



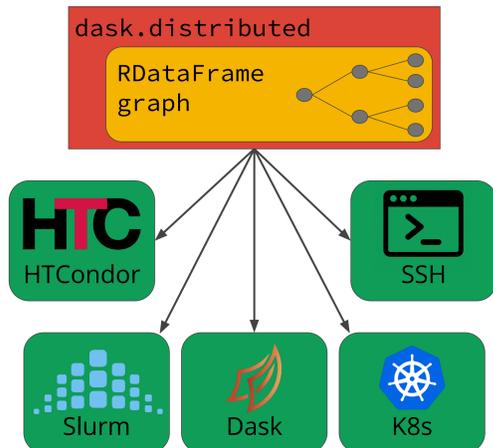
# Distributed RDataFrame: Dask on HPC, caching, optimization

V. E. Padulano, I. Kabadzhov, E. Guiraud, E. Tejedor  
EP-SFT CERN

<https://root.cern>

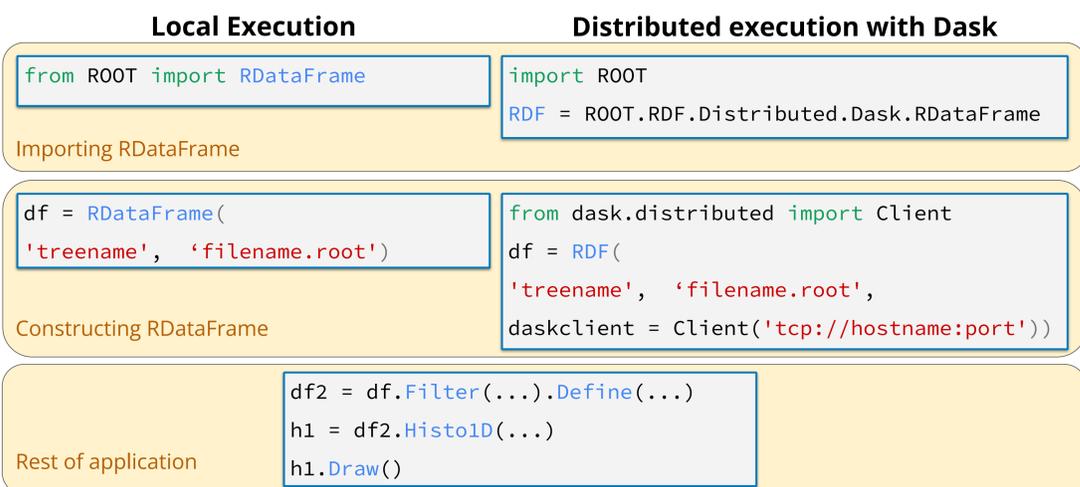
## Dask for HEP distributed computing

- **New** backend for **Distributed RDataFrame**
- Access to **multiple resource managers** (e.g. HTCondor, Slurm) through Dask
- Leverages `dask.delayed` interface to build **map-reduce tasks** and send them to the executors



### Programming model

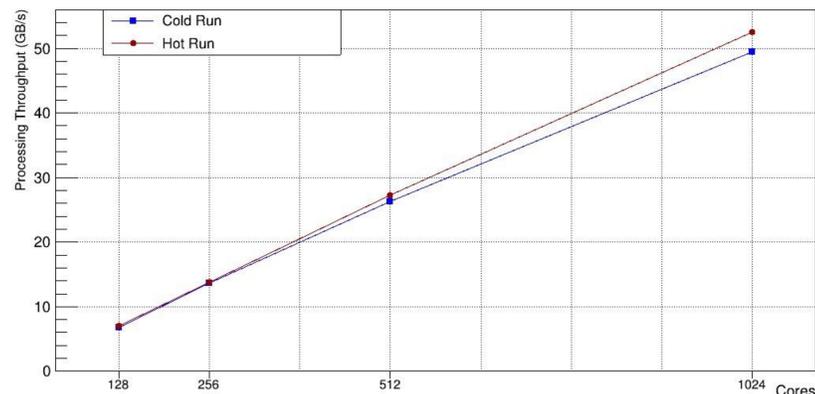
- **Minimal changes** when moving from local to distributed
  - Configuration of connection to scheduler
  - Main workflow remains the same



## Exploiting an HPC cluster

**9 TB processed**  
in **3 min** on **1024 cores (32 nodes)**

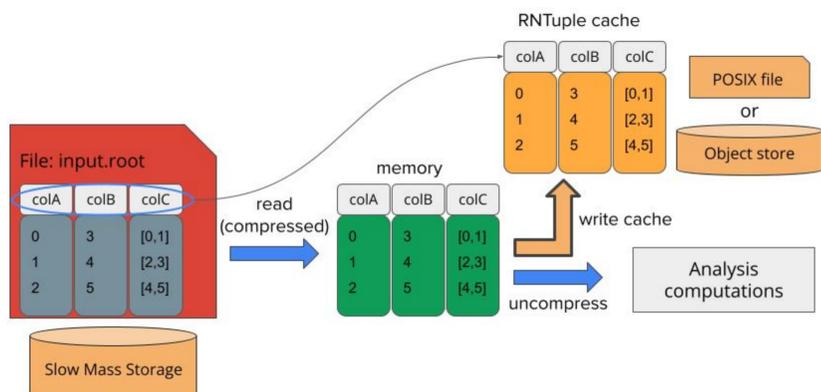
Over **52 GB/s**  
of peak throughput



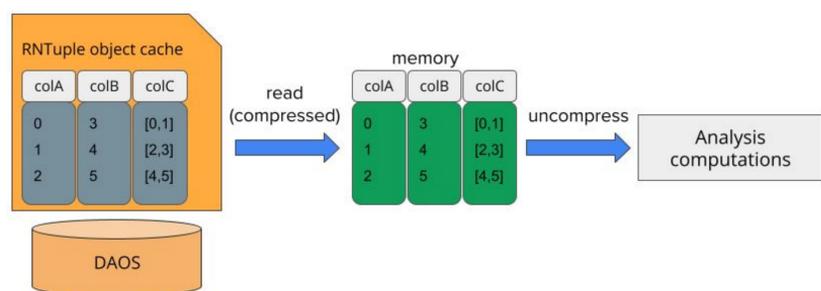
- HPC cluster tests (32 cores per node)
  - **OpenData NanoAOD** analysis with distributed RDataFrame
  - Running Dask computations as **Slurm** jobs
- First run (cold) pays some initialization cost, subsequent runs (hot) achieve even higher throughput

## Transparent caching on object stores

- R&D on caching input data with RNTuple
- Caching **independent** of the final **storage system**
- Read a remote POSIX file and cache it to a fast **object store** (e.g. **Intel DAOS**)
- Caching overhead compensated by fast reading from cache
- **High throughput** achieved with DAOS: 24 GB/s with 48 cores



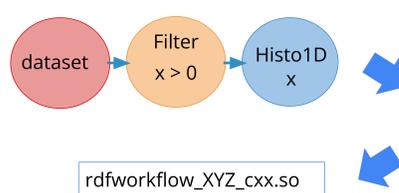
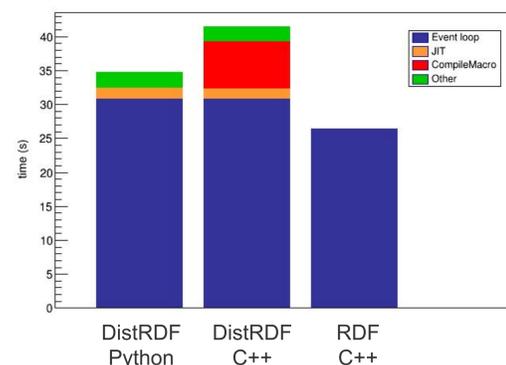
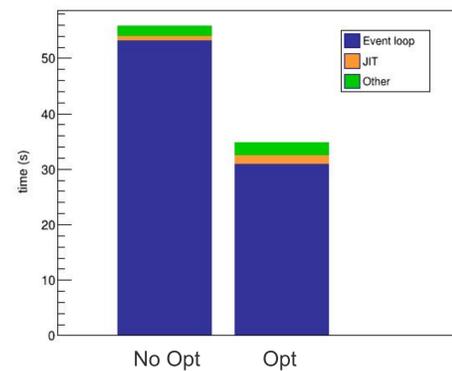
(a): creation of RNTuple cache object



(b): reading from the cached RNTuple

## Task execution optimization

- Distributed RDataFrame tasks rely heavily on just-in-time compiled code
- **Enabling cling optimizations** largely improves task execution time in most cases
  - At the expense of some increase in jitting time
- **Generating a C++ workflow** in tasks and statically compile it does not represent a big performance boost w.r.t. optimized jitted code
  - And it adds non-negligible compilation overhead



```
Result GenerateWorkflow(RNode &headnode) {
  auto df1 = headnode.Filter("x > 0");
  auto res1 = df1.Histo1D("x");
  ...
}
```