Fermilab U.S. DEPARTMENT OF ENERGY Office of Science

Data-parallel analysis supported by HDF5

Marc Paterno

HDF BOF @SC19

Fermilab

"Fermilab is America's particle physics and accelerator laboratory.

We bring the world together to solve the mysteries of matter, energy, space and time."

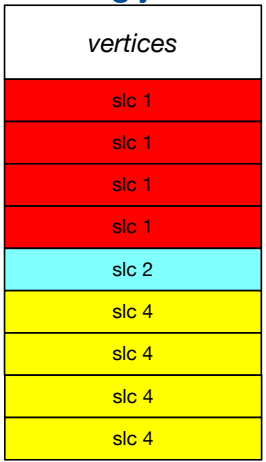
How we are currently using HDF5

- Most of HEP does not have a long history with HDF5.
  - The accelerator modeling community has been using more than 10 years (since v 1.6).
- We use it to store *tabular* data (HEP calls them *ntuples*)
  - for interactive analysis by scientists
  - for input to machine learning algorithms
- We are exploring its use for moderately large data sets
  - current analysis target is about 2 TB.
- HEP scientists program in (1) C++ and (2) Python
- One of our projects is concentrating on analysis using *pandas*, employing HDF5 to support easy-to-use parallel reading of datasets.

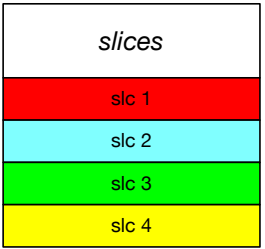
What are data storage looks like

- Tabular data
  - Use *group* to represent a table; read it into a *DataFrame*.
  - Use *dataset* to represent a column.
- Read table into a *pandas* DataFrame.
- Why not use support of HDF5 from *pandas*?
  - Our data sometimes contains multi-dimenional arrays, which don't seem well-supported by *pandas*.
  - Storing data using simple HDF5 concepts allow us to read the data with other tools as well.
- A complication: files have hundreds of tables, which do not all align. We need the equivalent of *joins* between tables — but joins are expensive.

Avoiding joins



Group verices by slice, to make first selection



Use vertex summary with slice information, for second level of selection.

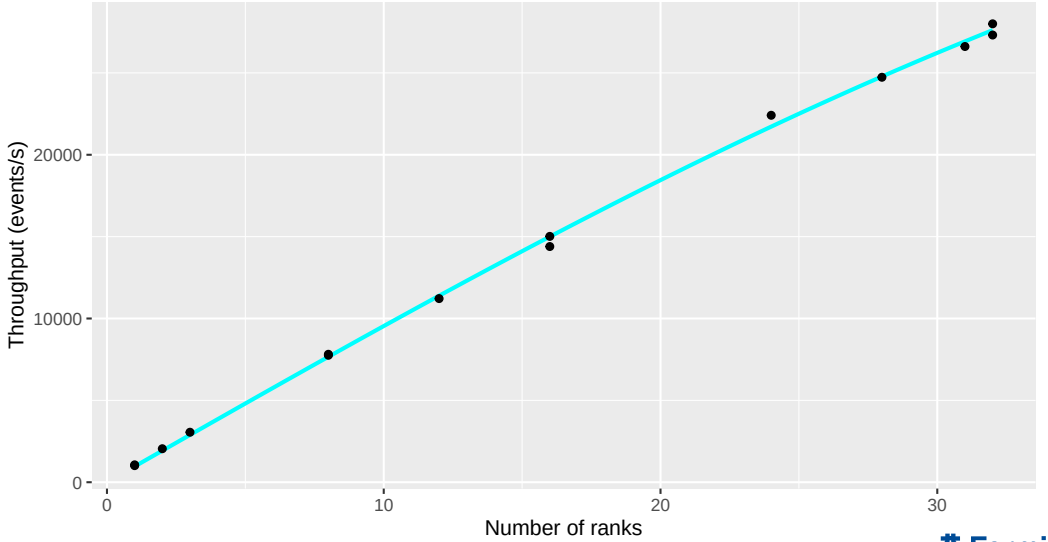
Fermilab

What analysis code looks like

- Analysis programs are MPI-based
- Physicists see *almost* no MPI code — they write as if for a serial process
  - Sometimes need a trivial reduction, adding results from the many ranks
- Framework core code handles data parallelism

```
def kNueCVNCut(tables): # input is a dict of DataFrames
    df = kCVNe(tables)  # apply a different named selection first
    dfRHC = df[kRHC(tables)==1] >= kNueCVNRHC # RHC-class data...
    dfFHC = df[kRHC(tables)!=1] >= kNueCVNFHC # FHC-class data...
    return pd.concat([dfRHC, dfFHC])          # Our result is the sum
```

Some performance measurements (32 core AMD K10 (Barcelona) server)



Fermilab

What could make this easier

- We have to keep *index columns* (datasets) in each table (group) to let us align the data.
- We keep another index column to let us calculate spans of slices to be processed by each MPI rank.
- A clever indexing scheme would make this task easier. . .
- Efficient *sparse array* storage would make this trivial.