Integrazione di una smart cache Italiana federata per CMS

INFN Perugia:
D. Ciangottini
M. Tracolli
D. Spiga

INFN Pisa:
T. Boccali
G. Bagliesi

INFRASTRUTTURA TESTBED

CNAF:
D. Cesini
A. Falabella

INFN Bari:
G. Donvito

INFN Legnaro:
M. Biasotto

Outline

- Pattern di accesso ai dati in job di analisi CMS
- XCache: descrizione tecnologia
- XCache per data-lake
- Testbed INFN
- CachingOnDemand (a.k.a. XCache as a Service)
- Work in progress

Introduzione

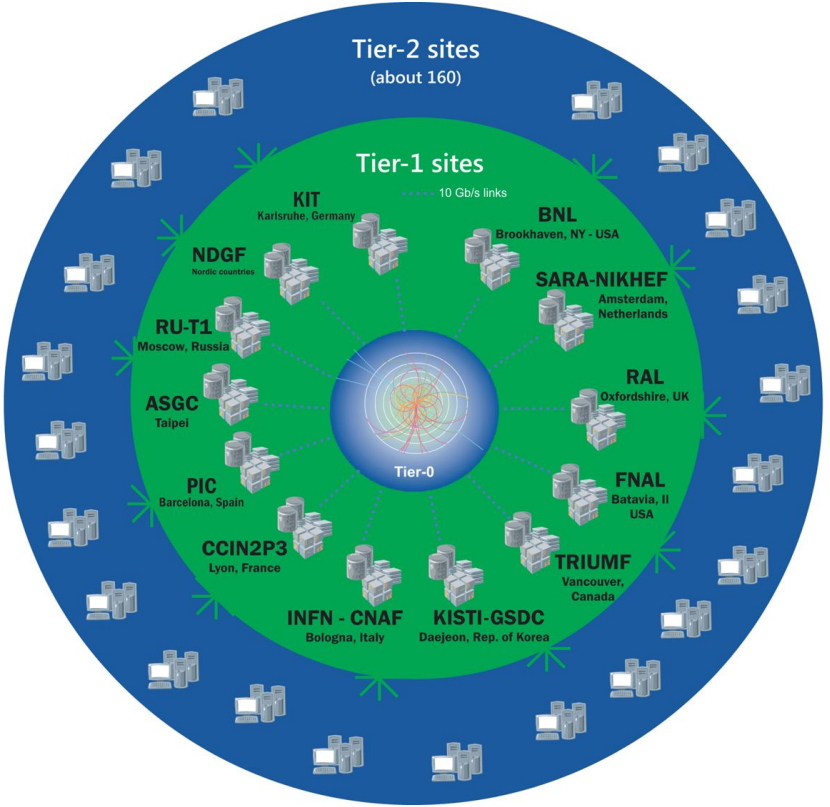
- Obiettivi:
  - Integrazione di un **layer di cache** (basato su XCache) in CMS
  - **Stimare i benefici** prodotti dalla soluzione proposta
- Motivazioni:
  - utilizzare la rete nazionale per **ridurre la quantità di dati e repliche** ai T2 italiani grazie a un **layer di unmanaged storage**
  - **ridurre il costo operazionale** per il mantenimento degli storage
    - **introducendo procedure automatiche**
    - facendo uso di **risorse unmanaged**

Contesto attivita’

- **CMS collaboration**
  - XCache seamless integration per ottimizzazione accesso ai dati per job di analisi
- **WLCG DOMA** (data organization, management, access):
  - XCache come soluzione core per implementazione di un primo modello Data Lake
- **eXtreme DataCloud**
  - CachingOnDemand: deployment automation per XCache su risorse cloud
- **ESCAPE**
  - cache come data access tool per data lake in comunità scientifiche
- **IDDLS**
  - sperimentazione di infrastruttura nazionale per data lake

CMS modello attuale “in a nutshell”

- Struttura gerarchica **gestisce centralmente il contenuto degli storage ai computing sites** (Tier)
- I job **girano nel sito in cui sono i dati richiesti**
- **L’accesso ai dati remoto avviene in casi particolari:**
  - fallback in caso di failure della lettura locale
  - overflow di job verso siti vicini piu’ liberi
  - gli user specificano di ignorare la data location



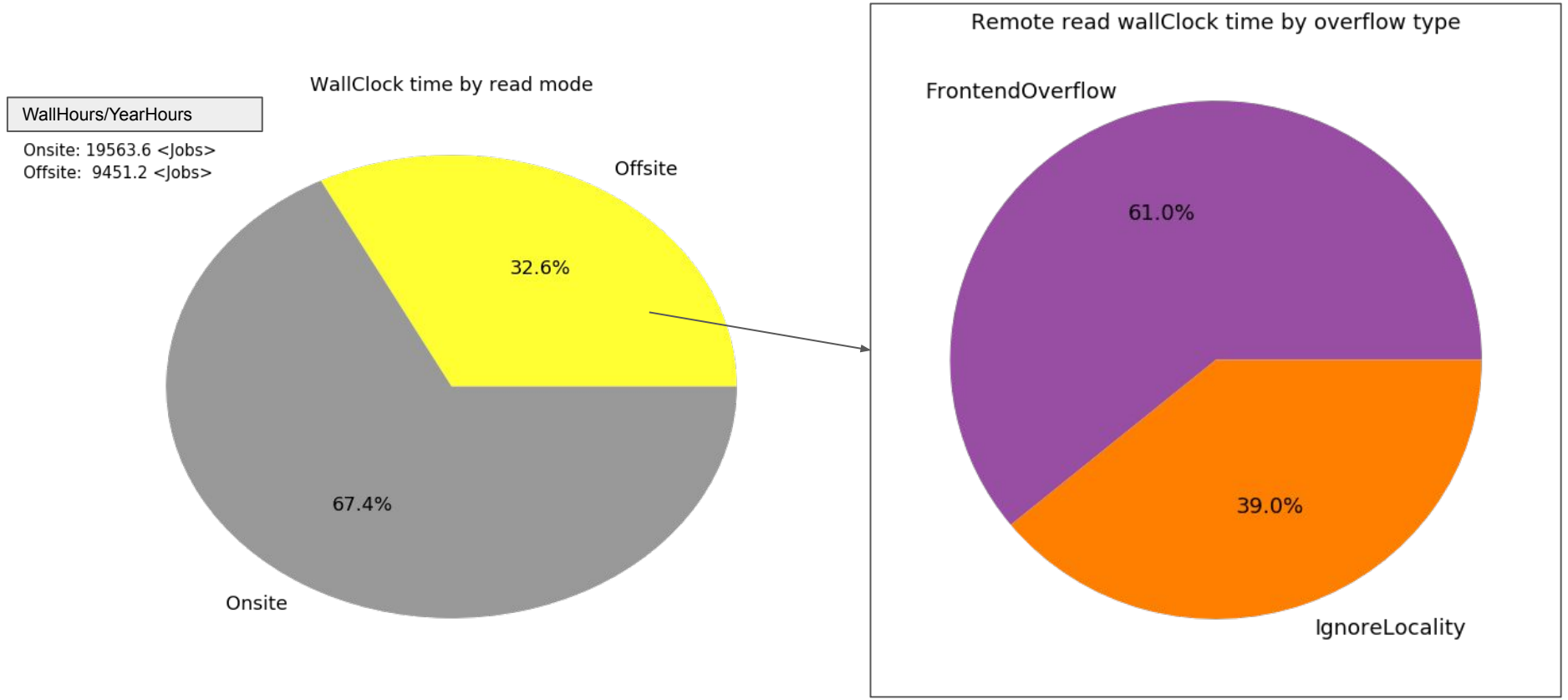
Metriche di accesso ai dati CMS

- **Job di analisi CMS:** situazione 2018
- HTCondor ClassAds
- Focus su wall, CPU time e CPU Efficiency
- Interessante confrontare performance dei job con lettura locale contro i due tipi principali di lettura remota 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"

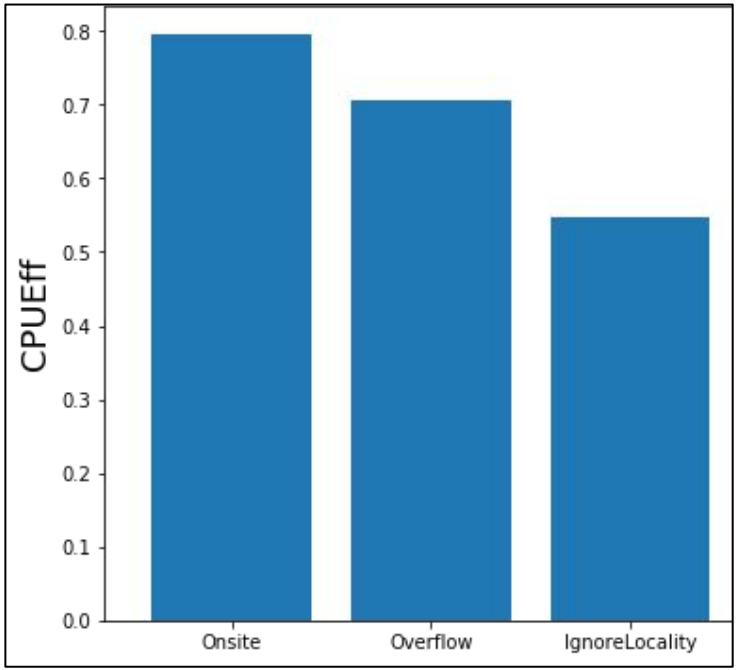
"User can configure their jobs to ignore the data location and for them to run on a custom list of sites"

Frazione di lettura remota



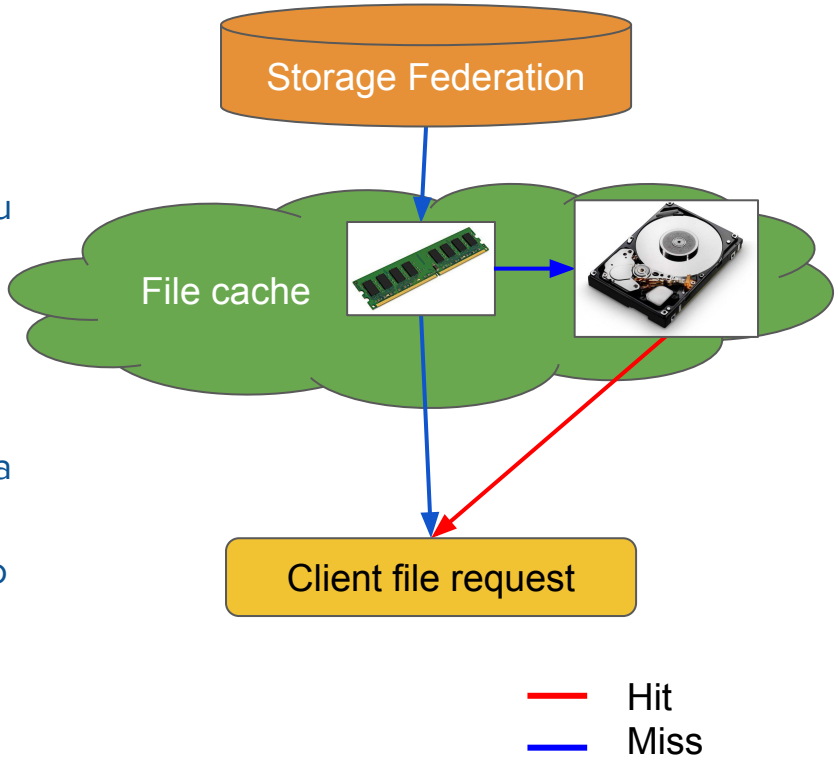
CPUEff per modalità di lettura

- In media la CPU efficiency presenta un significativo peggioramento nel caso di lettura remota
- Per ora ~OK visto che la lettura remota sta ancora su circa il 15% del totale di workflow utenti:
  - si perde ~5% della cpu time rispetto ad avere tutto in locale
  - ma si guadagna spazio disco
- In scenario data-lake o comunque in scenari dove la % di remote la perdita diventa importante



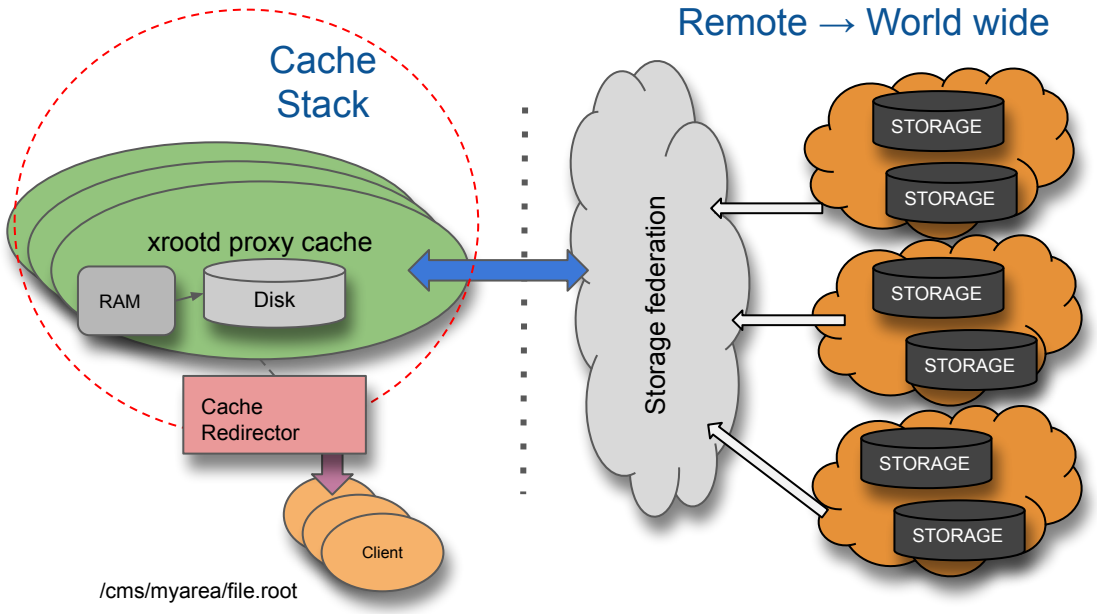
Usare la soluzione XCache

- Proxy server con capacità “read-ahead” in caso il file non sia già disponibile nello storage e/o in memoria
  - i dati, una volta serviti in streaming, vengono messi in una coda di scrittura per essere salvati su disco
  - in caso di overload viene evitato il salvataggio su disco e procede in modalità proxy
- Più server possono essere “federati” sotto un unico namespace attraverso un redirector dedicato ➔ una sola copia del file sotto lo stesso cluster
- Scelta LRU per file da rimuovere una volta che lo spazio occupato supera il valore di “High water-mark” (configurabile)



High level view of our setup

Schema setup XCache



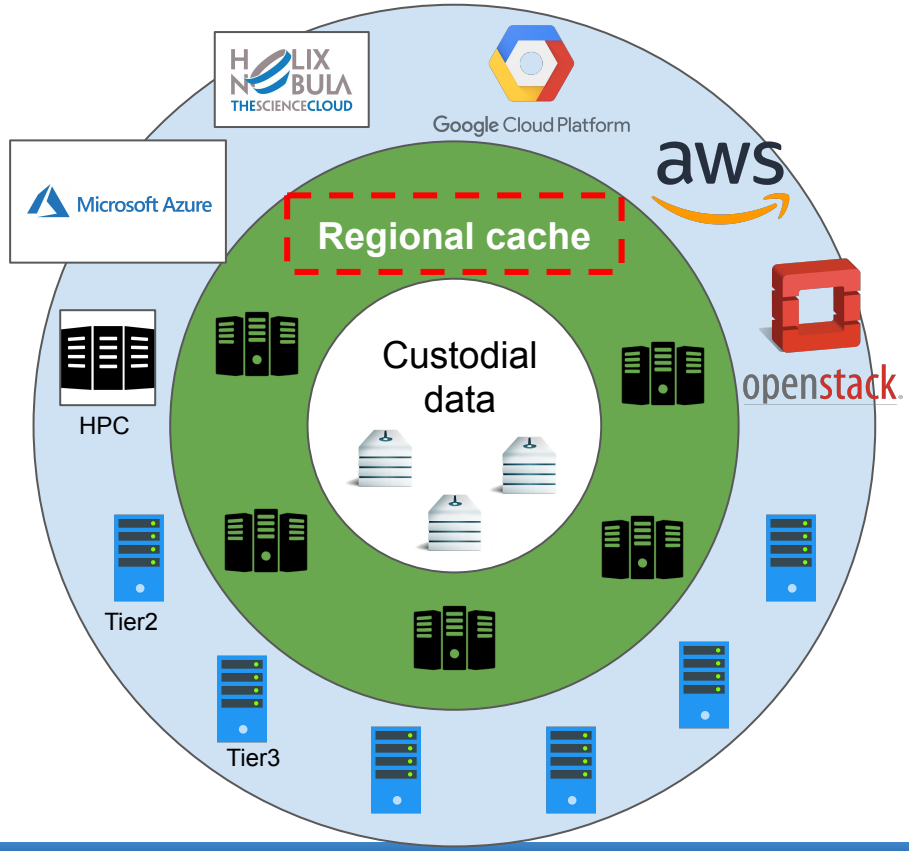
Cache role in Data Lake model at WLCG

Pochi siti “custodial” distribuiti world-wide

- i computing Tier **accedono direttamente al sito custodial più vicino**

L’idea e’ di usare le cache per:

- rete geo-distribuita di **unmanaged storages**
  - **storage non ridondato**
- common namespace tra cache servers
  - **no data replication**
- **request mitigation** verso custodial sites

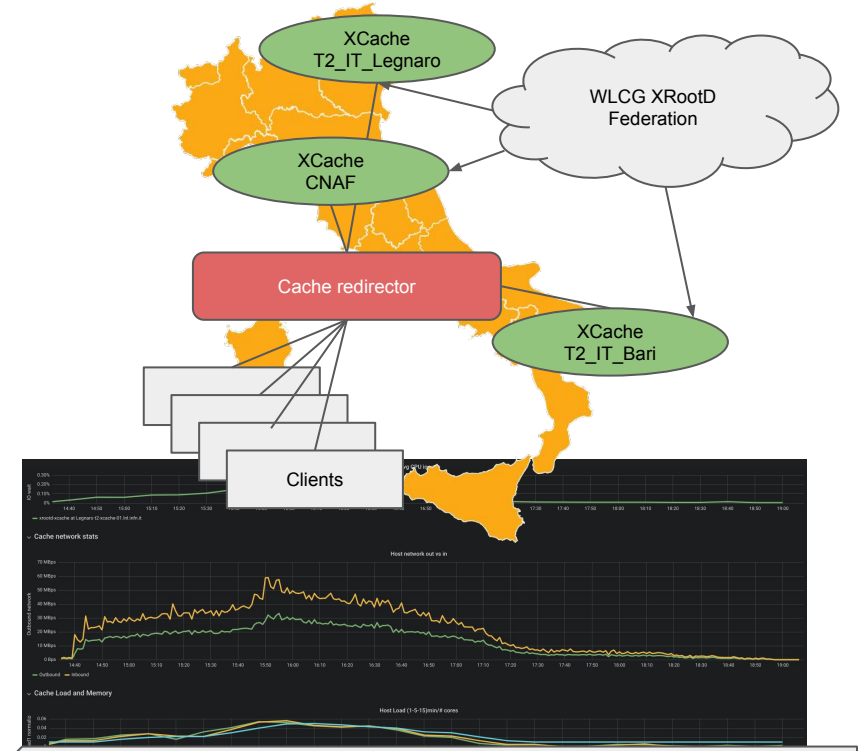


CMS XCache testbed @INFN

- L'idea e' di creare un testbed con **risorse volontarie** (no nuovo hw dedicato) per valutare la fattibilità e la funzionalità' del setup
- **Integrazione per l'utilizzo con CMS workflows e' stata trasparente per gli utenti**

Prototipo funzionante da meta' 2018 su 3 Tiers (CNAF, Bari, Legnaro) con cache redirector @CNAF.

Task di analisi reali utilizzano l'infrastruttura in maniera trasparente, ma a scala limitata (~60 client), non essendo lo scopo del setup attuale

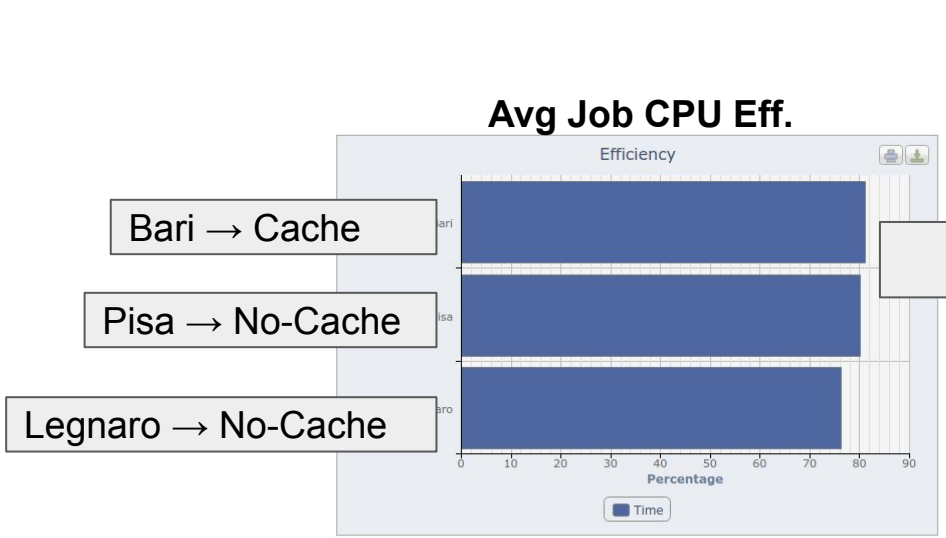


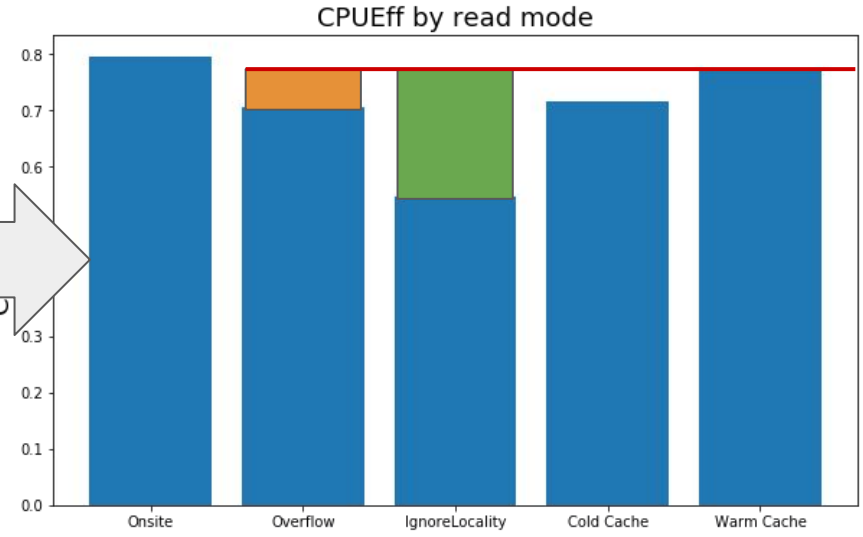
N.B. infrastructure provisioning grazie alla collaborazione con CNAF, Legnaro and Bari

Esempio di integrazione

Test di funzionalità e prime misure

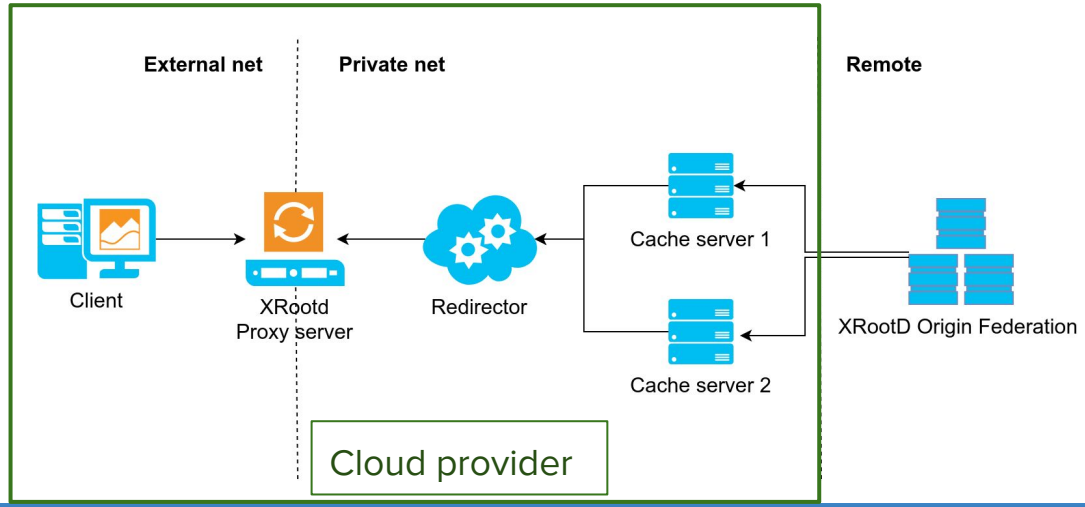
- CMS test tasks sottomessi a T2_IT_Bari partendo da **cache vuota**
- **Nessuna evidenza di penalità in CPU eff** partendo da cache vuota
  - anzi lieve miglioramento grazie a read-ahead





Installare lo stack: XCache on demand

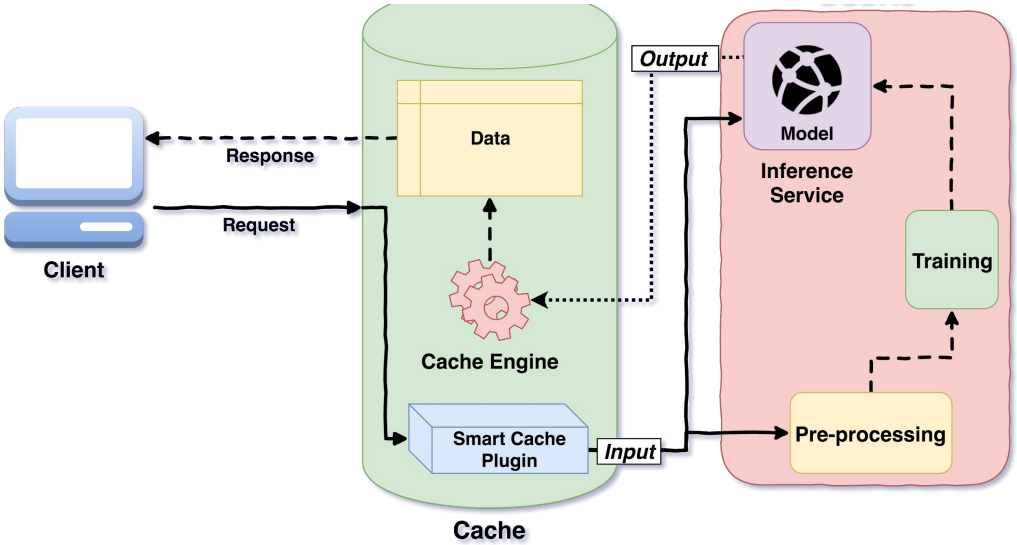
- Sono state create procedure automatiche per l'istanziazione dello **stack XCache on-demand**
  - **ricette ready-to-use** per installazione sia bare metal che cloud
- Documentazione disponibile per:
  - Ansible
  - Docker compose
    - (DEMO in doc)
  - Kubernetes (DEMO in doc)
    - Helm chart con sidecar per proxy renewal



Ricette in CachingOnDemand

Attività in corso

- **Simulazione comportamento cache** usando storico CMS
- **pianificazione di un testbed** production-like
- Integrare modell **ML-based per smart caching decision**



PoC funzionante con modello ML Dummy, ma verificata la fattibilità

Grazie ad attività PhD di M. Tracolli @PG sia per infrastruttura che modellizzazione

Conclusioni

Dati storici in CMS confermano utilità sistema di cache e permettono una prima stima del guadagno.

Primi test funzionali e misure di performance su workflow reali sono state effettuate sul testbed nazionale

Istanziazione dinamica di una cache su cloud resources e' completamente automatizzata e replicabile su cloud sia pubbliche che private