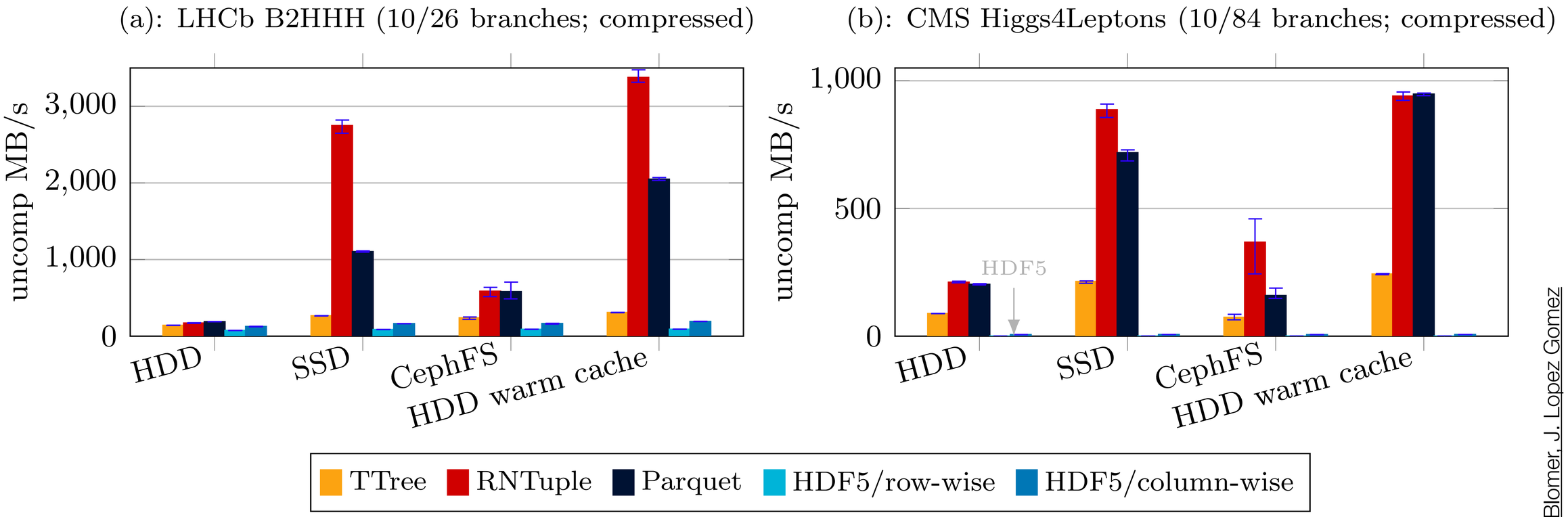
ROOT @ SWIFT-HEP #4

Axel Naumann axel@cern.ch for the ROOT Team
2022-11-01

ROOT Rejuvenation

- Investing in major upgrades of core components, gradual roll-out
  - I/O: RNTuple **faster** + **smaller** than anything else for HEP data; **reliable**; **sustainable** for the next 20+ years
  - RDataFrame: analysis graph abstraction hiding I/O and scheduling. **Python** first, **event**-based logic, **performance** optimized
  - SOFIE, RooFit
- Others sequenced afterwards: small team, focus on topics with major impact, few at a time

RNTuple



J. Blomer, J. Lopez Gomez

RNTuple

- **Higher throughput**: reduced code + features; async, prefetching, parallelism
- **Reliable**: explicit error handling
- **Robust**: clear interfaces
- **Compact**: state-of-the-art layout for HEP data; smart compression
- **Sustainable**: documentation, using today's best practices

RDataFrame

- RDataFrame: express analysis simply, run it efficiently

- Separates internal data flow (I/O, bulk processing, scheduling) from analysis definition

- RDataFrame as generator!

Python

```
nominal_hx = (
   df.Vary("pt", "RVecD{pt*0.9, pt*1.1}",  ["down", "up"])
      .Filter("pt > k")
      .Define("x", someFunc, ["pt"])
      .Histo1D("x")
)
hx = ROOT.RDF.VariationsFor(nominal_hx)
hx["nominal"].Draw()
hx["pt:down"].Draw("SAME")
```

proceed as usual, as if working with nominal values only

obtain all variations

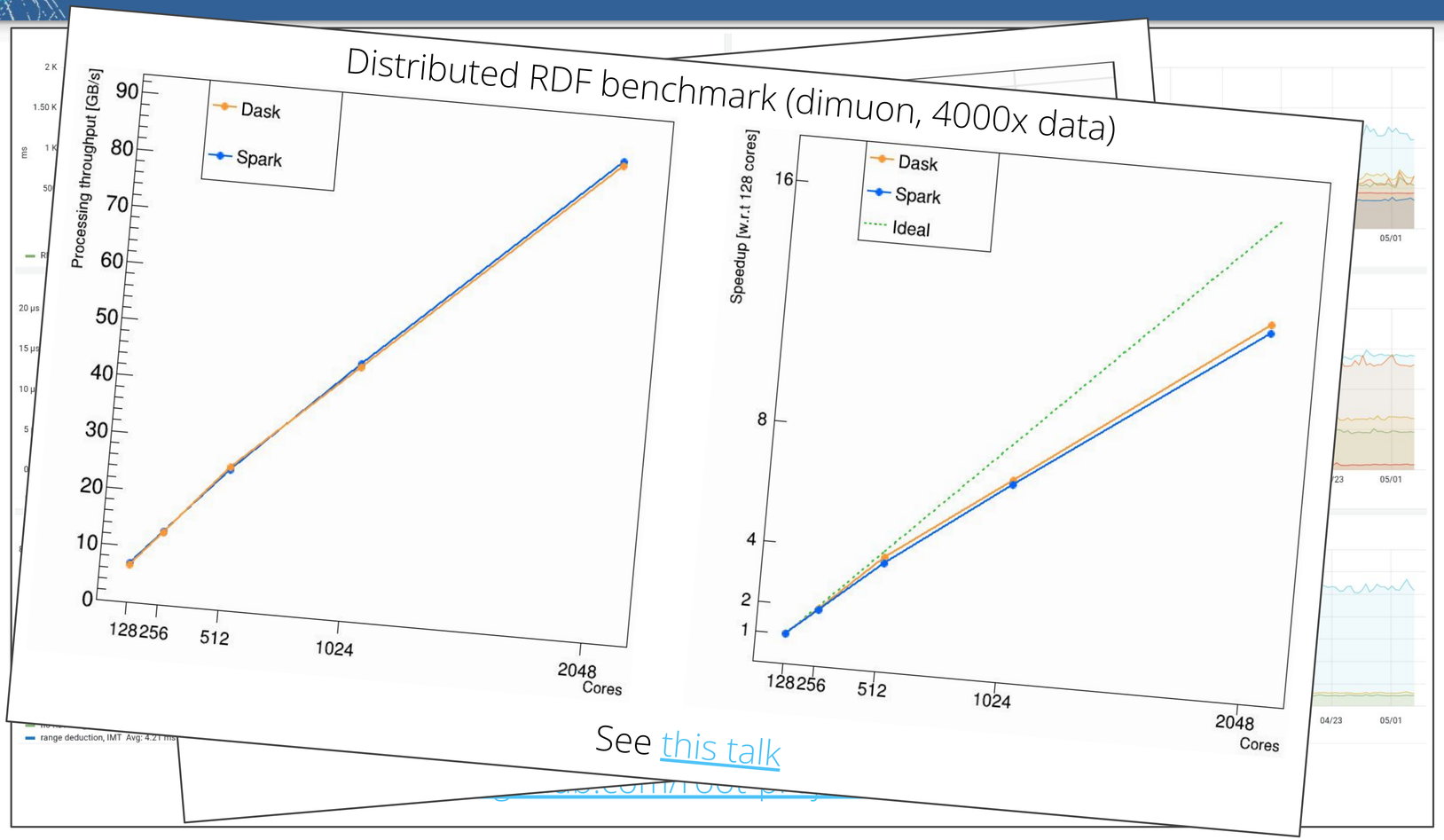
Material by V Padulano

Distributed RDataFrame

- Fairly independent from scheduler: Dask, Spark,...
- Workflow: can schedule multiple analysis graphs in parallel, `RDataFrame.RunGraphs`
- "Manual" checkpointing through `RDataFrame.Snapshot`
  - Challenge: want "sticky tasks", re-scheduled where checkpoint is stored, versus worker / filesystem lifetime. Dask R&D!
- Scales without changes to analysis code

Distributed RDataFrame

We care about performance. A lot.



See this talk

Material by V Padulano

Distributed RDF vs Workflow

- Files? Weights? Sample name? and whatever else you want to provide
  - Vision: `RDatasetSpec`
  - Let's create a standard for JSON metadata!

- Users report 0 failure rate in practice:
  - Robust, low memory consumption, no "random crashes"

Summary: The New ROOT

Performance targets

1 PB of (compressed) data, of which 100 TB are actually read by the analysis.
We expect the analysis will be able to run in A. 10 minutes on a cluster of 64 nodes,
or B. 4 hours on a single beefy machine with 128 cores.

Throughput required: A. ~3 GB/s/node or B. ~100 MB/s/core for read+processing.

- need hardware setup that can sustain such throughput
- cannot afford reading more than what's strictly needed
- must make good use of the hierarchy of storage options
  - remote
  - large shared storage at the level of the computing facility (xcache, high-bandwidth object stores)
  - small user-level storage

59

E. Guiraud, *HEP analysis in the LHC Run 3 Era*, ACAT 2022

Material by E Guiraud

How?

- **Fast** C++ foundation: faster than alternatives. Not just "fast enough, today": think zero-waste, lunch vs all afternoon, also 10 years from now
- Operated through seamless Python interface: part of the **ecosystem**
- Event-based analysis logic: affordable thanks to efficiency.
  "**Associate** leading-pT muon to two leading jets"
- **Scaling** from TTree::Draw() replacement to flooding 1000-core facility, with the exact same analysis code
- **Support + sustainability**

Data Delivery

- Goal: delegate, with application knowledge
- Recommendation: xrootd + XCache + RNTuple
- Example: xrootd, object store [1,2]
- XCache as transparent, multi-tier cache [1]
  - Stores local checkpointing snapshots!
  - No synchronization points, keeps parallelization opportunities

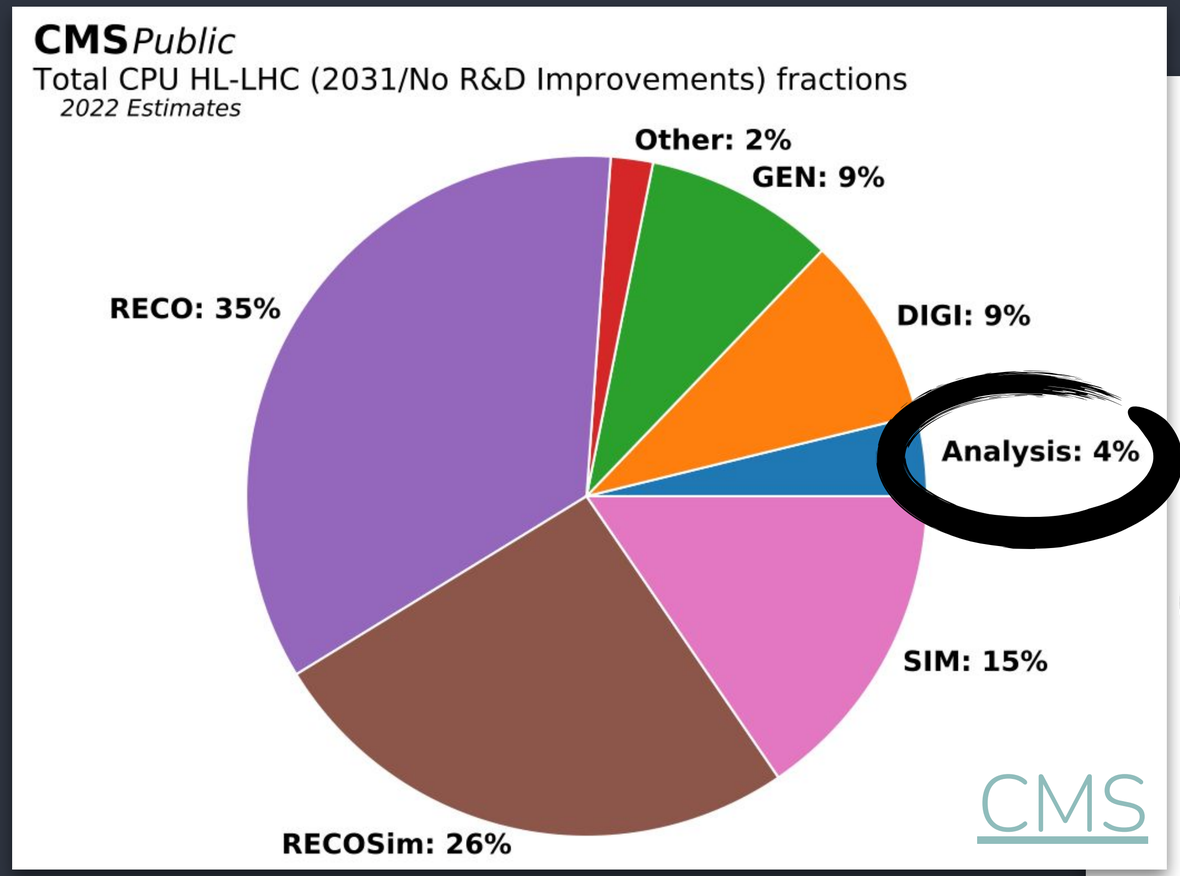
CC BY-SA 3.0/DE, Hans Adler

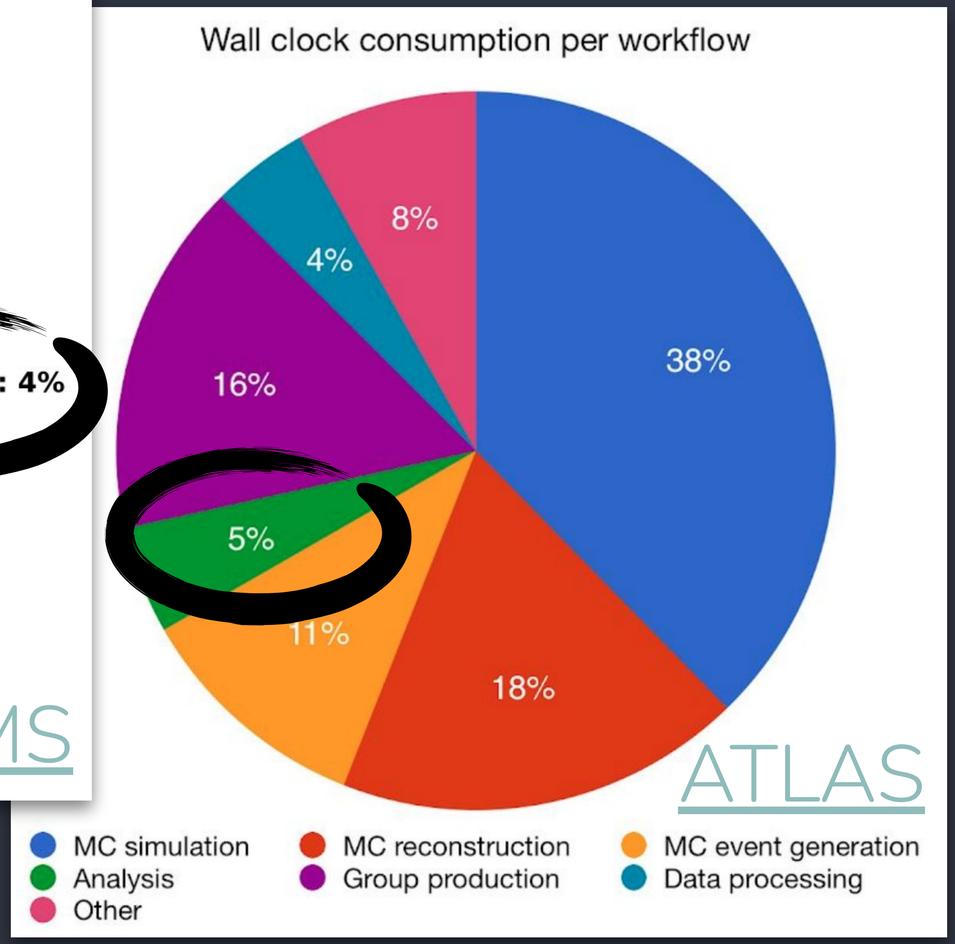
CC BY-SA 2.0, kees torn

The Perfect Analysis Facility

- Good analyses will saturate I/O
  - Fast network (async prefetch) and storage
- Parallelism needs 2GB/core
  - Analysis, scheduler, daemon, etc

Interactive vs Batch





Material by E Guiraud

Interactive vs Batch

- Symbiotic (parasitic?) with batch systems: zero-waste / highest efficiency

- R&D line: side-load to long-running jobs, reduced priority with cgroups
  - Scenario: ATLAS interactive analysis (I/O limited) next to ATLAS reco (CPU limited, known memory usage)

What brings me here?

- Please keep door open for physicists to choose
  - Let's test drive RDataFrame on your AF
  - Multiple solutions vs local optimum
- Unsolicited recommendations:
  - Be aware of overhead of layers that you buy in: benchmark with and without
  - Build a flexible setup: not everyone loves Jupyter; we don't know the future

Where to go from here?

- We have a story: fast, simple, robust computing
- Sizable R&D tasks: happy to collaborate, hand out responsibility - but let's coordinate for max impact and sustainability!
- Gradual transition from R&D to roll-out
  - Early adopters define optimization goals, feature scope / priorities [Josh] [INFN] [Bamboo] [xAOD-RDF] [CMSSW-RNTuple]
  - Trying to cover all experiments with limited resources

Where to go from here?

- We don't talk enough with our UK friends: **let's change that!**
  - How can we create a constructive, effective feedback loop?

- axel@cern.ch
- ROOT Forum
- ROOT devs' Mattermost