

A. Adelman, A. Gsell, B. Oswald, T. Schietinger, PSI, Villigen, Switzerland  
W. Bethel, J.M. Shalf, C. Siegerist, K. Stockinger, LBNL/NERSC, Berkeley, CA, USA

## Abstract

Significant problems facing all experimental and computational sciences arise from growing data size and complexity. Common to all these problems is the need to perform efficient data I/O on diverse computer architectures. In our scientific application, the largest parallel particle simulations generate vast quantities of six-dimensional data. Such a simulation run produces data for an aggregate data size up to several TB per run. Motivated by the need to address data I/O and access challenges, we have implemented H5Part, an open source data I/O API that simplifies the use of the Hierarchical Data Format v5 library (HDF5). HDF5 is an industry standard for high performance, cross-platform data storage and retrieval that runs on all contemporary architectures from large parallel supercomputers to laptops. H5Part, which is oriented to the needs of the particle physics and cosmology communities, provides support for parallel storage and retrieval of particles, structured and in the future unstructured meshes. In this poster, we describe recent work focusing on I/O support for particles and structured meshes and provide data showing performance on modern supercomputer architectures like the IBM POWER 5.

Modern large-scale parallel simulations produce data volumes that are on the orders of TBs. One of the main challenges is how to access this data efficiently and how to share it among scientists of specific communities. In order to address these problems we have developed H5Part, a high-performance data API that is particularly targeted for the accelerator modeling community.

H5Part uses HDF5 as the underlying storage format which has the following benefits:

- 1) **Machine independence:** No byte-swapping is necessary for accessing binary data created on different machines.
- 2) **Language independence:** Data can, for instance, be written using the Fortran API and read using the C/C++ API.
- 3) **Self-describing:** Data is accessed by names rather than position. For instance, read the values of the dataset  $p_x$ .
- 4) **High-performance:** Data is stored in native binary format and is only automatically translated if the machine that reads the data requires a different format.
- 5) **Parallel I/O:** Data is written in parallel into a single file using MPI-I/O.

## Data models

H5Part supports two types of data models. One data model is used for storing particle data as 1-dimensional datasets. The other model is used for storing field data as 3-dimensional datasets (block structured data). Both data models support parallel reading and writing of the respective data.

### H5Part: Particle Data

The data model for particle data allows storing multiple timesteps where each timestep can contain several datasets of the same length. By definition, each timestep must have the same number of datasets. Typical particle data consists of the 3-dimensional Cartesian positions of particles ( $x, y, z$ ) as well as the corresponding 3-dimensional momenta ( $p_x, p_y, p_z$ ). These 6 variables are stored as 6 HDF5 datasets. The type of the dataset can be either integer or real. H5Part also allows storing attribute information for the file and the time-steps.

```
if (not parallel);
  filehandle=OpenFile(filename,mode)
else
  filehandle=OpenFile(filename,mode,mpicomm)
SetNumberOfParticles(filehandle);
loop(step=1,NSteps);
  (compute data)
  SetStep(filehandle,step)
  WriteData(filehandle,fieldname1,data1)
  ...
  WriteData(filehandle,fieldname<n>,data<n>)
CloseFile(filehandle);
```

Simplified pseudo code for storing particle data with  $n$  timesteps. Note that if a file is opened in parallel, the data is evenly partitioned and written in parallel based on the number of particles.

### H5Block: Block-Structured Data

H5Block inherits all particle features of H5Part but also supports 3D scalar fields and vector fields. Examples of these two types of data are the *potential* and *electrical field* of a particle simulation. Note that in H5Part, the 1D dataset is evenly partitioned among the parallel processors based on the number of particles. Since H5Block stores 3D datasets, the data partition strategy among the processors must be specified explicitly.

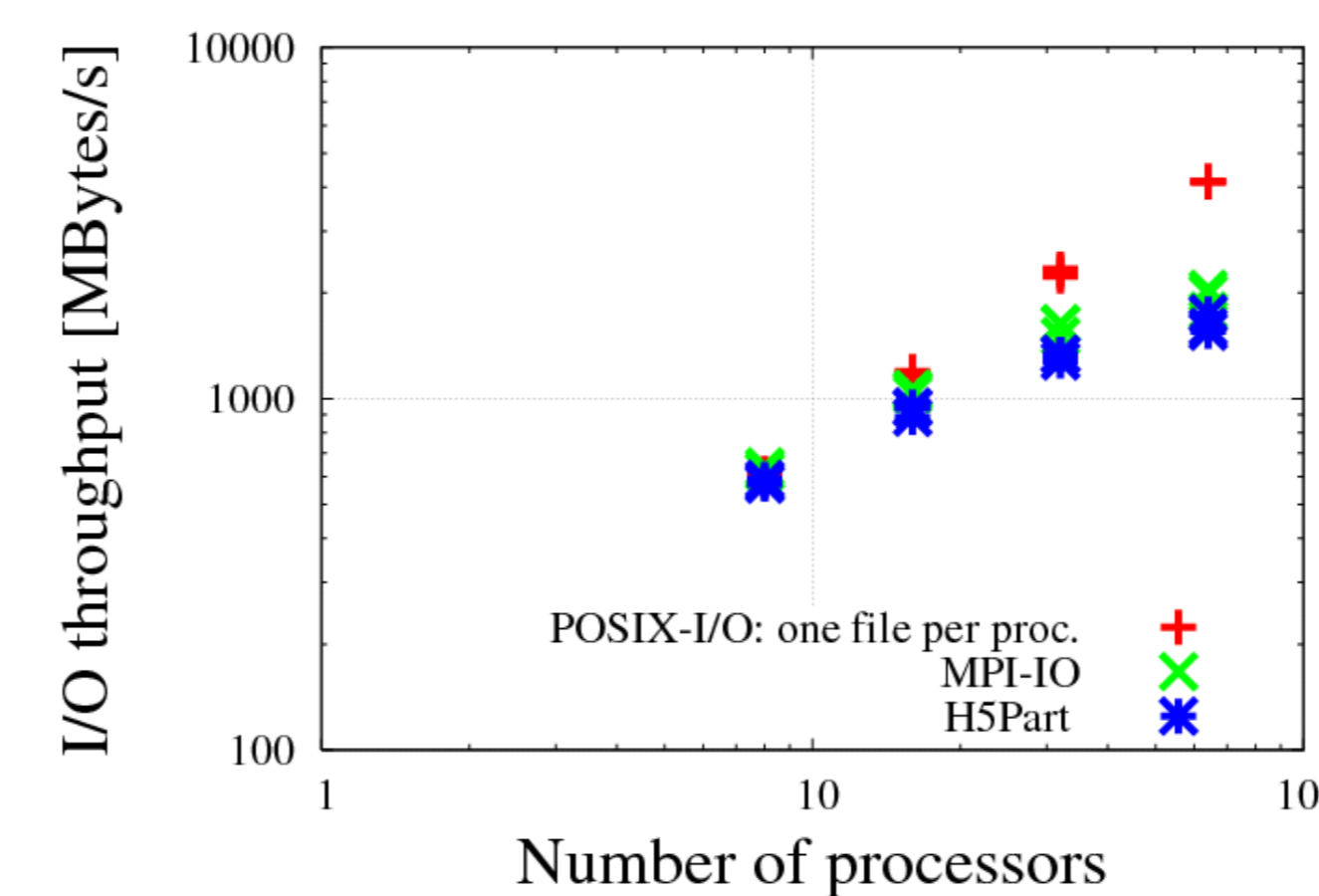
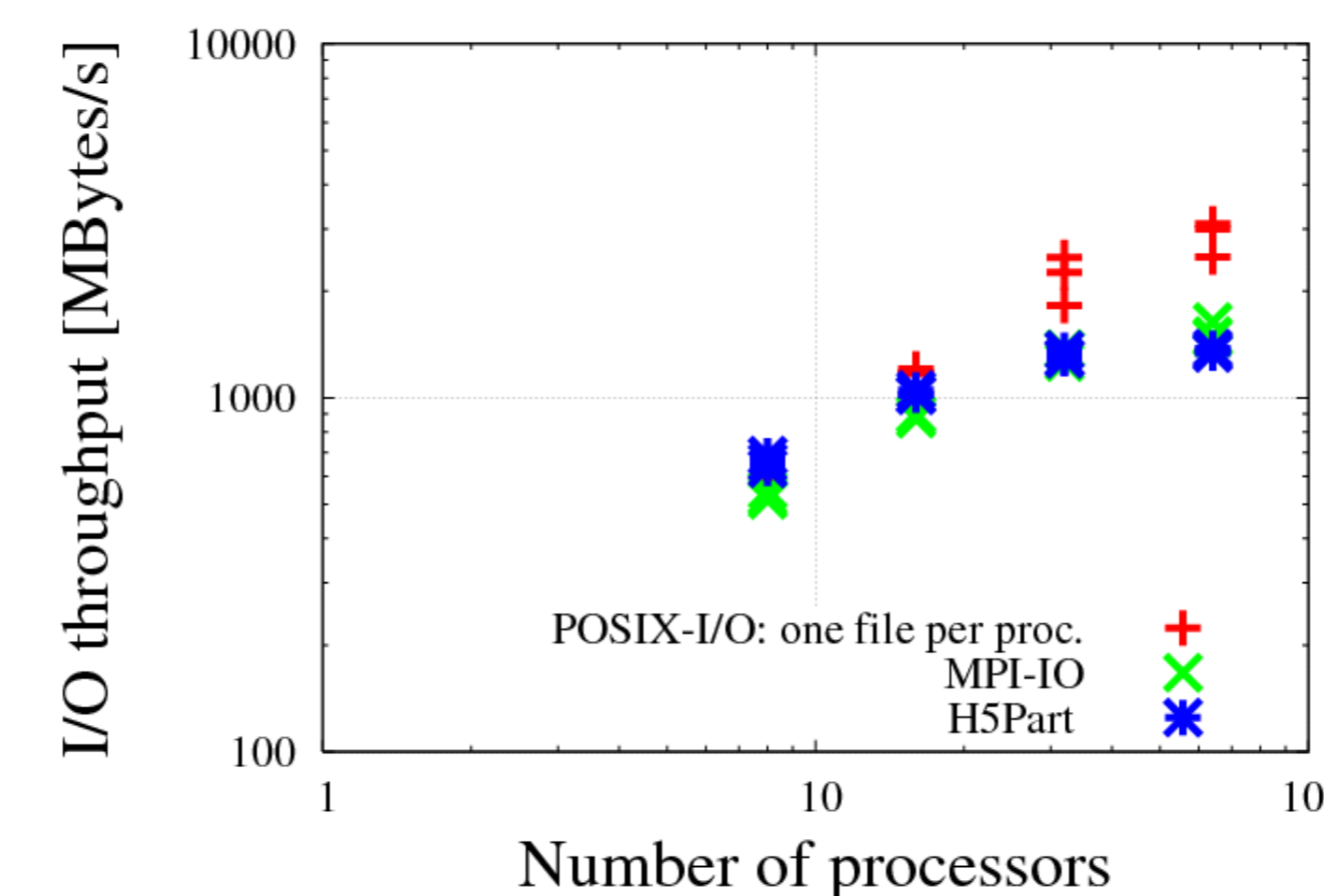
```
fh=OpenFile(filename,mode)
if processor 0
  Define3DFieldLayout(fh,0,15,0,15,0,63);
else
  Define3DFieldLayout(fh,0,15,0,15,64,127);
Write3DScalarField(fh,fieldname1,data1);
CloseFile(fh);
```

Pseudo code for partitioning a 3D scalar field with dimensionality 16, 16, 128 among two processors.

## Performance

Comparison between H5Part and MPI-I/O:

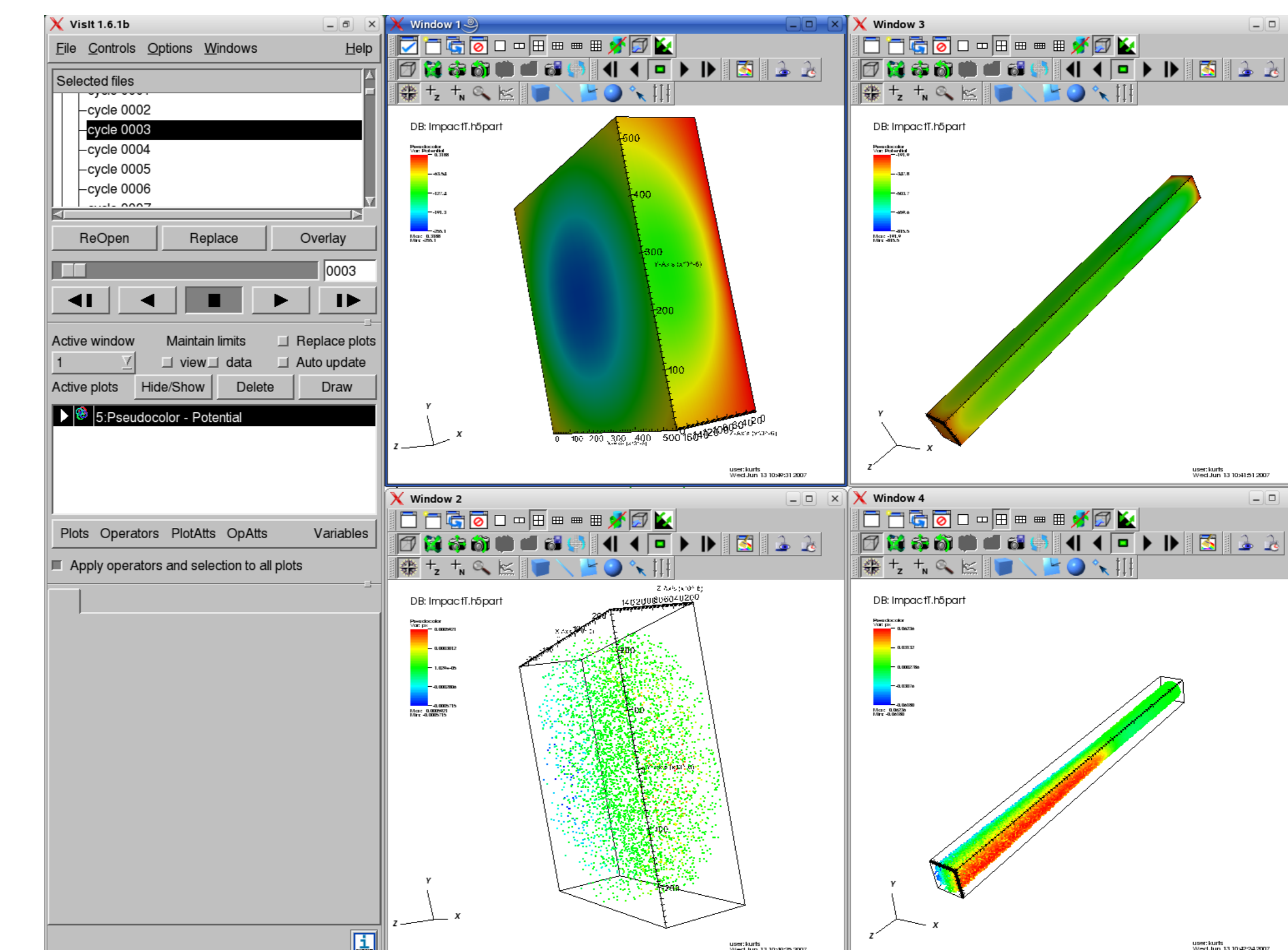
- IBM p575 POWER 5 system (up to 8 nodes);
- 8-64 CPUs;
- GPFS filesystem;
- non-collective I/O (faster than collective I/O in H5Part);
- $10^8$  particles with 6 attributes for  $\rightarrow$  5 timesteps (constant)
- $\rightarrow N_{CPU}/2$  timesteps ( $N_{CPU} = 8,16,32,64$ )



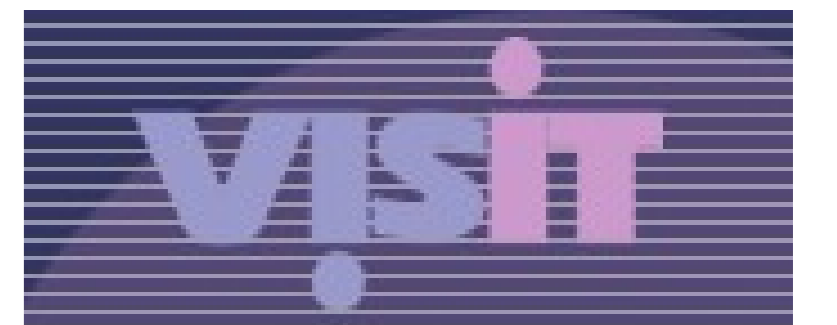
Performance of H5Part compared with POSIX and MPI-I/O, where the total size of the written data is constant (top plot) or increases with the number of processors (bottom plot).

## Visualization

Visualization matters! Data that cannot be looked at in an acceptable time frame are essentially lost.



Example: visualization of the scalar field "potential" (left frames) and the particle data " $p_x$ " (right frames) for two timesteps.



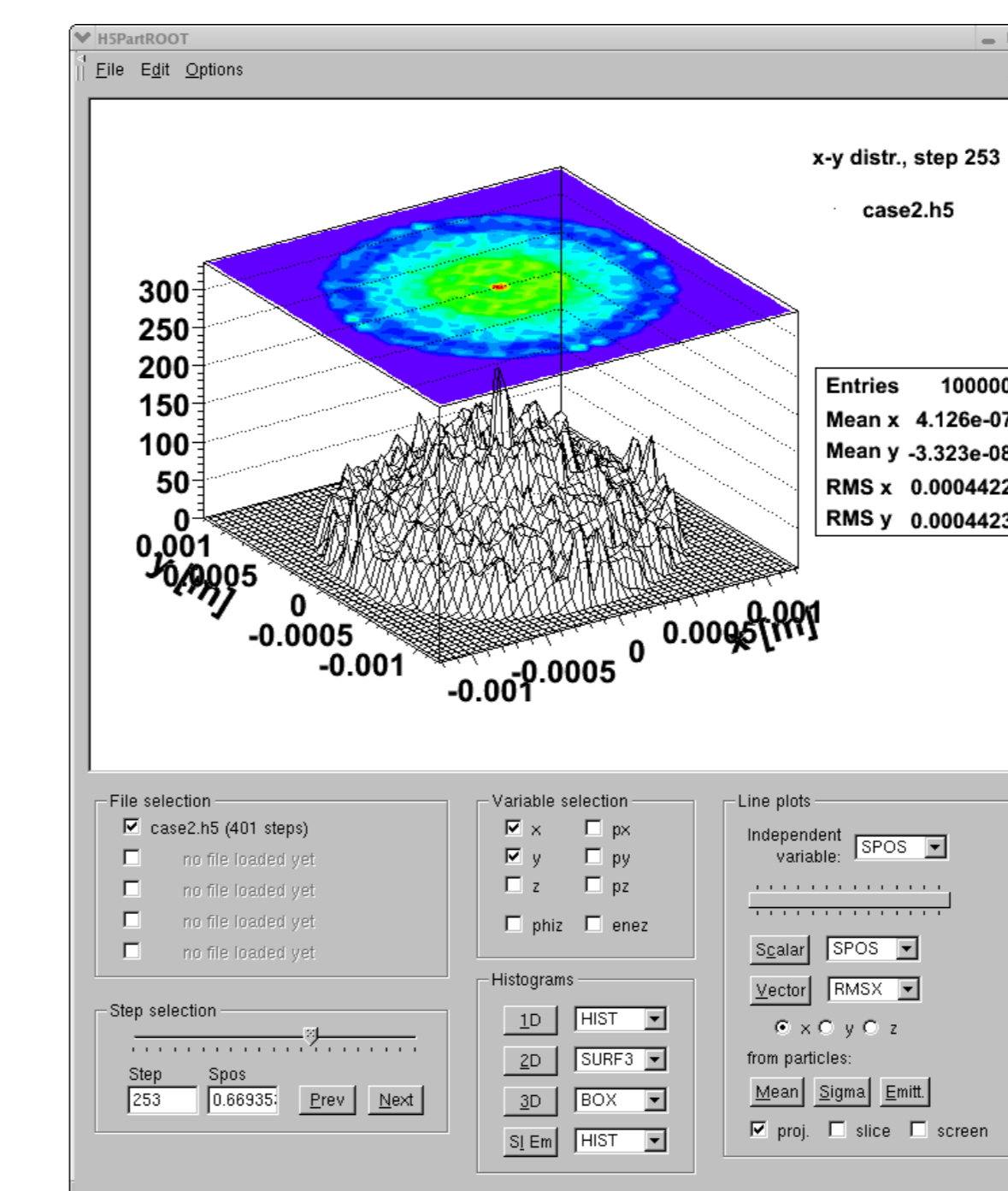
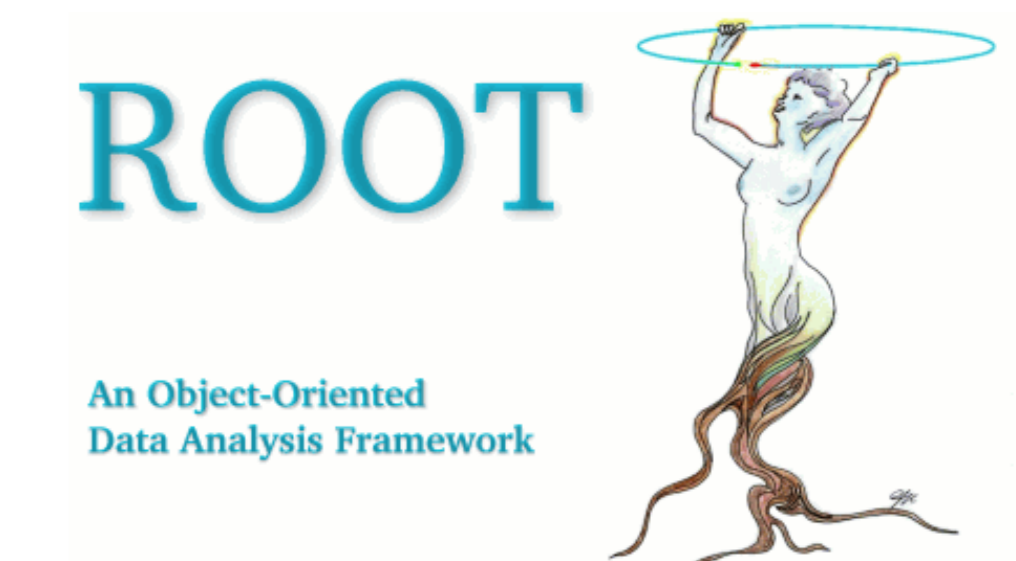
## VisIt

H5Part plugins available for reading and visualizing both particle and field data.

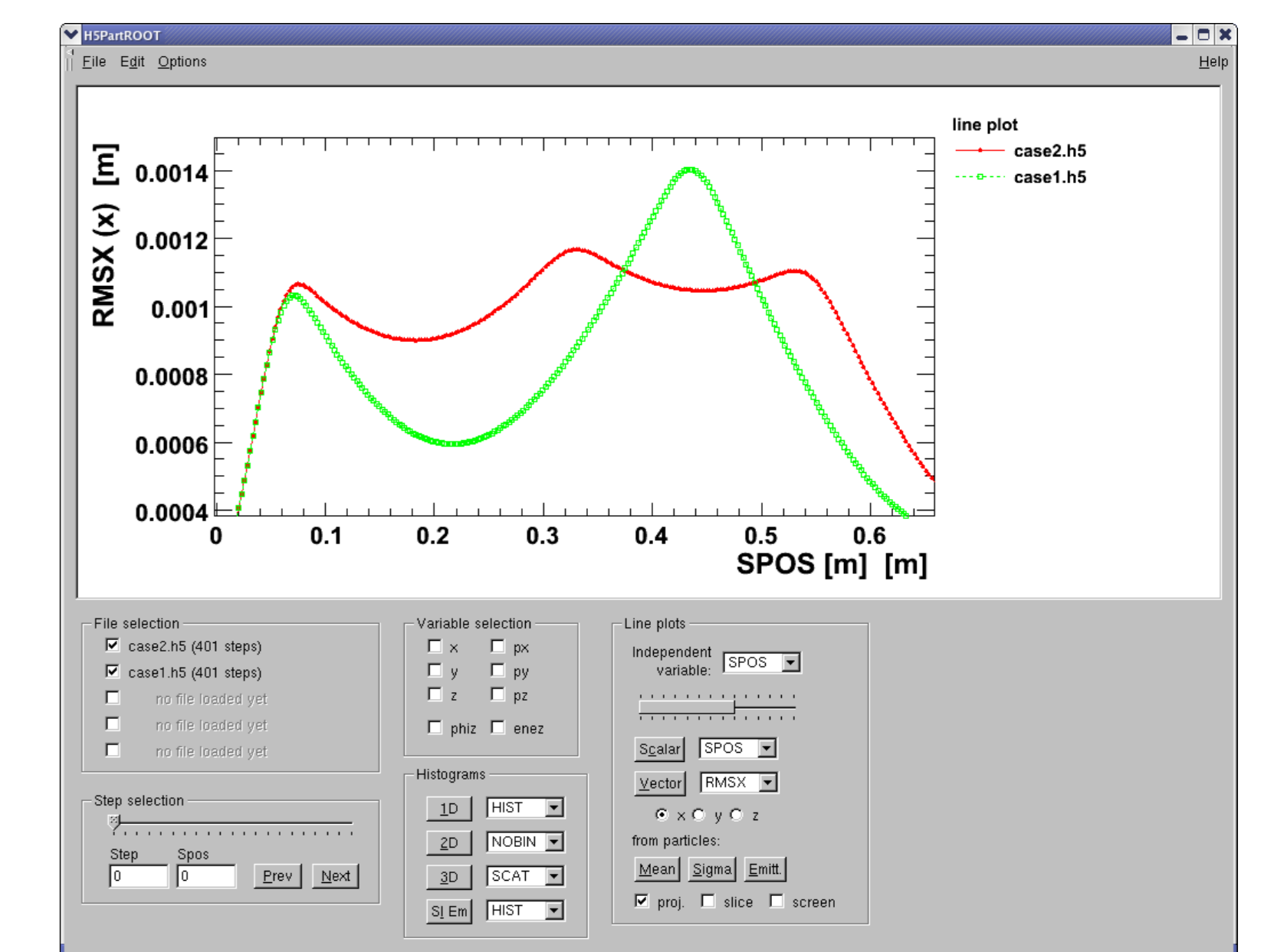
## H5PartROOT

Custom tool based on the ROOT object-oriented analysis framework. Two modes of operation:

- GUI for quick visual access to data
- shared library to be loaded in be used in interactive or batch ROOT session (macros).



Example 1 : x-y distribution of a bunch.



Example 2 : transverse beamsizes comparison between two setups.