PandAna

A Python Analysis Framework for Scalable High Performance Computing in High Energy Physics

Micah Groh

On behalf of Norm Buchanan, Derek Doyle, Jim Kowalkowski, Marc Paterno, Saba Sehrish

Needs and Goals

Experiments in HEP need to analyze millions of events from detector data and reconstruction algorithms.

Analysis frameworks provide tools for selecting signal events and extracting physics parameters from the data.

Next-generation experiments are approaching exabyte-scale dataset sizes requiring use of HPC facilities to process efficiently.

Goals of PandAna:

1. Utilize modern data science tools for analyzing tabular data.

HDF5, MPI, Pandas, Numpy

2. Provide implicit parallelism.

3. Scale from laptops to super computers.

4. Reduce time-to-physics.

PandAna

PandAna is an a python framework for analyzing tabular data, the “tidy data” model.

Experiment data is stored in flattened tables in HDF5 files.

Each table has a few index columns and any number of data columns.

Parent Particle Table

Event	Energy	X	Y	...
1	0.4	123	465	...
2	1.1	215	315	...
3	1.5	102	211	...
4	2.1	375	104	...
5	0.8	283	345	...
...				

Daughter Particle Table

Event	PartId	Energy	Nhit	...
1	0	0.1	30	...
1	1	0.3	45	...
3	0	0.3	12	...
3	1	1.0	87	...
3	2	0.2	25	...
...				

Traditional ROOT-based analyses would read each row, one at a time.

In PandAna, data is read from tables into pandas DataFrames to be analyzed in a columnar analysis.

Data Parallelism

Data parallelism is implicit to the PandAna framework.

When launching a process using *mpirun* each MPI rank sees disjoint data to be read from a given file.

Analyses can run in parallel up to the number of events being analyzed.

Event	Energy	X	Y	...
1	0.4	123	465	...
2	1.1	215	315	...
3	1.5	102	211	...
4	2.1	375	104	...
5	0.8	283	345	...
		...		

Rank 1

Rank 2

Result from each rank can be combine in an MPI reduction.

Usually summing histogram bin contents.

Var

Two main PandAna tools: Variables and Cuts.

Var represents a computed quantity of interest.

When a table is accessed a pandas DataFrame is initialized.

Access the ‘parent’ table…

And the ‘energy’ column.

```
def kParentEnergy(tables):
    EnergyDF = tables['parent']['energy']
    return EnergyDF
kParentEnergy = Var(kParentEnergy)
```

Return type is a DataFrame.

The function is stored within the class to be evaluated later.

PandAna will only read in the tables and columns used in the analysis.

The result of a *Var* (or *Cut*) is cached to be used multiple times within the analysis.

Cut

A *Cut* represents a selection criteria on the rows in a table.

Result is a single boolean per row in a table.

The ‘daughter’ table has two index columns (*event* and *daughter* index).

```
def kDaughtersContained(tables):
    PosDF = tables['daughter'][['x', 'y']]
    MaxPosDF = PosDF.abs().max(axis=1)
    ContainedDF = MaxPosDF < 500
    AllContainedDF = ContainedDF.groupby('Event').all()
    return AllContainedDF
kDaughtersContained = Cut(kDaughtersContained)
```

Use vectored functions within pandas and numpy to manipulate the DataFrame.

Group the DataFrame to yield one result per parent.

Complete script

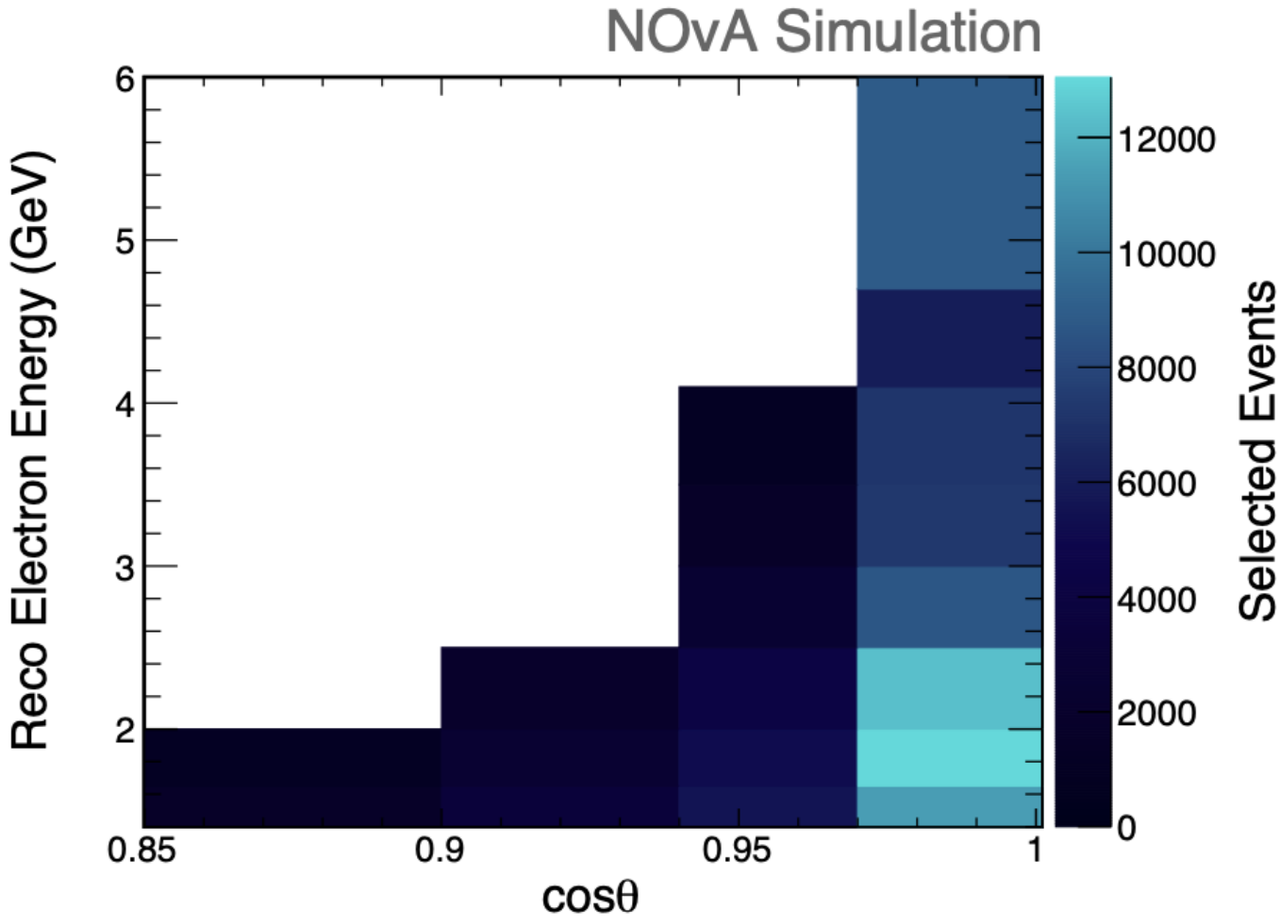
```
from pandana import *
from mycode import kParentEnergy, kDaughtersContained

tables = Loader('myfile.h5')

spec = Spectrum(tables, kDaughtersContained, kParentEnergy)

tables.Go()

n, edges = spec.histogram(bins=50, range=(0, 10), mpireduce=True)

if MPI.COMM_WORLD.rank == 0:
    print('Selected ', n.sum(), ' events')
```

Loader handles the construction of the tables from each input file.

After the call to *Go*, *Spectrum* contains a DataFrame of selected events which can be turned into a histogram or further analyzed.

No reference to file I/O or MPI within user code.

NOvA Example



10 Million ν interactions

Machine	Ranks	t (s)
NOvA VM	1	756
My Laptop	1	700
My Laptop	4	225
Fermi WC	1	658
Fermi WC	16	60
Fermi WC	64	20
Fermi WC	128	11

NOvA electron neutrino CC selection on a small sample.

Same code ran in all cases regardless of MPI ranks or location.

No dependence on compiled experiment code or libraries.

Did not use containers (docker, etc) - only python packages installable with pip.

Future Work

Future I/O improvements:

- Understanding HDF5 parameters.
- File concatenation (1 big file).

Want to move from data parallelism to task parallelism.

- Divide the needed computations amongst all MPI ranks.

Construction, optimization, and evaluation of an abstract syntax tree where each node represents once computation in the analysis.

Data Parallelism

Event	Energy	X	Y	
1	0.4	123	465	***Rank 1***
2	1.1	215	315	
3	1.5	102	211	
4	2.1	375	104	***Rank 2***
5	0.8	283	345	
	...			

Task Parallelism

Event	Energy	X	Y
1	0.4	123	465
2	1.1	215	315
3	1.5	102	211
4	2.1	375	104
5	0.8	283	345
	...		
	Rank 1	***Rank 2***	

Conclusion

PandAna is a python analysis framework for analyzing tabular HEP experiment data.

PandAna uses modern data science tools to build a fast and intuitive framework where users use efficient vector operations to manipulate arrays and tables.

Analysis is natively data parallel and scales from laptops to HPC systems with no needed code change or experience with parallel programming.

Code available on GitHub here - plan to release to PyPi soon.

Future improvements include optimized I/O, a better parallelism, and the use of an AST to accelerate analysis code.