

# Distributed RDataFrame:

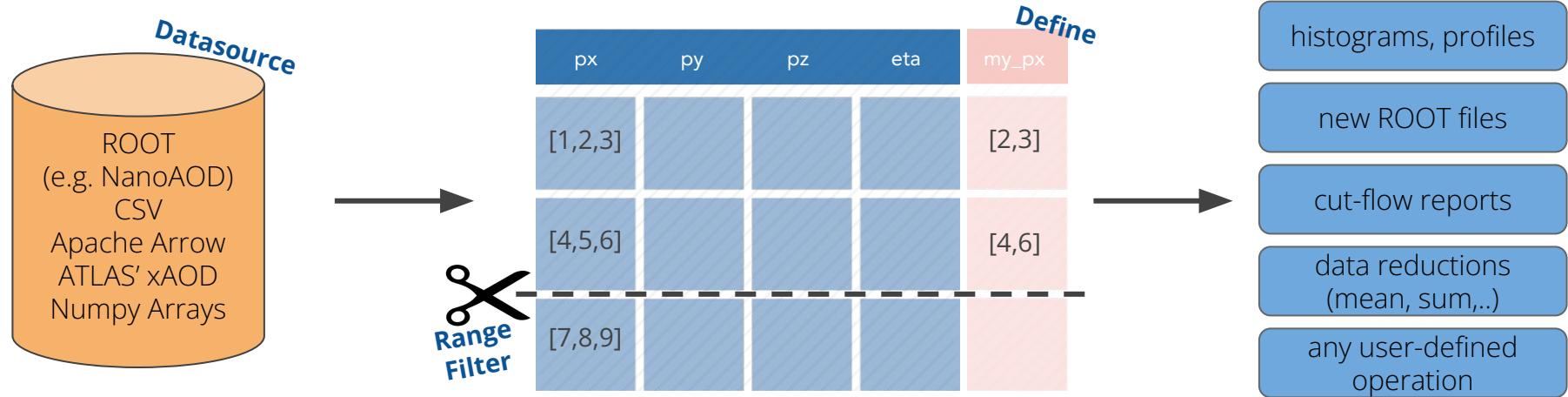
supported backends, latest improvements and future  
plans

ROOT  
Workshop  
2022

ROOT  
Data Analysis Framework  
<https://root.cern>



# RDataFrame analysis interface



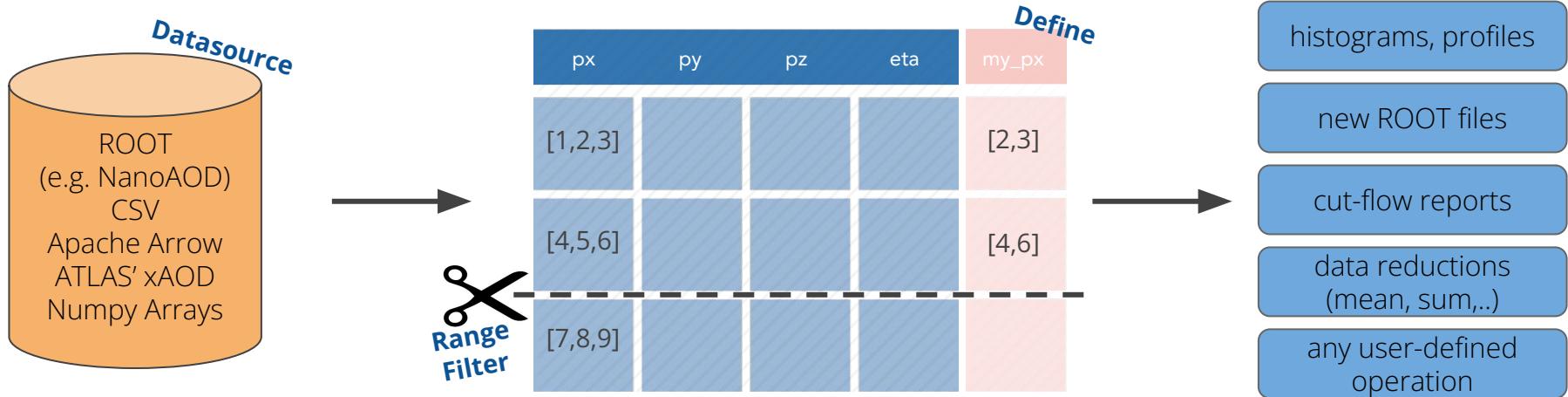
```
# enable multi-threading
ROOT.EnableImplicitMT()
df = ROOT.RDataFrame(dataset)
```

```
df = df.Range(2)
      .Define("my_px", "px[eta > 0]")
```

```
# filled in a single pass
h1 = df.Histo1D("my_px", "w")
h2 = df.Histo1D("px", "w")
```



# RDataFrame analysis interface



## Goal

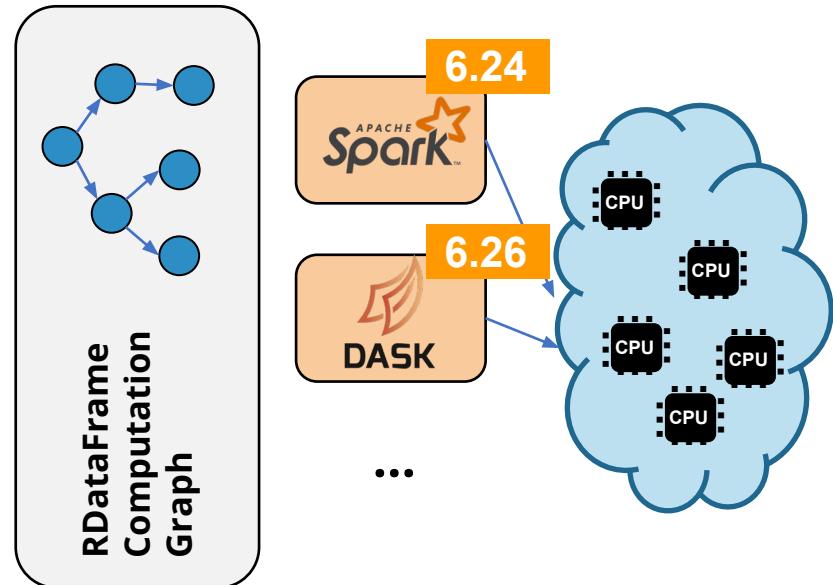
Best performance and ease of use across the board,  
for most (all?) HEP analysis use cases

[RDataFrame reference guide](#)

[RDataFrame tutorials](#)

# Going Distributed

- Enable **interactive large-scale distributed** data analysis with RDataFrame
- Can run with different schedulers
  - **Spark**
  - **Dask**
  - ...
- Analysis from start to end in a **single interface**





# Programming Model

## Local

```
from ROOT import RDataFrame
```

Importing RDataFrame

```
df = RDataFrame('treename', 'filename.root')
```

Constructing RDataFrame

```
df2 = df.Filter(...).Define(...)  
h1 = df2.Histo1D(...)  
h1.Draw()
```

Rest of application

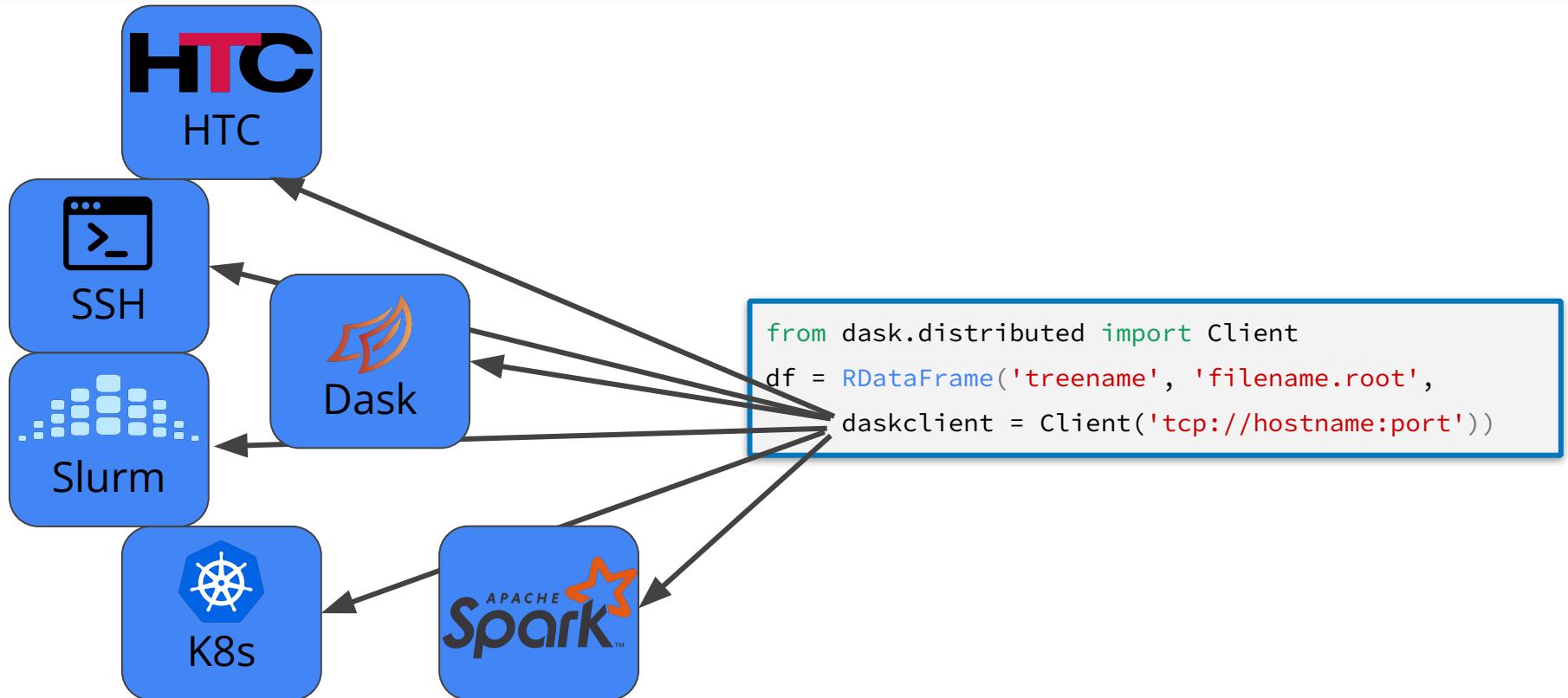
## Distributed

```
import ROOT  
RDataFrame = \  
ROOT.RDF.Experimental.Distributed.Dask.RDataFrame
```

```
from dask.distributed import Client  
df = RDataFrame('treename', 'filename.root',  
daskclient = Client('tcp://hostname:port'))
```

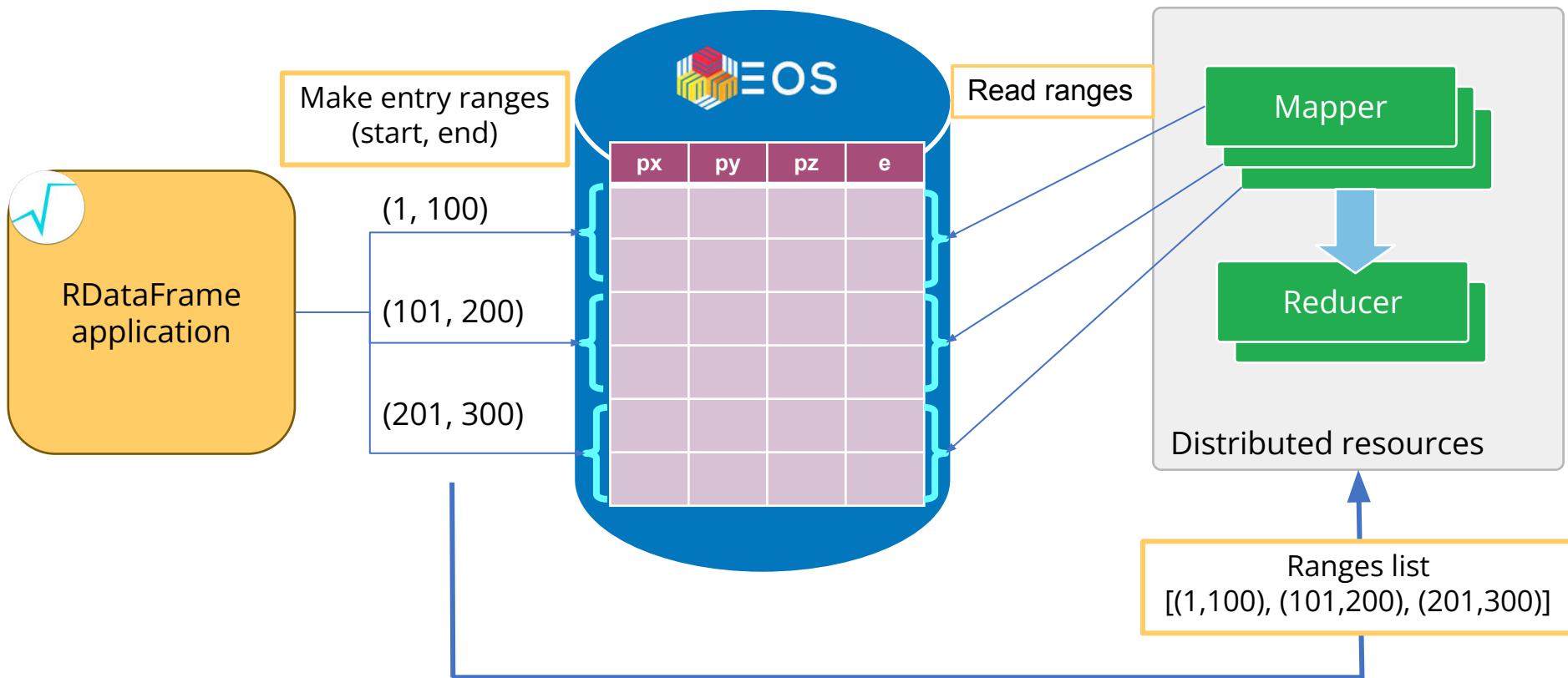


# Programming Model



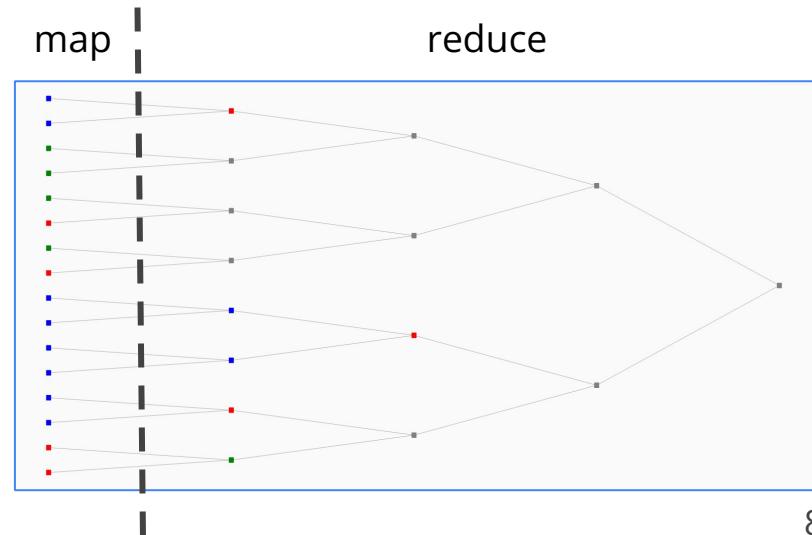
[[maybe\_unused]]

# Behind distributed RDataFrame



Dask vs Spark for distributed RDataFrame:

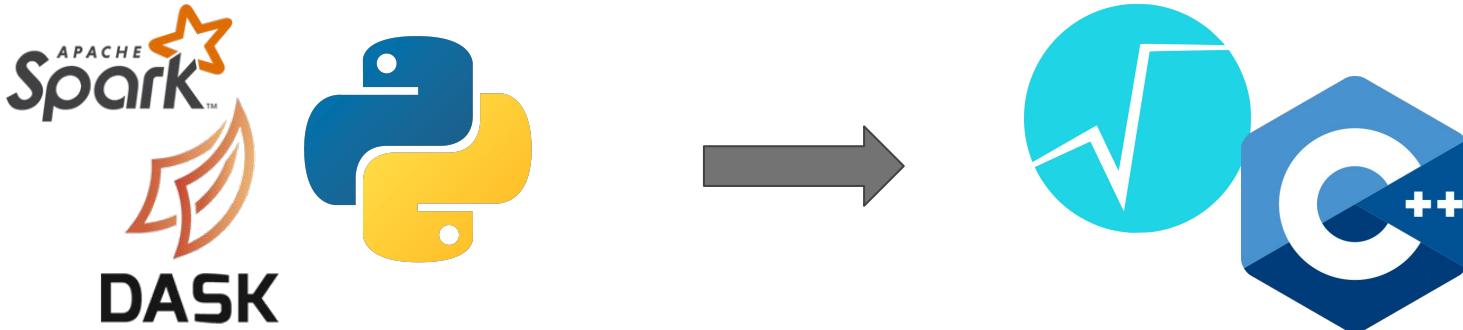
- ▶ dask.delayed vs spark.map & spark.treeReduce
- ▶ Same MapReduce tree pattern, different API
- ▶ Spark processes in stages
  - first map, then reduce
- ▶ Dask can reduce two completed while others are still





# Distributed C++

- ▶ Python API, C++ event loop
- ▶ Leveraging full power of ROOT I/O
- ▶ Let Spark/Dask/HTCondor/Slurm.... take care of scheduling and resource management





# Supported API

Subset of RDF methods are supported in distributed mode:

- AsNumpy
- Count
- Define
- DefinePerSample
- Fill
- Filter
- Graph
- Histo[1,2,3]D
- HistoND
- Max
- Mean
- Min
- Profile[1,2,3]D
- Redefine
- Snapshot
- Sum

## ROOT.RDF.Experimental.Distributed.initialize

- ▶ Define your own function
- ▶ Runs once at the beginning of each distributed task
- ▶ For example nice if you want to declare C++ code quickly

```
import ROOT
def f():
    ROOT.gInterpreter.Declare("""
        #ifndef MYFUN
        #define MYFUN
        int myfun(){ return 42; }
        #endif
    """)
```

Using header guards avoids  
redefinitions in the workers

```
ROOT.RDF.Experimental.Distributed.initialize(f)
```

Improved interface for user code distribution:

- ▶ Distribute C++ headers
- ▶ Distribute shared libraries
- ▶ Distribute code snippets (i.e. a “distributed”  
gInterpreter.Declare)

```
ROOT...distribute_code("")  
int myfun(){ return 42; }  
")")
```

```
ROOT...distribute_headers("myheader.hpp",  
restrict_to_df=mydf)
```

## Other improvements:

- ▶ Automatically pick a good number of partitions for a certain computation graph
- ▶ Aggregate logs from workers



# Coming soon

## Injecting Python functions in distributed RDF via Numba

```
import ROOT

def f(x: int) -> bool:
    return x > 10

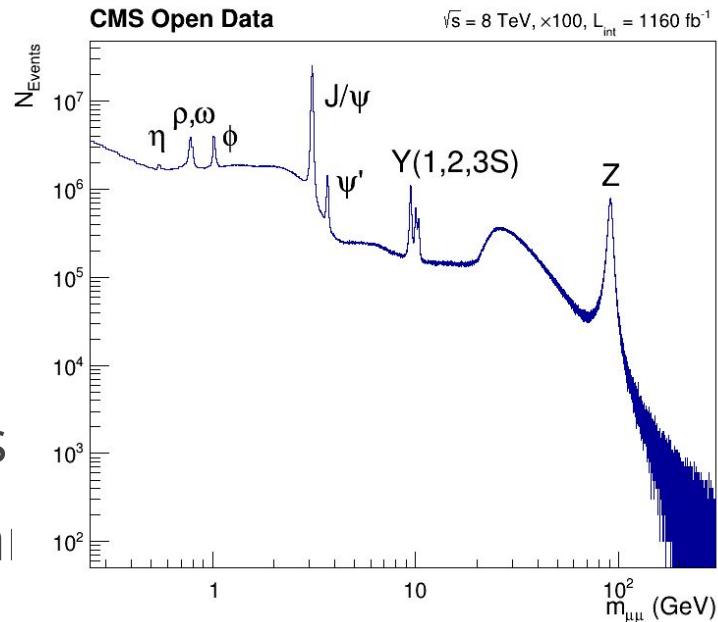
RDF = ROOT.RDF.Experimental.Distributed.Dask.RDataFrame
df = RDataFrame('treename', 'filename.root', daskclient = Client('tcp://hostname:port'))

df.Filter(f, ["mycol"])
```



# Test setup: dataset

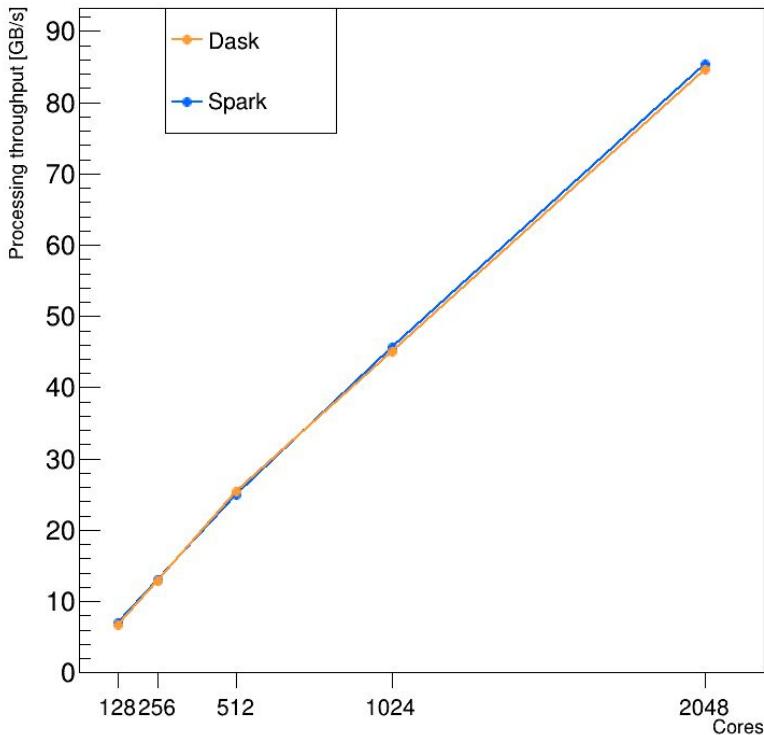
- ▶ Dimuon analysis
- ▶ 4000x original dataset
  - **8800 GB**
  - ZLIB compressed
- ▶ Data is **node-local** during analysis
- ▶ **100%** read and processed in the al



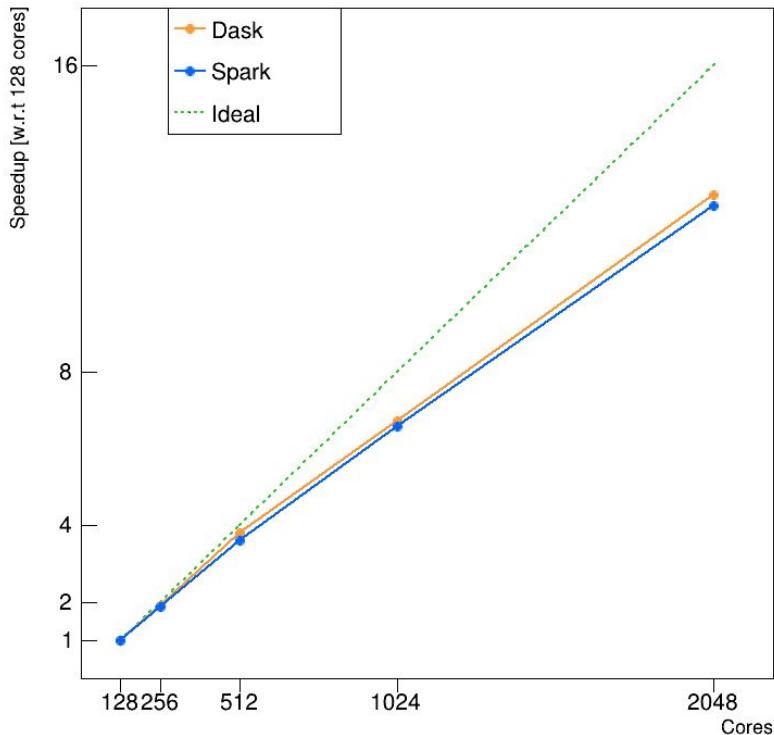


# Spark vs Dask comparison

processing throughput



speedup



- ▶ **Distributed RDataFrame** is here
  - Requires minimal changes w.r.t. local application
  - Task scheduling with Spark or Dask (for now!)
- ▶ Run interactively in a Python notebook or in a batch system with a Python script
- ▶ **Coming soon:**
- ▶ Still experimental and in development, **your feedback is welcome!**