Boosting Performance of Data-intensive Analysis Workflows with Distributed Coordinated Caching

C Heidecker, R F von Cube, M Giffels, G Quast, M Sauter and M J Schnepf

KIT - Karlsruhe Institute of Technology, Germany

E-mail: christoph.heidecker@kit.edu

Abstract. Data-intensive end-user analyses in high energy physics require high data throughput to reach short turnaround cycles. This leads to enormous challenges for storage and network infrastructure, especially when facing the tremendously increasing amount of data to be processed during High-Luminosity LHC runs. Including opportunistic resources with volatile storage systems into the traditional HEP computing facilities makes this situation more complex.

Bringing data close to the computing units is a promising approach to solve throughput limitations and improve the overall performance. We focus on coordinated distributed caching by coordinating workflows to the most suitable hosts in terms of cached files. This allows optimizing overall processing efficiency of data-intensive workflows and efficiently use limited cache volume by reducing replication of data on distributed caches.

We developed a `NaviX` coordination service at KIT that realizes coordinated distributed caching using `XRootD` cache proxy server infrastructure and `HTCondor` batch system. In this paper, we present the experience gained in operating coordinated distributed caches on cloud and HPC resources. Furthermore, we show benchmarks of a dedicated high throughput cluster, the Throughput-Optimized Analysis-System (TOpAS), which is based on the above-mentioned concept.

1. Introduction

The performance of data-intensive workflows is limited by the data transfer rate [1]. This is especially significant within a distributed computing infrastructure, where workflows must access data from remote when it is not available locally. We can avoid bottlenecks regarding data transfer rates by bringing data processing and storage resources close enough together [2]. Applying this *data locality* on various scales, e.g. within regional computing clusters or on per-host basis, allows for individually tuning established computing infrastructures for data-intensive workflows.

Caching data for realizing data locality fits especially well for HEP workflows that repeatedly access large amount of data as input. If the throughput for accessing remote data is limited, repeatedly accessing cached input data leads to an overall optimization of data throughput, and, thus, the CPU efficiency of workflows waiting for data. While increasing the efficiency of data processing and thus reducing the processing time, caching reduces the load for shared storage and network resources.

Conventional caches deployed in a distributed infrastructure might not necessarily be used in an efficient way. Repeated processing of workflows that might run on different hosts can

cause duplication of data in multiple caches. We want to avoid duplication, since it wastes limited cache space, decreases the cache hit rate, and, thus, reduces the total data throughput rate [2]. Therefore, it is essential to coordinate data placement and schedule workflows to the most suitable host in terms of data locality.

2. Coordinating distributed caches

We focus on increasing the efficiency of the WLCG Tier 3 computing infrastructure for HEP workflows including institute and opportunistic resources. There are two main challenges when coordinating caches in a distributed computing infrastructure: data selection and job to cache coordination. We need to select data relevant for caching, since we only need to speed up jobs that are suffering from insufficient data transfer rate. Other jobs will indirectly profit from resources that are freed faster. Furthermore, we need to coordinate jobs to the most suitable host in terms of data locality. To foster data locality, we need to influence the batch system, especially the scheduling jobs to resources. By scheduling of jobs to resources, we directly influence the placement of data in caches, and, thus, also coordinate data in distributed caches.

3. NaviX — Implementation of job and data coordination

We realized the coordinated distributed caching concept by implementing the coordination service `NaviX` [3]. It adds job and data coordination to an `HTCondor` batch system [4] using an `XRootD` [5] caching infrastructure. This section gives a short overview about the working principle of `NaviX`. A more detailed description can be found in [6].

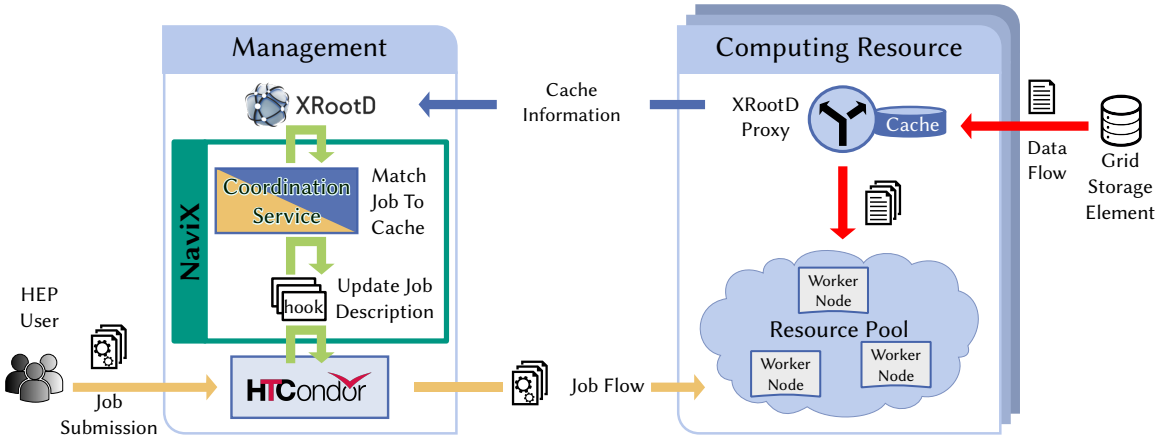

Figure 1. `NaviX` orchestrates an `XRootD` caching proxy server infrastructure and an `HTCondor` batch system to coordinate jobs to the most suitable computing resource in terms of data locality.

We require an `HTCondor` batch system and concentrate on data being transferred via the `XRootD` protocol. Both are widely used within the HEP community. Since `HTCondor` does not natively consider data locality for job to resource scheduling, we need to include this information by manipulating submitted jobs before they are scheduled. `XRootD` already provides caching via so-called *caching proxy servers* [8] and allows to transparently redirect clients accessing files to such a caching proxy server [1]. A caching proxy server reads the local copy if the file is already cached, or it streams the accessed file while caching it on the fly for repeated access. Such `XRootD` caching proxy servers can be placed within a distributed infrastructure. As shown in Figure 1, each submitted job triggers the `NaviX` coordination service to coordinate the job based

[1] The transparent redirection of data transfers to the caching proxy server requires `XRootD` version 4.7 and higher.

on data locality. Using information about the input data of the job as well as files already cached within `XRootD` proxy servers, `NaviX` calculates how well each cache fits to the job. Based on the calculated data locality score, it updates the job description to influence the scheduling of `HTCondor`. The decision is updated periodically while jobs are waiting for resources to become available. We need the user to specify the list of input files required for each job, since this can not be reliably extracted from the job itself.

4. Benchmarking prototype setups for distributed caching

Benchmark results of a first prototype system for coordinated distributed caching were presented at the CHEP conference in 2018 [6]. We showed that `NaviX` successfully coordinates jobs to the most suitable hosts in terms of data locality. It reduced the duplication of data in distributed caches and improved the overall data throughput by coordinating jobs to the cached data. The experiences of operating this prototype system allowed us to test the suitability of coordinated distributed caching on different kinds of resources. In this section, we present benchmark results of a dedicated high throughput cluster, an HPC cluster, and a cloud resource when using distributed caches coordinated by `NaviX`.

4.1. A dedicated high throughput cluster

We installed the *Throughput Optimized Analysis System* (TOpAS), a dedicated high throughput cluster at the WLCG Tier 1 center GridKa. This cluster consists of 11 worker nodes and is designed for data-intensive end-user analysis workflows benefiting from fast network connection to WLCG storage resources. Each worker node has a single 1TB NVME SSD, and all worker nodes share a 1PB distributed filesystem, both intended for caching.

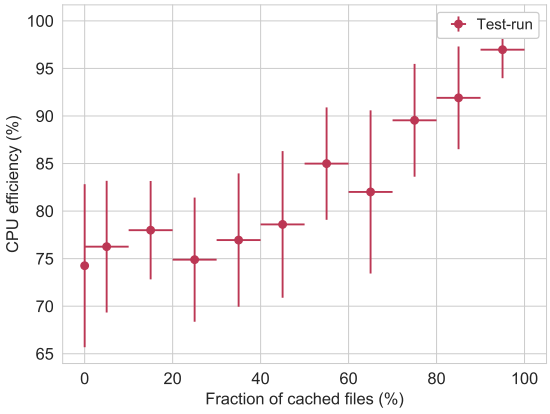

Figure 2. HEP analysis benchmark for the TOpAS cluster using 9 coordinated NVME SSD caches and accessing data stored on remote Tier 2 center. The dots show the mean value of 420 jobs processed in parallel, the error bars represent the variance. [7]

First benchmarks of the TOpAS setup use the NVME SSDs as coordinated cache to test their usability as fast level-one cache. We measure the benefit for a data-intensive CMS jet energy calibration workflow, which serves as an estimate for benefit for HEP analysis workflows. The test jobs accessed data from remote Tier 2 Grid storage elements to evaluate the benefits of caching within the WLCG structure. We repeatedly process the same jobs while increasing the fraction of cached files in steps of 10%. This allows scanning for an optimal working point, where jobs profit from optimally combining the data transfer rate from remote Grid storage elements via network with direct access to local cache.

As Figure 2 shows, processing cached data directly from built-in SSD increased the CPU efficiency of the workflow by about 20% compared to accessing remote Tier 2 Grid storage elements. When data transfer from remote Grid storage elements is limited, the high data transfer rate of built-in caches accelerate the jobs. Speeding up the workflows by 20% CPU efficiency means, that we can save the same fraction of costs for CPU cores. A realistic speed-up factor $f_{speed-up}$ of the cluster depends on fraction $\left(\frac{n_{access}-1}{n_{access}}\right)$ of cache read accesses compared to the total number of file accesses n_{access}. Comparing the costs for additional SSD caches

($cost_{SSD} \approx 200$€) with additional worker nodes ($cost_{wn} \approx 5000$€) required for compensating the lost CPU hours, caching helps to save about $\left(1 - \frac{1-f_{speed-up}}{f_{speed-up}} \cdot \frac{cost_{SSD}}{cost_{wn}}\right)$ percent of money. Even if we assume an average cache hit rate of 80-100% and only 2 file accesses in total, we get a realistic speed-up factor of $f_{speed-up} = 0.1$, which means that we save about 64% money. Additionally, caching allows for reducing the load for the network and Grid storage infrastructure by accessing data internally within the cluster. This allows us to reduce costs while freeing resources for other WLCG users.

4.2. Boosting shared computing infrastructures

We, additionally, investigated the benefits of coordinated distributed caching for shared computing infrastructures such as cloud and HPC infrastructures. In contrast to TOpAS, we have to face fluctuating performance of shared storage, computing and network resources, which complicates optimizing data throughput due to congestion of resources.

We successfully used resources at NEMO HPC cluster at Freiburg [9, 10] and OpenTelekom cloud [1]. When processing data-intensive workflows on these resources, we observed inefficient CPU utilization caused by limited data transfer rates between the processing hosts and the Grid storage elements [11]. We expect an improvement of the data transfer rate, when placing caching proxy servers directly inside the HPC cluster and the OpenTelekom cloud resources.

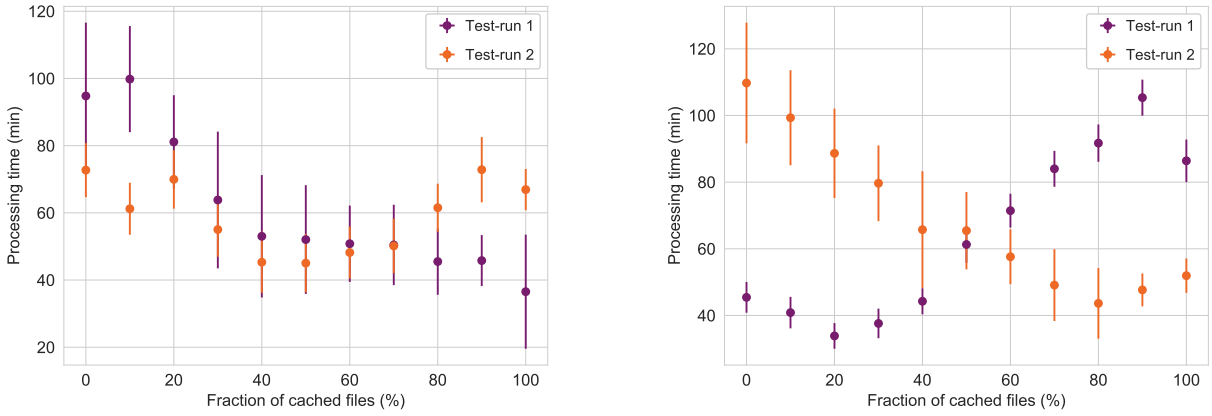

Figure 3. Maximum achievable performance using a cache within the shared NEMO HPC center (left) and the shared OpenTelekom cloud (right). The dots show the mean value of 420 jobs processed in parallel, the error bars represent the variance. [7]

For measuring the performance gain caused by caching, we used test jobs that only read data without further processing. Again, we repeatedly processed the same jobs while increasing the fraction of cached files in steps of 10% to scan for an optimal working point. We repeated the measurement after one week, and observed different behavior for both, NEMO and OpenTelekom. The results of the test runs are shown in Figure 3. Since both are shared resources, the performance of the setup varies over time according to the load caused by other users. At NEMO, the achievable data throughput is influenced by the load on the network and the distributed file system used as cache volume. Especially at OpenTelekom, we observe a different behavior. While the I/O rate of the cache volume was the limiting factor for the first test run, limited network transfers slow down the second test run. Since we have no detailed insight into how OpenTelekom provides storage and network in virtual machines, we can not to determine the exact reasons. We will have to balance performance of network with the cache volume and need to adapt this decision to the evolving conditions at the different systems due to

congestion. Whether an increase in performance can be achieved with caching therefore depends strongly on the resource provider and the utilization of resources. Caching enables the use of resources that provide storage volume suitable for caching purpose available for processing of data-intensive HEP analysis workflows. In particular, HPC centers, in which distributed file systems are usually quickly accessible, can be easily adopted for HEP use.

5. Conclusion

The coordinated distributed caching concept utilizes caches in a distributed computing infrastructure and coordinates workflows to the most suitable host in terms of data locality. We have shown that coordinated distributed caches can improve the overall processing efficiency for data-intensive jobs on specific computing infrastructures.

This concept was realized on the dedicated high throughput system TOpAS. We observed a performance improvement for a data-intensive HEP workflow when accessing data from built-in caches instead of remote Tier 2 WLCG Grid storage resources. Here, high data transfer rate improves the data throughput of jobs and reduces the processing time. By increasing the CPU efficiency of jobs, caching optimizes the overall data throughput of the cluster, and reduces costs. Furthermore, we should also consider freeing resources for other WLCG users by caching as much data as possible.

Additionally, we tested caching at resources not dedicated for HEP usage such as the NEMO HPC center and the OpenTelekom cloud. Making these resources available for processing HEP workflows, allows for adding additional CPU resources and, thus, facing increasing need for computing resources. Here, we observed highly fluctuating performance caused by resources being shared among different users. When adjusting selection of data for caching and the scheduling of jobs to cached data to the computing infrastructure and resource provider, we enable efficient processing of data-intensive HEP workflows.

References

[1] Schnepf M J et al. 2019 Dynamic Integration and Management of Opportunistic Resources for HEP, *Proceedings of the 23rd International Conference on Computing in High Energy and Nuclear Physics* (to be published)

[2] Fischer M et al. 2016 Data Locality via Coordinated Caching for Distributed Processing *Journal of Physics: Conference Series* **62** 012011

[3] Sauter M, Heidecker C, NaviX [software], version 1.0, 2017. Available from `https://gitlab.ekp.kit.edu/ETP-HTC/NaviX` [accessed 2019-05-13]

[4] Thain D et al. 2005 Distributed computing in practice: the Condor experience, *Concurrency and Computation: Practice and Experience* Volume **17** 24 32356

[5] Dorigo A et al. 2005 XROOTD/TXNetFile: A Highly Scalable Architecture for Data Access in the ROOT Environment, *Proceedings of TELE-INFO'05* 46:1–46:6

[6] Heidecker C et al. 2019 Advancing throughput of HEP analysis work-flows using caching concepts, *Proceedings of the 23rd International Conference on Computing in High Energy and Nuclear Physics* (to be published)

[7] Sauter M 2019 Increasing efficiency of HEP workflows by coordinated distributed caching, *Karlsruhe Institute of Technology: Institute of Experimental Particle Physics* (to be published)

[8] Bauerdick L A T et al. 2014 XRootd, disk-based, caching proxy for optimization of data access, data placement and data replication, *Journal of Physics: Conference Series* **513** 4 042044

[9] Meier K et al. 2016 Dynamic provisioning of a HEP computing infrastructure on a shared hybrid HPC system, *Journal of Physics: Conference Series* **762** 012012

[10] Heidecker C et al. 2019 Dynamic Resource Extension for Data Intensive Computing with Specialized Software Environments on HPC systems, *Proceedings of the 5rd bwHPC-Symposium* (to be published)

[11] Schnepf M J et al. 2018 Mastering Opportunistic Computing Resources for HEP, *Journal of Physics: Conference Series* Volume **1085** 032056