

CMS experience of caches in production

XRootD Workshop 2019

[D. Ciangottini*](#)

T. Boccali

D. Spiga

M. Tracolli

for the INFN Cache working group

J. Balcas

E. M. Fajardo Hernandez

F. Wuerthwein

M. Zvada

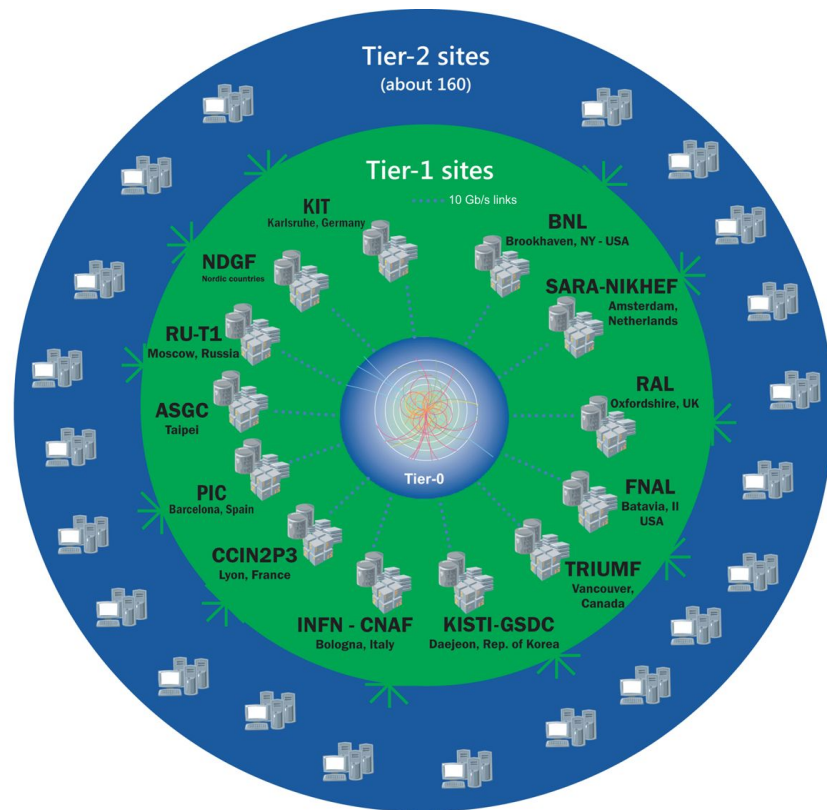
for the SoCal Cache working group

Outline

- CMS analysis data access patterns
- Status XCache @US - SoCal
 - status
 - problems and plans
- Status of INFN Xcache cluster
 - status
 - plans
- Conclusions

CMS current model in a nutshell

- Hierarchical **centrally managed storages at computing sites** (Tier)
- Payloads **run at the site that stores** the requested data
- **Remote data access** already technically supported
 - fallback to remote in case of local read failure
 - overflow of jobs to near sites
 - users that manually ask to ignore the locality of the data



Data access metrics description

Want to maximize CPU utilization:
 $\text{efficiency} = (\text{CPU time}) / (\text{wallclock time})$

N.B. ideally, the two would be identical
+ jobs that fail impact overall efficiency

- **CMS user analysis jobs** in the whole 2018
- HTCondor ClassAds
- Studies focused on job wall and CPU time and CPUEfficiency
- Comparison of scenario with reading at the local site (“LOCAL”) and different type of remote read in CMS (Overflow and IgnoreLocality)

If the site storing the data is busy then the jobs are sent to a near and less busy one

User can configure their jobs to ignore the data location and for them to run on a custom list of sites

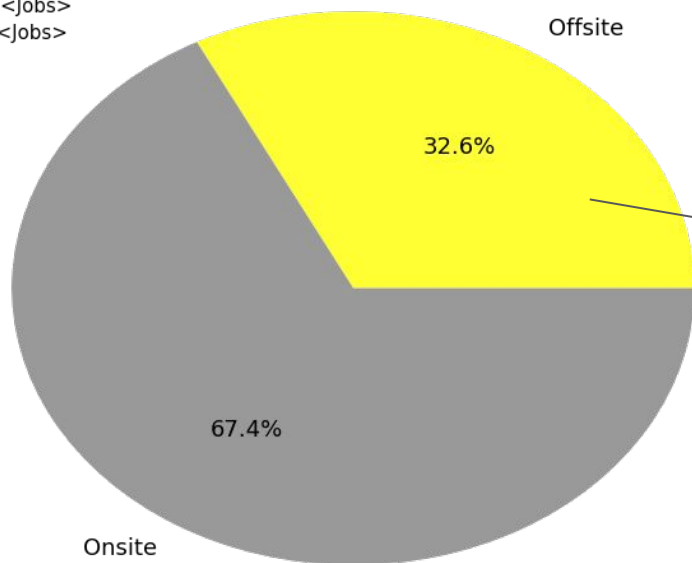
CMS data access: 2018 remote read fraction

WallHours/YearHours

Onsite: 19563.6 <Jobs>

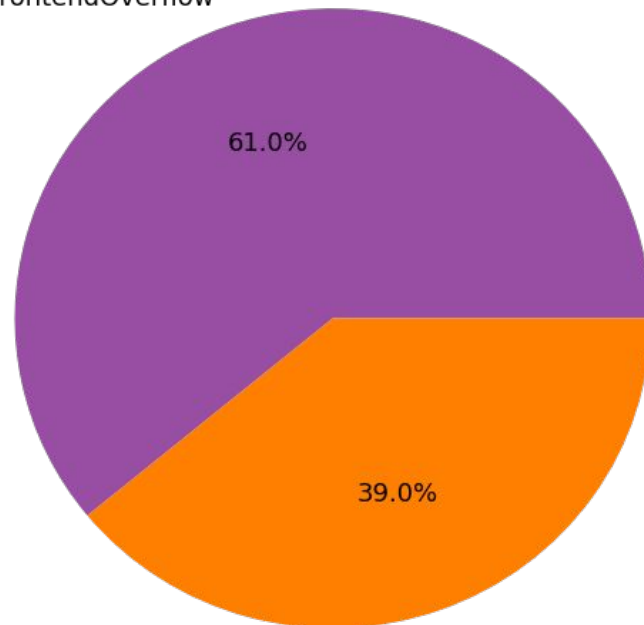
Offsite: 9451.2 <Jobs>

WallClock time by read mode



Remote read wallClock time by overflow type

FrontendOverflow



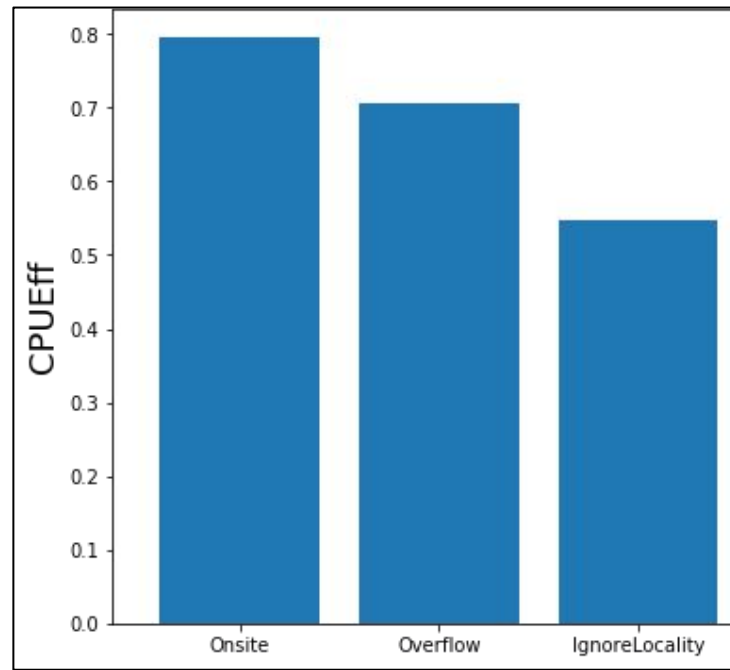
IgnoreLocality

CMS data access: 2018 CPUEff by read mode

- On average significant loss of CPU efficiency passing from local read to remote one
- Still, for now OK since the fraction of remote read time is $\sim 1/3$ of the total
 - considering the beneficial effect of reducing the amount of file replicas around
- Increasing the fraction of remote read it is what we expect in the future though

Evaluating a caching solution:

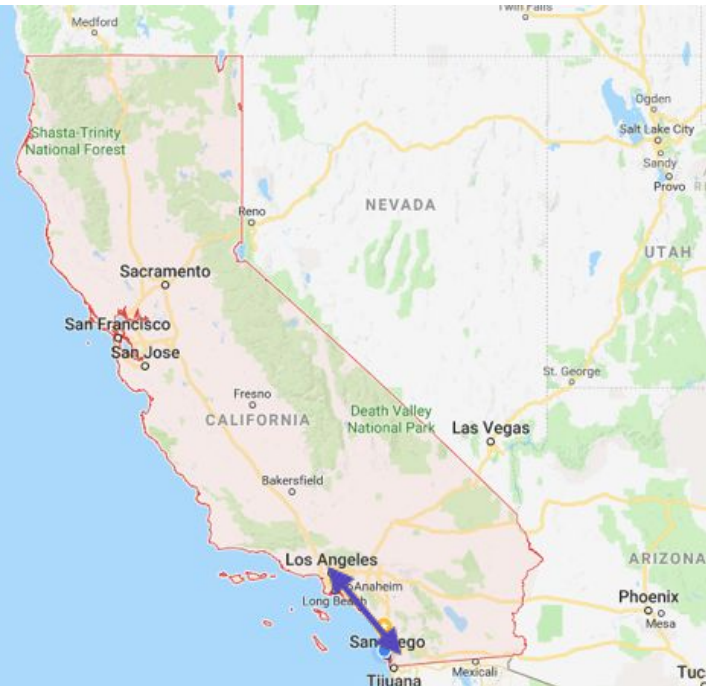
SoCal and INFN have two working XCache “federation” to investigate the functionalities and the performance





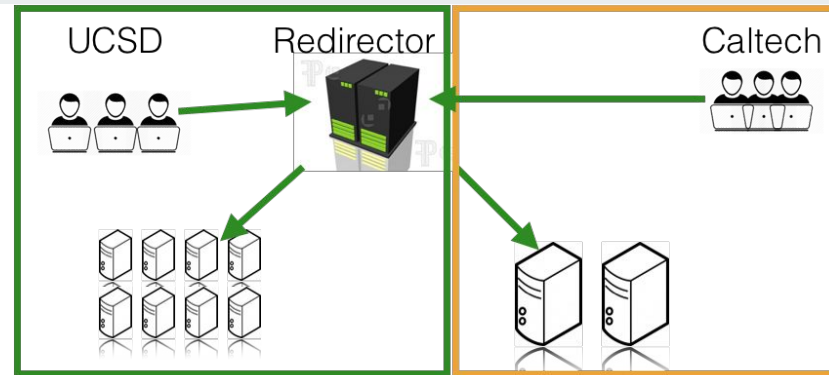
SoCal Cache idea: Two caches become one

- 120 Miles
- 100 Gbit/sec
- 3ms



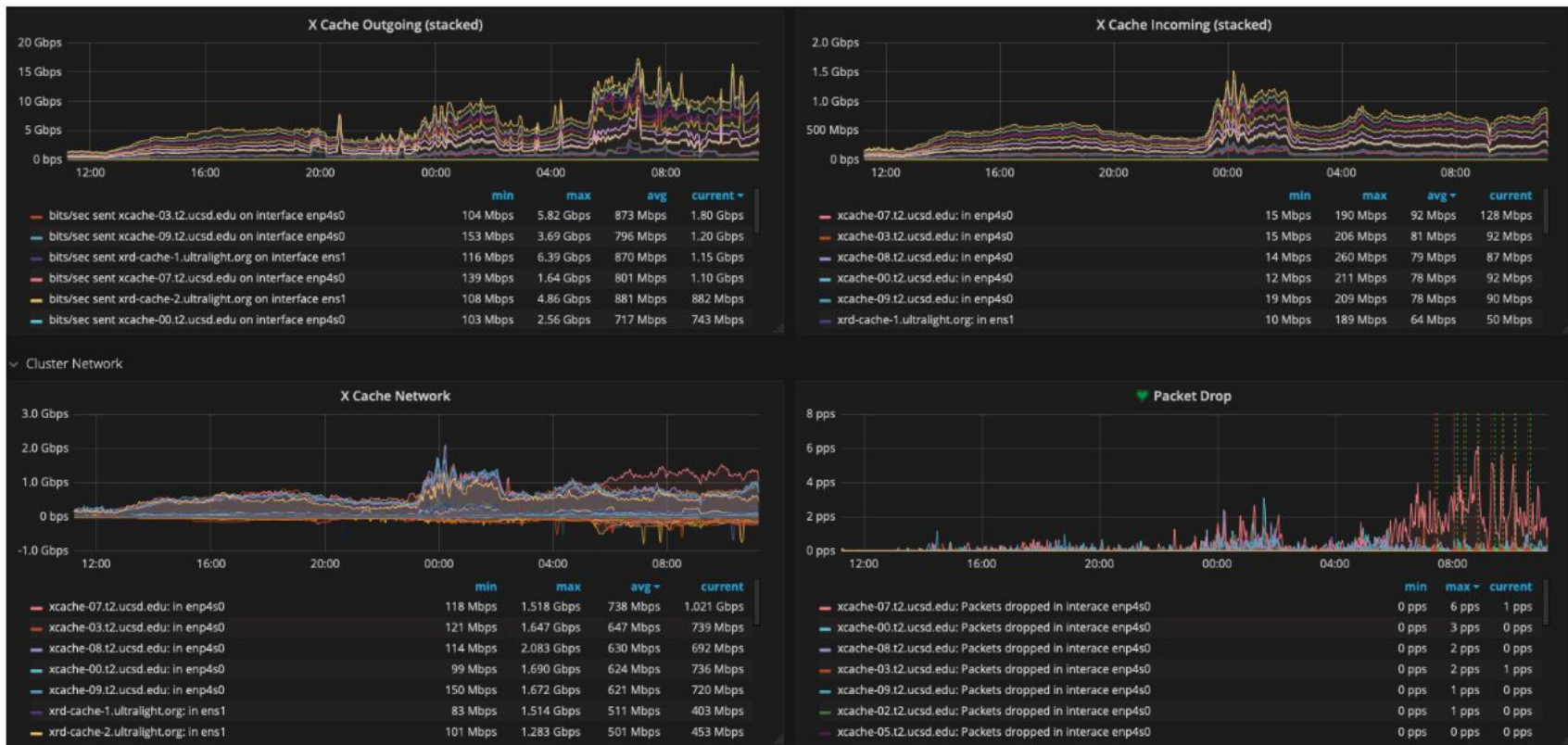
	UCSD	Caltech
Nodes	11	2
Disk Capacity node	12x2TB = 24TB.	30 x 6TB disks (HGST Ultrastar 7K6000)
Network Card	10 Gbps	40 Gbps
Total Capacity	264 TB	360 TB

Current SoCal Deployment



Datasets	Size (TB)
<code>/*Run2016*-03Feb2017*/MINIAOD</code>	182.8
<code>/*RunIISummer16MiniAODv2-PUMoriond17_80X_*/MINIAODSIM</code>	502.5
<code>/*RunIIFall17MiniAODv2*/MINIAODSIM</code>	211
<code>/*-31Mar2018*/MINIAOD</code>	137.9
Total	1041

Combined Monitoring



Good cache hit/miss ratio (average 10:1 with peaks to 15:1)




Some operational problems (CMSSW Multisource)

- In general CMSSW multisource is envisioned to be as greedy as possible. (i.e grab the data from as many different xrootd servers as possible.
- This caused problems on our deployment when several independent caches are behind a single redirector.
- At the worst we had several copies of the same files

We were wasting up to 73% of the space of the cache.

Num Replicas	% of total files
4 or more	26.3
3	17.5
2	33
1	23.2



Matevz & Andy provided a quick fix.

At the redirector level the option to limit how many “redirects” for each job.

```
cms.sched maxretries 0 nomultisrc
```

How it looks now:

Num Replicas	% of total files
4 or more	0.000629
3	0.01
2	0.33
1	99.6



Some operational problems (IPv6)

- The AAA origin redirector for CMS was dual stacked and the individual caches had ipv6 enabled but no global IPv6 address.
- The XCache acting as the client was trying to talk to origin using Ipv6 but timeout after **15 min.**
- This was causing the jobs to timeout (CMS timeout is 3600 secs) and looking for a new sources but together with the previous patch this was causing jobs to fail
- We fixed this problem by dual stacking the cache hosts. However the default timeouts should be changed either on xrootd or at the xcache (OSG packaging).



Current Work and Future Work

1. Kubernitizing the deployment. Igor and some students are working on getting a CMSCache docker image and kubernetes deployment for one cache.
2. The future will be to replace all the current bare metal with pods
3. The last round of scale tests happened with Xrootd 4.4 lots of code changes to the cache since then. We will be doing a new round of scale tests this summer.
4. Improvement to the monitoring to be able to calculate cache hit/miss rate.

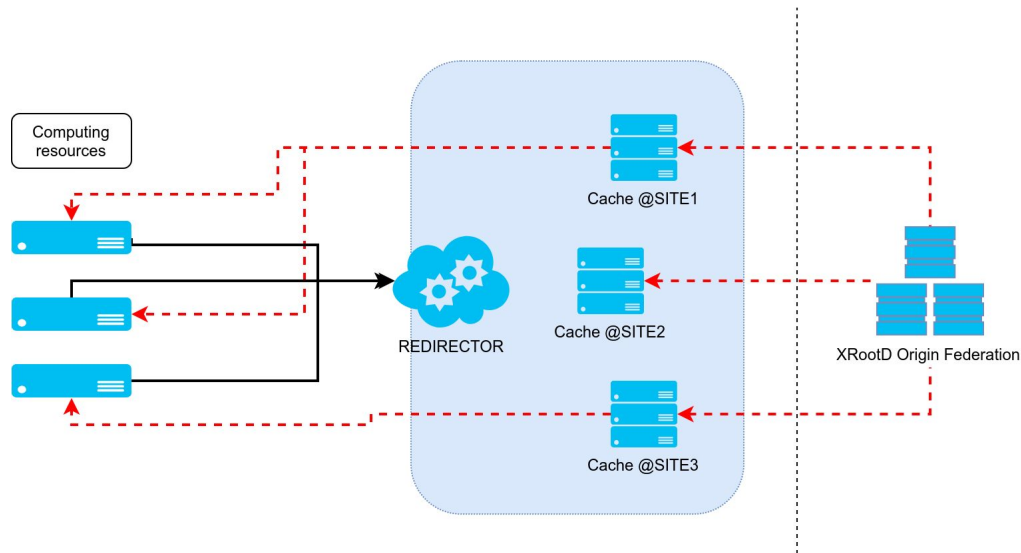
INFN distributed cache: recap

Prototyping a cache model for analysis jobs:

- enabling “**storage-less**” T2s and efficient **scale out over opportunistic resources**

Using cache for a **geo-distributed layer approach:**

- geo-distributed **network of unmanaged storages**
- common namespace (**no data replication** across cache)
- request mitigation** to custodial sites



Activity context

- **WLCG DOMA** (data organization, management, access):

- XCache as core solution for Data Lake model implementation in WLCG
- interest in historical data access modellization



- **eXtreme DataCloud:**

- CachingOnDemand: deployment automation of XCache stack on cloud resources
- performance evaluation



- **ESCAPE**

- cache as data access tool for scientific communities data lake



- **IDDLS**

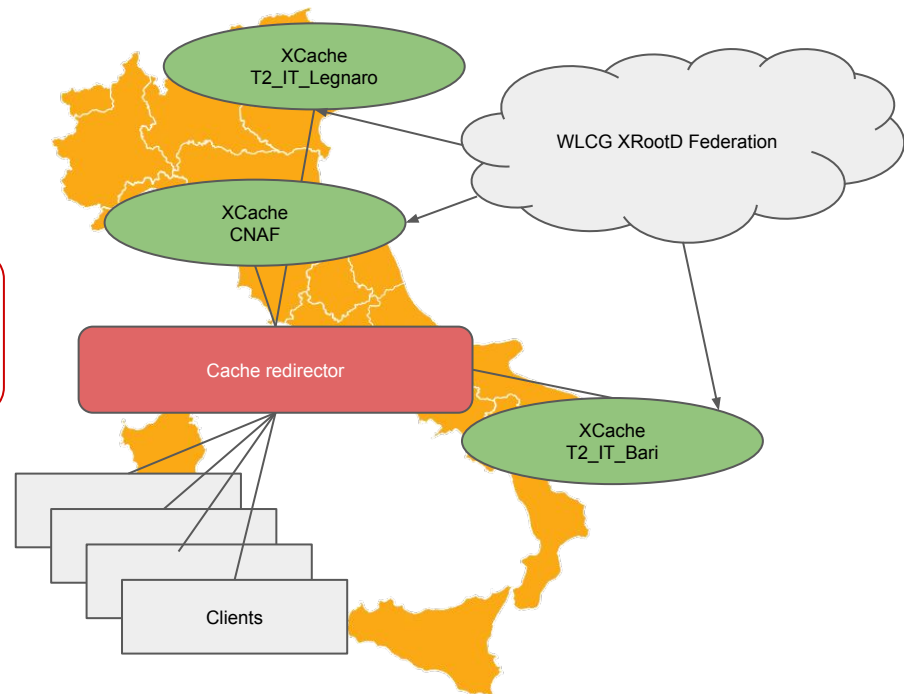
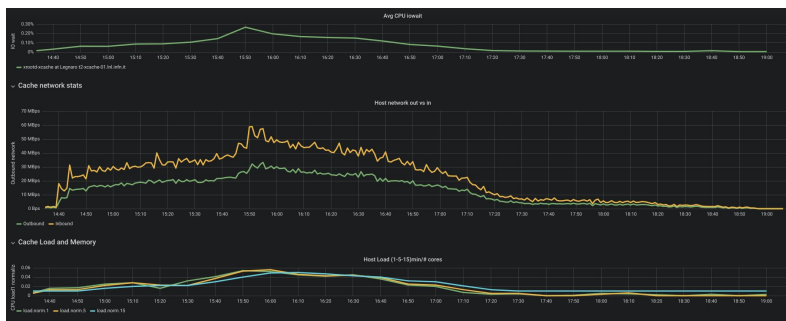
- INFN+GARR national experimentation for a data-lake infrastructure

INFN distributed cache status

N.B. infrastructure provisioning in collaboration with CNAF, Legnaro and Bari working group

- Using **already available resources** (with minimal requirements) on **volunteer Italian Tiers** for a distributed **cache-layer PoC**
- Setup **integrated with CMS workflows**

Working prototype since mid-2018 with a limited amount of real tasks using it.

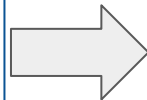


Cache server integrated monitoring - so far host based metrics - xrd metadata not yet there

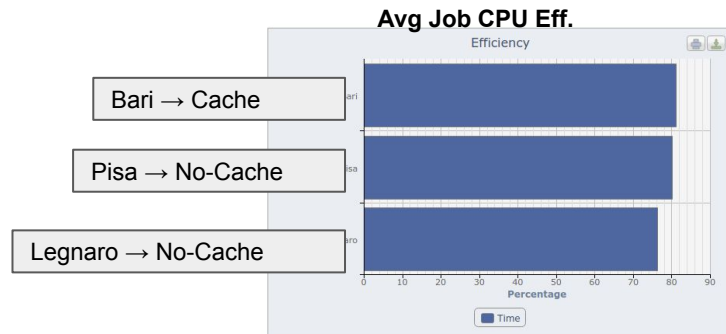
INFN distributed cache: tests

Current **functional test** setup:

- CNAF XCache redirector federating 3 servers:
 - CNAF XCache server (5TB spinning)
 - T2 Bari XCache server (10TB gpfs)
 - T2 Legnaro XCache server (22TB spinning)
- **Redirecting part of the CMS analysis workflows** to contact National redirector
 - **based on dataset name requested**



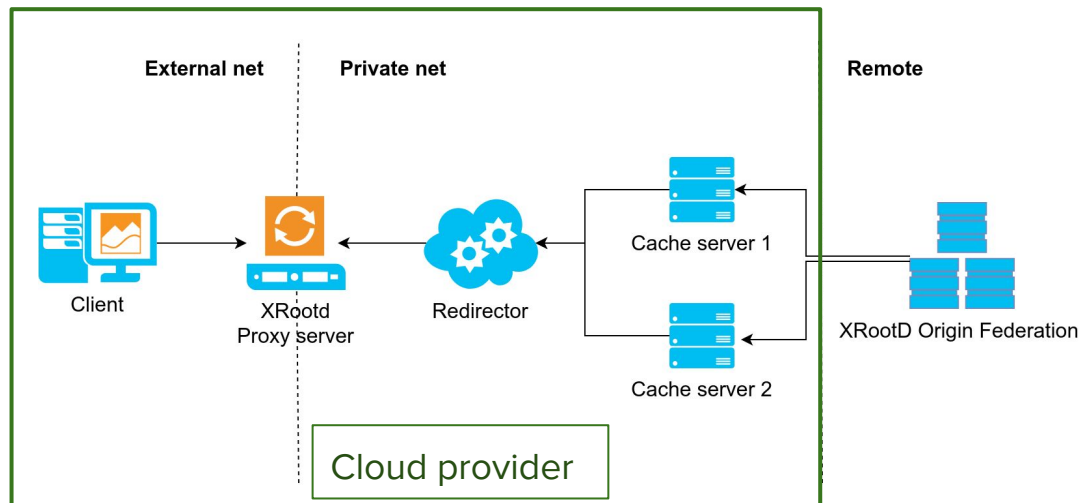
- **Satisfied with the functionalities for CMS workflows**
 - **latency hiding** effect on job CPUEff is noticeable for higher latency links



Question: In case of heterogeneous resources, a “smart” load balancing would be useful. Is it possible/in pipeline?

XCache on cloud resources

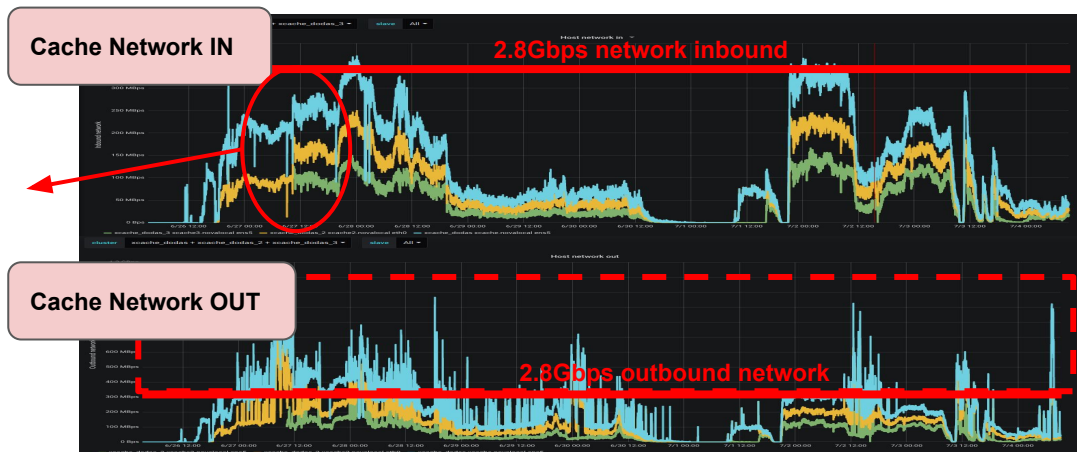
- Automatic procedures for the creation of an **XCache cluster on-demand**
 - **ready-to-use recipe** for both bare metal and cloud environment/orchestration
- Deployment available with:
 - Ansible for bare metal installation
 - Docker compose
 - good end-to-end example for new communities
 - Kubernetes
 - deployment at scale on cloud resource
 - DODAS
 - TOSCA templates for end-to-end automatic deployment on cloud resources



Scale test on cloud resources

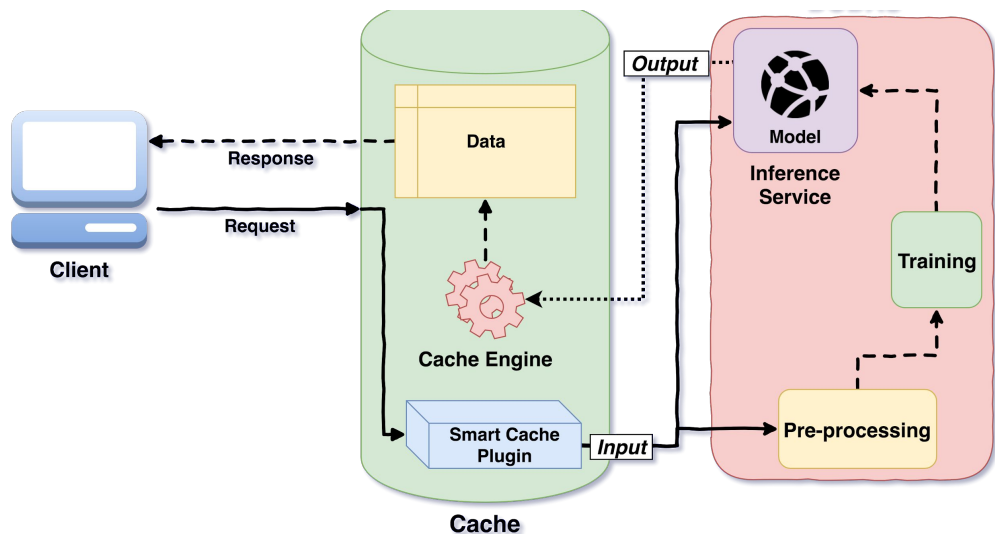
- Complete cache cluster deployment (3 cache servers) on OpenTelekomCloud resources
 - co-located with computing resources deployed with DODAS
- **Real CMS analysis workflows on cloud resources (2 volunteer users)**
 - 2k concurrent jobs
 - ~150k of users jobs completed reading from standalone cache cluster deployed at OTC

Seamless
scale out
from 2 to 3
servers



Additional activities

- Tested XCache on cloud deployment for **AMS use case:**
 - Moving **toward a production ready system**
- Interested in Token-based auth model as alternative to the current one
- A first **INFN proof-of-concept** implementation to enable **smart data cache at CMS has been deployed**



Conclusion

- Data access patterns studies confirm the expectations, latency impact on CMS analysis jobs is not negligible and may increase in the future
- XCache shows satisfactory functionalities from latency hiding to disk space optimization
 - CMS user analysis datasets have been proved to be manageable at scale
- For opportunistic resources a container based solution has been tested
- Good overall experience in terms of support and fixes providing in case of issues