29 October - 1 November 2018
Amsterdam, the Netherlands

14th eScience
IEEE INTERNATIONAL CONFERENCE

Distributed and on-demand cache for CMS experiment at LHC

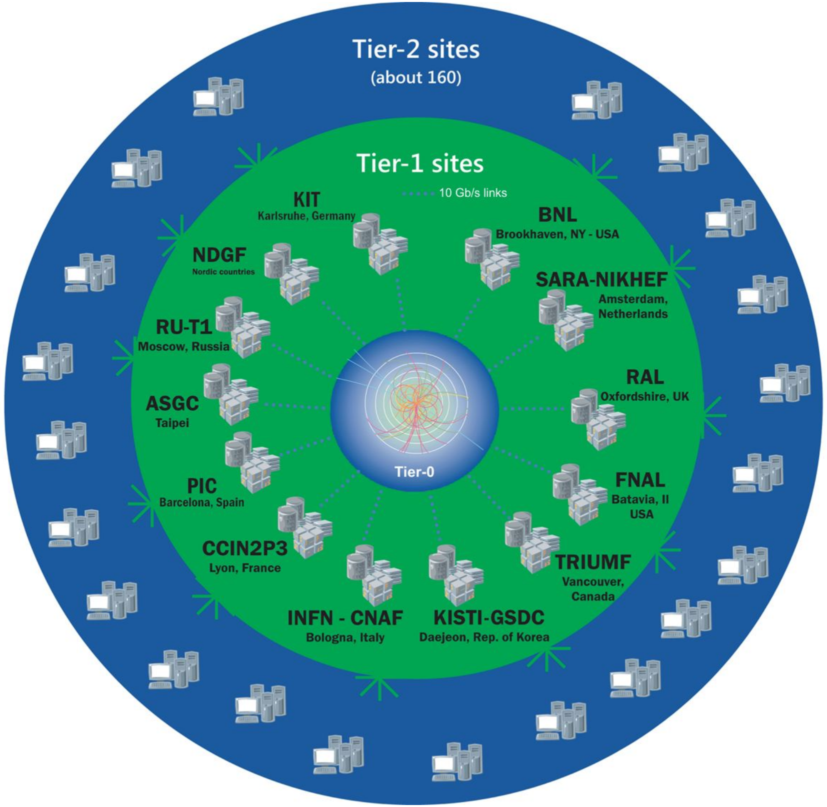
Diego Ciangottini
on behalf of CMS Collaboration and INFN-Cache team

Outline

- Introduction
- 2 scenarios of evaluation
  - cache on ephemeral storage for opportunistic resources
  - geo-distributed cache with unmanaged storage
- Performance results
- Conclusion and future activities

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



Extension: dynamic resource provisioning

Scenario 1

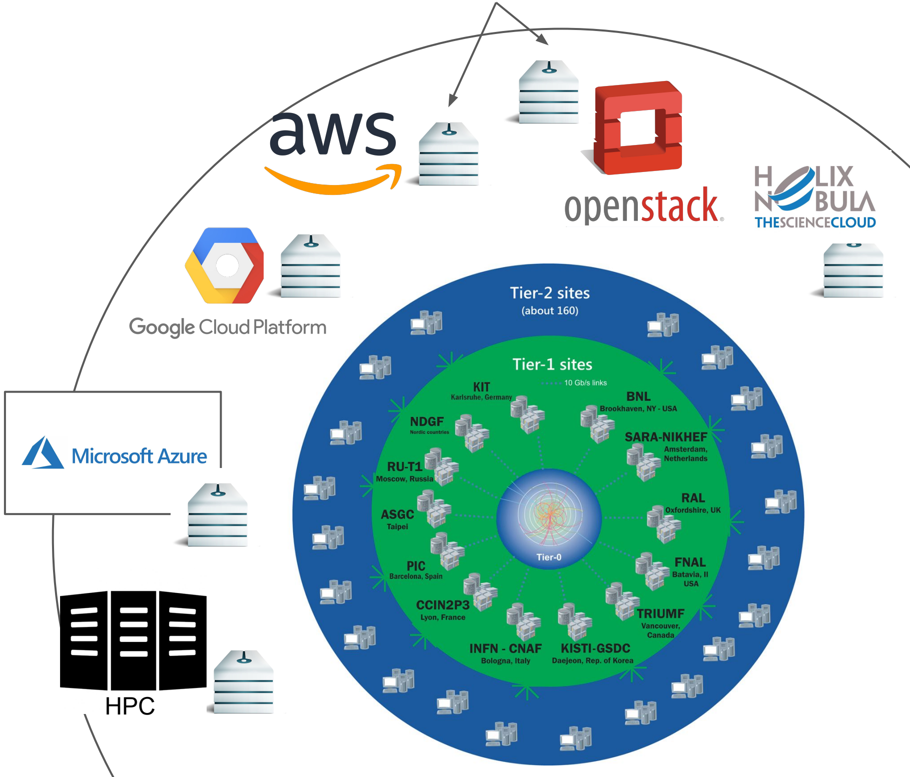
Computing resources are **opportunistically deployed on cloud/HPC resources**

- **storage not necessarily available**
  - ○ remote read **latency**
  - ○ **I/O inefficient**

The **cache** introduction may offer:

- **ephemeral storage for hot data** near the computing provider
- **optimized wan access**, only for data not already on the cache

On-demand Cache resources



Cache layer in data-lake for HL-LHC

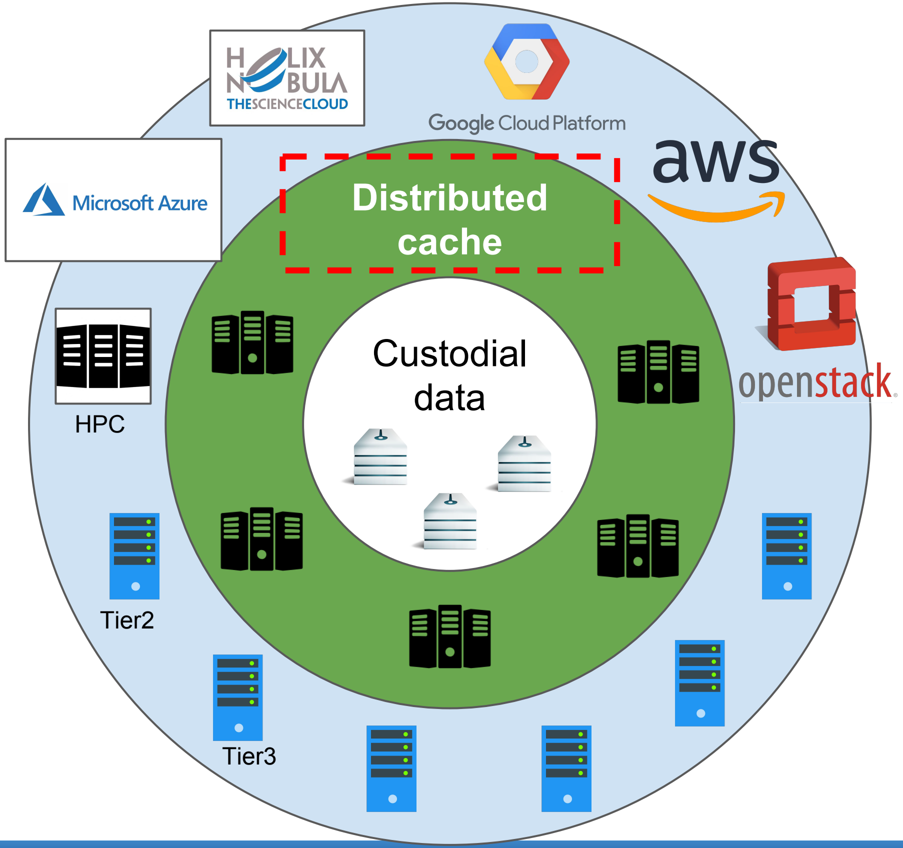
Scenario 2

Few world-wide custodial centers with data replica managed by the experiment

- Computing Tiers **access data directly from closest custodial center**

Using **cache for a Content Delivery Network approach:**

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



Technology: XCache evaluation

Two scenarios for evaluation:

- cache on ephemeral storage for opportunistic resources
- geo-distributed cache with unmanaged storage

XCache technology have been used in both of the activities:

- Part of **XRootD** technology already widely used in WLCG for **federating storages**
  - Storage resources are accessible for **any data, anywhere at anytime (AAA)**
  - XRootD infrastructure **spans all of the Tier-1 and Tier-2 sites in EU and US CMS**

XCache mechanics

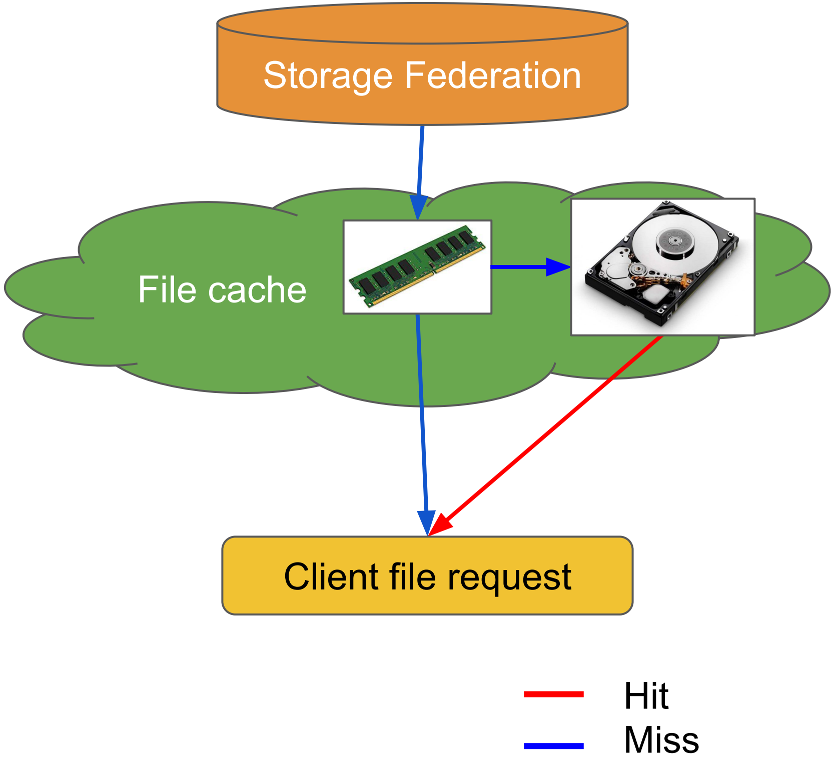
Open File

1. *Cold cache:* remote open through storage Federation
2. *Warm cache:* opens file on local disk

Note: remote open is only initiated if/when a requested block is not available in the cache.

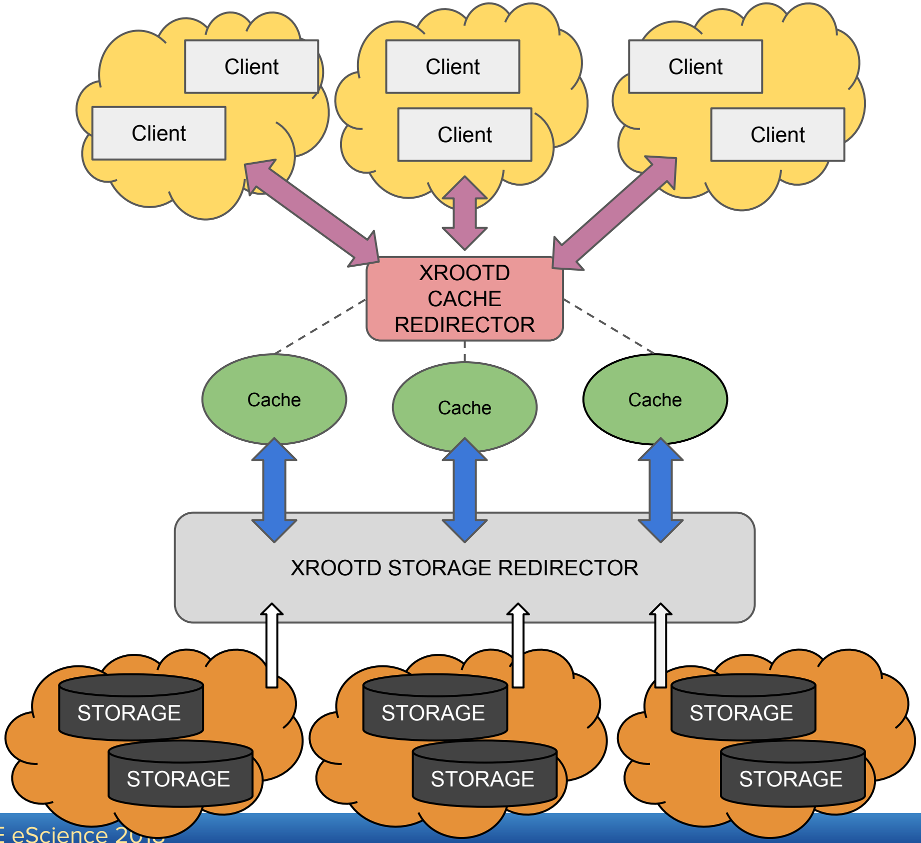
Read File

1. If in RAM/disk➞serve from RAM/disk
2. Otherwise request data from remote and
   a. **serve it to the client**
   b. **write it to disk via write queue** (this way data remains in RAM until written to disk)



Clustering with xrootd cache redirector

- Through the XrootD redirection is possible to **federate caches in a content-aware manner**
  - redirect client to the cache that actually have file on disk
- **Loadbalancing:** If no cache has the requested file, **a round robin selection of cache server is used** (*configurable*)

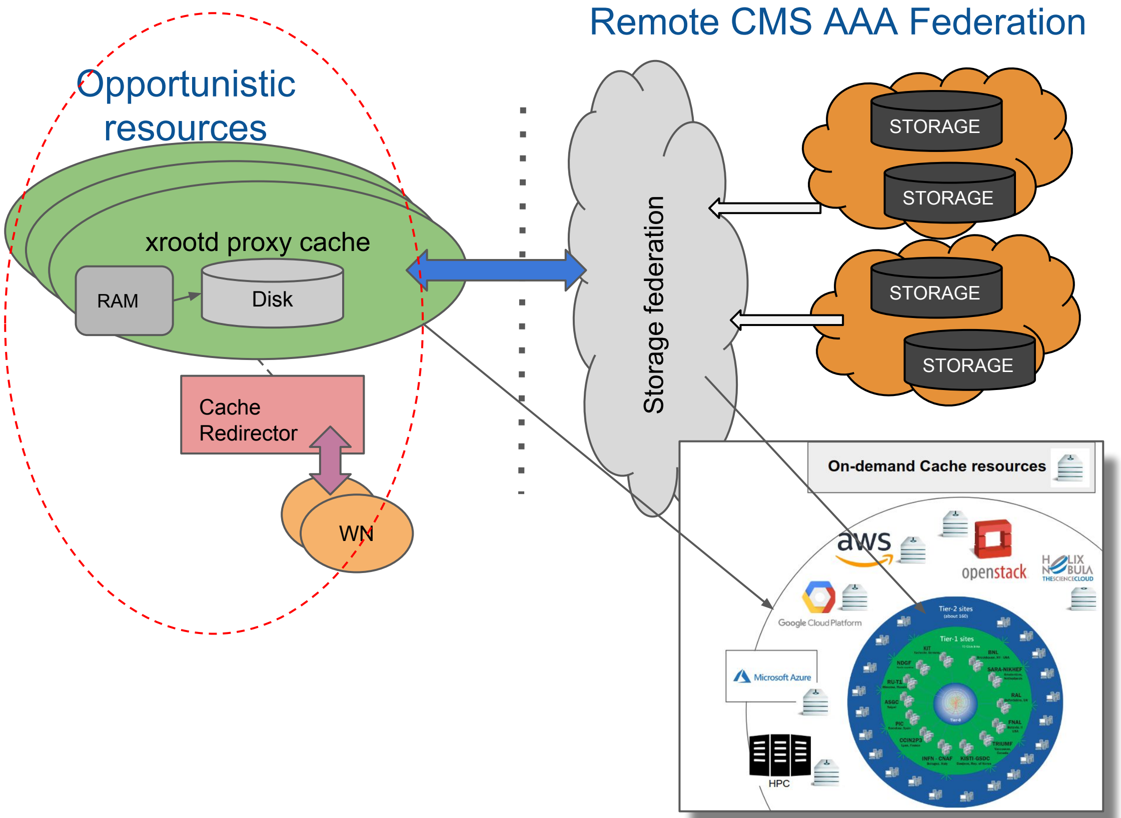


Cache for opportunistic resources

Scenario 1

In case of **computing on opportunistic resources** the remote data access pattern can be improved providing:

- an **on-demand cache layer near cpu resources (same cloud provider)**
  - **scaling horizontally**
  - **manage caches in a content-aware manner**
    - redirect client to the cache that currently have file on disk

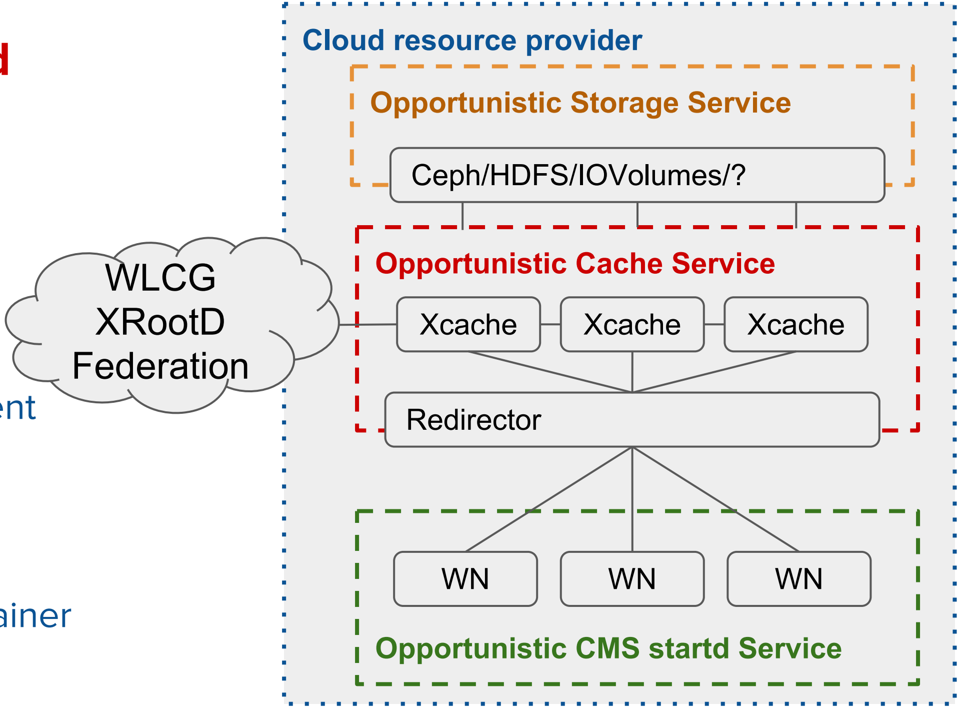


Testing with CMS workflows

Scenario 1

(*) https://dodas-ts.github.io/dodas-doc/ EOSC-hub

- **Real CMS analysis workflows on cloud resources (2 volunteer users)**
  - 2k jobs @OpenTelekomCloud (OTC)
  - ~150k of users jobs completed reading from standalone cache cluster deployed at OTC
- **DODAS (*)** have been used for:
  - same configuration for setup on different cloud providers
  - automated deployment through:
    - Ansible for infrastructure
    - K8s or Mesos/Marathon for container orchestration



Results

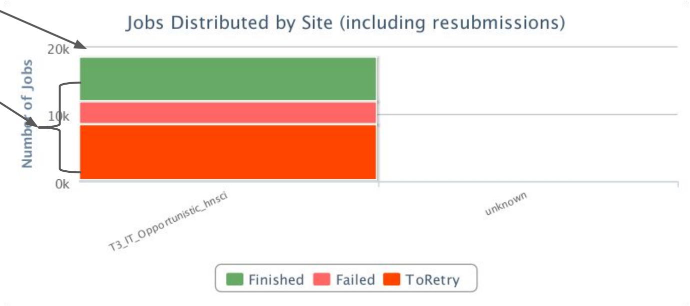
Scenario 1

All in all good performances

- **partial healing for high latency** remote access failures (timeout)
- **local-like performances when a cache hit occurs**
- **on-demand deployment recipes** and easy maintenance

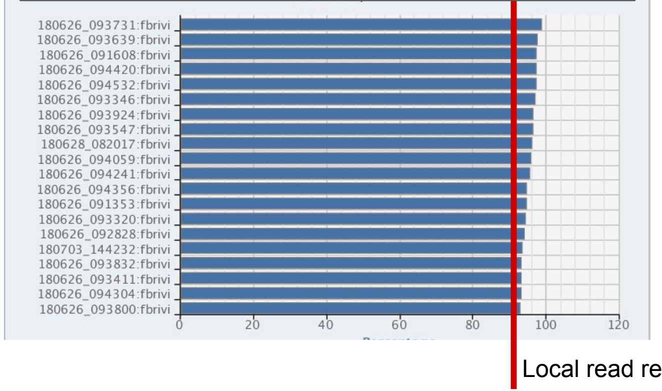
Effect of the cache

Failure for latency



No cache overhead observed

Cache hit - Avg CPU efficiency



Local read reference

Automated deployment through:

- Ansible
- K8s (soon also in helm)
- Mesos/Marathon

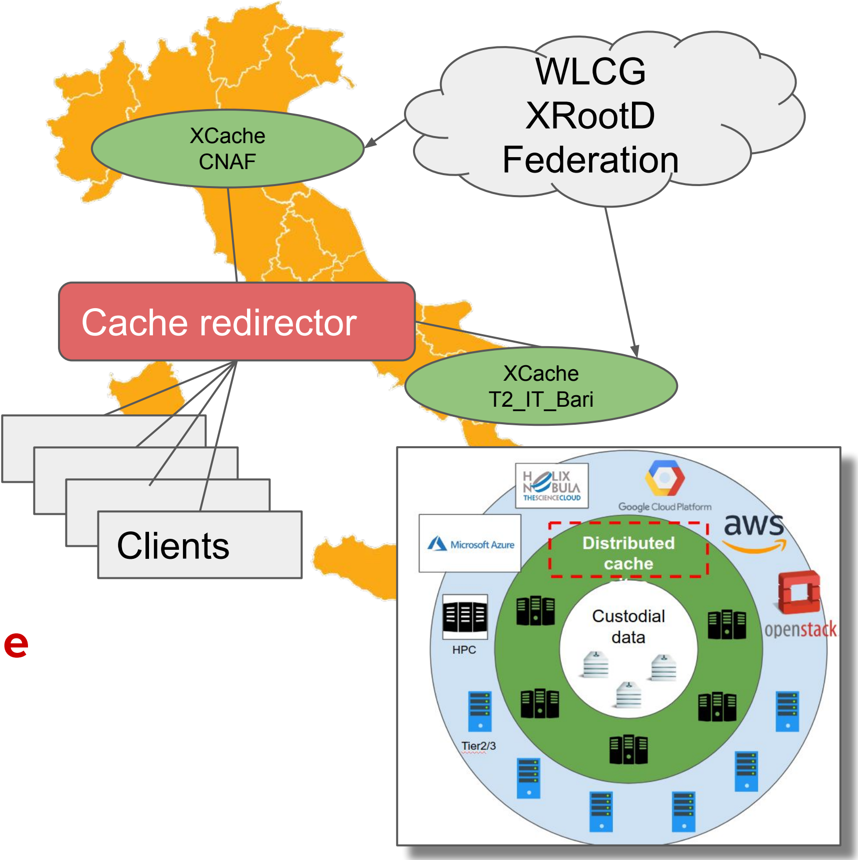
eXtreme DataCloud

https://cloud-pg.github.io/CachingOnDemand/

Distributed testbed deployment

Scenario 2

- Deployment a **geo-distributed cache:**
  - Clients contact the **cache redirector**
  - Redirector **steers client to**
    - the **cache that actually have file** on disk
    - **If no cache has the requested file, a round robin selection** of cache server is used
- Network of **unmanaged storages for hot data**
- One line configuration tweak on computing resources allows to **seamlessly integrate the distributed cache on CMS workflows**



Distributed testbed deployment: testbed

Scenario 2

Current **functional test** setup:

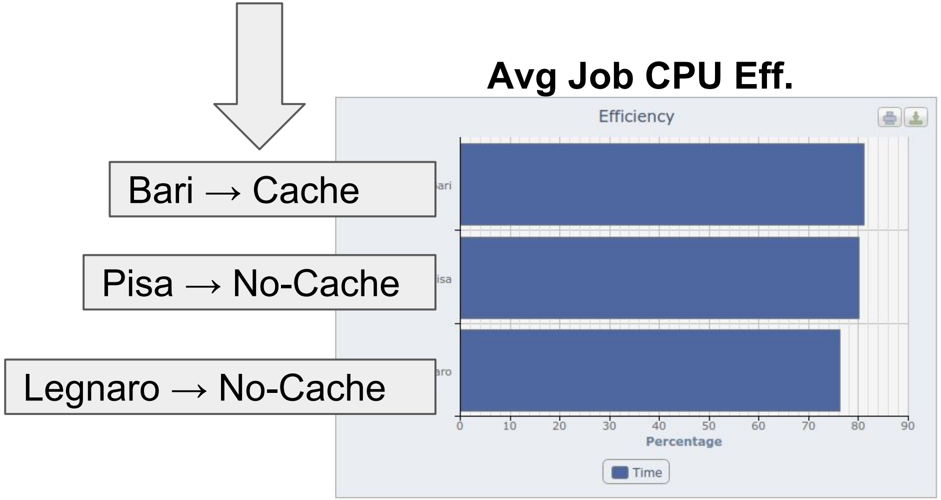
- CNAF XCache redirector federating 2 servers:
  - CNAF XCache server (5TB)
  - T2 Bari XCache server (10TB)
- **Redirecting part of the CMS analysis** workflows to contact National redirector
  - based on dataset name requested
- **2 more sites** (Tier2 at Pisa and Legnaro) are planning to join the testbed

Italian XCache federation: functional checks

Scenario 2

- Test tasks submitted to T2_IT_Bari with **empty cache**
- Comparing jobs running at Bari (pointing to cache) with “Ignore locality” ones on other sites

Avg Job CPU Eff.



Bari → Cache

Pisa → No-Cache

Legnaro → No-Cache

No penalty in CPU eff in case of empty cache

Performances of jobs **reading from empty cache** is comparable with **remote reading.**

Conclusions and plans

In the context of
DOMA-Access WG

- Two analyzed scenario have been presented:
  - cache for dynamic resources
  - distributed cache layer for HL-LHC data-lake model
- Performance evaluation motivates further activities
  - **on-demand deployment** and easy maintenance
  - **partial healing for high latency** remote access failures
    - no penalty in case of empty cache
    - local-like performances when an hit occurs

Work in progress:

- evaluate **cache benefits within CMS computing model through simulation**
- **smart (ML-based) data fetching and request routing** based on real-time and historical information
- **deployment in production** @INFN

Thank you

Backup