Results of the Benchmarks for the DCOTF and XCache

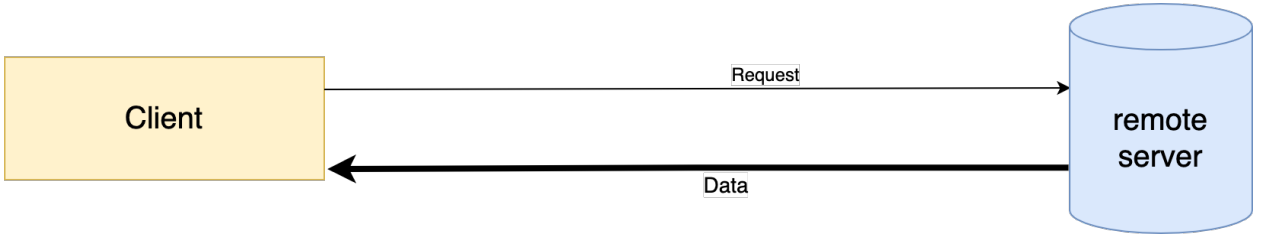
Volker Lindenstruth, Kilian Schwarz, Paul-Niklas Kramp, Dirk Hutter,
Serhat Atay
Goethe University, Frankfurt
GSI, Darmstadt
11 May 2021

Singularity Container Image Management

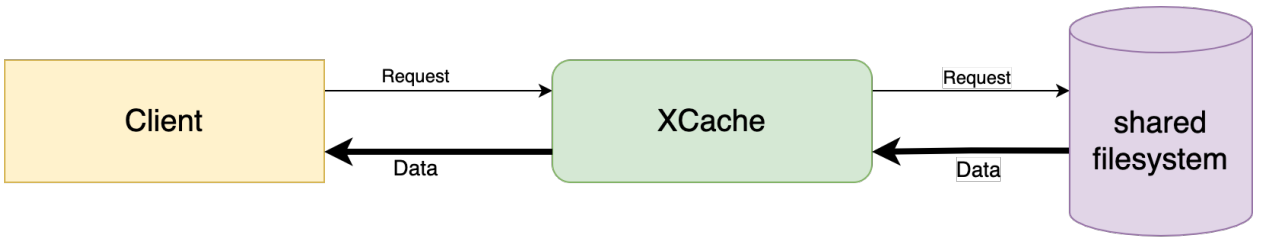
- A shell script (only for ALICE experiment)
  - to fetch the binaries from cvmfs repository for a specific ALICE environment tag and
  - to build a container image with the binaries included
- Prerequisites to run the script
  - A linux operating system (tested for centOS)
  - alice.cern.ch cvmfs repository mount
  - singularity (with superuser or with --fakeroot option enabled)
- Usage
  - sudo ./singularity_alice.sh -t <environment_tag> -o <working_directory> -y
    - -t/--tag specify a valid environment tag, for example "AliPhysics/vAN-20201018_ROOT6-1”
    - -o/--output specify a working directory, default: current directory
    - -y/--yes Yes to folder clean up. it will clean the content of working directory.
    - -h/--help displays help
- Continuous integration
  - Gitlab runner (being resurrected at GSI)
    - checks if the script is successfully run
    - checks if the container image is successfully run
    - checks if the environment is set inside the container
    - (later) pushes the image to a registry to distribute (continuous deployment)
- Source code with detailed documentation: https://git.gsi.de/atay/alice_singularity
- Generalization for other experiments (next milestone by 30 June 2021)
  - Continuous integration agent is general and can be used for other experiment
  - However, requires similar script for each experiment
  - Any volunteer to write a similar script for another experiment?

Cache Scenarios for Benchmarks

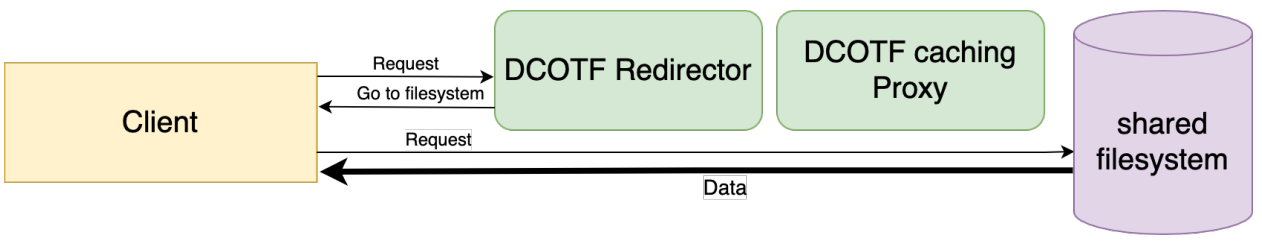
- No cache



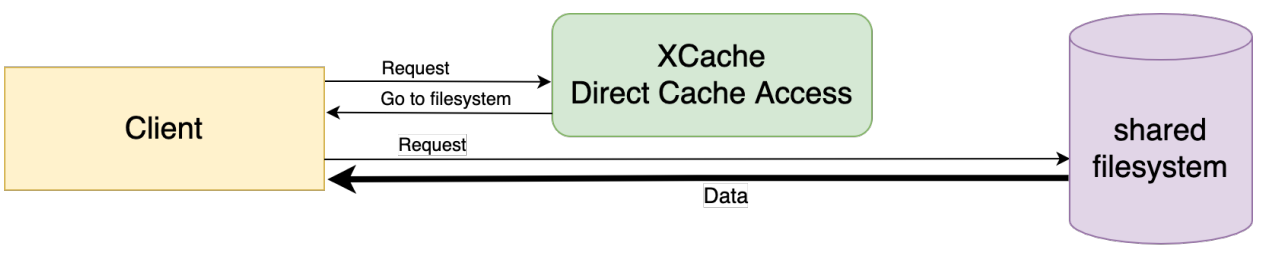
- XCache, data in cache



- DCOTF, data in cache

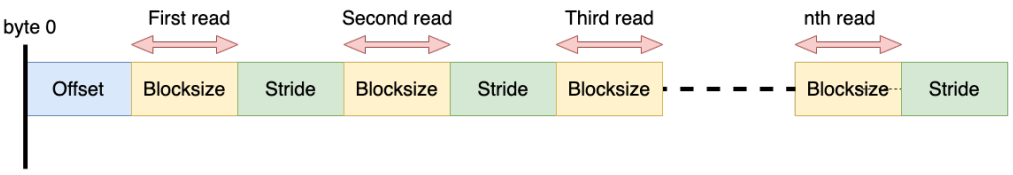


- XCache DCA, data in cache



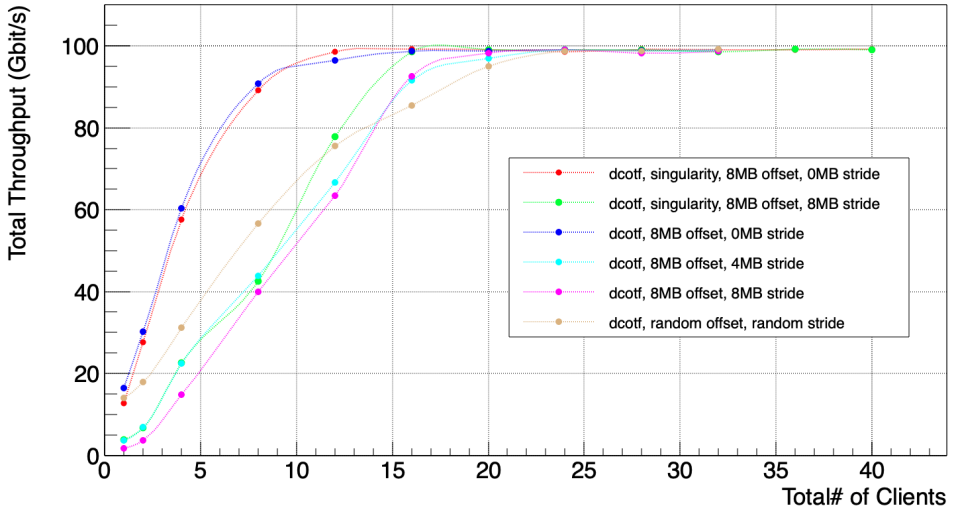
Generic Read Benchmark with XrdCl

- In Frankfurt (Goethe-HLR)
  - SLURM scheduler
  - Local cache location under BeeGFS filesystem at Goethe-HLR with infiniband fabric
- Caching node: node48-004, client nodes: node48-[001-003]
  - 2x (20 core, 40 thread) CPUs, 192GB RAM, EDR link (100Gbit/s)
- Blocksize: 8MiB (8.38MB)
- Plots in Gbit/s (10^9 bits)
- Datasets at GSI Lustre filesystem with 10Gbit/s link to Goethe-HLR
  - Files filled with "/dev/urandom" at Lustre
    - 80 x 50GiB files (for DCOTF and XCache DCA) or
    - 160 x 25GiB files (for XCache, memory flush with 200GB file between SLURM jobs)
  - with different names (to avoid memory caching in the node)
  - with exactly same contents
- Reading algorithm
  - No multiple read of the same blocks,
    - offset in each iteration: offset + (blocksize + stride) * counter
  - measuring interval: 1 second (# blocks / second)
  - Measurement time: 20-100 second

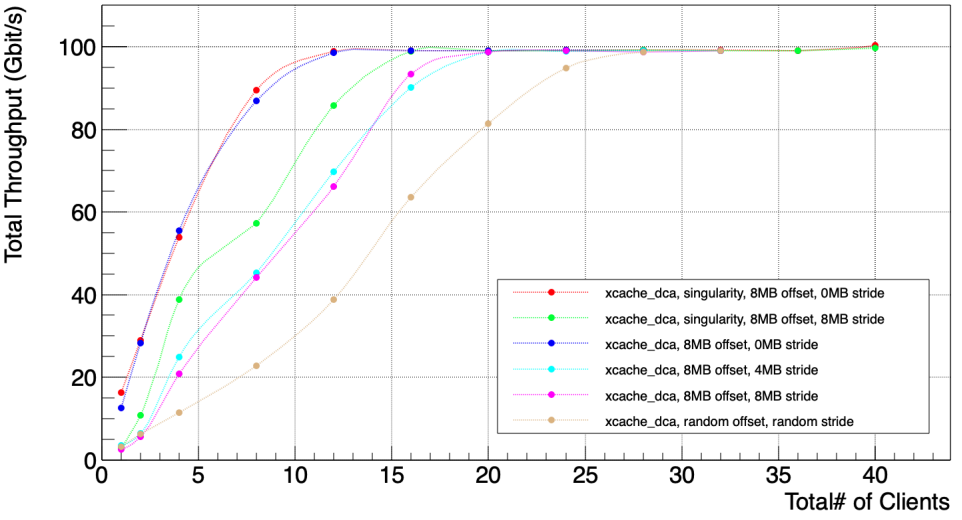


- Time stamp and read interval matching
  - Reading time and interval are matched with respect to system time when summing up clients

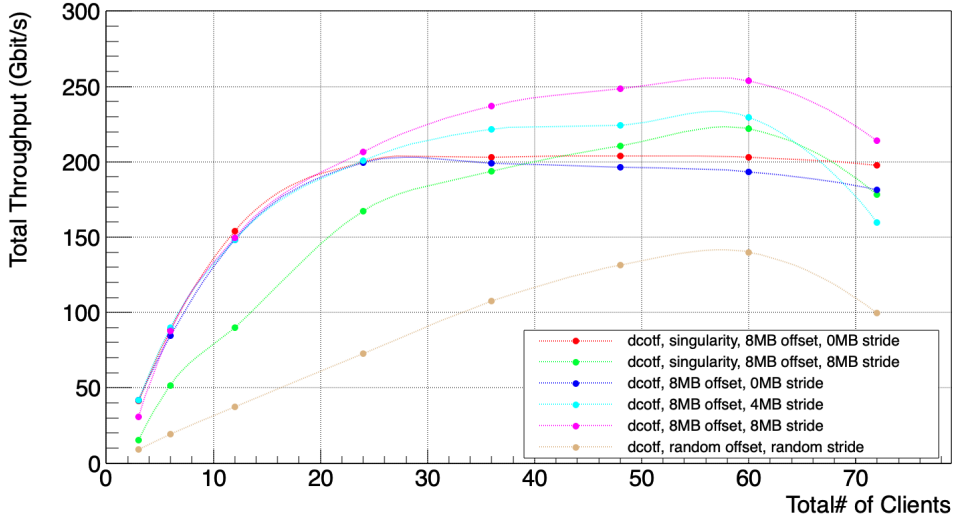
DCOTF, 1 NODE



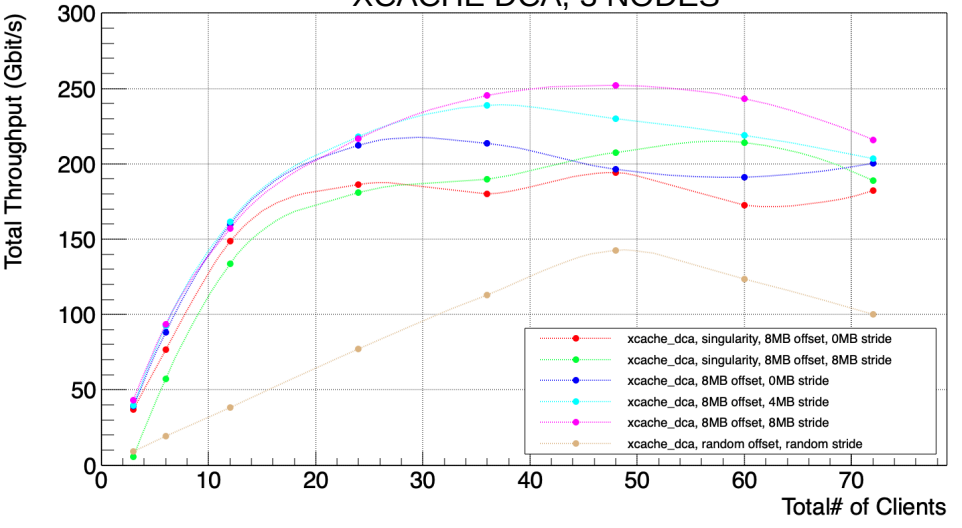
XCACHE DCA, 1 NODE



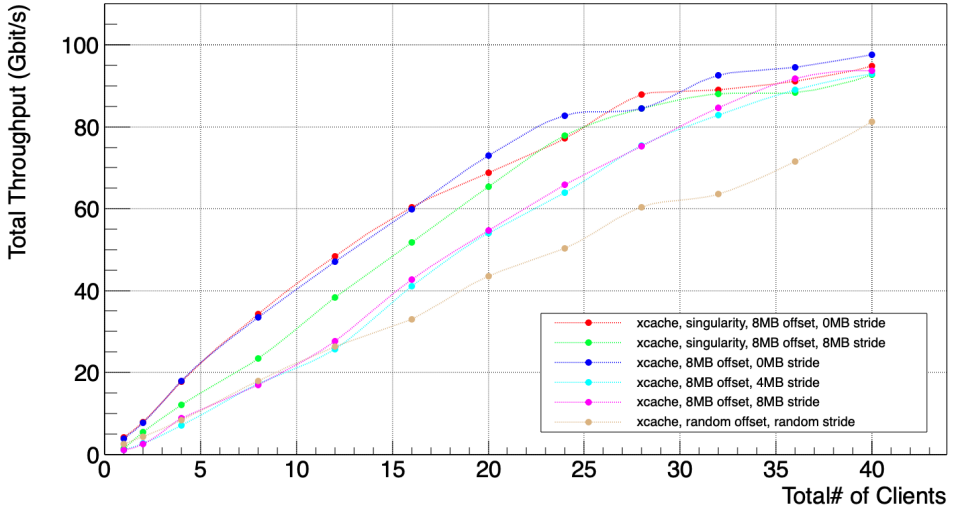
DCOTF, 3 NODES



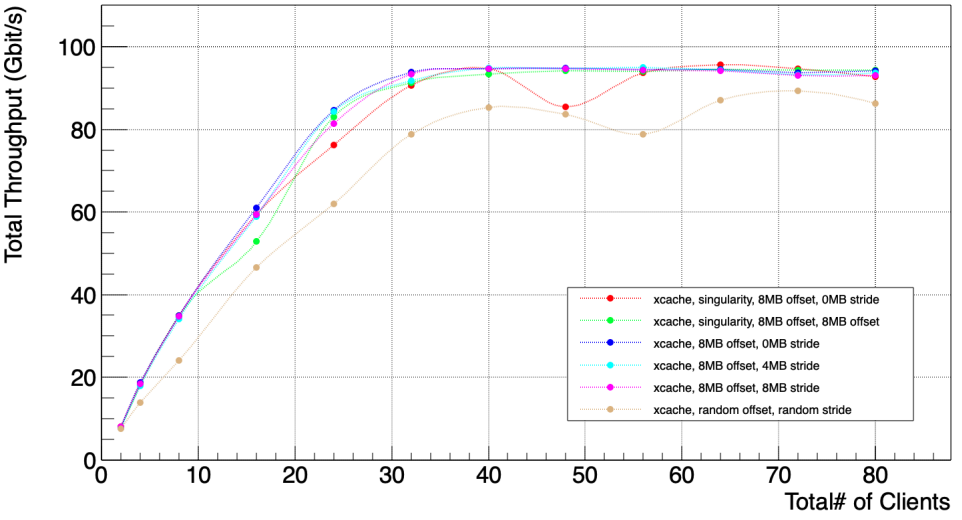
XCACHE DCA, 3 NODES



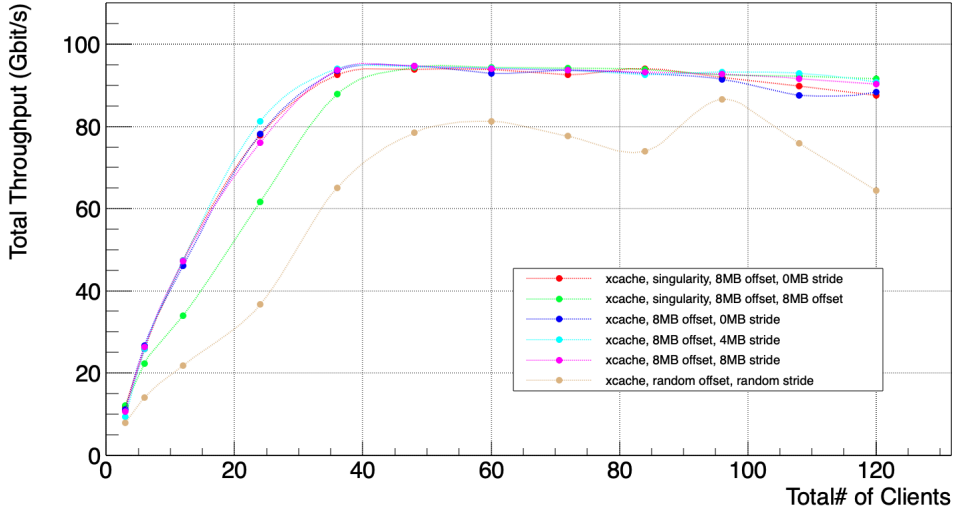
XCACHE, 1 NODE



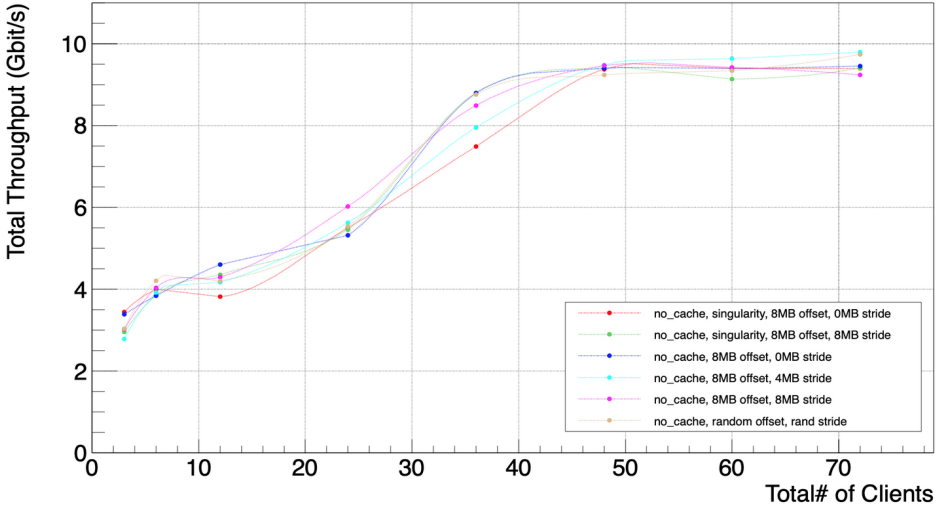
XCACHE, 2 NODES



XCACHE, 3 NODES



NO CACHE, 3 NODES



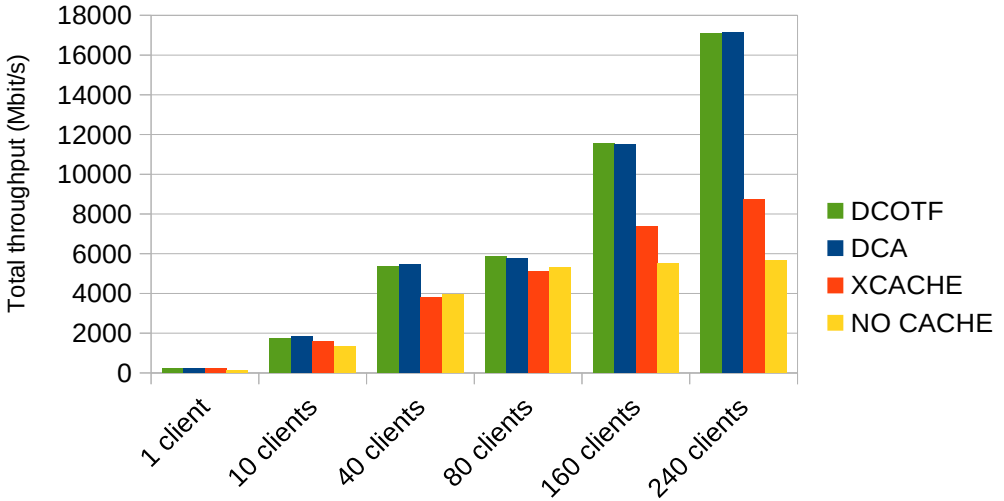
Results for Micro Benchmark

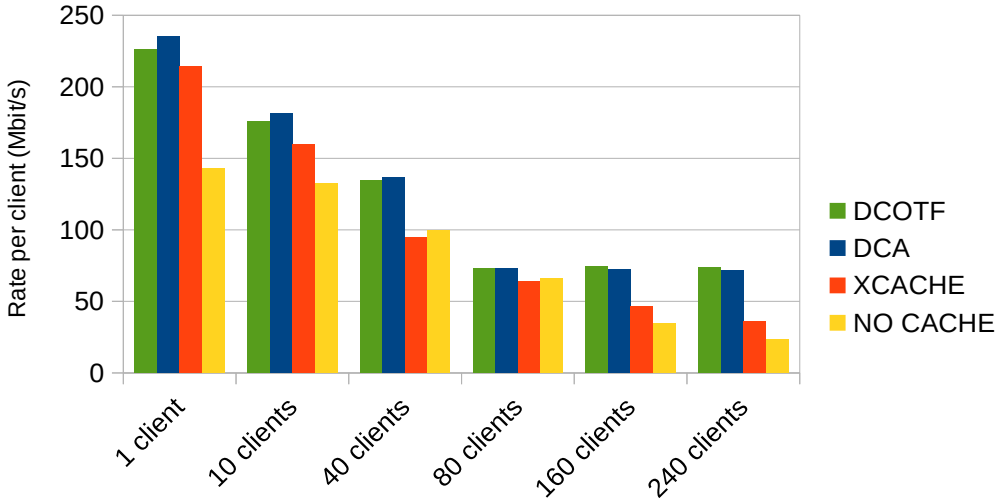
- Successful saturation of the node and filesystem bandwidth for DCOTF and XCache DCA
- XCache server with slightly lower bandwidth of the caching node (~95Gbit/s)
- Saturation 10Gbit/s link between GSI and Frankfurt without cache
- Highest rate per client with 0MB stride
- Similar results with singularity container
- Lower read performance with random stride
- Filesystem bandwidth
  - Highest observed: 251Gbit/s (during easter holiday)
  - Depends on the actual usage of the filesystem by other users
    - No filesystem throughput history available to compare. (admins do not store it)
  - Aggregated filesystem BW on the Goethe-HLR website: 216 Gbit/s (27 GB/s)
    - It does not pose any kind limitation of filesystem according to admins
  - Reduction of the filesystem bandwidth with higher number of clients (for # of client>~60)

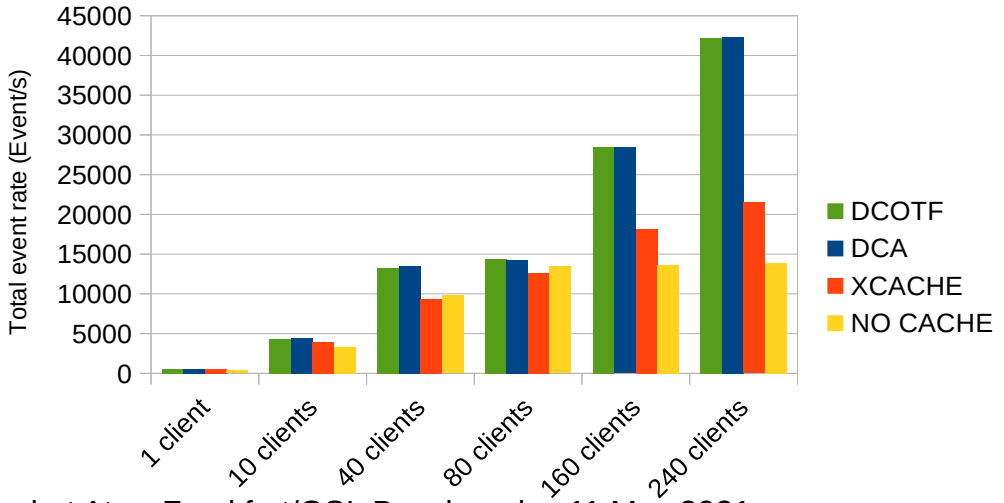
Analysis Benchmarks (only in Singularity)

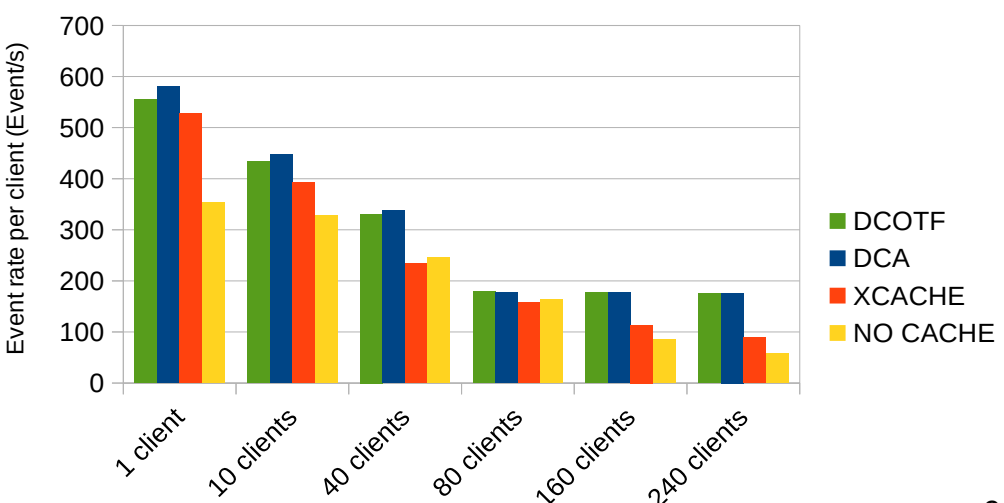
- In Frankfurt (Goethe-HLR)
  - SLURM scheduler
  - Local cache location under BeeGFS filesystem at Goethe-HLR with infiniband fabric
- Caching node: node48-004, client nodes: node48-[001-003]
  - 2x (20 core, 40 thread) CPUs, 192GB RAM, EDR link (100Gbit/s)
- “No cache” measurements with old router for 10Gbit/s link
  - router at GSI → router in Frankfurt → worker nodes: ~6Gbit/s due to CPU limitation of router in Frankfurt
- Analysis setup
  - ALICE AOD and ALICE ESD datasets
    - 10530MB / 25990 events AOD data (CPU intensive analysis tests)
    - 31457MB / 17290 event ESD data (data intensive analysis tests)
  - 1, 10, 40 and 80 client on single node (node48-001)
  - 160 clients on 2 nodes, 240 clients on 3 nodes
- Sample analysis reads a branch from data and fill a histogram
  - removed filling histogram to increase throughput
  - now only reads events from a branch and does nothing further (>80% of each file)

ALICE AOD Data in Singularity

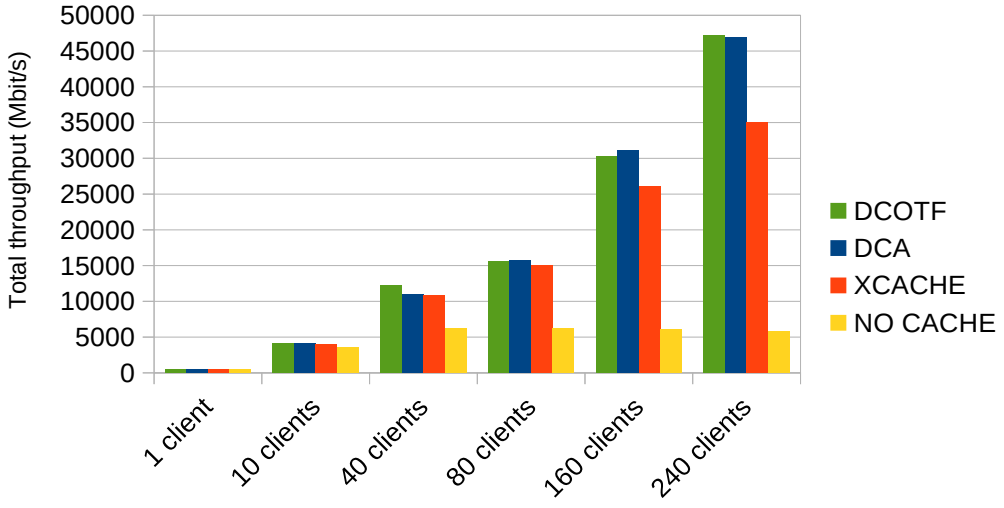


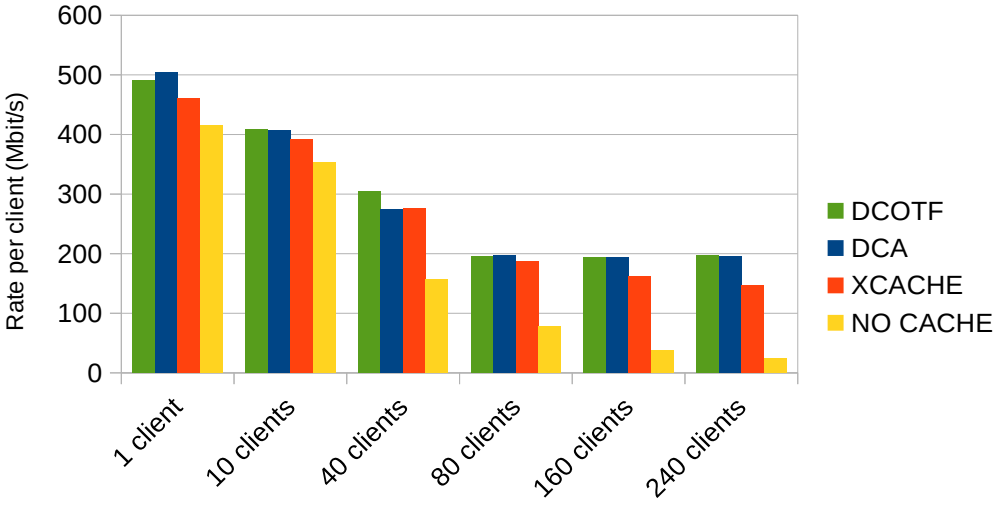


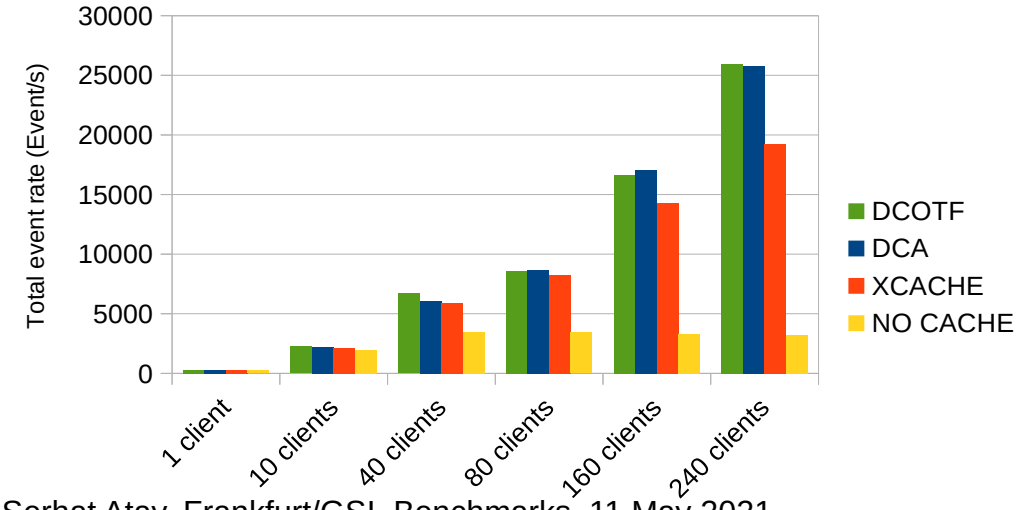


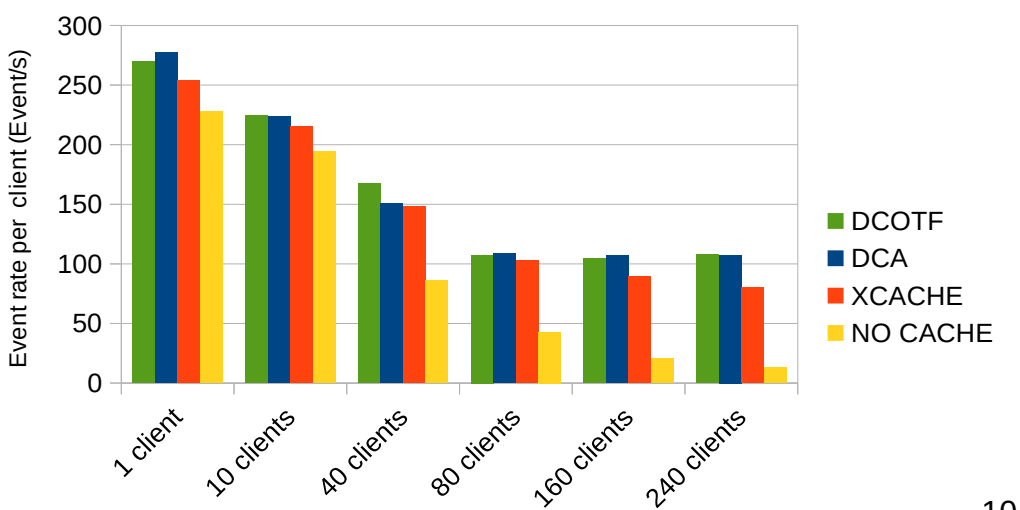


ALICE ESD Data in Singularity









Results for Analysis Benchmark

- Similar performance DCOTF and XCache DCA
  - Comparable performance with XCache up to # of total threads on the caching node (80 clients)
- AOD data
  - ~17Gbit/s achieved (240 tasks)
  - ~47k event/s achieved (240 tasks)
  - Maximum ~240Mbit/s per client (1 client) and ~72Mbit/s after CPU saturation
  - Maximum ~580 event/s per client (1 client) and ~190 event/s after CPU saturation
- ESD data
  - ~47Gbit/s bandwidth achieved with ESD data (240 tasks)
  - ~26k event/s achieved (240 tasks)
  - Maximum ~490Mbit/s per client (1 client) and ~200Mbit/s after CPU saturation
  - Maximum ~280 event/s per client (1 client) and ~110 event/s after CPU saturation
- “No Cache” is limited to 6Gbit/s, thus has the worst performance among all

Summary and Outlook

- Environment specific standalone ALICE singularity image script is ready with image management system
- DCOTF and XCache DCA has a similar performance when reading data
  - XCache DCA is natively supported by XRootD
  - DCOTF requires a special XRootD cms plugin and a modification on the caching plugin to alter UNIX permissions of the files in the cache
- XCache creates a bottleneck in terms of CPU and bandwidth for the cached data in a shared filesystem
- Containerization doesn’t affect the performance
- HTCondor jobs extending over multiple HPC centers in singularity (in progress)

Thanks