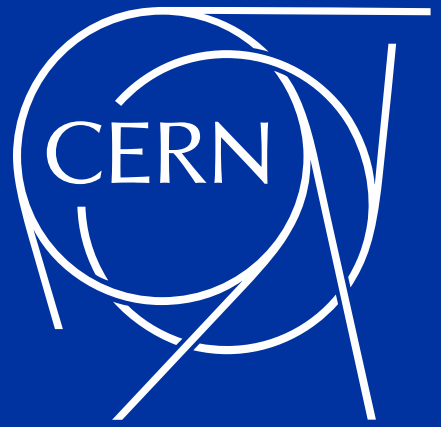


I/O performance studies of analysis workloads on production and dedicated resources at CERN

A. Sciabà, J. Blomer, P. Canal, D. Duellmann, E. Guiraud, A. Naumann, V.E. Padulano, B. Panzer-Steindel, A.J. Peters, M. Schulz, D. Smith

CHEP2023, 8-12 May 2023

Introduction

- **Data Analysis at the LHC is evolving**
  - Very compact data formats (NanoAOD, Physlite) allow to replicate years of data on a single facility
  - New analysis frameworks (Coffea, RDataFrame) allow to use columnar data analysis concepts on both local and distributed resources
  - Services like Xcache or ServiceX can significantly reduce I/O latency and save processing time
  - Interactive analysis using notebooks adds extra convenience
- **Two different types of resources**
  - High performance nodes (many cores, lots of SSD storage) are used for interactive work
  - Analysis facilities as dedicated clusters with software and services to enable interactive distributed analysis
- **CERN providing the former to experiments, and evaluating choices for the latter**
  - A working group on analysis at CERN started in 2021 and produced a final report

Excerpts from the 2022 studies

Experiment	Completed jobs	Running jobs	CPU time	Wallclock time
ATLAS	100K	2500	70 years	80 years
CMS	600K	8000	140 years	170 years

CERN Grid analysis in a 7-day period in October

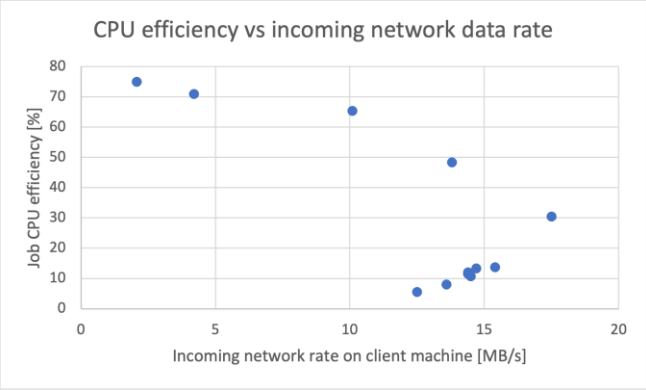
	LXBATCH	LXPLUS
ATLAS	4400 cores	100 cores
CMS	8800 cores	70 cores

Average number of busy cores in a 7-day period

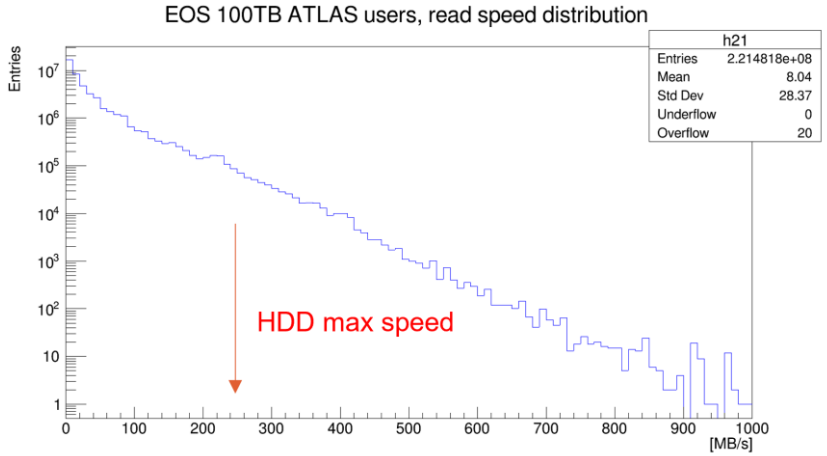
	SWAN	Spark/YARN
ATLAS	8.5K hours · cores	18.5K hours · cores
CMS	16K hours · cores	22K hours · cores

Wallclock time in a 10-day period

Interactive analysis still very small fraction



Studied saturation for a single HDD of Xcache



Average read rates very low and dominated by small reads
Reads from memory caches but < 1% of accesses
EOS capable of hundreds of MB/s
Concurrent access to same HDD is the norm
Batch analysis I/O not intensive, no bottlenecks in EOS
Additional caching layers not needed

Workload	Network limit	Disk limit (96 HDD)	Disk limit (16 SSD)
top-xaod	14000	40000	20000
cms-skimmer	1500	1800	2800
aggregate gradients	20	130	60

Number of clients supported by the storage system heavily dependent on the application
Optimal storage architecture needs to be tuned as a function of the applications

Name	Experiment	Data	Events
top-xaod	ATLAS	1 DAOD_PHYS file	35000
cms-skimmer	CMS	640 NANOAODSIM files	670M
Aggregate gradients	CMS	189 root files	

Experiment workloads used as I/O benchmarks

- ATLAS top-xaod: very low I/O, storage has no impact
- CMS aggregate gradients: multithreaded, significant I/O but high read rates, HDD-based Xcache already optimal
- CMS skimmer: low I/O, no cache needed

Conclusions

- Access to local storage is best
- Xcache is not always useful, and HDDs may be enough
- Scalability depends heavily on workload

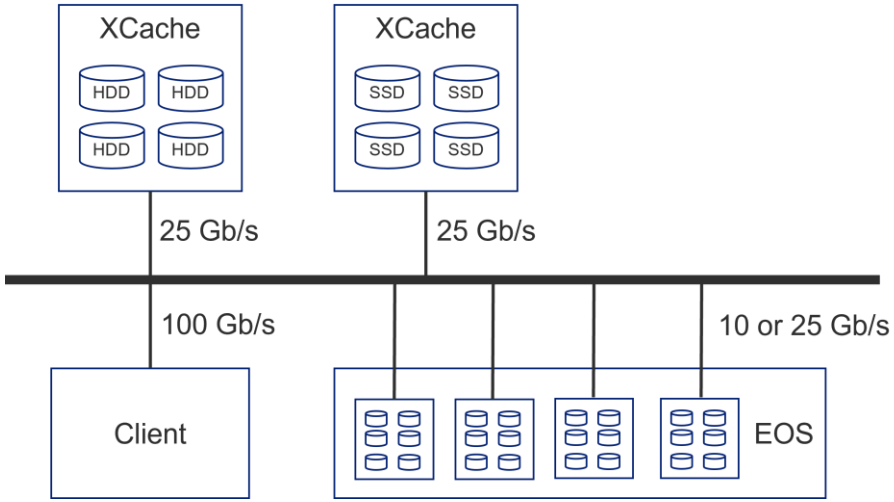
- Infrastructure at CERN can handle well analysis
- EOS is operating well below saturation
- Performance studies are important and should continue

Current work

- **Goal is to collect analysis workloads and tools to measure I/O performance in different storage configurations and levels of parallelism**
  - The scope is CERN but it could be easily extended
  - The final objective is to optimize resource allocation
- **Several workloads available now or soon**
  - ROOT's *rootreadspeed* I/O benchmark
  - ROOT's RDataFrame benchmark
  - IRIS-HEP Analysis Grand Challenge, both Coffea and RDataFrame implementations
  - A real CMS analysis using Coffea by A. Novak
  - A real CMS analysis using RDF by T. Tedeschi
  - The list will expand! Please contact us if you want to contribute
- **Almost all input data is in TTree format**
  - RNTuple is expected to show different behavior

Testbed setup and metric measurement

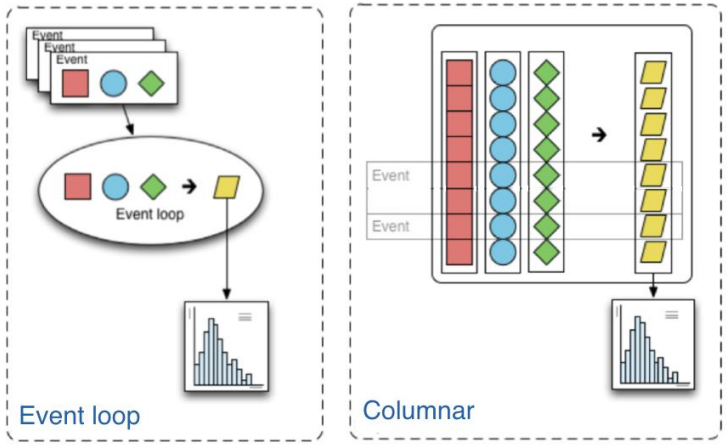
- **High performance client node**
  - Two AMD EPYC 7702 (128 cores)
  - 1 TB of RAM
  - 20 SSD of 4 TB each (of which 10 in RAID0)
  - 100 Gb/s connection
- **Two Xcache nodes**
  - Two Intel Xeon Silver 4216 (32 cores)
  - 192 GB of RAM
  - One with ~ 1 PB in HDD, the other with 32 TB in SSD
- **Storage systems**
  - EOS at CERN (EOSCMS and CERNBOX)
  - Xrootd server at UNL



- **HSF PrMon tool to measure performance**
  - Wallclock time
  - CPU time
  - Read bytes (from storage or network)
  - Time spent in data processing
  - CPU (pseudo) efficiency
    - CPU time / (wallclock time × workers)
  - Average read data rate
    - read bytes / processing time

Analysis Grand Challenge ttbar analysis

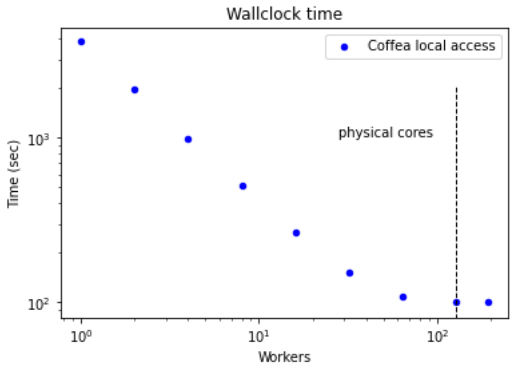
- **Simplified analysis from CMS used as technical demonstrator in IRIS-HEP**
  - Input dataset 3.6 TB, 2300 ROOT files, 1.5 GB/file consisting of CMS 2015 Open Data
- **Columnar analysis paradigm**
  - Distributed using a map-reduce concept
- **Original Coffea implementation**
  - ROOT-less, parallelism via Python futures or Dask
- **RDataFrame port (talk)**
  - ROOT-based, parallelism via implicit multithreading, Dask and other

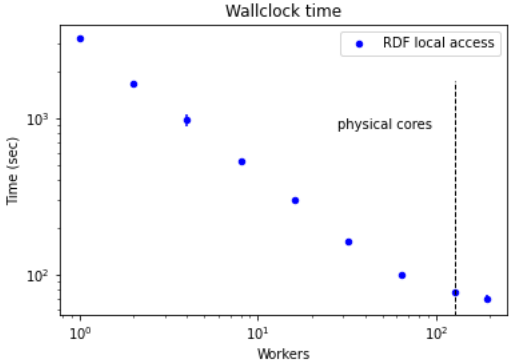


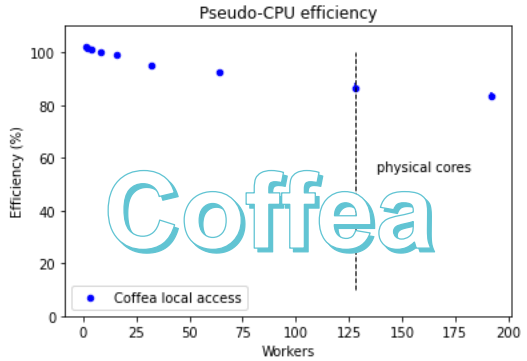
- **Measure performance and scalability**
  - Local parallelism on client node
  - Data read from local node vs. directly from EOS via xrootd vs. via an XCache instance
  - NOT a comparison between Coffea and RDF
    - Simply, different workloads with different behaviors

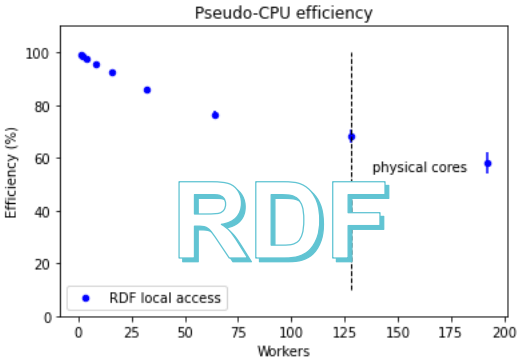
Local access

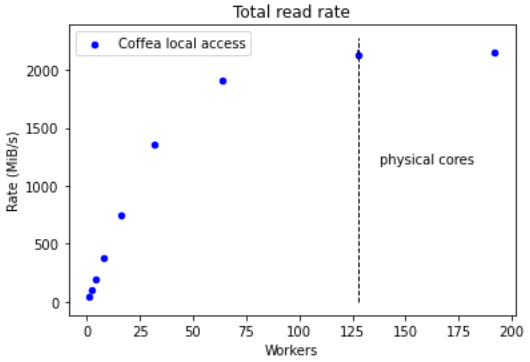
- **Scalability is excellent**
  - Some bottleneck appears for high numbers of workers
  - Overcommitting does not help for Coffea, but it increases throughput for RDF
    - Indication that Coffea is more CPU-constrained
- **The CPU efficiency comparably high**
  - I/O not a strong bottleneck
- **Local, fast SSD storage is always going to work well**
  - Aggregate read rates up to 3 GB/s

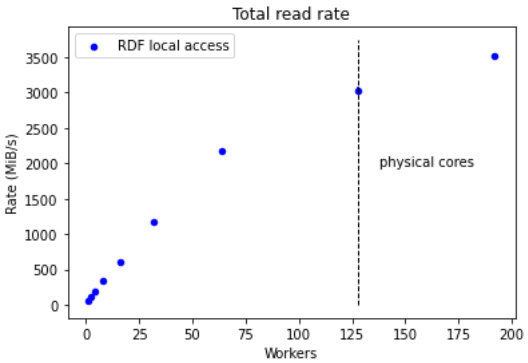






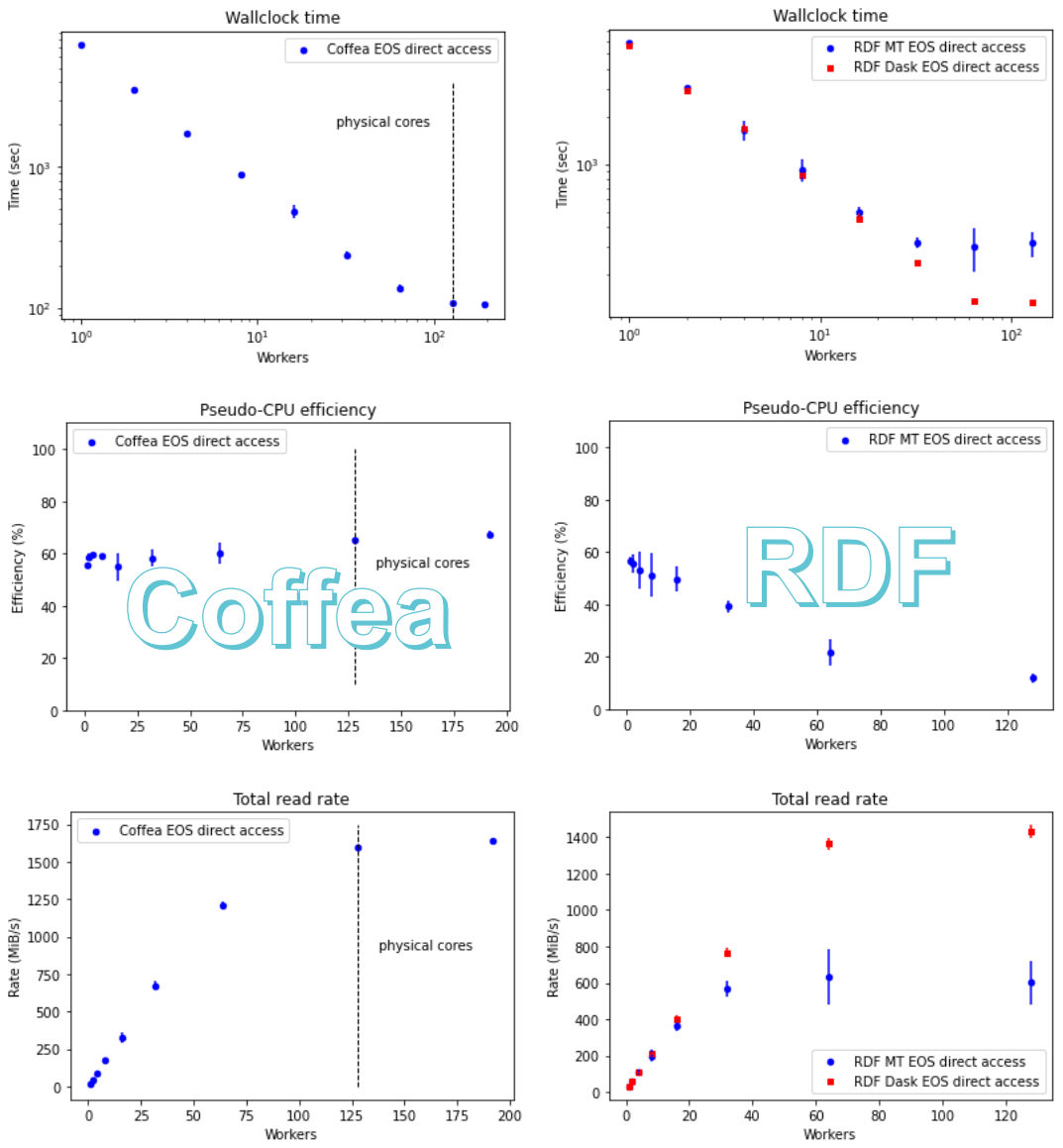






Direct access to EOS

- **Scalability still good when parallelism is via multiprocess**
  - RDF implicit multithreading does not perform well with xrootd and many threads
  - RDF via Dask is multiprocess, scales better
- **CPU efficiency practically constant around 60% with multiprocess parallelism**
  - I/O time is not negligible anymore but no bottlenecks
- **Two EOS instances tested**
  - EOSCMS by default, but CERNBOX produces similar (slightly worse) results



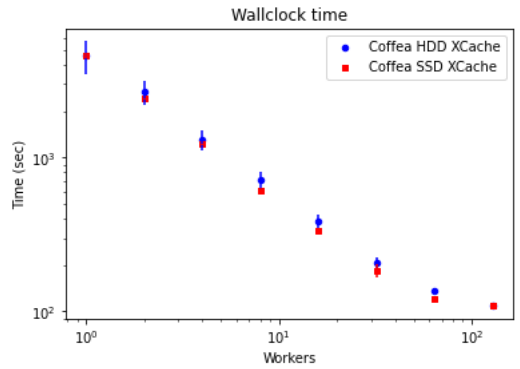
HDD/SDD-based XCache

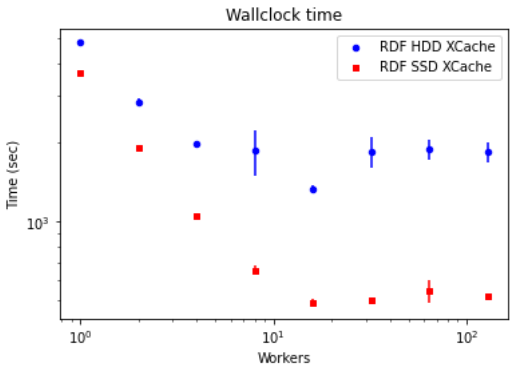
- **Compared performance of direct access to Nebraska and CERN, cold cache and warm cache**

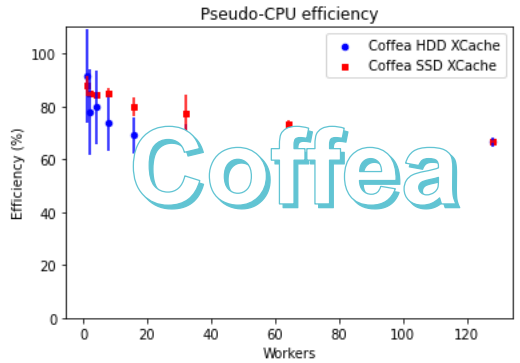
Coffea + HDD XCache: wallclock time (s)			
Site	Direct	Cold	Warm
Nebraska	442 ± 16	608	133 ± 6
EOSCMS	139 ± 8	325	137 ± 3

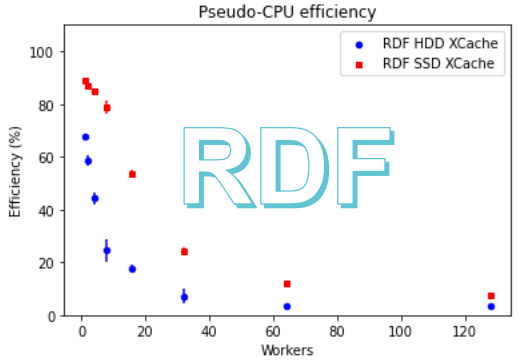
RDF MT + HDD XCache: wallclock time (s)			
Site	Direct	Cold	Warm
Nebraska	5470 ± 910	18841	1531 ± 78
EOSCMS	323 ± 95	8149	1558 ± 400

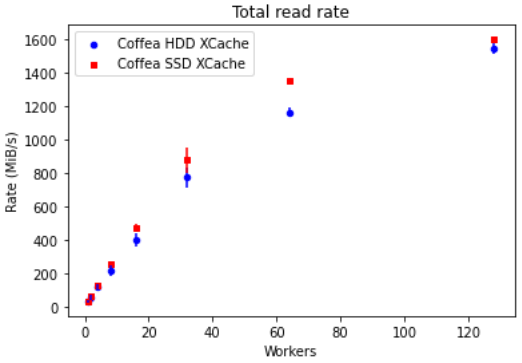
- **Performance results**
  - A cold cache is slower than direct access!
    - Due to sparse file access and network latency
  - Multiprocess scales very well
    - HDD XCache almost as good as SSD XCache
  - RDF multithreaded scales very poorly “out of the box”
    - All connections multiplexed into one ⇒ bottleneck!
    - SDD XCache helps a lot, but scalability is still broken

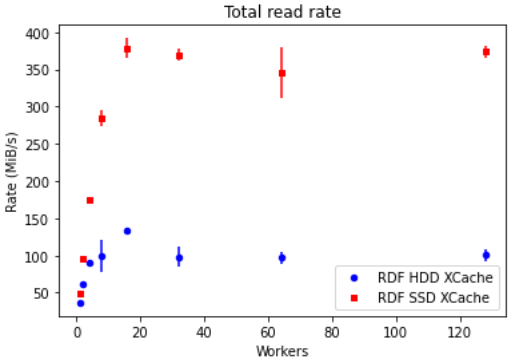






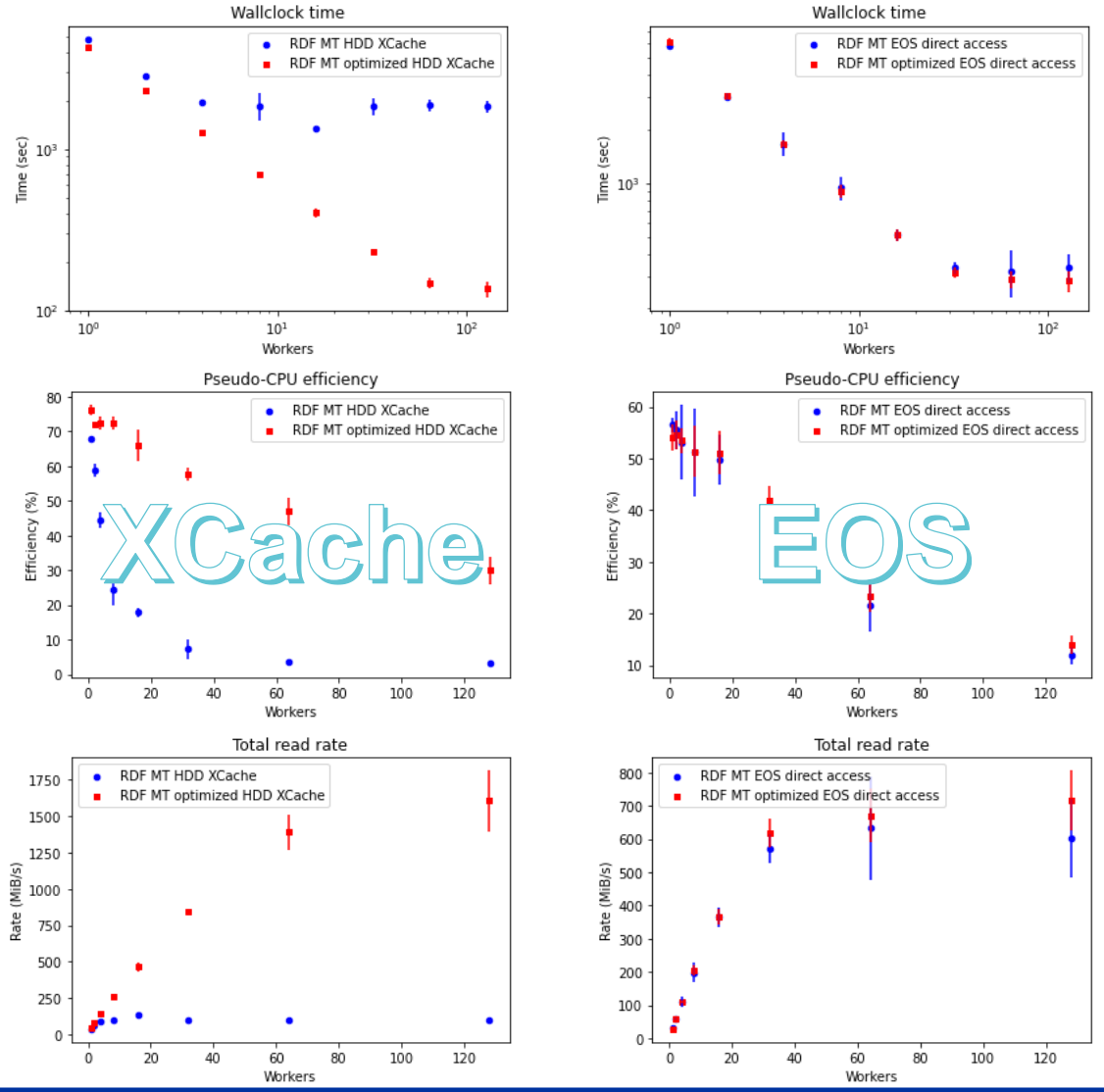






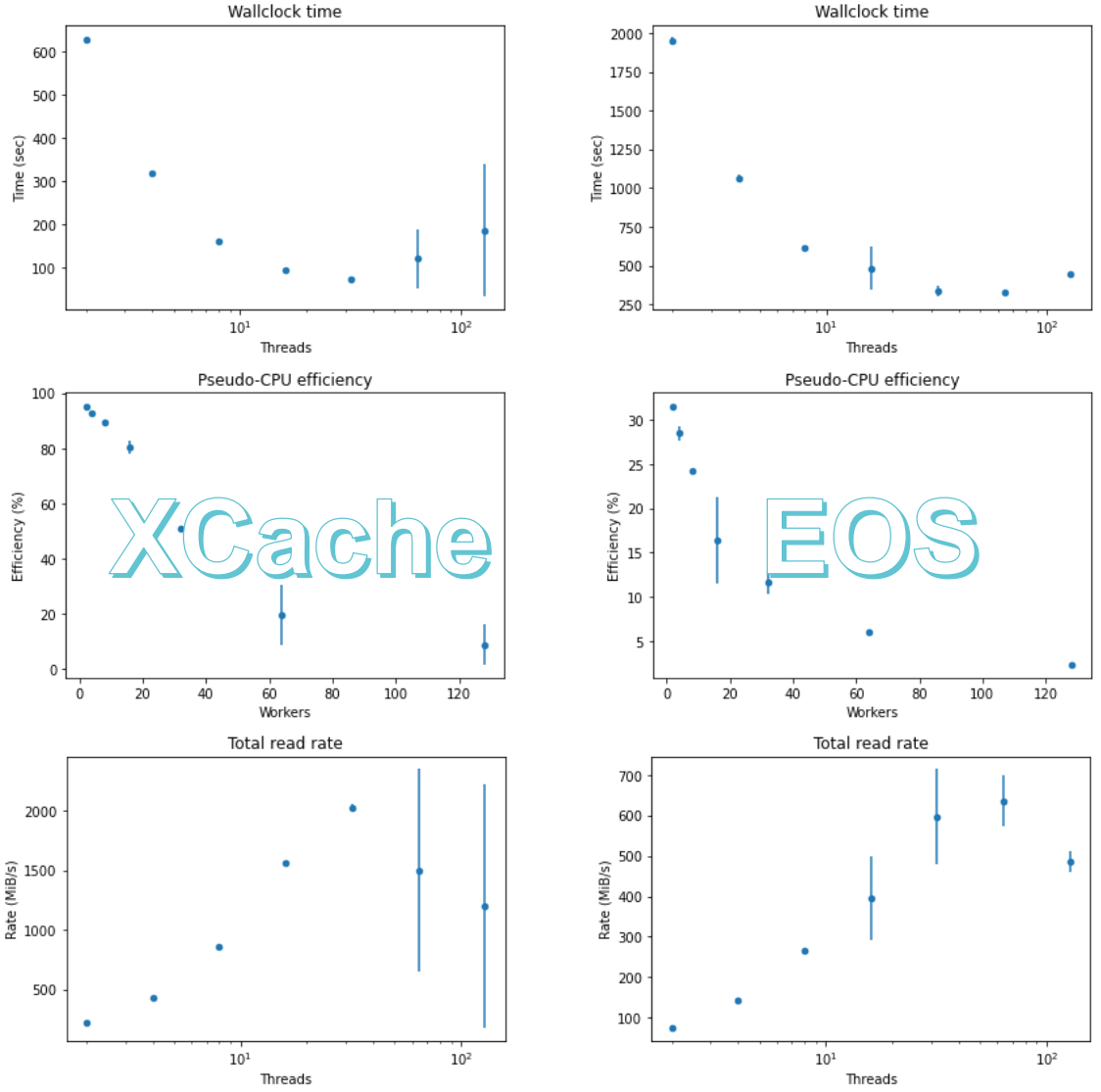
RDF + Xrootd performance optimization

- **Scalability with ROOT multithreading and XCache can be improved**
  - XRD_PARALLELEVTLOOP=10 on the client largely improves Xrootd performance
  - Prevent the connection multiplexing by adding different client names to the file names
    `root://client1@eoscms.cern.ch//eos/myfile.root`
- **Enormous impact when reading from XCache**
  - Obvious as it is a single server and multiplexing a big bottleneck
- **Effect negligible when reading from EOS**
  - Files already spread over hundreds of disk servers, multiplexing irrelevant



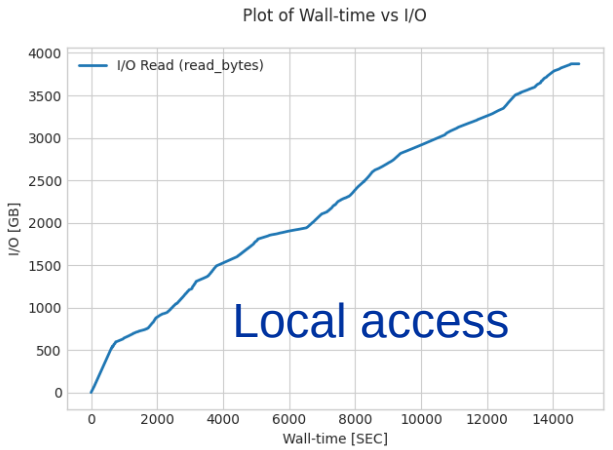
Synthetic ROOT I/O performance

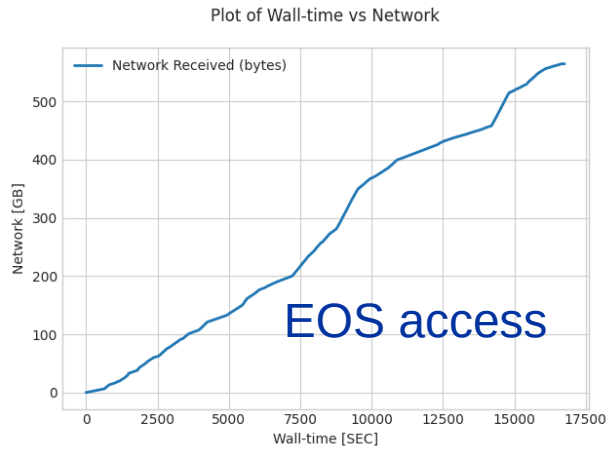
- ***rootreadspeed* CLI tool**
  - Created exactly to identify I/O bottlenecks when reading ROOT files locally or via Xrootd
  - Reading only the branches used by the AGC RDF workload for comparison purposes
- **Excellent performance with XCache**
  - To be better understood
- **EOS performance and scalability similar to EOS RDF**
  - About 1.5 faster as it does only the I/O



CMS NanoAOD analysis using Coffea

- **Real world Higgs analysis from A. Novak**
  - First observation of ggH → cc
  - Running on NanoAOD
    - Tests using 2017 data, 6 TB, average file size 160 MB
- **Performance comparison in different scenarios**
  - Local access
  - Direct EOS access
  - XCache





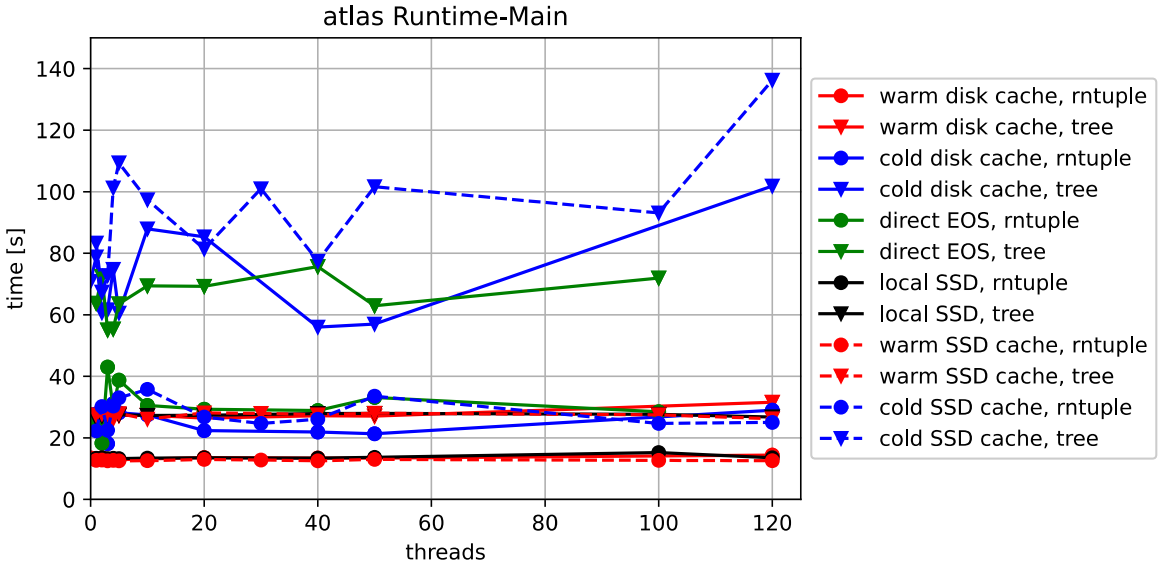
	Wallclock time (h)	CPU efficiency (%)	Read rate (MiB/s)
Local access	4.1	103	270
Direct EOS access	4.7	87	34.5
Cold XCache	4.6	89	34.7
Warm XCache	4.8	82	34.1

- **Results for 64 workers**
  - CPU limited
  - Caching layer irrelevant
  - I/O is modest

Tests with J. Blomer's Virtual Probe Station :

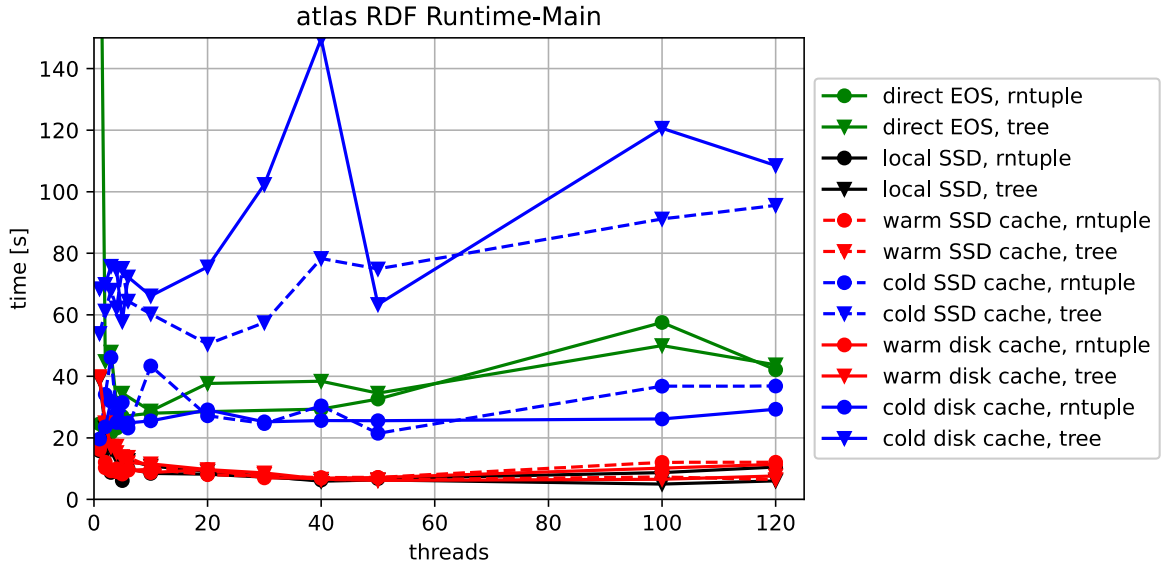
- **Several sample analyses using direct access with TTree and RNTuple**
  - Also support for RDataFrame and implicit multi-threading (prmon readings not conclusive)
- **ATLAS:** H --> gg 2020 open data (cf. tutorial df104). Medium dense reading (~25%, 12/81 features)
  - TTree: 9.9GB, RNTuple: 4.4 GB
- **CMS:** 2019 ttjet nanoAOD (cf. tutorial df102). Sparse reading (<1%, 6/1479 features)
  - TTree: 20GB , RNTuple: 13.5GB
- **Lhcb:** LHCb run 1 open data (B mass spectrum). Dense reading (>75%, 18/26 features)
  - TTree: 5.3 GB , RNTuple: 5.3 GB
- **H1**: ROOT's H1 standard analysis (cf. df101 tutorial).
  - TTree: 13.3 GB , RNTuple: 10 GB Medium dense reading (~10%, 16/152 features)
- Measurements with different storage setups (PrMon and internal metrics (too many to show..))

- **Cold HDD cache and direct EOS are close**
- **Warm SSD cache is on par with local SSDs**
- **Data formats matter, little gain from concurrency**
- **TTree is sensitive to external activities**



Direct access TTree and RNTuple

- **RDataFrame shows similar patterns, profits from using multiple cores up to 50**
- **For RNTuples on SSDs, RDataFrame is, using 40 cores, more than twice as fast as "classic" processing**



RDataFrame-style processing

Future plans

- **More workloads in the pipeline**
  - T. Tedeschi's CMS analysis using RDataFrame
  - New version of the AGC workload using NanoAOD
- **Obtain workloads from other LHC experiments**
- **Scale out tests to batch using Dask**
- **Simulate multiuser environment**
- **Provide documentation to allow others to run the same tests**
  - With some limitations, e.g. only Open Data datasets

Conclusions

- **Designing an analysis facility requires to test several workloads in several hardware, client and server configurations**
- **Compared performance and scalability under different (favorable) scenarios**
  - A high-performance node with no other users offers the maximum obtainable throughput per core
  - Direct access to EOS comparably performant, no server-side bottlenecks observed
    - It is a shared facility, busy with other workloads, but performance is rather consistent
  - A caching layer (XCache) does not provide additional benefits to direct access to EOS
    - SSDs are better but not necessarily much better
    - It will be shared in production
  - Quasi-real workloads like the AGC ones are a very valuable tool, but real workloads are also essential for testing
- **Plans to extend the testing and analysis activity**
  - New workloads
  - New ideas
- **Will work on making RDF/XrdCl multithreading more scalable without the need for additional tuning**

Acknowledgements

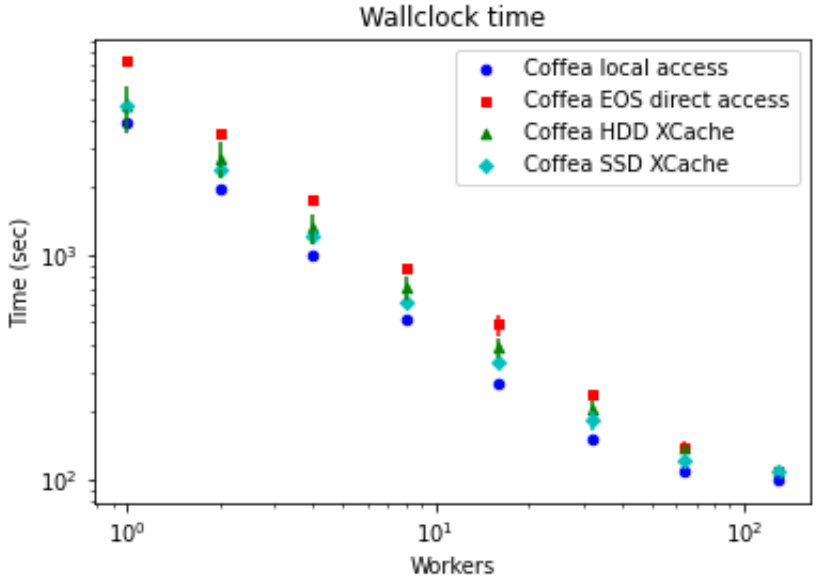
- **The authors would like to thank**
  - B. Bockelman, S. Campana, X. Espinal, A. Held, O. Shadura, C. Lange, P. Lenzi, L. Mascetti, A. Novak, E. Sindrilaru, J. Pivarski, T. Tedeschi and others for many useful discussions and suggestions

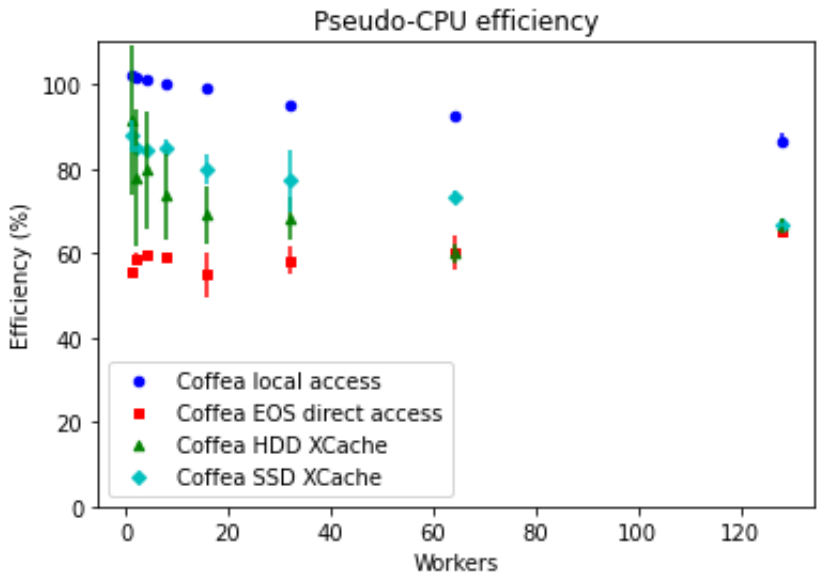
References

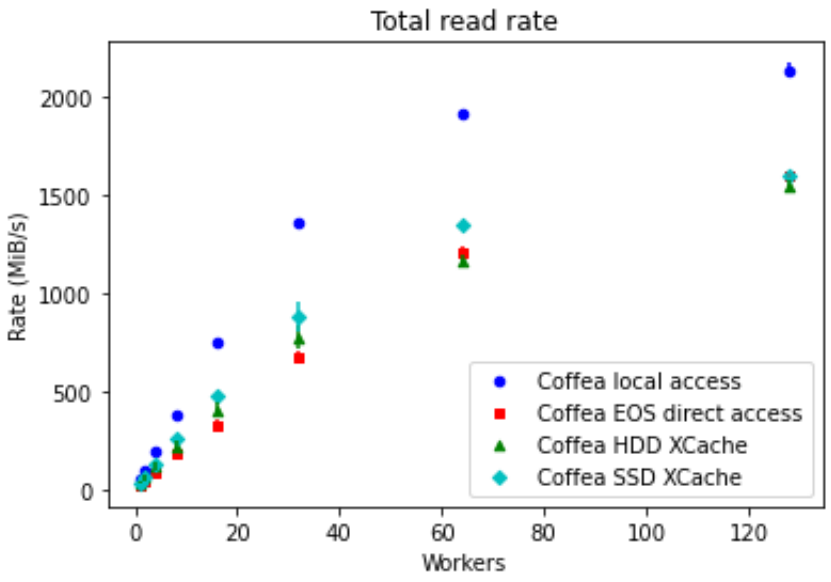
- **Analysis for LHC experiments at CERN: https://zenodo.org/record/6337728**
- **PrMon: https://github.com/HSF/prmon**
- **Analysis Grand Challenge: https://github.com/iris-hep/analysis-grand-challenge**
- **Coffea: https://coffeateam.github.io/coffea/**
- **AGC RDF implementation: https://github.com/andriiknu/RDF/**

Backup slides

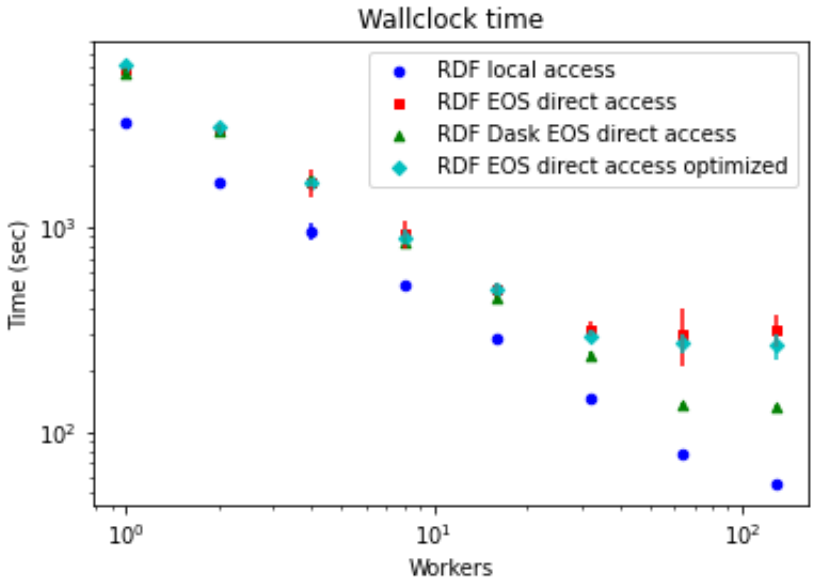
Coffea: all configurations

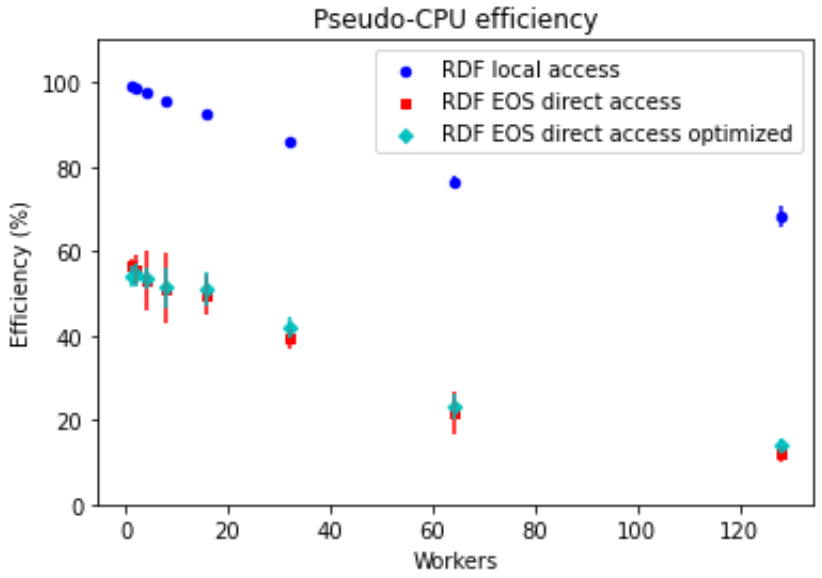


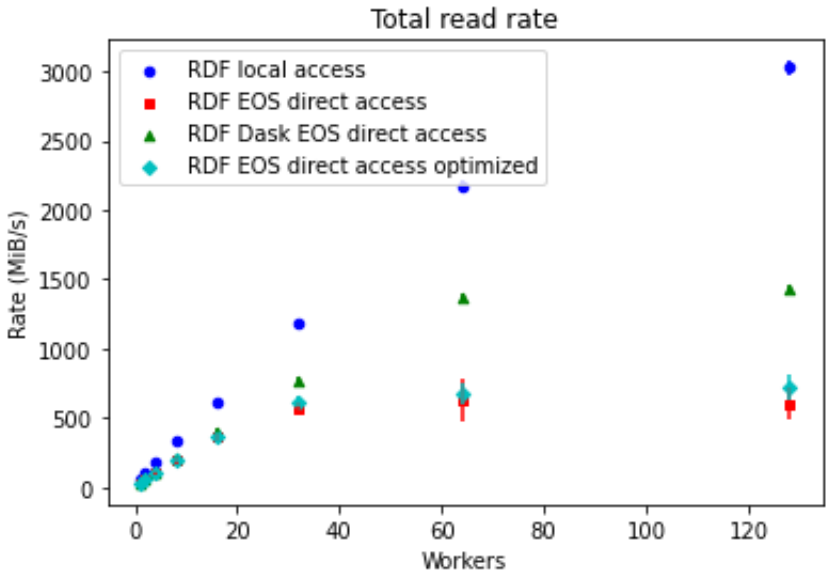




RDF: all non-Xcache configurations







RDF: all XCache configurations