National distributed disk cache for CMS@LHC: Status and Progress

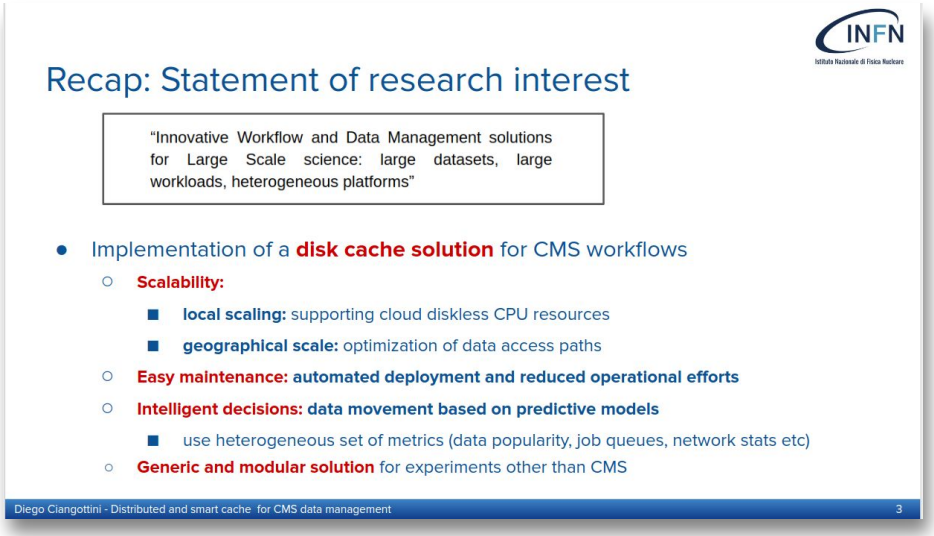
Diego Ciangottini[a]

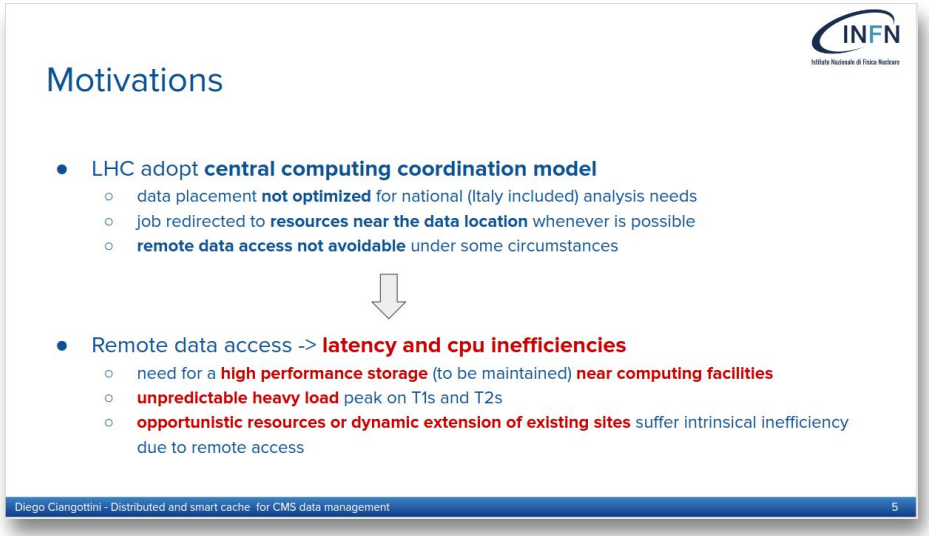
a) INFN, Perugia

Outline

- Activity motivations remind
- XRootD cache architecture
- Current status
- Next steps
  - Deployment of a first National distributed testbed
- Long term view
  - Data-lake integration

Motivations





CSN1 meeting 21-23 Feb

https://agenda.infn.it/getFile.py/access?contribId=27&resId=0&materialId=slides&confId=14896

A cache system to support several scenarios

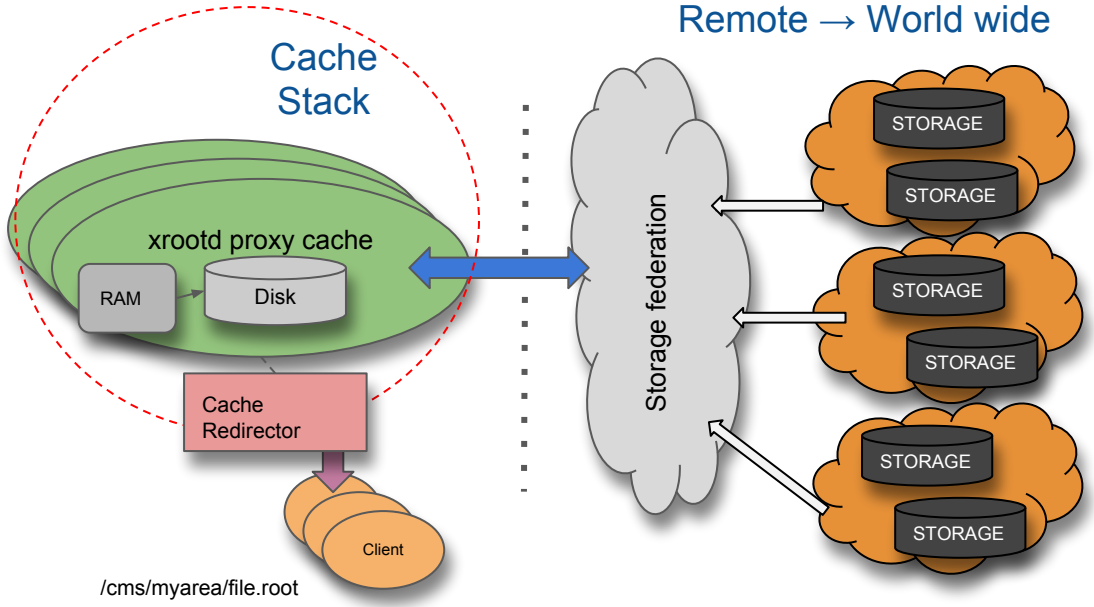
- **Leverage national networking to reduce total maintained storage resources**
- **“Data-lake” approach:**
  - **Interposing cache** on top of **a central custodial site**
- **Opportunistic Computing:** to bring **not pledged resources** in the computational model
  - From the experiment point of view: to integrate cloud based resources with **zero effort**
- **Dynamic Site Extension:**
  - **Peak of usage** or more in general buying **external cloud resources:**
    - see activities like: Aruba, Microsoft Azure, HNSci project etc

NOTE: The presented activity lives within the CMS Data Management project

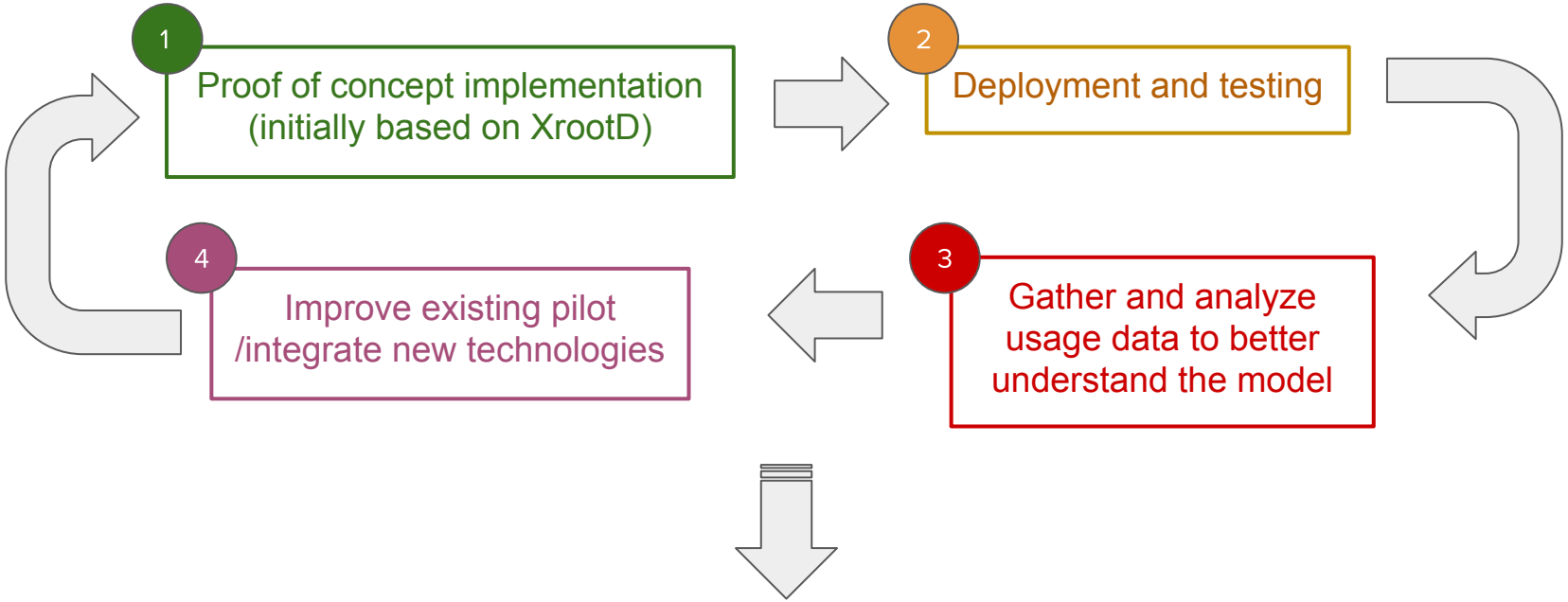
XCache implementation

XRootd based implementation of a disk proxy-cache tool

- XRootD infrastructure **spans all of the Tier-1 and Tier-2 sites in EU and US CMS**
  - **well known protocol at computing sites**
- **Multiple storage backend** support and optimization
- **Easy integration on current LHC computing model**

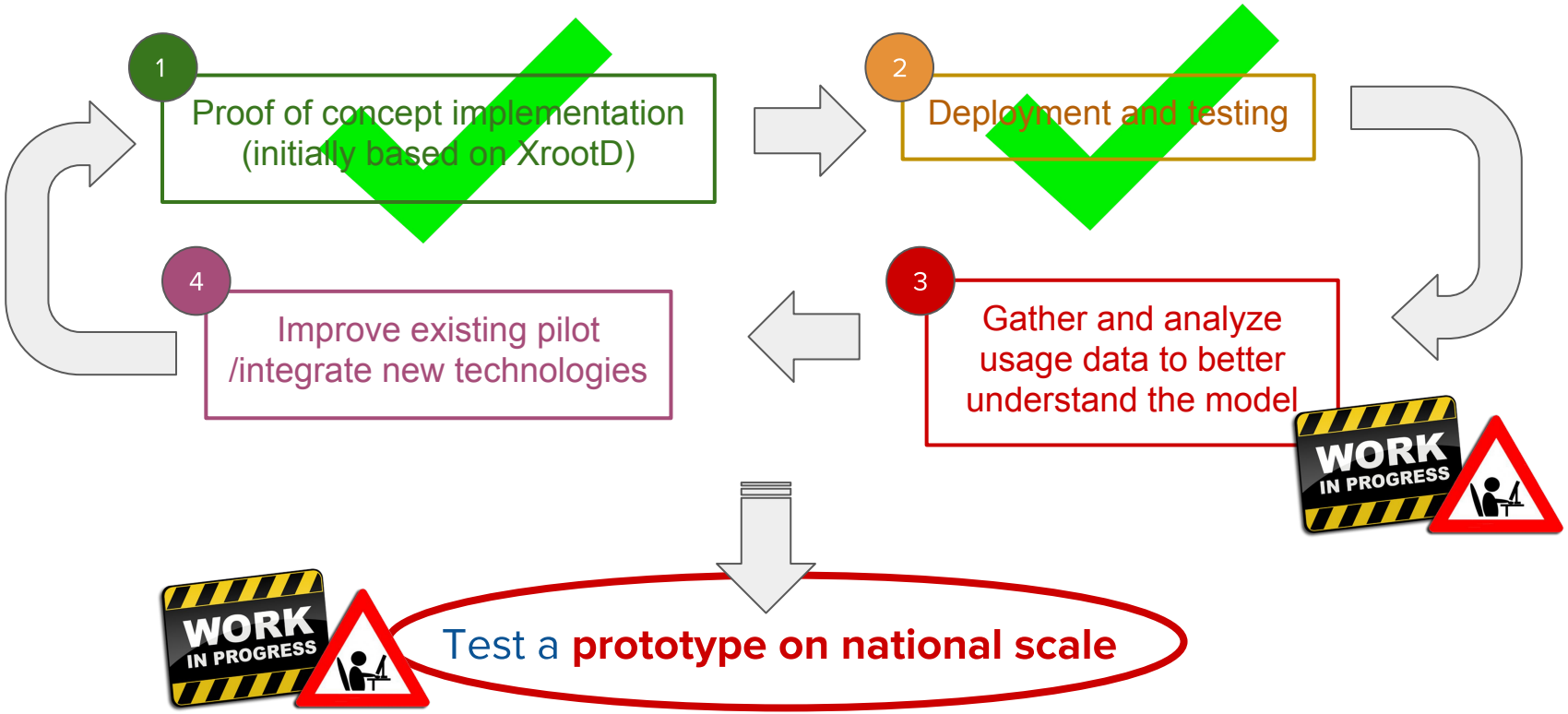


Work strategy



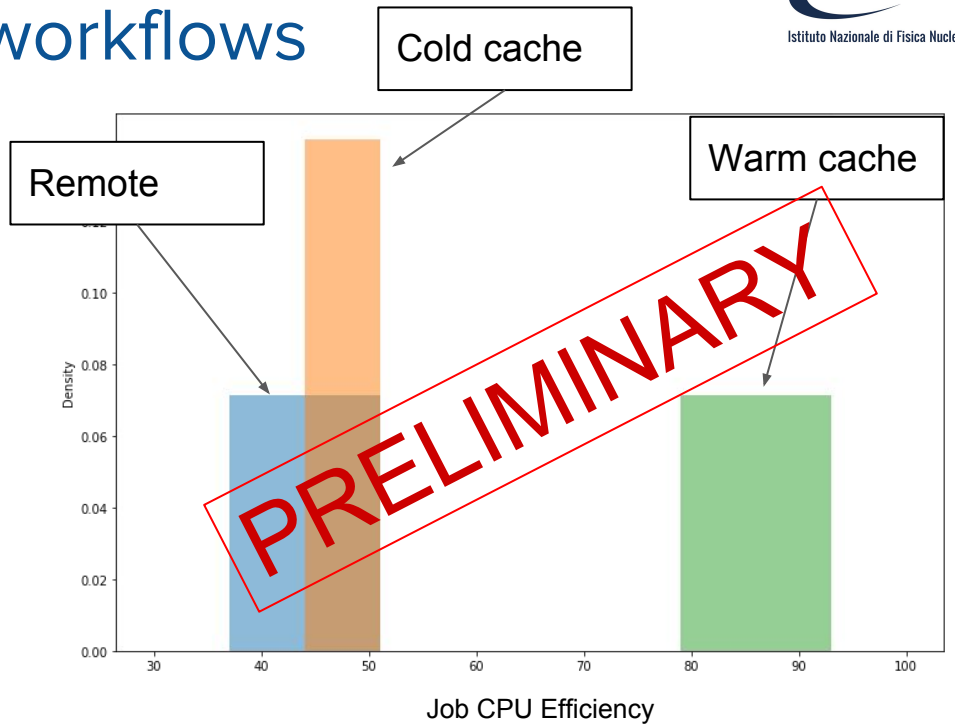
Test a **prototype on national scale**

Work strategy: status update



Testing cache with CMS workflows

- A first **proof of concept** has been developed and **first tests with CMS analysis workflows** are ongoing.
  - **DODAS environment** to provide a recipe for an **automatic XrootD cache deployment on heterogeneous cloud resources**
  - utilize “any cloud provider” with almost zero requirements and a **simple text configuration file.**
    - **Open Telekom Cloud (OS based)** resources used for preliminary results



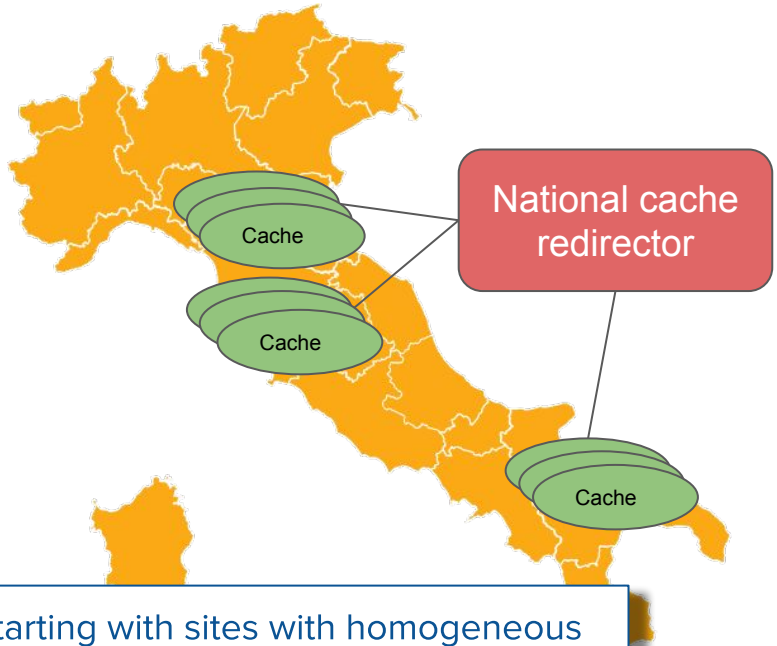
Everything behaves as expected.

Remember that the amount of efficiency gain is heavily **workflow dependent**

National testbed deployment

- **Objective: to deploy a national level cache**
  - **geographically distributed** cache servers
  - **heterogeneous resources and providers**
  - Leverage **national networking to optimize the total maintained storage** resources
- **Collection of important data for evaluating the benefits on a realistic scenario**

Already contacted CNAF, Pisa and Bari to support **the deployment of a national testbed for Xcache federation → Agreed!!**



Starting with sites with homogeneous resources (gpfs-storm).
Then extending to other sites (e.g. Legnaro) on a second step.

National testbed deployment activity

- Reproduce on national scale the **same architecture and tests as on cloud resources**
  - cluster of cache servers federated under a dedicated redirector
- Write a **technical proposal for activity coordination**
- **Functional tests** with **no dedicated high IO hardware** (e.g. ssd etc)
  - in general no additional costs, using hardware that is already in place

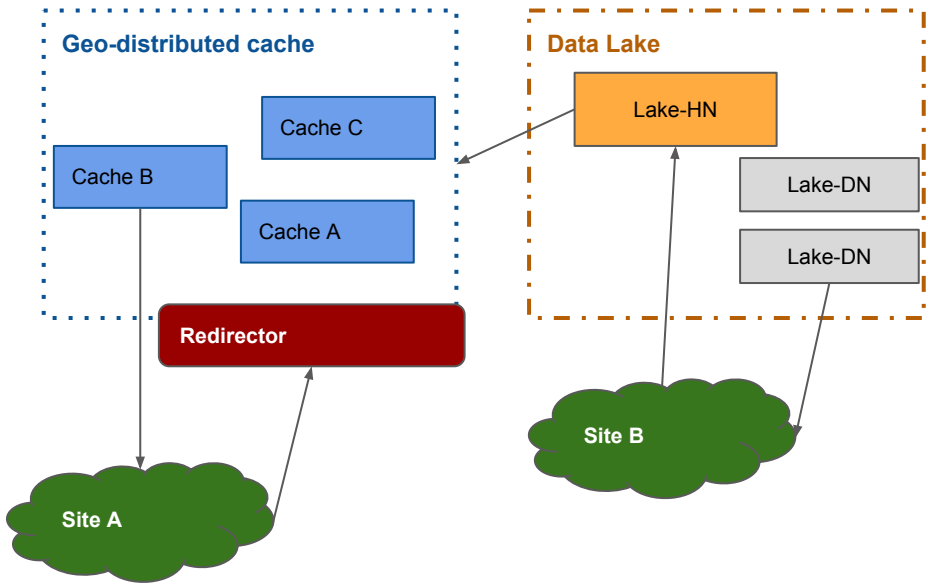
XCache and Data-Lake

The activity on Italian testbed can be **a first benchmark for future solutions** proposed in an LHC data-lake scenario.

Synergies:

- outlook on **projects with similar motivations and objectives** (**eXtremeDataCloud** and others if any will come up)
- with **CERN investigation of XCache application for internal EOS cache mechanism.**

- Lake-HN : central service of the data lake (namespace, metadata etc)
- Lake-DN : storing data and under the management of Lake-C



Summary

- **INFN XCache for CMS@LHC is proceeding as for roadmap**
  - first phase ready ➔ automation and local cluster test
- The technology used is **based on XRootD**
  - **multi-backend storages** ➔ generic application
  - every **system XRootD compliant can use XCache**
  - possible to be extended **beyond CMS boundaries**
- **Existing synergy with pure HTTP approches** (Sonja talk today)

My two major milestones for 2018:

- **Tests with national testbed**
- **Dynamic site extension:**
  - e.g. **overflow:** CMS payloads assigned to CNAF pledged resource ➔ redirected seamlessly to cloud resources deployed with DODAS

Backup

Motivations and features: recall

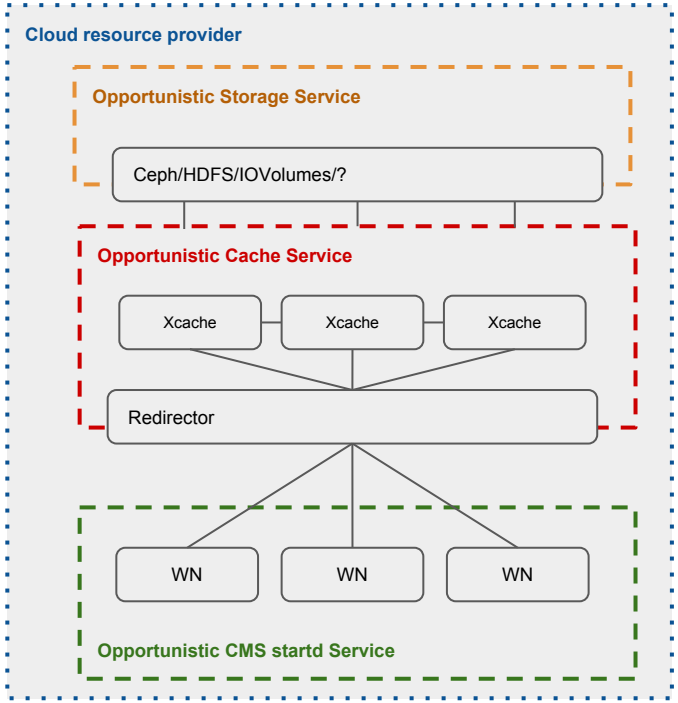
- **Reduce latencies / Improve efficiency** on remote data access
  - enhance CPU efficiency and job success rate
- **Reduction of traffic load**
  - mitigate the load on custodial storages
- Optimization of data access for **opportunistic resources**
  - e.g. sites extension, public cloud etc..

Important features:

- **Scalability**
- **Easy maintenance**
- **Intelligent decisions**
- **Modular solution**

Architecture outlook

- **Modularity**
  - factorized applications
  - cache on top of existing storages
  - seamless scaling
- **Packaging**
  - docker images
  - health-checks for self-healing implementation
- **Plug-in**
  - cache algorithms
  - clustering data distribution



XCache in CMS

- **Reduce latencies / Improve efficiency** on remote data access
  - enhance CPU efficiency and job success rate
- **Reduction of traffic load**
  - mitigate the load on custodial storages
- Optimization of data access from **opportunistic resources**
  - e.g. sites extension, public cloud etc..

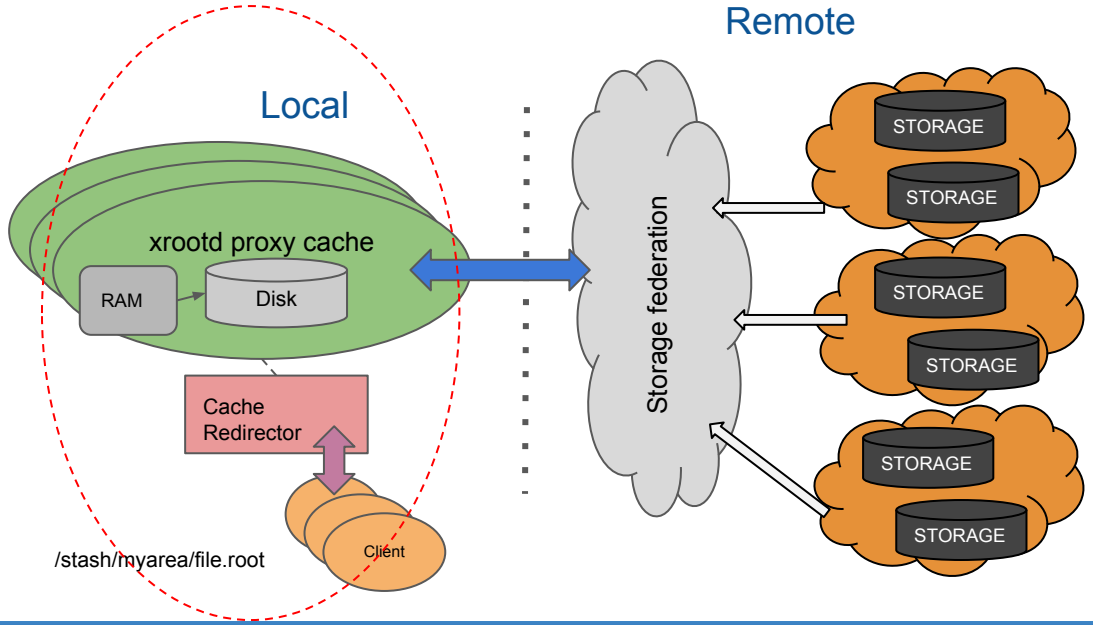
Main features

- **Scalability:**
  - **local scaling:** supporting cloud diskless CPU resources
  - **geographical scale:** optimization of data access paths
- **Easy maintenance:**
  - automated deployment and reduced operational efforts
  - self-healing
- **Intelligent decisions:**
  - data movement based on predictive models
- **Modular solution:**
  - easy to extend and pluggable

Local site scenario

e.g. opportunistic sites and remote site extension

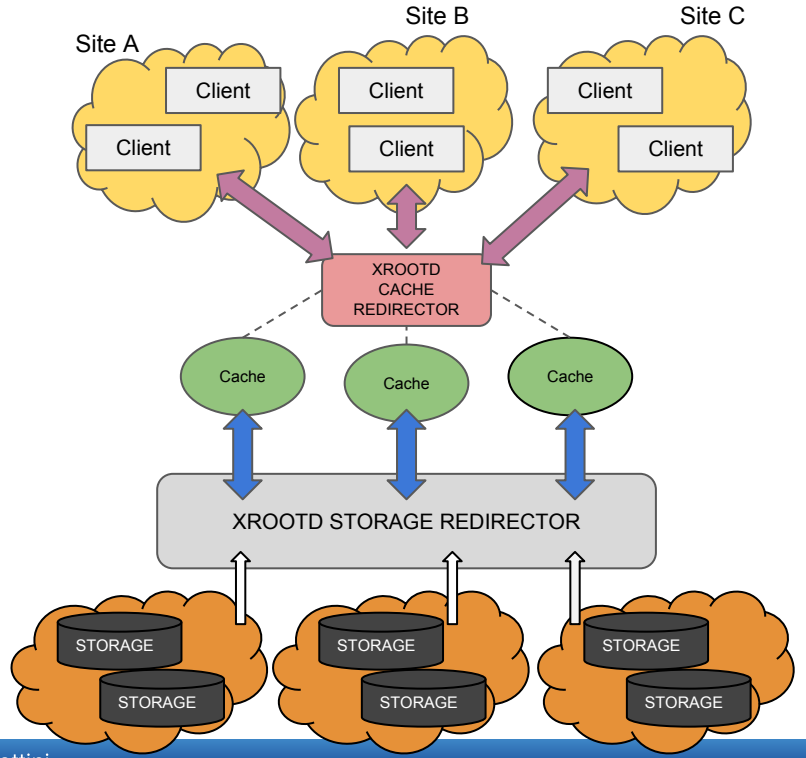
- Create a **cache layer near cpu resources**
- Bring it up **on demand**
- **Scale horizontally**
- **Federate caches in a content-aware manner**
  - redirect client to the cache that currently have file on disk



Distributed scenario

Geographically distributed cache

- The very same technology used on local scenario can be **geo-distributed**
- Use **ephemeral storages to enhance jobs efficiency**
- Leverage high speed links to **reduce the total amount of allocated space**



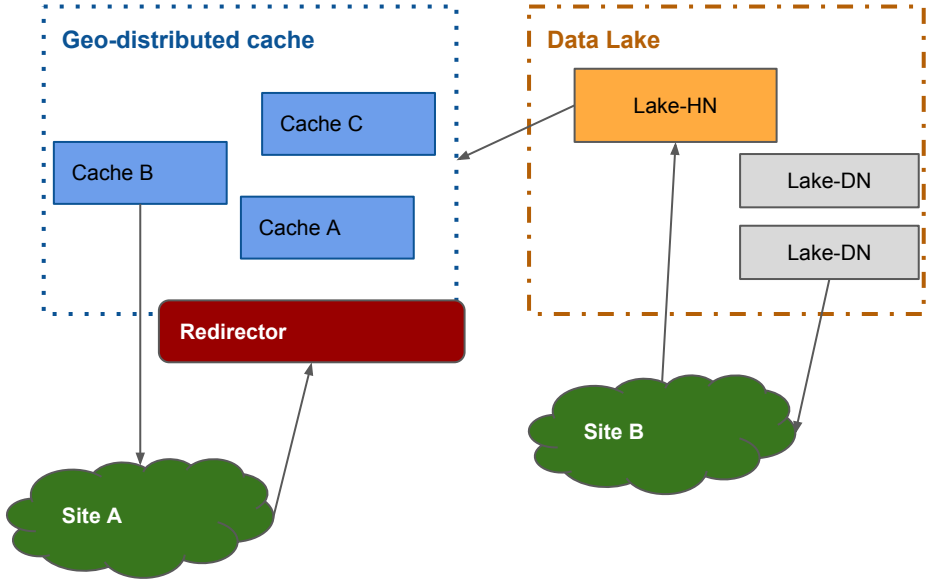
Side note on future development

Data lakes model

- **Geo-distributed scenario** is also **part of a “data lake” model**

- **creating a multi-site cache layer**

- Lake-HN : central service of the data lake (namespace, metadata etc)
- Lake-DN : storing data and under the management of Lake-C



Current state of the work

- Implemented **local site scenario:**
  - Preliminary functional tests
  - **Local scale scenario test on cloud resources**
    - deployed on private and public cloud
    - setup automation
    - CMS workflow test

Configuration and integration overview

Example

```
sudo docker run -v $PWD/config:/etc/xrootd cloudpg/xrootd-proxy --config /etc/xrootd/xrd_test.conf
```

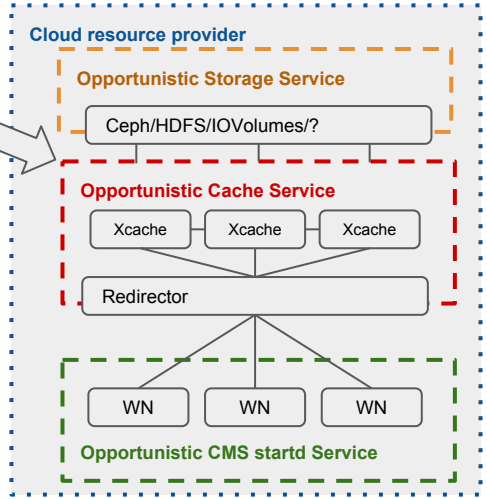
- **CMS Xcache Docker container** has been setup to allow an easy deployment
  - passing a **complete xcache config file**
  - or setting **caching parameter as arguments/env**
  - healthcheck call implemented

A Docker Compose configuration file is available to **orchestrate the deployment of a local test instance.** The stack contains a test remote server, a cache instance and cache redirector (preliminary docs here)

- then a variety of recipe for **orchestration tools have been evaluated**:
  - **docker swarm, k8s and marathon services** (redirector+caches)
  - config and scale services with compose-like recipes

swarm orchestration kubernetes MARATHON

- **Open issue: authentication**
  - the cache server **authenticate with remote storage through its own credentials**
  - **no user cred forwarding**
  - **no token authentication yet**



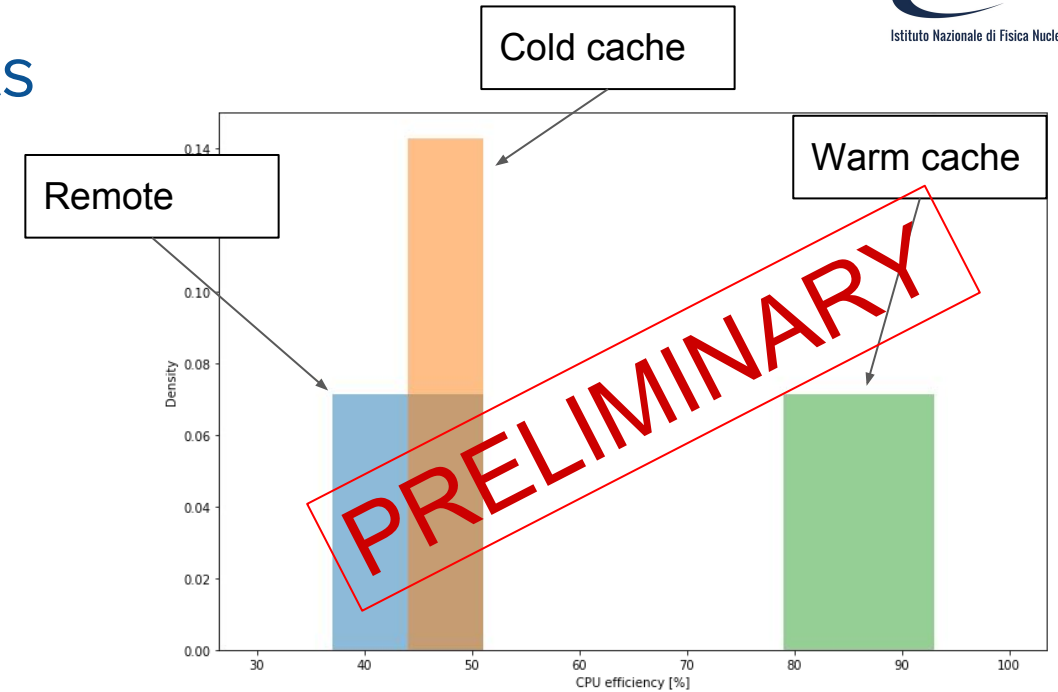
Tests on local site scenario

- **Tests with CMS analysis workflows**
  - **DODAS** service have been used
    - same configuration for setup on different cloud providers
    - automated deployment

- **Measurement and comparison of CPU efficiency with:**
  - remote data access
  - cold cache (missing file)
  - warm cache (file found)

First benchmark results

- **1 cache server**
  - Open Telecom Cloud
  - VM 16 cores and 32GB RAM
  - 500GB HighIO flavor
  - Cache config:
    - prefetch: 0
    - block size: 512k
    - origin: xrootd.ba.infn.it
- **4 WNs** over the same internal network
- **IO test with CMS analysis workflow**
  - 4 jobs
  - reading vertex associated tracks information
  - **ad-hoc setup to enhance the network latency effect**

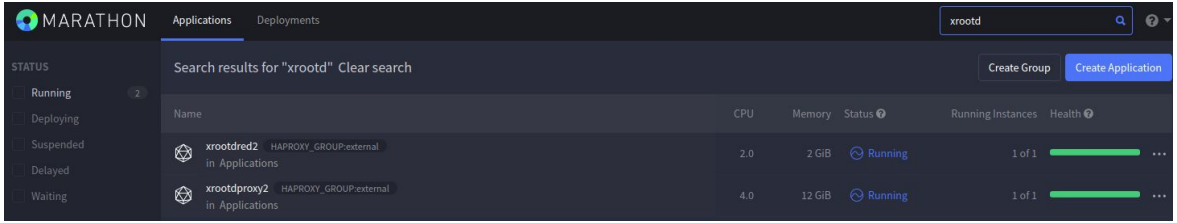


Everything behaves as expected.

Remember that the amount of efficiency gain is heavily **workflow dependent**

Cache management and monitoring

Scaling up and down cache servers dynamically

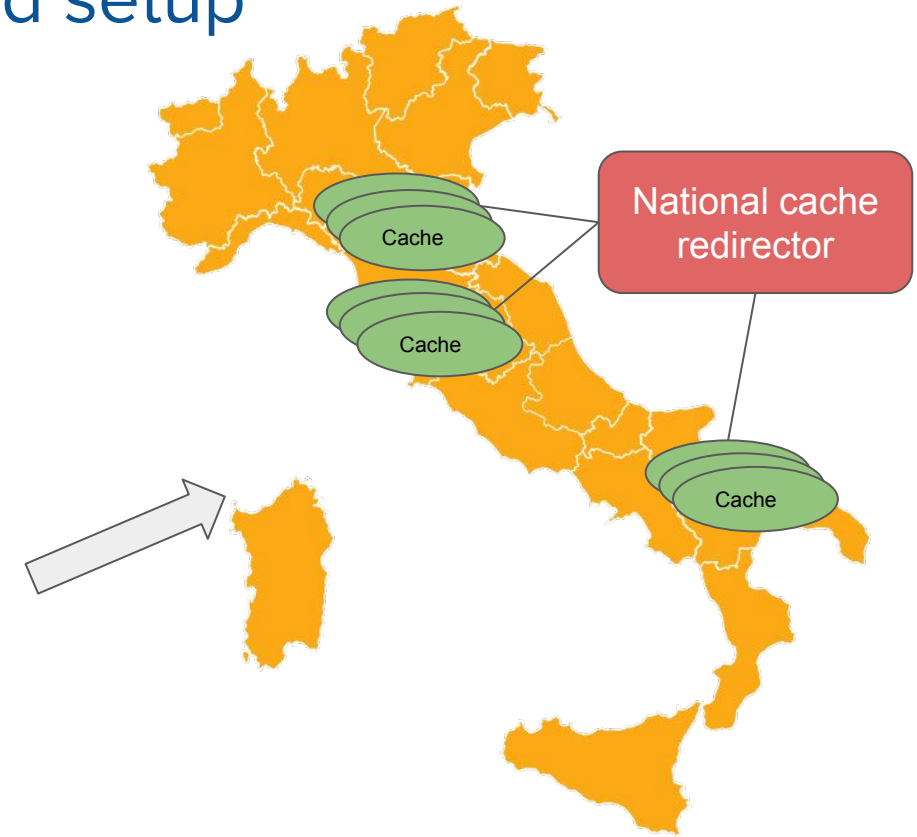


The deployment provides an **integrated monitoring stack** based on Elasticsearch Beats and Grafana Dashboards

Planned activity: distributed setup

NEXT STEPS

- get **quantitative evaluation** of cache performance on **local scenario**
  - different WFs, configuration, backend etc
- **distributed scenario prototype (national level)**
  - **geographically distributed** cache servers
  - **heterogeneous resources and providers**

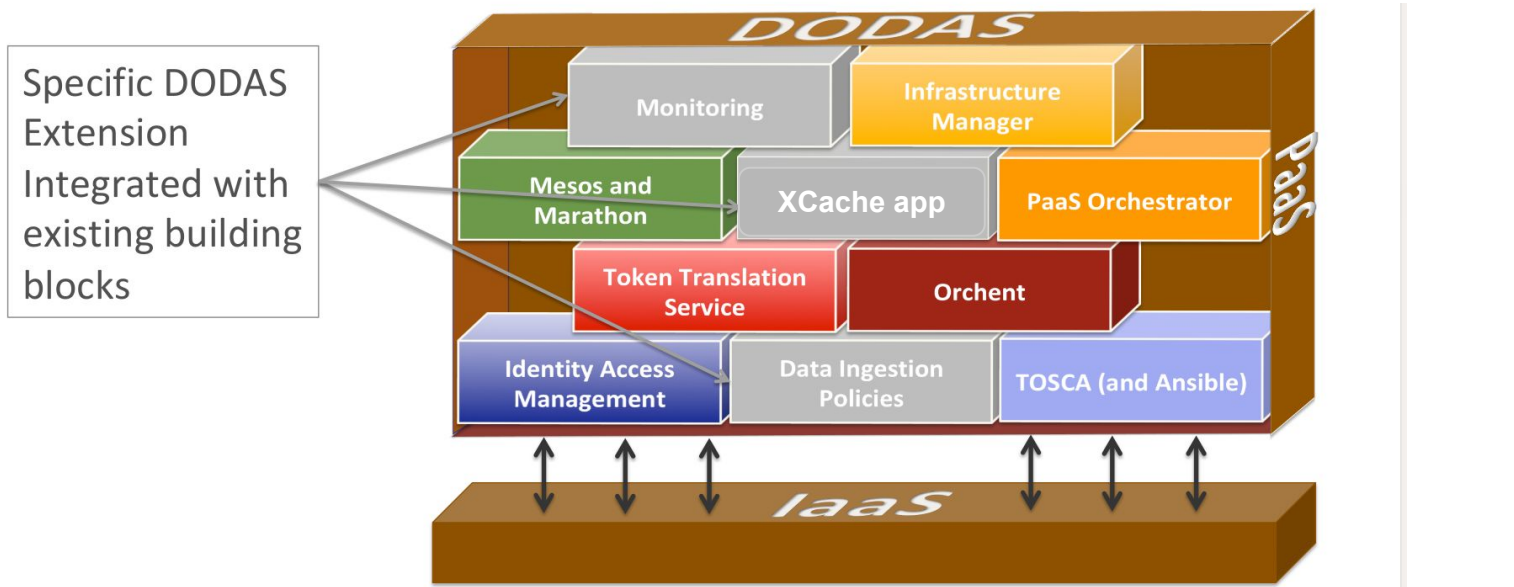


Summary

- Deployed a first prototype of **XCache instance on private and public cloud provider**
- Started to **measure performances on a benchmark workflow**
- **Recipes for automatic deployment** of XCache on cloud resources
- Do we find sinergies with similar activities in CMS ?

IaaS overlay: DODAS integration

- Service for generating over cloud resources an on-demand, container based application deployment.



Tests on local scale scenario

What's needed?

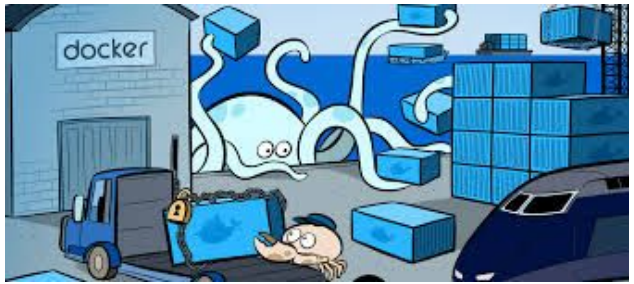
- **XRootD cache packaging**
  - **Docker container** with different configuration
- **Containers orchestration**
  - different solution available: **docker swarm, k8s and marathon pods** (redir+caches)
- **Working at PaaS level**
  - tests with **DODAS integration**

CMS XCache packaging

A **CMS Xcache Docker container** has been setup to allow an easy deployment

Example

```
sudo docker run -v $PWD/config:/etc/xrootd cloudpg/xrootd-proxy --config /etc/xrootd/xrd_test.conf
```



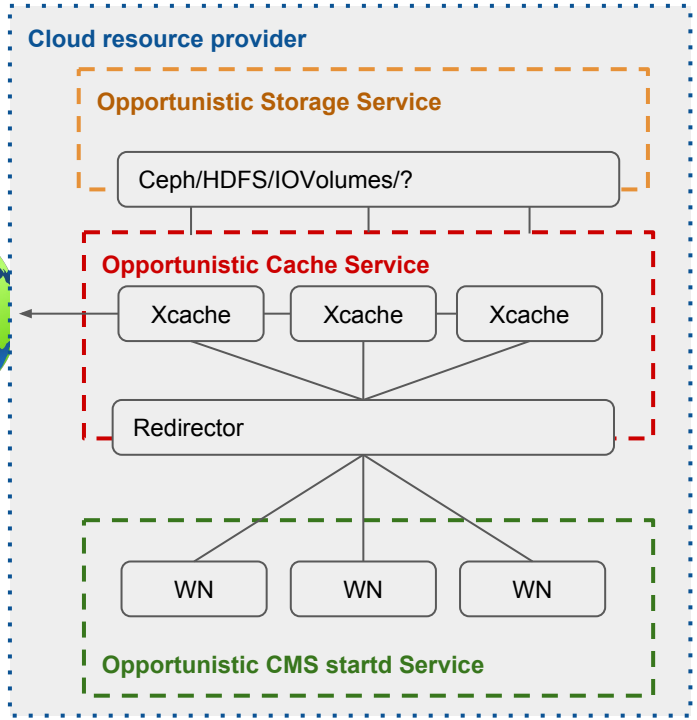
A Docker Compose configuration file is available to **orchestrate the deployment of a local test instance.**

The stack contains a test remote server, a cache instance and cache redirector (preliminary docs here)

Orchestration

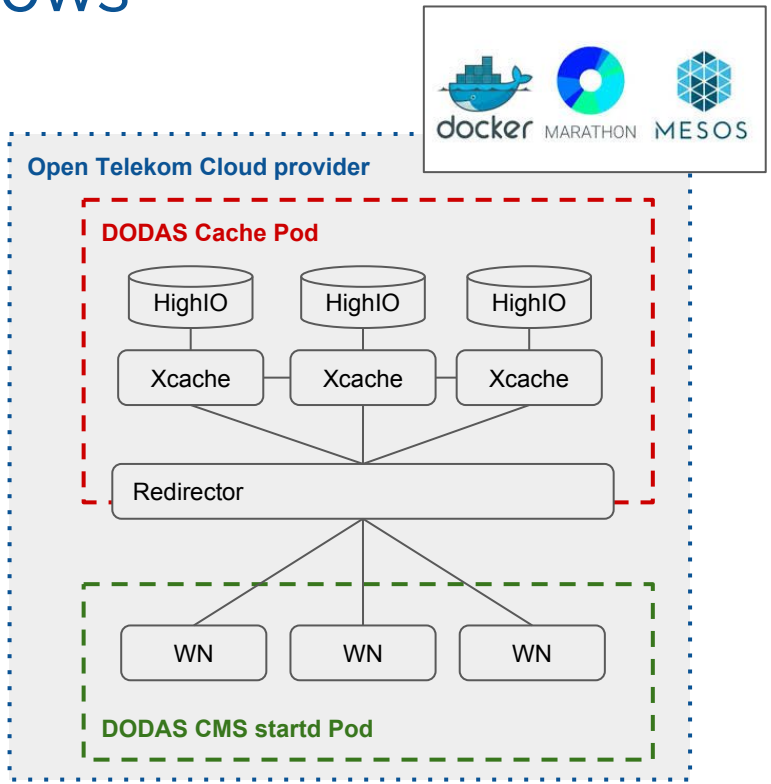
A variety of orchestration tool can be used for the deployment:

- **Docker swarm**
  - deploy and scale services on multiple nodes using docker-compose recipes
- **Kubernetes or Mesos+Marathon**
  - deploy and scale cache services containers as pods with a compose-like recipes



Testing cache with CMS workflows

- A first **proof of concept** has been developed and **first tests with CMS analysis workflows** are ongoing.
  - **DODAS environment** to provide a recipe for an **automatic XrootD cache deployment on cloud resources**
  - utilize “any cloud provider” with almost zero requirements and a **simple text configuration file.**
    - **Open Telekom Cloud (OS based)** resources used for the following results



Cache in CMS: XRootD based cache

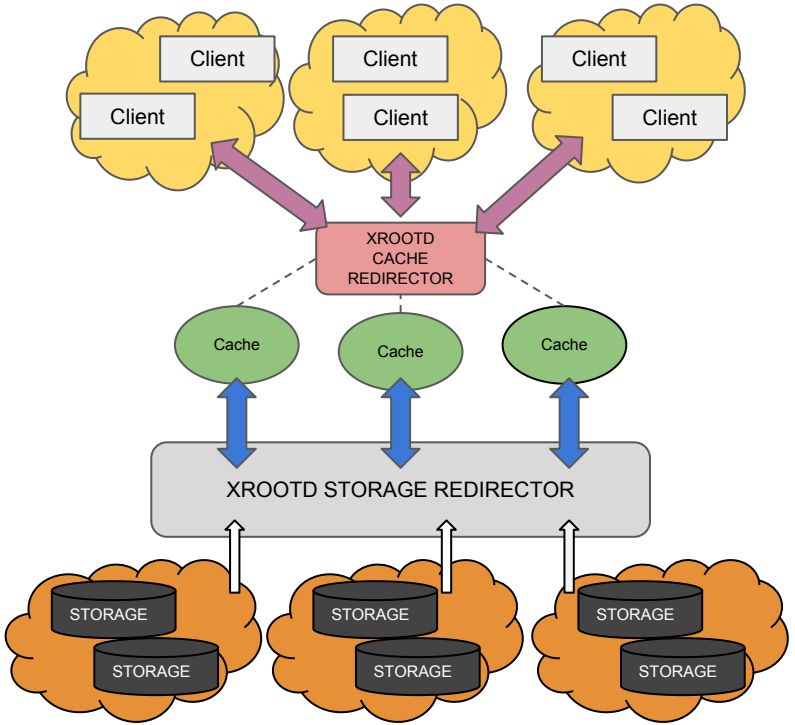
- CMS model can integrate **XRootD caches (XCache) seamlessly**
  - XRootD is widely supported at T1s and T2s
  - XCache is modular and pluggable
  - cluster many caches with a cache redirector
  - already under evaluation by various activities within WLCG

Cache keywords

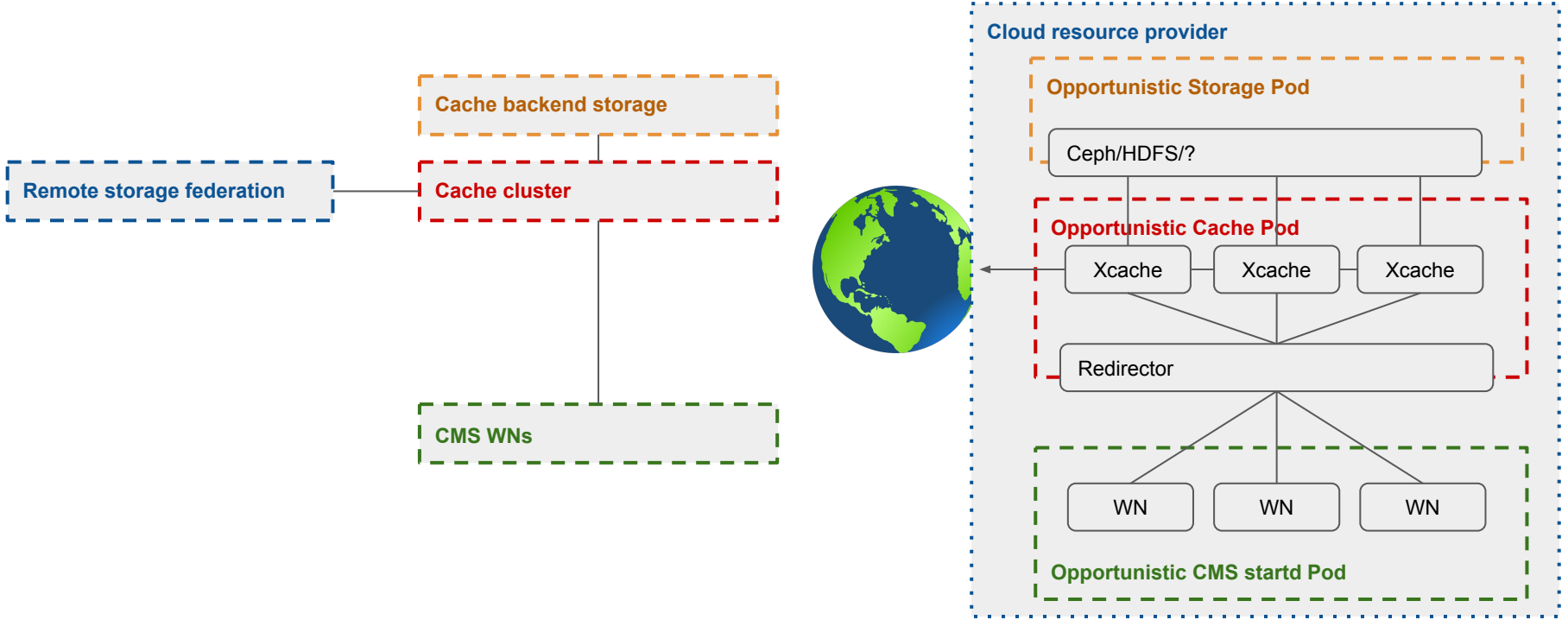
- **Cache metadata**
  - stores details about already downloaded blocks and all local accesses.
- **Prefetching**
  - cache can issue advance read requests to reduce read latency
- **Decision plugin**
  - allows users to configure which parts of namespace are to be cached.
- **Cache purging**
  - e.g. high/low water mark algorithm to start/stop purging. Plugins for smarter algos should be possible

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*)
- **Overloading:** If a file present on **one cache is requested by many clients**, it can be **allowed to be duplicated on other servers**



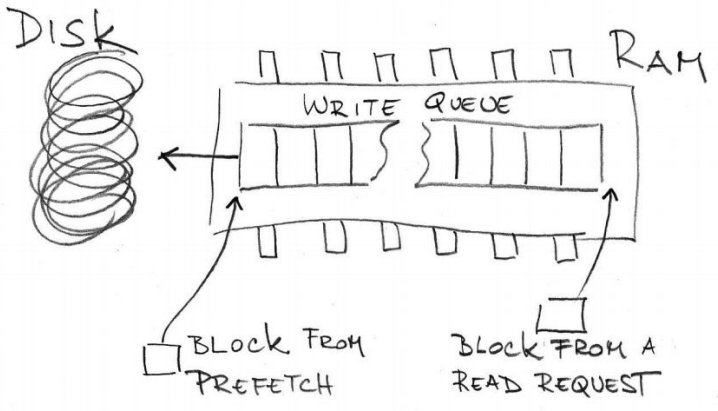
Vision on caching for opportunistic resources



XrootD cache mechanics: write queue

- **Prefetched buffers** are put to the **beginning of the write queue**
  - assume they will be needed at a later time so **RAM should be vacated as soon as possible.**
- **Buffers obtained to serve outstanding read requests** are put to the **end of the write queue**
  - assume they will be needed to serve future read requests so they should be **kept in RAM as long as possible.**

From A. and M. Matevz talk ICEPP2016



XRootd, disk-based, caching proxy for optimization of data access, data placement and data replication. A. T. Bauerdick, L & Bloom, K & Bockelman, B & Bradley, Dan & Dasu, S & Dost, Jeffrey & Sfiligoi, I & Tadel, A & Tadel, Matevz & Wuerthwein, Frank & Yagil, A. (2014). *Journal of Physics: Conference Series. 513. 10.1088/1742-6596/513/4/042044.*

XrootD cache mechanics: overview

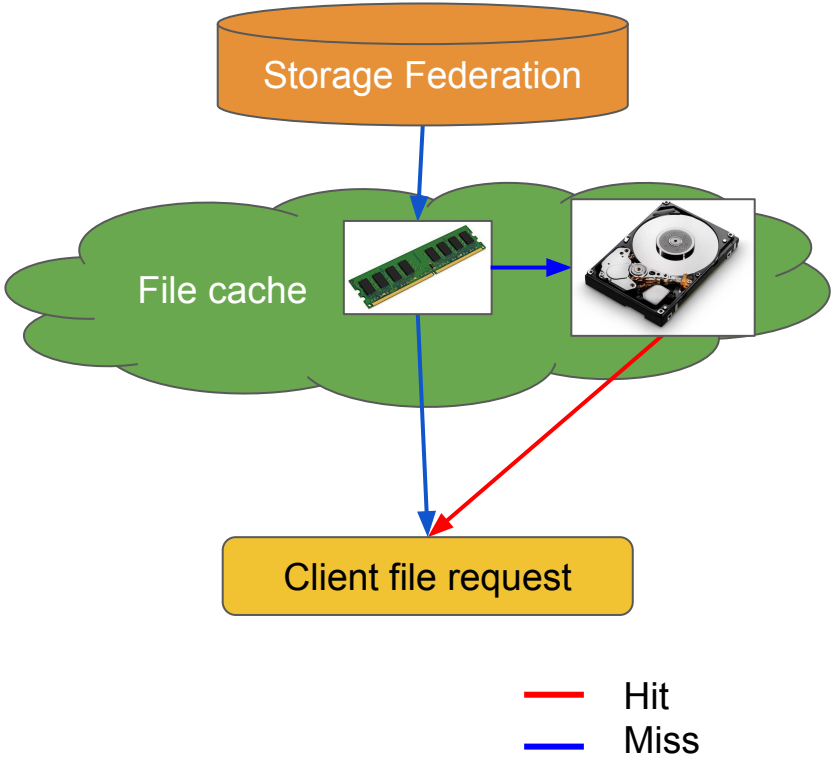
Open File

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)



Dynamic On Demand Analysis Service

Dynamic On Demand Analysis Service (DODAS) is a solution developed in the context of INDIGO-DataCloud project.

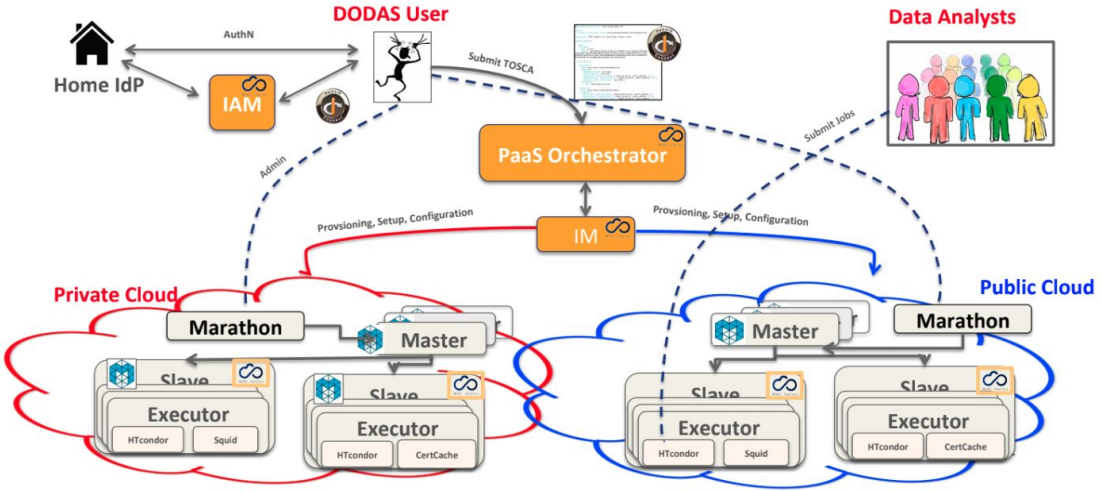
It is a service for generating over cloud resources an on-demand, container based application deployment.

That includes solutions that spans from a standalone HTCondor batch system to a Big Data processing cluster

To support user tailored computing environments

To automate configuration and deployment of custom services and/or dependencies

To define input parameters and customize the workflow execution



XDC scenario

- Lake-C : The central service of the data lake, holding namespace, metadata and making scheduling decisions. The “head node”.
- Lake-MS : A service storing data and under the management of Lake-C. The “disk node”.