PROCEEDINGS OF SCIENCE

Integration of the Italian cache federation within the CMS computing model

Diego Ciangottini*
INFN, Sez. Perugia, Italy
E-mail: diego.ciangottini@pg.infn.it

Giuseppe Bagliesi
INFN, Sez. Pisa, Italy
E-mail: giuseppe.bagliesi@pi.infn.it

Massimo Biasotto
INFN, Sez. Legnaro, Italy
E-mail: massimo.biasotto@lnl.infn.it

Tommaso Boccali
INFN, Sez. Pisa, Italy
E-mail: tommaso.boccali@cern.ch

Daniele Cesini
INFN, Sez. Bologna, Italy
E-mail: daniele.cesini@cnaf.infn.it

Giacinto Donvito
INFN, Sez. Bari, Italy
E-mail: giacinto.donvito@ba.infn.it

Antonio Falabella
INFN, Sez. Bologna, Italy
E-mail: antonio.falabella@cnaf.infn.it

Enrico Mazzoni
INFN, Sez. Pisa, Italy
E-mail: enrico.mazzoni@pi.infn.it

Daniele Spiga
INFN, Sez. Perugia, Italy
E-mail: daniele.spiga@pg.infn.it

Mirco Tracolli
INFN, Sez. Perugia, Italy
E-mail: mirco.tracolli@pg.infn.it

PoS(ISGC2019)014

The next decades at HL-LHC will be characterized by a huge increase of both storage and computing requirements (between one and two orders of magnitude). Moreover we foresee a shift on resources provisioning towards the exploitation of dynamic (on private or public cloud and HPC facilities) solutions. In this scenario the computing model of the CMS experiment is pushed towards an evolution for the optimization of the amount of space that is managed centrally and the CPU efficiency of the jobs that run on "storage-less" resources. In particular the computing resources of the "Tier2" sites layer, for the most part, can be instrumented to read data from a geographically distributed cache storage based on unmanaged resources, reducing, in this way, the operational efforts by a large fraction and generating additional flexibility.
The objective of this contribution is to present the first implementation of an INFN federation of cache servers, developed also in collaboration with the eXtreme Data Cloud EU project. The CNAF Tier-1 plus Bari and Legnaro Tier-2s provide unmanaged storages which have been organized under a common namespace. This distributed cache federation has been seamlessly integrated in the CMS computing infrastructure, while the technical implementation of this solution is based on XRootD, largely adopted in the CMS computing model under the "Anydata, Anytime, Anywhere project" (AAA).
The results in terms of CMS workflows performances will be shown. In addition a complete simulation of the effects of the described model under several scenarios, including dynamic hybrid cloud resource provisioning, will be discussed. Finally a plan for the upgrade of such a prototype towards a stable INFN setup seamlessly integrated with production CMS computing infrastructure will be discussed.

International Symposium on Grids Clouds 2019, ISGC2019
31st March - 5th April, 2019
Academia Sinica, Taipei, Taiwan

*Speaker.

1. Introduction

In the current computing model of the CMS [1] experiment at LHC [2], the recorded and simulated data are distributed and replicated via a central management system across a worldwide network of custodial storage sites. There are different typologies of datasets created for different purposes that have to be managed with different policies. For instance, the most frequently used samples might be kept on disk ready to be accessed with the lowest latency possible, while custodial data and ,more in general, legacy data that are rarely accessed can be stored on tape facilities. In this model the computing payloads are sent and distributed across the Grid sites based on the input data location (preplacement) in order to reduce inefficiencies from transfer latencies. Nevertheless there are few cases where CMS payloads run at one site while reading data from a remote location:

- *fallback* mechanism for which, at certain sites, if the local storage is for some reason unable to serve the requested file, the request is redirected to the remote XRootD CMS federation.
- *overflow* mechanism that consists in steering jobs assigned to a busy site towards a near one with idle slots reading remotely from the original tier.

This model is more focused on coherence and data availability guarantee than on operational and disk space optimization. On the other hand, a centrally managed approach can be intrinsically disk inefficient and slow on adaptation for the pace at which the user needs may change. A huge increase of both storage and computing requirements are foreseen for the HL-LHC era (between one and two orders of magnitude) and new kind of resources are covering a crescent amount of needs (e.g. private or public cloud and HPC facilities). As a consequence, the CMS experiment is looking for the optimization of the centrally managed space and the CPU efficiency of the jobs that will eventually run on "storage-less" resources. In particular "Tier2" sites, for the most part, can be instrumented to read data from a remote source, eventually enabling the use of a geographically distributed cache storage based on unmanaged resources. Consequently a reduction of the operational efforts for maintaining managed custodial storage and an increase in the flexibility of the system is expected. Among several R&D activities willing to explore new approaches and models for data access and data organization, one is the exploitation of data caches. The cache system will appear as distributed and shared file system populated with the most requested data; in case of missing information data access will fallback to the remote access. Already in the current implementation, a fraction of accessed data is read from a remote storage and this is a first motivation for this activity. Moreover in a possible future scenario where a data-lake model would be implemented, a protection layer against peaks of request to the centrally managed storage might be a key factor along with the control on data access latency. The cache storage used for such a layer will be by definition "non-custodial", thus reducing the overall operational costs.

The objective of this contribution is to present the first integration experience of an INFN federation of cache servers spanning over the CNAF Tier-1, Bari and Legnaro Tier-2s that provided unmanaged storage organized under a common namespace with the use of XCache technology. The technical implementation of this solution is based on XRootD [3], largely adopted in the CMS computing model under the "Anydata, Anytime, Anywhere project" (AAA) [4]. First studies on

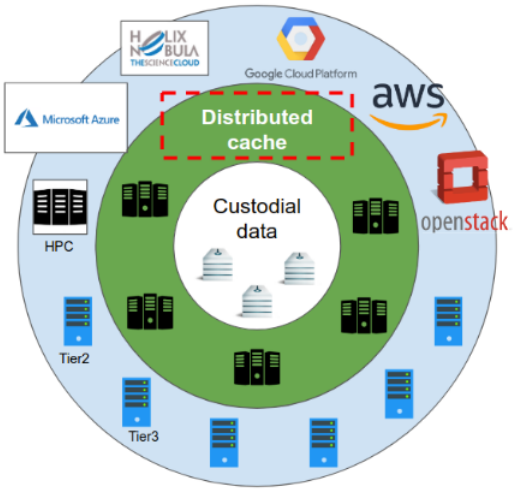

Figure 1: Representation of a data lake implementation with a distributed cache layer. Few worldwide sites have replicated data and both ordinary computing sites and dynamic resources access those data through a network of geo-distributed cache servers.

CMS monitoring data regarding the analysis jobs access pattern over Italian Tier2's will be presented, leading to an estimation of the possible improvements provided by the introduction of the proposed solution at production scale for the Italian Tier2's. In addition the first measurements in terms of performances on real CMS workflows will be shown. Finally a brief description of a Proof-of-Concept deployment of a "smart decision service" platform will be given. XCache server has been instrumented to interact with the service enabling caching management through ML-based algorithms.

2. CMS data access patterns

Several metadata of the CMS jobs [5] are collected and stored on a dedicated monitoring system. For the purpose of this work, data relative to the information of the input datasets requested by each job have been analyzed for the whole 2018. Data format considered for the following results are the most common ones requested by the analysis workflows, namely MINIAOD (recorded data) and MINIAODSIM (simulation). Only user jobs requiring datasets in these formats and running in one of the italian Tier2's have been analyzed and two main categories have been created based on the data access pattern. In fact, as previously described, payloads can read data either directly from the storage at the site where they are running or remotely from another site on the grid. Therefore these two categories go under the label of "Local" and "Remote" in the results that will be presented. In addition to normal operational fallback procedures, remote access can also be triggered manually by users. A typical use case is a user that needs some data that are on a "busy" site (few free running slots) and he wants to run at his local Tier2-3 computing resources. The metrics evaluated in this work are the following:

- sum of walltime: sum of all the running times considering the jobs running of the previous month
- sum of CPU time: sum of all the CPU times considering the jobs running of the previous month
- CPU efficiency: the ratio of "sum of CPU time"/"sum of walltime"
- number of hits: number of unique user workflows requiring a dataset

These metrics have been evaluated for each day over a "moving window" spanning over the previous month; in other words, each day on the following plots collects the information of all the jobs running in the month before. This approach allowed us to get a view of the overall behaviour in terms of access patterns on a reasonable time scale, without suffering for daily fluctuation that, when it comes to user driven activities, might be very significant. Fig. 2 shows the comparison of the CPU efficiency for jobs reading data locally with respect the remote mode. Local deep aside, where few big user tasks are running on particularly bad links, the average CPU loss in remote mode is around 15%. This information has to be combined with the total amount of CPU hours each reading mode covers. In Fig. 3 the amount of running time in remote mode is on average 2.5E9 seconds per month (equivalent to around 1k core/month), which is around 30% of the local one.

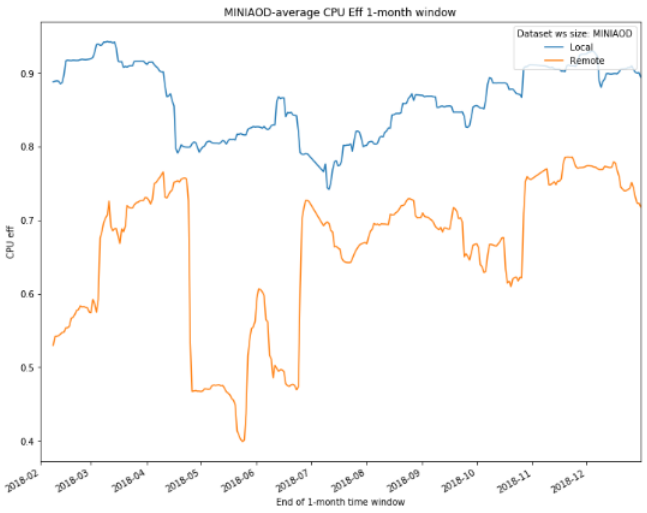

Figure 2: Comparison of the CPU efficiency for jobs reading data locally with respect to the remote mode

In Fig. 4 the total amount of MINIAOD data stored at the Italian Tier2's compared to the amount of data requested by user jobs over one month period is shown. There is a clear feature here, where the requested data size is consistently below the stored data available at the moment by a factor of around 20%. In addition in Fig. 5-2 the results of the studies on data hit rates are shown. These results show how a vast majority of the running time is spent reading data that are accessed by more than one user workflow over a month and, in terms of volume, around two thirds of the requested data are likely requested again in the same time window.

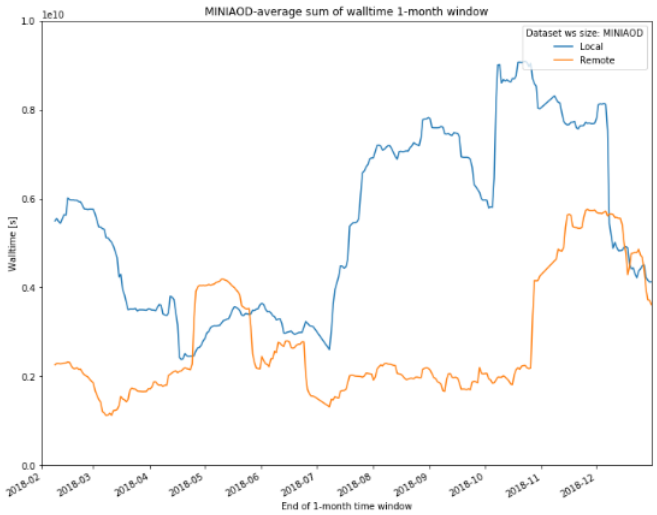

Figure 3: Amount of running time in local mode compared to remote

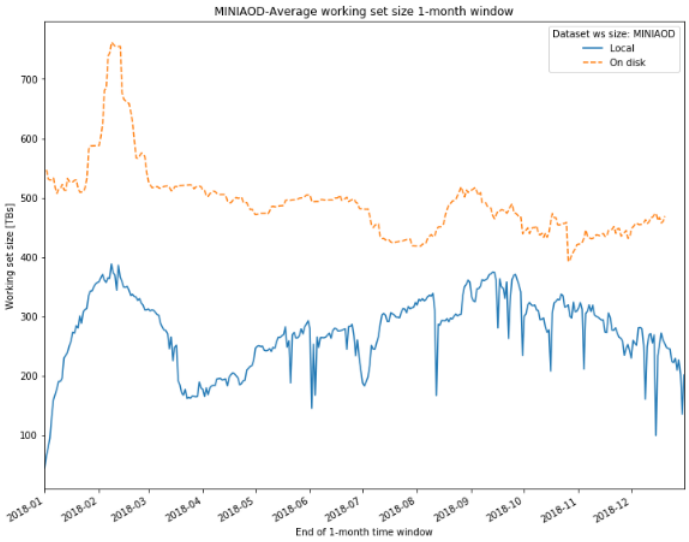

Figure 4: Total amount of data stored compared to the amount of data requested by user jobs over one month period

3. XCache geo-distributed CMS federation: Italian testbed

In a scenario as the one described by the results above, we investigated the possibility of seamlessly integrating a geo-distributed cluster of cache servers in the CMS workflow, leveraging the national high-bandwidth network to optimize the amount of disk space for user analysis inputs and, in addition, to provide good performance in terms of CPU efficiency for user payloads. A proof of concept has been deployed using resources provided by 3 Italian computing sites (namely Legnaro and Bari Tier2s and CNAF). The setup (schematically summarized on Fig. 7) consists in creating a cache federation thanks to the XCache technology [6] where all the 3 servers register

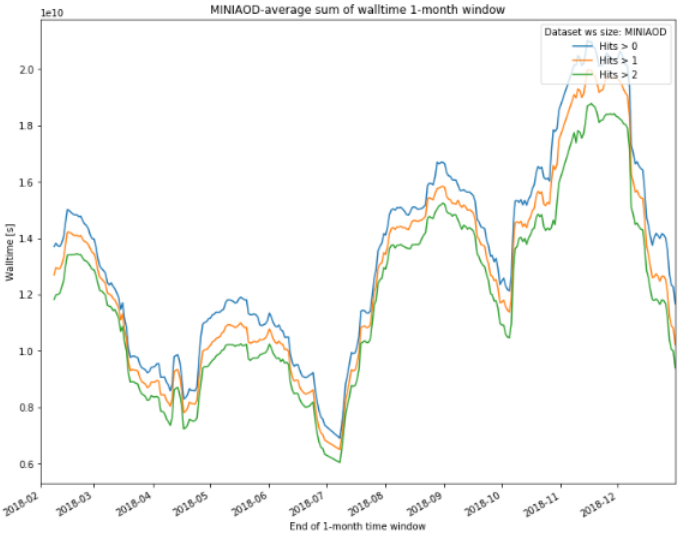

Figure 5: Time spent by jobs requesting data accessed grouped by number of unique user workflows over one month: at least one (blue), at least two (yellow) or three (green).

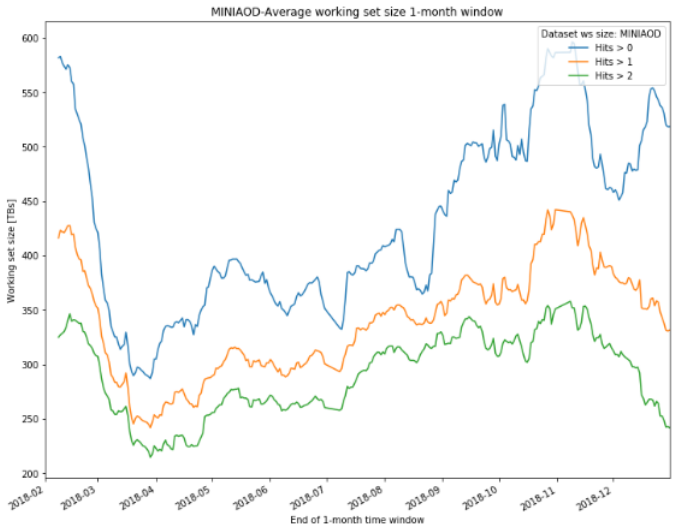

Figure 6: Size of the requested data grouped by number of unique user workflows over one month: at least one (blue), at least two (yellow) or three (green).

themselves to a cache redirector that is responsible of steering the client request and, in this sense, load balancing the cluster. Since the servers mediate the requests for a file that is available on the current CMS XRootD remote federation, the setup is also meant to be propaedeutic for studies on the effect of a cache layer in a data-lake scenario where data are stored in few remote custodial sites.

We performed tests using real user workflows that consists in a data reduction for original format to a series of tuples filtered by analysis needs. This is a good benchmark since it represents a significant portion of user use cases. Input data for the workflow under study are stored at DESY

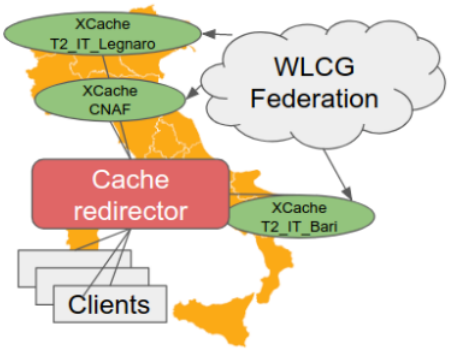

Figure 7: Schematic representation of the INFN distributed cache prototype

Scenario	Time wasted	CPU Efficiency
Remote with No Cache	7%	78%
Cold Cache	3%	87%
Warm Cache	2%	92%
Local with No Cache	2%	94%

Table 1: Time wasted and CPU efficiency of real user analysis workflow in different

Tier-2. We measured performances for the same payload under different scenarios:

- Remote with No Cache: run payloads at available Italian Tier2s while reading input data remotely from original site.
- Cold Cache: run payloads at available Italian Tier2s and seamlessly use the regional cache cluster for input data. Starting from an empty cache, so testing performances of proxy while caching mode.
- Warm Cache: after Cold Cache test, re-run the same workflow to test performances when data already in cache are requested.
- Local with No Cache: run payloads where data are stored (DESY in this case).

In Tab. 1 you can see the results of these measurement showing the time wasted on jobs that failed and that were eventually re-tried along with the CPU efficiency calculated as the total job running time over the total CPU time.

The results show how at the first request, when the cache servers do not have the input data cached on disk, the performances in terms of CPU efficiency are already 10% better than the case in which the remote access is done directly from the computing node to the original site. This is possible thanks to the latency hiding effect introduced by the cache read-ahead capability. In fact when a file not present on disk is requested, the server starts to behave as a proxy providing the requested block of the file while reading ahead the successive blocks in separate streams. When the same file is requested again (as in the Warm Cache case) the performances improve by an additional 5% moving near the scenario in which files are read as they were in the local storage of

the site where the job runs. Combining this result and with the historical metrics analysis shown on the previous chapter, it is possible to depict a first expected scenario for the presented setup in production:

- 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).

3.1 XCache automated deployment on cloud resources

The presented activity involved also the creation of automatic procedures to enable a fast and effortless deployment of cache clusters in a cloud environment. In fact dynamic resource provision is a crescent use case that by construction requires a remote read mode for the jobs. So providing an easy deployment of a scalable cache system near the computing nodes is an important tool to increase, with a minimal effort, the efficiency of jobs running on this kind of resources.

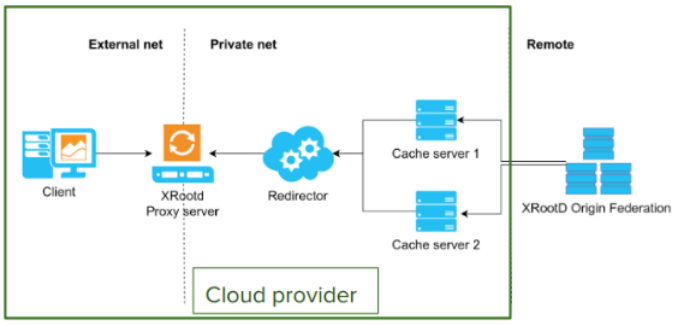

Figure 8: Schema of the components deployed for using a caching on-demand system on cloud resources

The setup infrastructure is shown in Fig. 8, where the clients that run the payload can be instructed to request data to a cache system deployed on the same cloud provider and thus with low latency. The cache stack consists in: a proxy server to function as bridge between the private network of the cache and the client. This server will simply tunnel the request from cache servers. a cache redirector for federating each cache server deployed. If a new server is added, it will be automatically configured to contact this redirector for registration a configurable number of cache servers, the core of the tool that are responsible for reading-ahead from remote site while caching. This setup has been tested on different cloud providers. It is also been tested at a scale of 2k concurrent jobs on Open Telekom Cloud resources in the context of HelixNebulaScience Cloud [7] project. In the context of the eXtreme Data-Cloud project [8], a collection of recipes have been produced for the automatic deployment of a cache service on demand using different automation

technology. For bare metal installation an Ansible [9] playbook is available that can deploy either directly on host or through docker container the whole stack. For those who use docker swarm for container orchestration, a docker-compose [10] recipe is also available as for Kubernetes where an Helm [11] chart is provided. All these solutions have been integrated in DODAS [12] and thus with very few changes the same setup can be automatically replicated in different kind of cloud resources.

4. Infrastructure setup for a Smart Cache Decision Service

The integration of the XCache servers with a ML-based external service to further optimize performances and disk space has been investigated. In particular recipes for the infrastructure deployment have been created and functionally tested to enable studies on different models. In Fig. 9 an overview of the components is shown, the infrastructure setup is possible with a simple configuration file describing the system for different cloud provider leveraging the DODAS [12] enabling technology. The work flow starts from the pre-processing of the input data, filtered and prepared for the ML training part. The data processed are used to increment or modify the current model implemented and this will involve further requests. The current input to the environment is submitted also to the inference service that responds with a suggestion for the cache. The output from the inference service is immediately used by the cache to produce a result and a response to the client request.

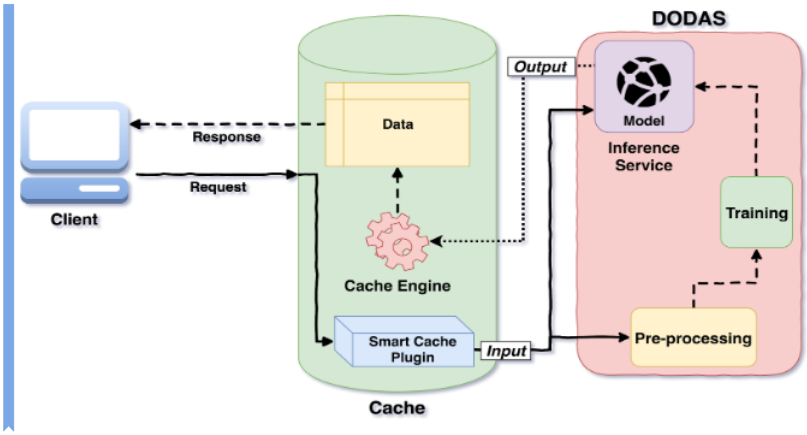

Figure 9: Complete infrastructure overview for the integration of a smart decision service for cache management.

The principal building blocks of the proposed pre-processing stack are Apache Spark [13], and Apache Hadoop [14] that are used, respectively, to process and manage data. The idea behind the architecture is that the configuration has to be as much modular as possible in order to allow the interaction with external and already existing data sources and pipelines when needed. The components provided in the stack are the following:

- data ingestion: sequence of modules responsible for collecting and manipulating metrics from a local or remote archive

- stream ingestion: sequence of modules responsible for collecting and manipulating metrics from streaming pipelines (e.g. message queues, sensors etc.)
- data manager: user custom modules to aggregate information coming from ingestion services. They can be either cron or spark jobs
- pre-processing: jupyter notebooks for the production of pre-processed dataframes, ready to be served as input to train ML models
- processed data store: local storage instance where pre-processed data are stored
- training notebooks: jupyter notebooks for the training and saving the trained models

The components are focused on providing on-demand services for training and exposing inference to different services, in this case the cache servers. The integrated deployment of a jupyterhub [15] allows the end users to interact with the ML frameworks through python notebooks. With such notebooks it is possible to create a model using the most popular ML framework python APIs and test their performances. It is also possible to define custom data pre-processing steps there without any further requirements. The Spark framework is integrated through the notebook and a context is already created for the users. So far TensorFlow [16] and Keras [17] frameworks are natively installed, others can be added with ease though. The output models can then be saved and uploaded on the TensorFlow as a Service (TFaaS) provided. The TFaaS [18] is a piece of the service offered by the cluster. It manages the models loaded into it and it performs the inference for the end users. The service has an HTTP API interface that is also exposed to external sources. That is the endpoint of the request from the XCache decision plugin, enabling the cache to use an external source to make smarter choices, delegating them to a ML based model. An end-to-end prototype has been tested on INFN cloud resources and it will be used as testbed for different model comparison in the near future.

5. Conclusions

In this work the motivations for integrating a cache system for the CMS computing model have been presented along with an estimation of the possible benefits. An INFN prototype is in place and seamlessly integrated with CMS workflows, it will also be used as a testbed for RD activities that aims to optimize the cache content management base on ML-base models. Future plans include the transition of the current prototype towards a production-like grade for a better understanding of the needs for a WLCG data-lake. At the same time a systematic evaluation of the performance of different ML models for the cache decision is in the pipeline, along with the creation of a common interface between the cache servers and the model inference endpoint.

6. Acknowledgments

The authors would like to thank the European Commission's Horizon 2020 research and innovation programme for financial support, under grant agreement RIA 777367.

References

[1] S. Chatrchyan et al. “The CMS Experiment at the CERN LHC”. In: JINST 3 (2008), S08004. DOI: 10.1088/1748-0221/3/08/S08004

[2] Lyndon Evans and Philip Bryant. “LHC Machine”. In: Journal of Instrumentation 3.08 (2008), S08001

[3] https://xrootd.slac.stanford.edu

[4] Kenneth Bloom et al. “Any Data, Any Time, Anywhere: Global Data Access for Science”, arXiv:1508.01443 [physics.comp-ph]

[5] V. Kuznetsov, “Gaining insight from large data volumes with ease”, arXiv:1811.04785 [physics.data-an]

[6] XRootd, disk-based, caching proxy for optimization of data access, data placement and data replication, CMS collaboration, J.Phys.Conf.Ser. 513 (2014) 042044

[7] http://www.helix-nebula.eu/

[8] http://www.extreme-datacloud.eu/

[9] https://www.ansible.com/

[10] https://docs.docker.com/compose/

[11] https://helm.sh/

[12] D. Spiga et al. “DODAS: How to effectively exploit heterogeneous clouds for scientific computations”, PoS(ISGC 2018 FCDD)024, DOI: https://doi.org/10.22323/1.327.0024

[13] https://spark.apache.org/

[14] https://hadoop.apache.org/

[15] https://jupyter.org/hub

[16] https://www.tensorflow.org/

[17] https://keras.io/

[18] https://github.com/vkuznet/TFaaS