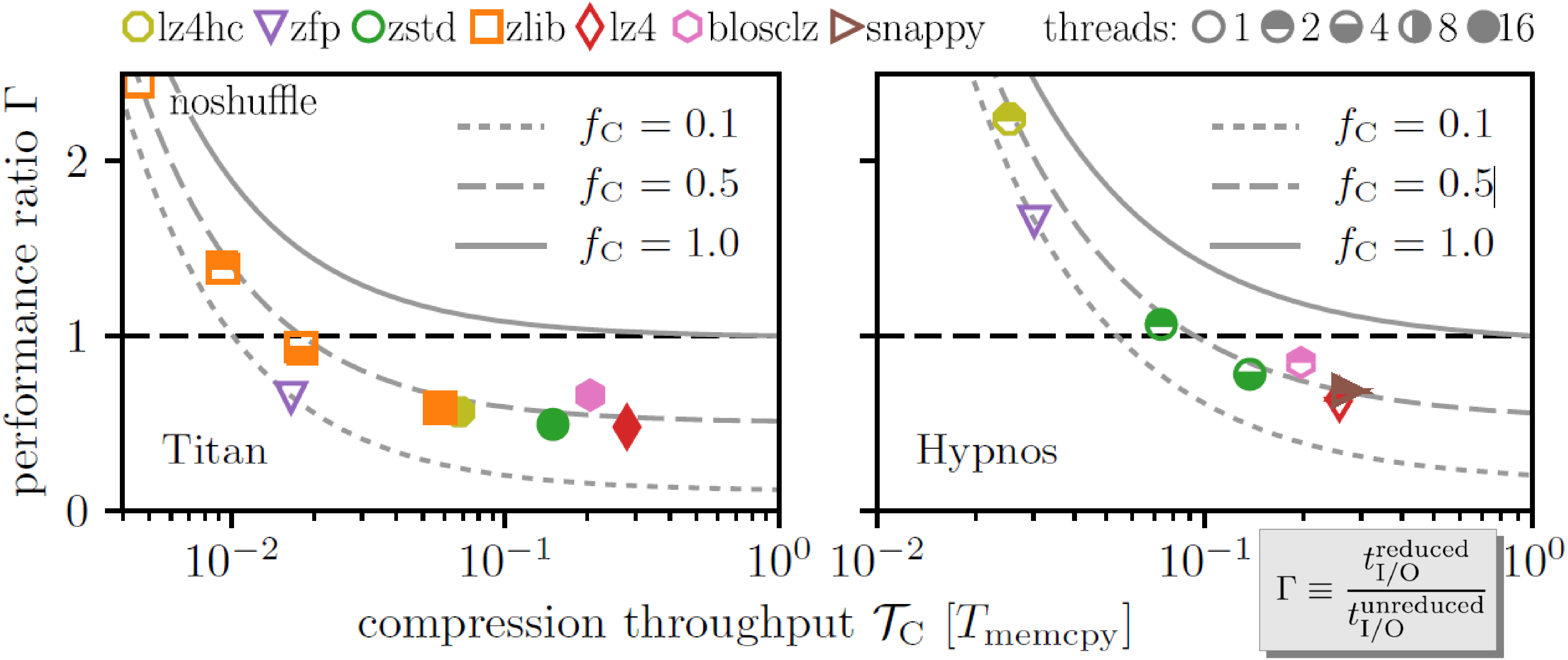
visualizing data written by a PIConGPU LaserWakefield simulation

Module environment at `/project/project_465001310/workshop_software/env.sh`
Materials at `https://github.com/alpaka-group/alpaka-workshop-slides/tree/oct2024_workshop`
Launch Notebook at `https://www.lumi.csc.fi/`

Read the instructions inside `next_steps.md`:

Data-driven Science

Just compress the data and everything will be fine ...?



A. Huebl et al., "On the Scalability of Data Reduction Techniques in Current and Upcoming HPC Systems from an Application Perspective", In: Lect. Notes Comput. Sci. 10524.4, pp.15-20 (2017)

Hands-On:

compression

Module environment at `/project/project_465001310/workshop_software/env.sh`
Materials at `https://github.com/alpaka-group/alpaka-workshop-slides/tree/oct2024_workshop`
Launch Notebook at `https://www.lumi.csc.fi/`

Continue with the instructions in the openPMD-viewer Notebook

TBD: Data workflow in a realistic application

CASUS
CENTER FOR ADVANCED
SYSTEMS UNDERSTANDING

CASUS
CENTER FOR ADVANCED
SYSTEMS UNDERSTANDING

Advanced Hands-On:

openPMD-api:
Span API

Module environment at `/project/project_465001310/workshop_software/env.sh`
Materials at `https://github.com/alpaka-group/alpaka-workshop-slides/tree/oct2024_workshop`
Launch Notebook at `https://www.lumi.csc.fi/`

Read the instructions inside `src/openPMDOutput.hpp`

TBD: Recap constant components

Advanced Hands-On:

openPMD-api:

Python API: constant components

Module environment at `/project/project_465001310/workshop_software/env.sh`
Materials at `https://github.com/alpaka-group/alpaka-workshop-slides/tree/oct2024_workshop`
Launch Notebook at `https://www.lumi.csc.fi/`

Read the TODO comments inside `write_parallel.py`

TBD: Parallel I/O, parallel file systems

Advanced Hands-On:

openPMD-api:

Python API: Parallel Writing

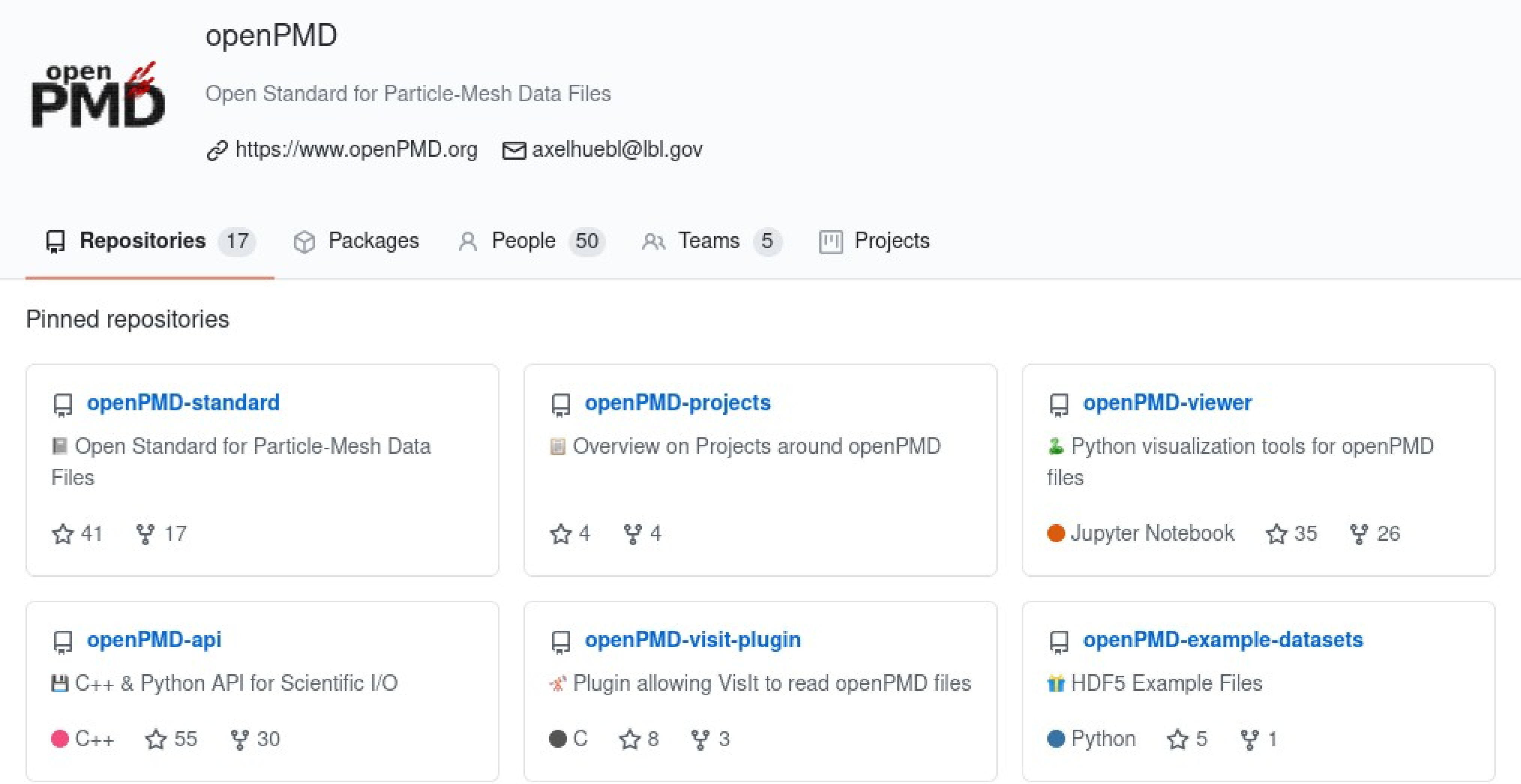
Module environment at `/project/project_465001310/workshop_software/env.sh`
Materials at `https://github.com/alpaka-group/alpaka-workshop-slides/tree/oct2024_workshop`
Launch Notebook at `https://www.lumi.csc.fi/`

Read the TODO comments inside `write_parallel.py`

Where to find us

→ head to https://github.com/openPMD/

CASUS
CENTER FOR ADVANCED
SYSTEMS UNDERSTANDING


openPMD

Open Standard for Particle-Mesh Data Files

https://www.openPMD.org axelhuebl@lbl.gov

Repositories 17 | Packages | People 50 | Teams 5 | Projects

Pinned repositories

openPMD-standard
Open Standard for Particle-Mesh Data Files
41 17

openPMD-projects
Overview on Projects around openPMD
4 4

openPMD-viewer
Python visualization tools for openPMD files
Jupyter Notebook 35 26

openPMD-api
C++ & Python API for Scientific I/O
C++ 55 30

openPMD-visit-plugin
Plugin allowing VisIt to read openPMD files
C 8 3

openPMD-example-datasets
HDF5 Example Files
Python 5 1

…and of course https://openpmd-api.readthedocs.io/

Community www.openPMD.org

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!