HL-LHC computing review: DOMA ACCESS

Community driven document, contributed so far by (please add your name here if it's missing): S. Jezequel, F. Wuerthwein, I. Vukotic, X. Espinal, O. Smirnova, T. Boccali, N. Hartman, (who is Tiramisu Mokka?)

Introduction:

The estimates for the data volumes and computing for the High-Luminosity LHC program (HL-LHC) show a major step up from the current needs and a program of work was established from the WLCG point of view to address this future challenge. In particular storage has been identified as the main challenge for HL-LHC due to the increasing use of disk volumes, and also the costs from the site perspective to operate and maintain complex storage systems.

The past experience has demonstrated that operating a permanent Grid storage and ensuring the data integrity by a local team of site administrators requires a significant manpower overhead dedicated to this activity whatever the storage capacity. Since LHC experiments have given the possibility to access remotely files for specific processing activities, some site administrators or countries have decommissioned their Grid storage to focus on the CPU operation. Reinforcing this trend by extending the possibilities for reliable remote access opens the possibility of a scenario where data can be concentrated in fewer sites and to easily integrate heterogeneous Computing facilities like HPC or Commercial cloud. This also gives the opportunity to orchestrate storages as a single entity with common policies and quality of services (QoS) [add ref/link to QoS doc] via a high level data management system (Rucio) [Add reference], enabling new possibilities for storage resources optimization.

One of the challenges is addressed by the DOMA ACCESS working group that evaluates potential new technologies and scenarios for data access in the analysis workflows still operating at the HL-LHC scale.

Networking is the technology domain where more innovation is expected and might be a good battle horse to offload persistent storage, relying more on remote access and leveraging data with different storage QoS. The main showstopper for remote data access is latency, driven by topological network distance, impacting CPUefficiency. This is the field where the working group has concentrated the efforts and started evaluating streaming caches.

Data caching infrastructures could maximize the potential of a datalake-like model (Fig.1), caches can effectively hide latency and offer file re-usability (the WLCG-datalake is introduced in the next chapter).

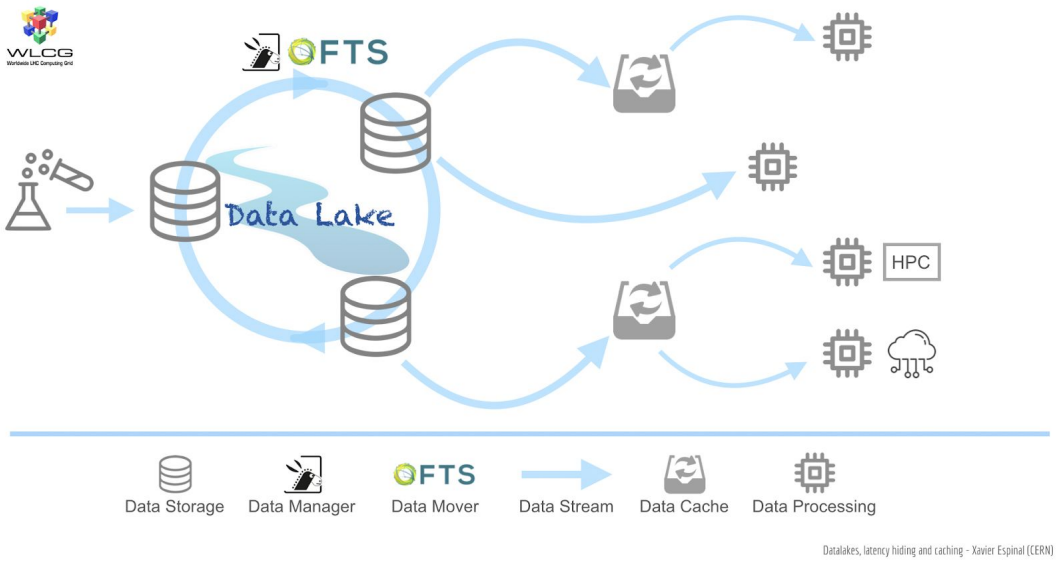

Fig.1: Sketch of caching models leveraging data access from a consolidated storage infrastructure labeled as WLCG-datalake

Future of analysis data: the *compact* datasets

CMS and ATLAS experiments are transitioning towards more compact datasets for analysis, which fit well with the concept of caching infrastructures. These analysis objects, which are expected to be accessed by sizable share of analysis streams, have event sizes in the order of few kB/event which is x50 less than today's miniAOD. Even with the higher trigger data rate, single full analysis datasets will be close to 2/0.5 PB for data/MC per version per year [ATLAS link][ATLAS Link2][CMS link].

The possibility to exploit these new analysis objects to promote less demanding storage to moderately small sites is being evaluated. Sites that do not have special interest in maintaining and operating a full storage system could opt for stateless storage (caches) or storage-less

approach. These sites could channel efforts towards computing resources (CPU, GPU, Machine Learning, etc.).
Engagement from the physics community to make use of this compact analysis datasets is crucial.

Storage consolidation: the WLCG-datalake:

The WLCG-datalake differs from the academic wikipedia's definition [add ref]. The idea is to achieve storage consolidation of the current storage systems at large centers, with the following characteristics:

- Common data orchestration with the possibility to define overall policies for different communities. This is a step forward towards an open and accessible data storage, not only for WLCG but for other sciences.
- Enable the possibility to benefit from different storage media, labelled as Quality of Service (QoS from now on). QoS together with the common orchestration layer allow to implement automated workflows for files, spanning the overall WLCG-datalake. This fosters sustainability and improves reliability (less sensitive to disk/node/rac/site failures).
- The model eventually can bring resource usage improvements. Allowing to leverage QoS and fit the data demands with storage properties (e.g. files remain on low latency media for a period, then transition to a lower cost QoS)
- The WLCG-datalake model enables new possibilities for sites wanting to shift their focus towards a CPU oriented goal. This is possible by introducing content delivery layer (now we are focusing on XCache). In this framework sites could eventually become stateless (running a stateless caching layer for sites that are *far* from the lake) or storage-less (relying on remote I/O and client side caching for sites *close* to the lake). Where here *far* and *close* are network topological terms.

The WLCG-datalake is aggregating a set of current computing infrastructures, providing together storage services to an identified set of user communities capable to carry out independently well defined tasks. Large centers providing storage in WLCG are the natural candidates to be part of the WLCG-datalake. These centers run different storage flavors (dCache, DPM, EOS, Ceph, etc.), the abstraction of the different deployments is achieved by the orchestration layer.

Topology could be defined by geography, connectivity, funding or a shared user community. This requires that their combined storage capacity and bandwidth can meet the demands of the designated task and that usage of the different sites is transparent to the users. This means that some form of trust relationship has to exist and a way to locate data is in place as a high level data management system (Rucio). *For a group of infrastructures which cannot reach sufficient storage capacity (Asia, South America), another organisation to be found.*
While access for users is transparent during job execution, the population and management of the storage within the WLCG-datalake is a planned and managed activity. This includes the transitions between QoS [reference to the QoS WG doc to be added] levels. These operations

are done on the granularity of the WLCG-datalake. Transfers between datalakes are handled by FTS and do not know about the internal datalake organisation.

Data access through caches: streaming, latency hiding and file re-usability

The functionality of a Cache in a data lake scenario is to stream data and provide read-ahead functionality. To minimise the complexity of the maintenance of a cache, its content is not published. For client tools and data management services (i.e. Rucio), it is fully transparent. To maximise the probability to access files already stored on cache, the possibility to restrict the potential caches hosting a file is being evaluated. The role of the local cache is twofold:

- Ability to read ahead to reduce the impact of latency and peak bandwidth requirements for the first reading of the file. In addition, only the relevant part of the file is cached.
    - *Caching is performed at the level of file blocks and not bound to only-root formats, however performances have been validated only for root-format files.*
- Reduce the required wide area network bandwidth by holding frequently used files in the cache.

The size of the cache depends on the data type being used at the site and the type of workloads assigned. The file reuse rate depends on the data type: analysis files are likely to be accessed several times while reconstruction files are single access and eventually could *wipe* the cache.

The optimal size of the cache is given by the frequently reused files and enough storage to serve the running jobs that process files only once. A second optimisation could be done with smart scheduling of tasks with respect to the site characteristics. Preliminary results, based on data popularity studies, indicate that a very modest size, compared to current T2 storage is sufficient (Fig. 2)

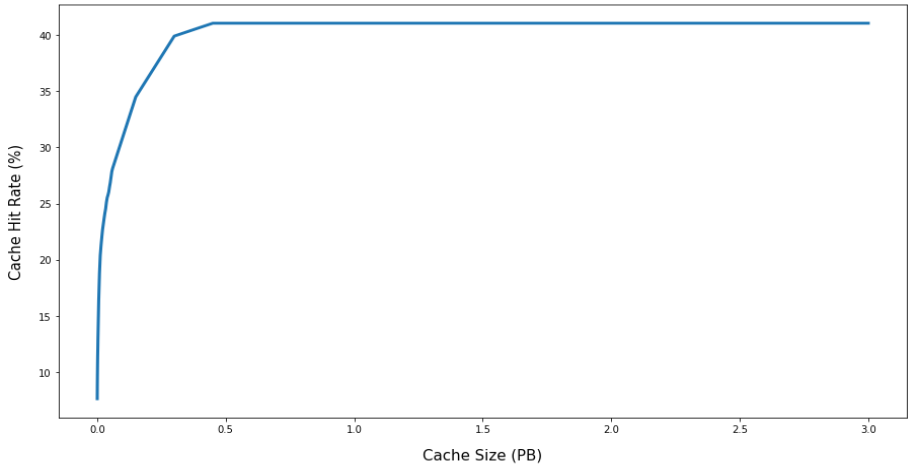

Fig.2: Cache simulation done using popularity data for a 28 day period at a medium sized Tier-2 (6 PB and 30k cores). The cache hit rate hints that 500TB cache (~10% of the total storage) would be enough to serve the local cluster for the considered period and workflows. LRU

algorithm was used in the simulation. Total IOPS: 1.4 M (average of 0.58 OP/s). Total data accessed: 1.5 PB (620MB/s).

These Caching infrastructures can be implemented in several ways. To name some of them:

- As standalone service with dedicated boxes (would need dedicated hardware)
- Cache tiering/federation: layer of caching nodes at the site working synchronously, connected to a second layer of caching (e.g. regional level) serving several sites.
- Distributed over all processing nodes at the site, ie. connected network of caches within the site (no dedicated hardware but fraction of the disk in every WN at the site)

The advantages of these different implementations are being studied for different sites, roles and use cases (next chapter)

Cache activity is driven by client activity, with clients requesting data from a cache. If the requested data is stored locally then the cache will serve the client's request using the cached data. If the requested data is not stored locally then the cache will request the data from the datalake. Existing caching technologies (ie. XCache) provide a rich set of tuning parameters to express preferences regarding the location of the data.

It has to be noted that on large sites with many active clients the I/O demands on cache nodes will be substantial.

Caching infrastructures may not be needed for sites with very low latency to the datalake, in this cases applications can do the read-ahead work, this is known as client buffering and is defined as the ability of the application to manage the I/O in order to achieve an intelligent cache behavior (streaming, read-ahead) in terms of managing latency and bandwidth.

Investments by CMS and ALICE have shown that by employing advanced asynchronous I/O techniques and tools to select and predict those subsets of data that will be used by the application, the throughput/efficiency of the workload can be maintained with losses below 5%, despite significant latency. Most of these techniques are available and integrated in the analysis frameworks that we currently use (ROOT), however care has to be taken to configure them correctly for the situation at hand.

Caches deployment and testing activities

Several studies to evaluate the use of caching infrastructures are conducted. Many initiatives in different countries have deployed and operate XCache technology for ATLAS and CMS. Reference WLCG workloads have been running on these infrastructures showing good ability for latency hiding even when data is read for the first time.

The experiment workflows are already implementing caching at the client level (TTreeCache root configuration). In this case the client *predictively* asks for more bytes (events) than needed

in the first cycle, as a result the next *expected* chunk is in memory and ready for the next cycle of the job.

This *predictive* read has been demonstrated to be very effective to hide latencies up to 10 ms (***source of this number which should be different between ATLAS and CMS ?***), and could be the simplest solution for sites that have good network connections to the data lake. On the other hand the degradation in CPU efficiency is noticeable as latency grows, this is where a caching layer plays a big role for hiding latency.

In WLCG we have more than 160 sites all with different roles and scopes e.g. different experiments, local communities, analysis groups, etc. and with a big variety of network topologies and latencies.

Caching infrastructures offers more flexibility for the sites. An example could be: a relatively small Tier-2 providing storage and computing to WLCG. In the current model they need to maintain and operate a full fledged storage system. In the working group we are studying the possibility to open new possibilities as mentioned before:

- Remote access (storage less site), if the site is close enough to a the datalake
- Caching layer (stateless storage), for latency hiding and eventual file reusability.

In this scope the working group has promoted the deployment of several caching models to operate in a region and on a site level. We are investigating three different approaches:

- High performance caching servers in USA to feed a region Southern California (SoCal) and Chicago
- Caching federation to feed data to regional sites (France and Italy examples)
- Site caching mechanism as stateless Tier-2 storage (Munich and Birmingham)

The results obtained confirm that caching is a promising mechanism to promote efficient usage of the storage.

- **SoCal:** The caching layer setup demonstrated that three sites: Riverside, Caltech and San Diego, can benefit of a common caching layer of 1 PB (c.f. with the old model where the site had to deal with 5 PB of stateful storage installation). This cache can serve 90% of the jobs/user request at 1/5th of the cost in hardware and alleviating the site to manage a complex storage service.

- **LMU:** The initiative demonstrated that an old disk pool node, with simple XCache deployment could serve up to 3k jobs of ATLAS production workflows reading data from the neighbour site in Munich (MPP). Operating with storage based on JBOD instead of RAID6 proved to make the server more robust for heavy load.
  Figure 2 shows a test with jobs reading from a closeby site in Hamburg (DESY) and from a far site in China (IHEP in Beijing). Caching via the XCache server showed to be able to hide more latency compared to caching on the worker node level (both with TTreeCache

and an xcache instance on the worker node) and could provide a similar running time for closeby and far remote reading already for the first time a file was accessed.

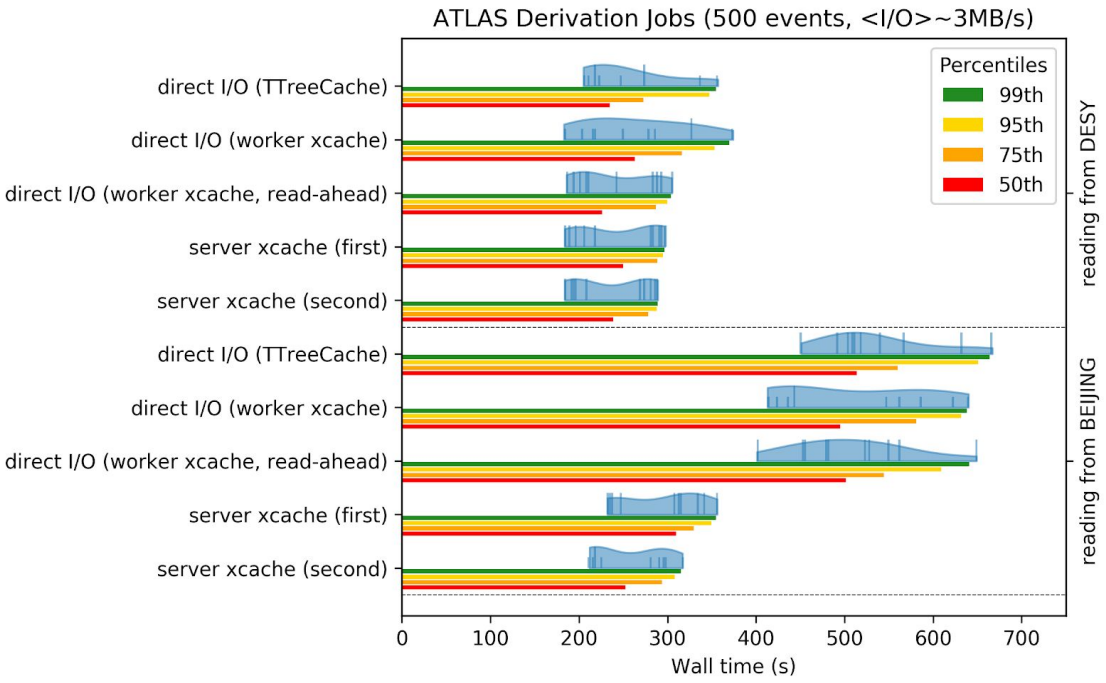

Fig2.

- **UK (BHAM) (input from Stéphane to be validated by ATLAS UK):** The Tier2 located at the University of Birmingham is an example of a Tier2 site which, in agreement with ATLAS recommendations, decided to simplify its operation by decommissioning its ATLAS storage and replace it with the storage located in Manchester. Although the small CPU capacity is too small to host analysis queue, an xcache layer used for production jobs was installed to validate the model that xcache infrastructure can be easily installed and maintained.

  After one year of operation, the site administrator confirmed the simplicity and reliability of this system. In 2019, the long term follow up was mainly done with simulation jobs where input files are read by 5 to 10 jobs. Over the last 3 months, 80% of the accessed capacity (20 TB total) was already available on the cache.

- **RU (PNPI) :**

  The site has evaluated the interest for xcache layer with CPUs located in PNPI (St Petersburg) with files located in JINR-Dubna

- **Italy (INFN):** The caching layer at INFN, deployed across 3 Italian CMS computing sites (namely Legnaro and Bari Tier2s and CNAF Tier1), has been used to serve input data for user analysis running on Italian Tier2s since mid 2018. The current setup is composed by a single cache redirector at CNAF that federates 3 the XCache servers:
  - CNAF with attached spinning disks for 5 TB and 64GB RAM, 10Gbps

- Bari and Legnaro with 10TB and 22TB over gpfs file system and 64GB RAM, 10Gbps

The setup demonstrated some improvements during the whole period that can be summarized as follow:

- the CPU loss reduced by around 10% for the first data access with cache with respect to direct remote read. Thanks to read-ahead capability (only <7% worse than local read in terms of CPU efficiency).
- within one month, around 40% of the data are usually accessed again, in which case the gain raise up to a 20%, similar to local read mode
- usage of "no replica" File Systems is possible for non custodial data as the cache servers. Thus at least a factor 2 in space available with respect to the actual setup (depending on the File System configuration used).
- Reduction of the overall WAN traffic and a correspondent cost efficient reduction of I/O wait time for remote data

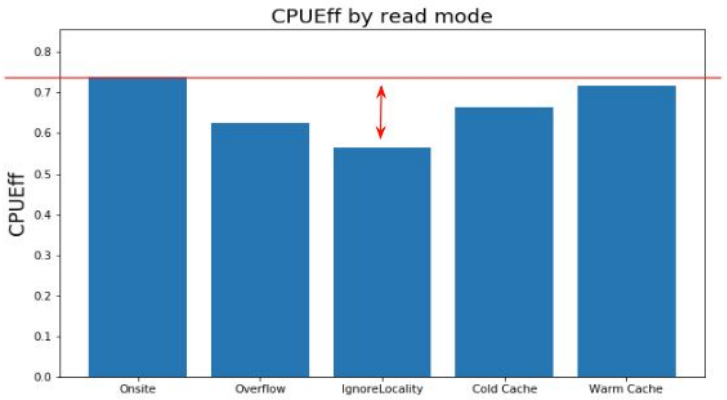

Fig3.

XCache: collaboration and projects

Rapid adoption of Xcache in the LHC community unsurprisingly brought up issues that we don't see in the testing environment. A few of them are deep in the software stack that request attention from experts. There are also long term works which we will not see immediate impact to the community, but are badly needed in order to lay the groundwork for the future. For this reason, CERN, SLAC and UCSD agreed to contribute voluntary efforts to work together and address those long term needs. So far we identified areas in data integrity and checksum algorithm, HTTP based cache, TLS in Xroot protocol as the initial projects for investigation. Given the voluntary nature of this loose collaboration, we do not set deadlines on these projects but we are paying close attention to make sure this collaboration stays alive and delivers. We will continue to identify other long term areas and define works accordingly.

Native caching with dCache

Many Tier-1 and Tier-2 sites use dCache as their storage system. The flexible dCache design allows to size their deployments to match site requirements - from a tiny single node setup, up to multi-petabyte installations, like BNL or KIT, or even geographically distributed deployments, like AGLT2 or NDGF. Though AGLT2 and NDGF operate over a decade, such distributed setups are exceptions. This is going to change with a growing demand of datalakes.

Caching and data placement was always the foundation of dCache design. However, to get operational distributed deployments a special configuration was needed. Moever, addition and removal of *cache* nodes on a remote site required operation intervention at the main site as well. Thus, such distributed setups more or less a static deployments, which is orthogonal datalake model expects.

The recent developments in dCache have introduced a concept of a *Zone.* A zone is a virtual container that collects multiple dCache services into a logical group. Those virtual groups are available in data placement and replication rules as well as for dCache inter-component communication. As those grouping rules can be defined to be dynamic, the *caching* nodes can be dynamically added and removed from the system without manual operation on the main site. Moreover, for internal data movement the standard dCache *pool-to-pool* copy is used, which provides data protection and integrity mechanisms, like checksumming or TLS.

With a fine grained configuration, an automatic data replication can be triggered from one zone into another one when data is located in a zone remote for the clients. Such data replication can be triggered for some selected protocols and allow direct access for the others to separate streaming read by HTTP and FTP from latency secetive random access by NFS or XrootD.

In a combination with a read-only namespace replica at the *caching site* a full access to the data can be provided when connection to the main site is lost without compromising authorization and data integrity.

This model can help smaller Tier-2s to become an integral part of their Tier-1s, but benefit from local expertise and technology used in place.

File usability and data access patterns

One of the key parameters to assess how effective we are in using storage is to measure the access frequency after data placement.

There are two extremes regarding data *thermodynamics*:

- cold data (aka WORN: Write Once Read Never) and

- hot data (aka WORM: Write Once Read Many)

But after studying data access patterns at several sites we observed that a large fraction of our files are neither cold nor hot. The analysis objects files seem to lose popularity with time and the access rate decreases significantly after days/weeks. The plots below show the file access rates and file popularity on a Tier-1 and a Tier-2 as a function of time

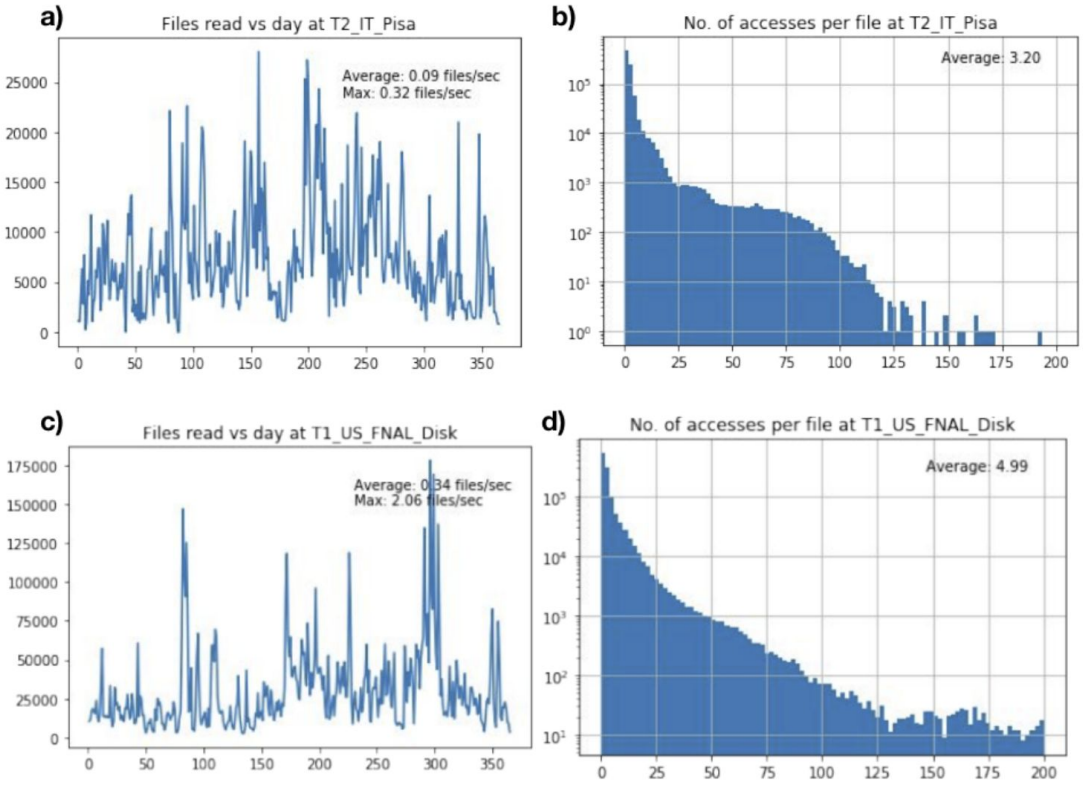

Fig.5 File popularity on a Tier-1 and a Tier-2 as a function of time (300 days).

The plots indicate that data is not accessed very often, it is most likely to be re-read within days after placement then the access drops substantially, almost two orders of magnitude.
This provides an indication whether this type of data could be better handled with caches; available when popular and replaced with newer files once the demand decreases.
This way the space on disk at the computing sites is optimised for data being actively used.
We also observed a fundamental difference between analysis and production data: analysis has higher re-use while production files have very few re-reads.
The net effect is that running the two workflows together at the same site has the effect of pushing analysis data out of the storage/cache, and these are the files with higher likelihood to be reused.
Changing the current model and favoring running predictable workflows at the sites with limited storage and schedule user analysis at the sites with larger storage might be also a solution.

Events As A Service

The proposed implementation of an Event Streaming Service relies on the existence of an edge service, running closed/attached/as part of the storage capable to prepare “collections of events” to be served to a client that would process them sequentially.
The advantage of this approach with respect the random access of events from multiple (or even the same) files is that the events are picked and prepared asynchronously with respect of the job and can be served to the job itself reducing the I/O latency.
In an ideal case, the experiments would not need to duplicate data after skimming, but they would simply define the skimmed event collections in terms of event metadata (file IDs and event IDs) and access them directly through this events service. A possible R&D in this would consist of:

- Consider an existing storage implementation and deploy in front of it a ROOT-aware capable of:
    - Receiving a list of (fileID, event ID) from a client
    - Read the events from the backend storage through a local protocol and organise them in event collections (files containing events) accessible via ROOT
    - Return the information to the client (fileID, eventID, event collection ID)

- Read the event collection with a ROOT client and process it. Compare the main metrics (event throughput, server I/O) with the case the events are processed with random access
- Consider a similar implementation where E2AS is implemented as an object store. The object store would not organise the events in files, but rather serve them as objects. Perform the same measurements as in 2)

There are several initiatives being evaluated right now: ServiceX (ATLAS) and Coffea (CMS) (... need input here… status summary of these activities)