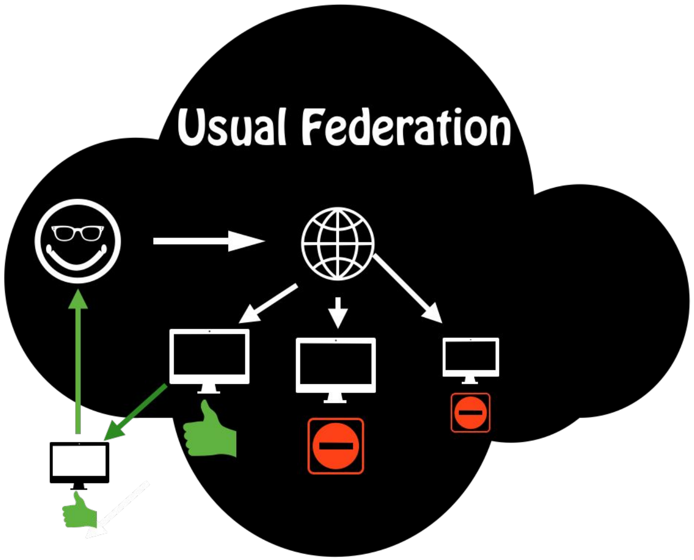
# Moving the California distributed CMS XCache from bare metal into containers using Kubernetes

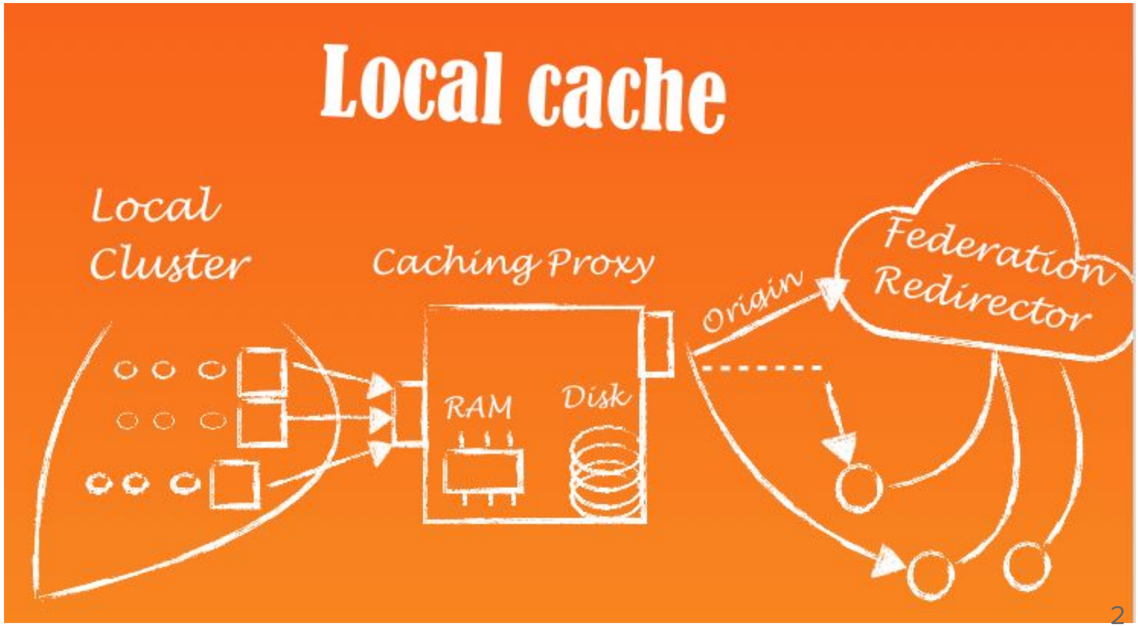
Edgar Fajardo[1], presented by Matevž Tadel[1]
Justas Balcas[2], Alja Tadel[1], Frank Würthwein[1], Diego Davila[1], Jonathan Guiang[1], Igor Sfiligoi[1]

Caltech[2], UCSD[1]

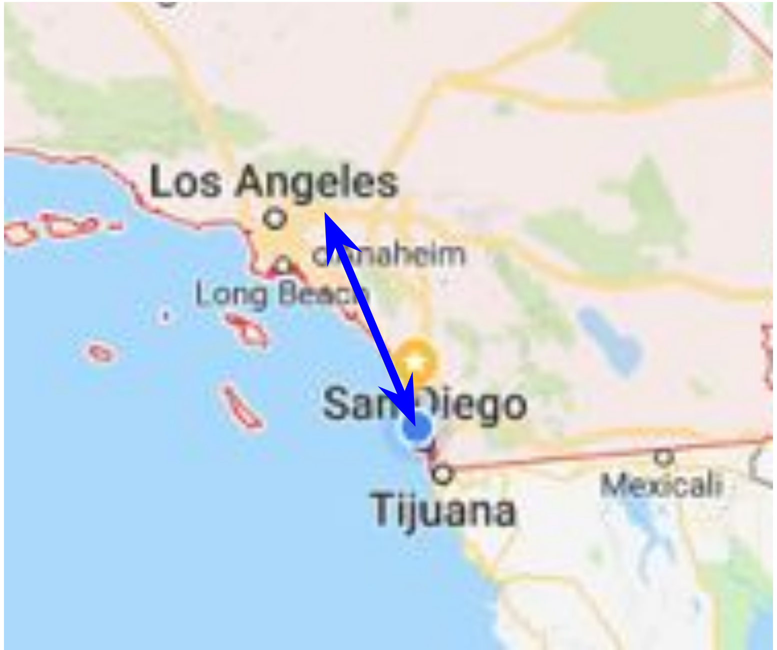
# What is an XRootD Cache? Caching Cluster!

Each node acts as an independent cache. But they can be clustered through cmsd to scale horizontally.



# Opportunity for two sites to merge some namespace and profit from closeness




UCSD ↔ Caltech link:

- 120 miles
- 100 Gbit/sec
- below 3 ms

# Recipe: How to setup an XCache for CMS at a given site(s)?

1. Decide on the namespace to cache. (We used /MINIAOD*)
2. Calculate the working set (the set of unique files of the namespace that are accessed on a given period).
3. Provision the storage for some fraction of data from step #2.
4. Install and configure XCache on top of disk from step #3.

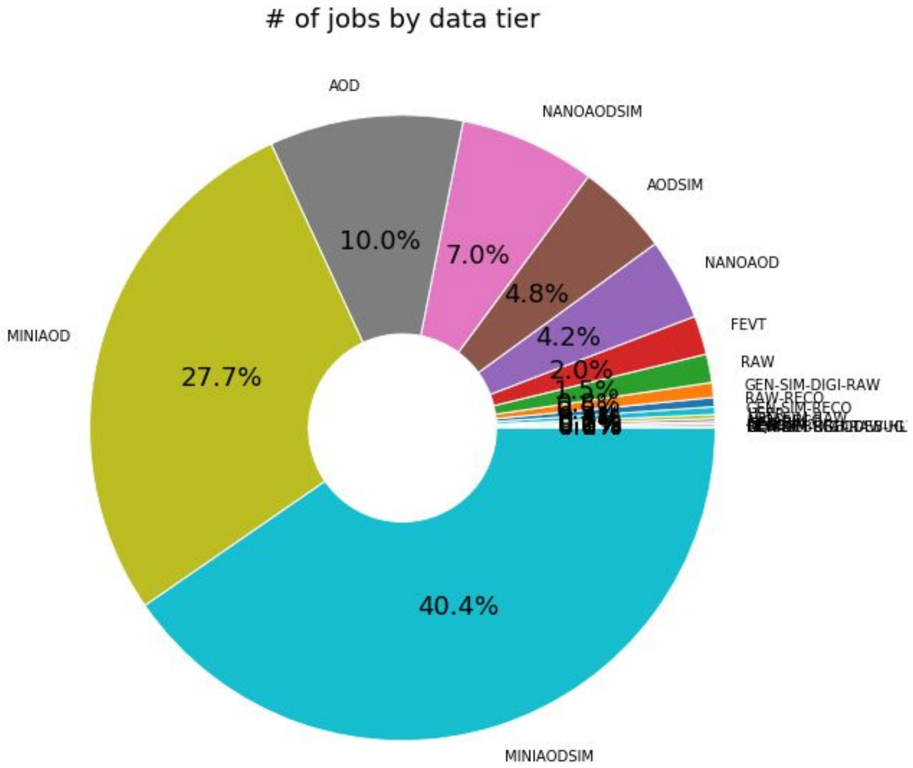
This is where kubernetes kicks in

# 1. Decide on the namespace

Plots from Diego Ciangottini, see May 15, CMS Computing & Offline meeting:

http://dciangot.web.cern.ch/dciangot/all_18/

http://dciangot.web.cern.ch/dciangot/all_19/




~70% of CMS analysis jobs used MINIAOD* in 2019

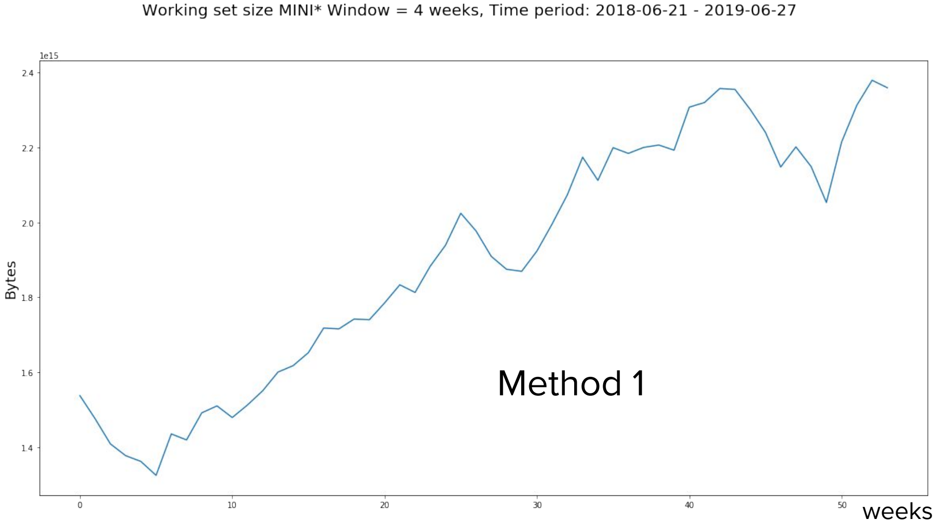
# 1. Decide on the namespace

| Dataset | Size (PB) |
|---|---|
| /*/Run2016*-03Feb2017*/MINIAOD | 0.182 |
| /*/RunIISummer16MiniAODv2-PUMoriond17_80X_*/ MINIAODSIM | 0.5 |
| /*/*RunIIFall17MiniAODv2*/MINIAODSIM | 0.2 |
| /*/*-31Mar2018*/MINIAOD | 0.14 |
| Total | 1.04 |
| /*/*/MINIAOD | 2.92 |
| /*/*/MINIAODSIM | 4.6 |
| Total | 7.52 |

Until end of september 2019 we had a more restricted namespace.

Now with more hardware we relaxed the constraints.

# 2. Estimating the working set for SoCal




## Method 1:

1. Look at the unique MINI* **data-sets** accessed globally (at all sites) within a four week window and calculate their size.
2. Move the window 1 week at a time for a year worth of data from the Global pool ClassAds
3. Results: The **monthly working set** is somewhere between 1.6PB and 2.4 PB

## Method 2:

1. Look at the unique MINIAOD* files accessed in SoCAL during the month of October.
2. Estimate the month working set as the size of all the unique files accessed during October
3. Results: 451TB

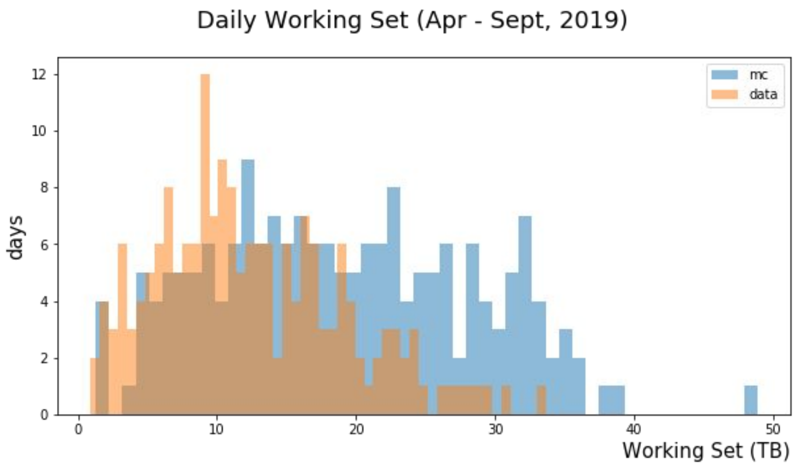
On going work to estimate the size and cost of the cache see Andrea's talk "**Data access pattern analysis and modeling**"

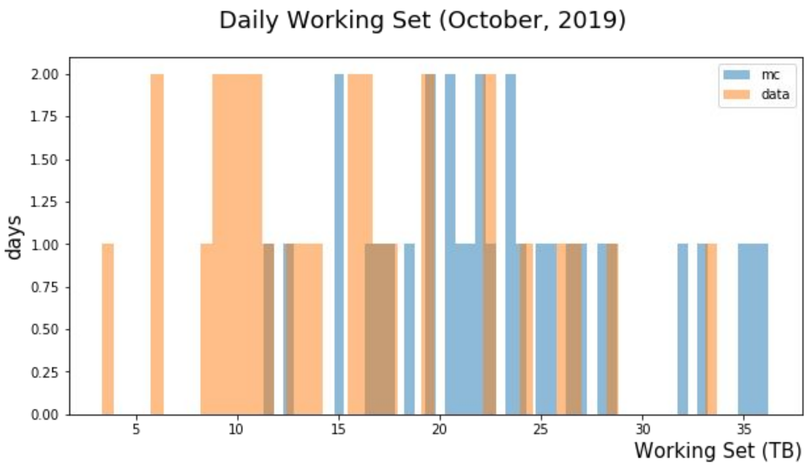
# 3. Provision disk infrastructure based on the needs in step #2.

| | UCSD | Caltech |
|---|---|---|
| **Nodes** | 11 (+1 JBOD) | 2* |
| **Disk Capacity per node** | 12 x 2TB = 24TB (+ 48 x 11TB) | 30 x 6TB (HGST Ultrastar 7K6000) |
| **Network Card per node** | 10 Gbps (+ 40 Gbps) | 40 Gbps |
| **Total Disk Capacity** | 264 TB (+ 528 TB) = **792 TB** | **360 TB*** |
| **TOTAL** | 792 TB + 360 TB* = **1,152 TB** | |

* Caltech has 1 JBOD (440TB) ready to be added anytime (currently in HDFS managed space)

# 3. Provision disk infrastructure (Working set for SoCal)



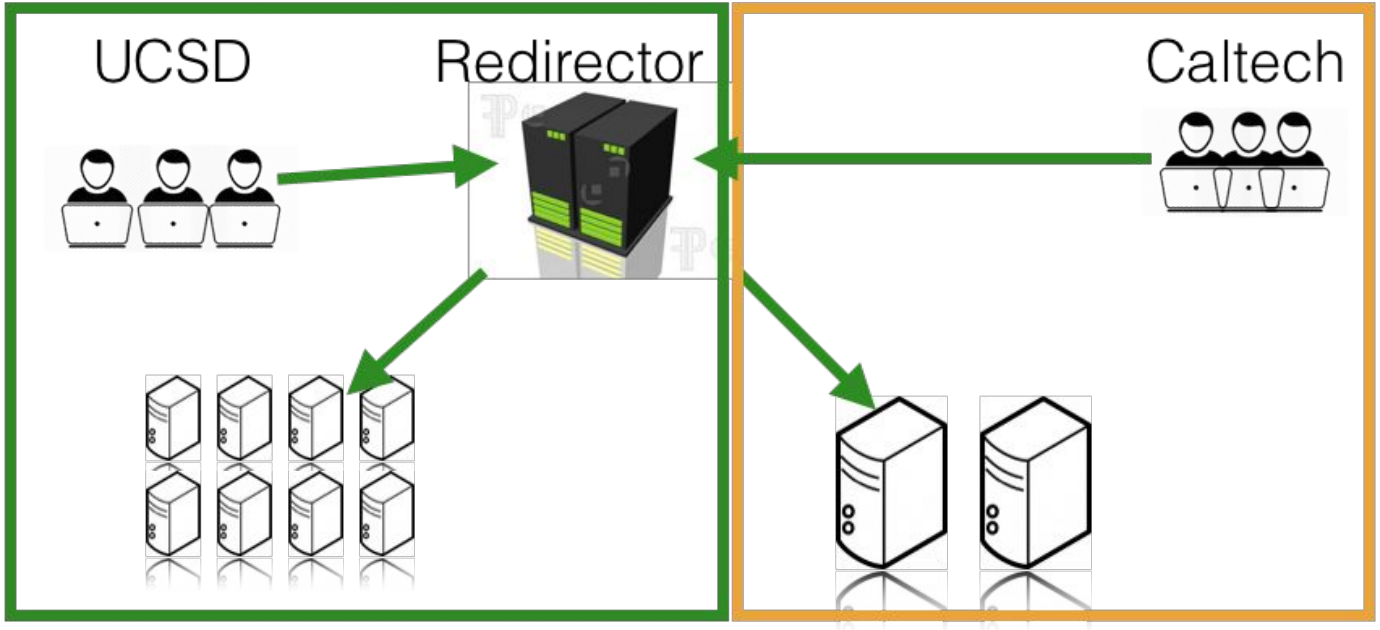




- The daily working set for SoCal was at most 80 TB = 48 TB MC + 32 TB Data.

- The daily working set for SoCal for October was at most:
  70 TB = 35 TB Data + 35 TB MC.

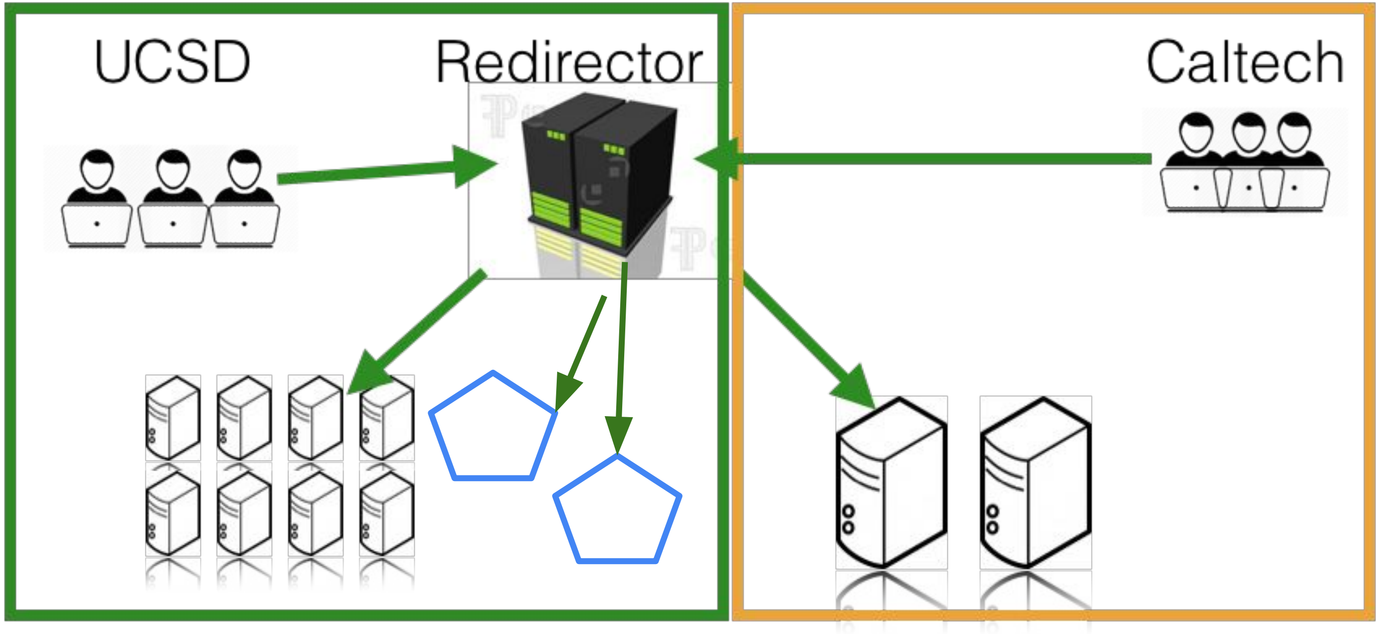
# 4. Install and configure XCache

2018 setup



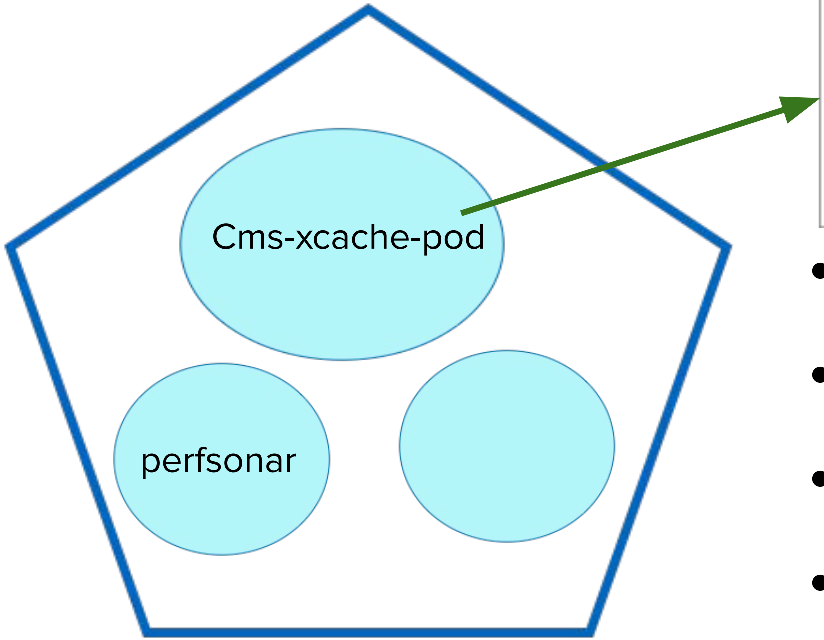

Each server acts as an independent cache and through the redirector they all work as a logical cache

# 2019 and forward Setup




All new XCaches are installed via kubernetes. Two of them currently installed this way.

# How do the pods look inside?




Physical machine with kubernetes installed


- CMS XCache image provided by OSG
  - Based on OSG CMS XCache RPM
- We added the extra configuration via kubernetes knobs
  - Our github repo with k8s yaml is here
- We are not using virtualized network capabilities: we bind the pod to the network interface.
- Authentication comes from the RPM and is done via LCMAPS
- We use the PRP kubernetes federation (See Igor's talk)

# Recipe: How to setup an XCache for CMS at a given site(s)?

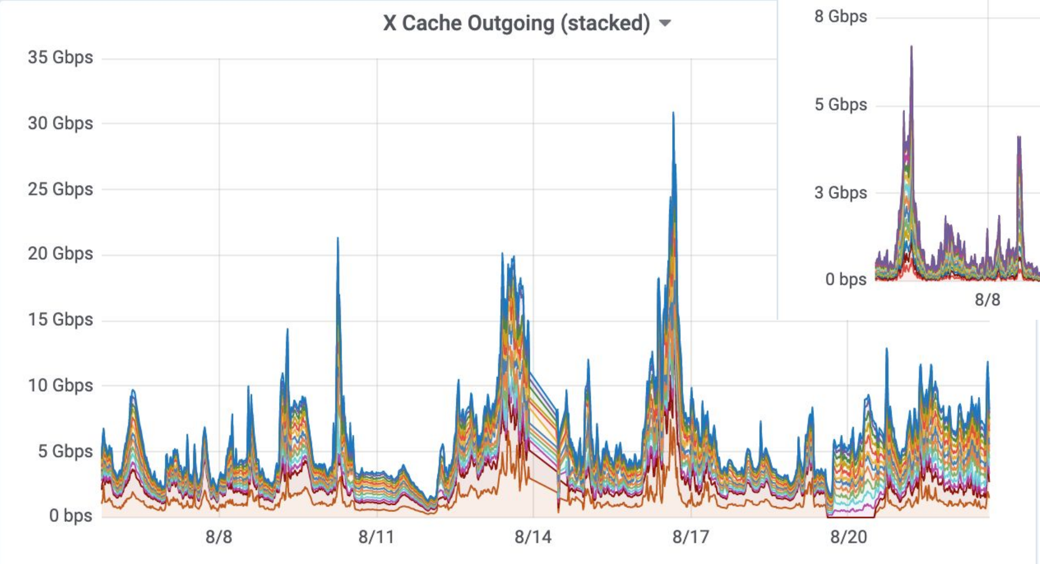
1. Decide on the namespace to cache. (We used /MINIAOD*)
2. Calculate the working set (the set of unique files of the namespace that are accessed on a given period).
3. Provision the storage for some fraction of data from step #2.
4. Install and configure XCache mentioned on step #3.
5. Monitor
6. Experiment with some changes
7. Compare
8. Conclude

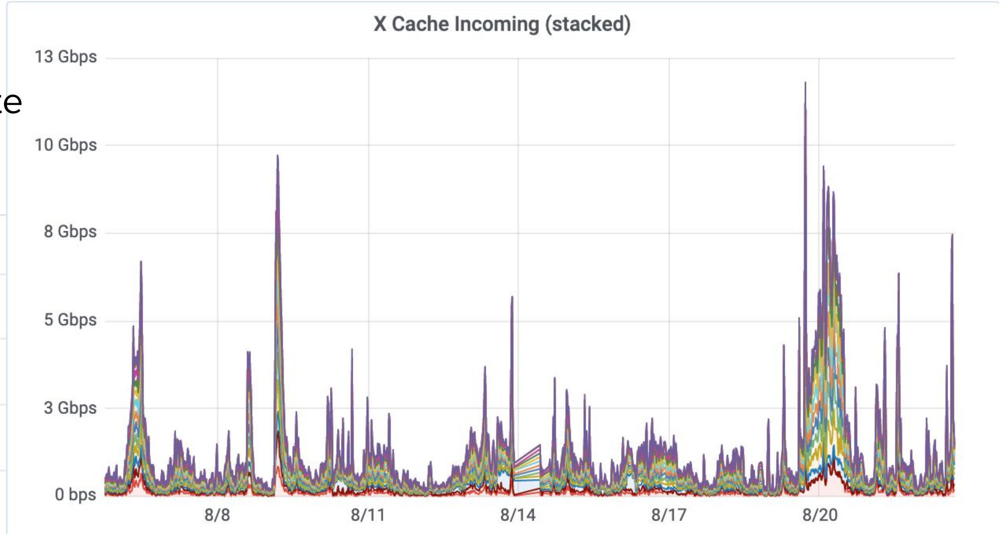
# Monitoring

- Right now monitoring comes from several sources:
  - Telegraf + influxDB + Grafana on  the bare metal
  - Monit job information (failure rates, IO times, etc.)
  - Kubernetes own network monitoring (Prometheus)
  - XRootD monitoring dashboards going through monit.

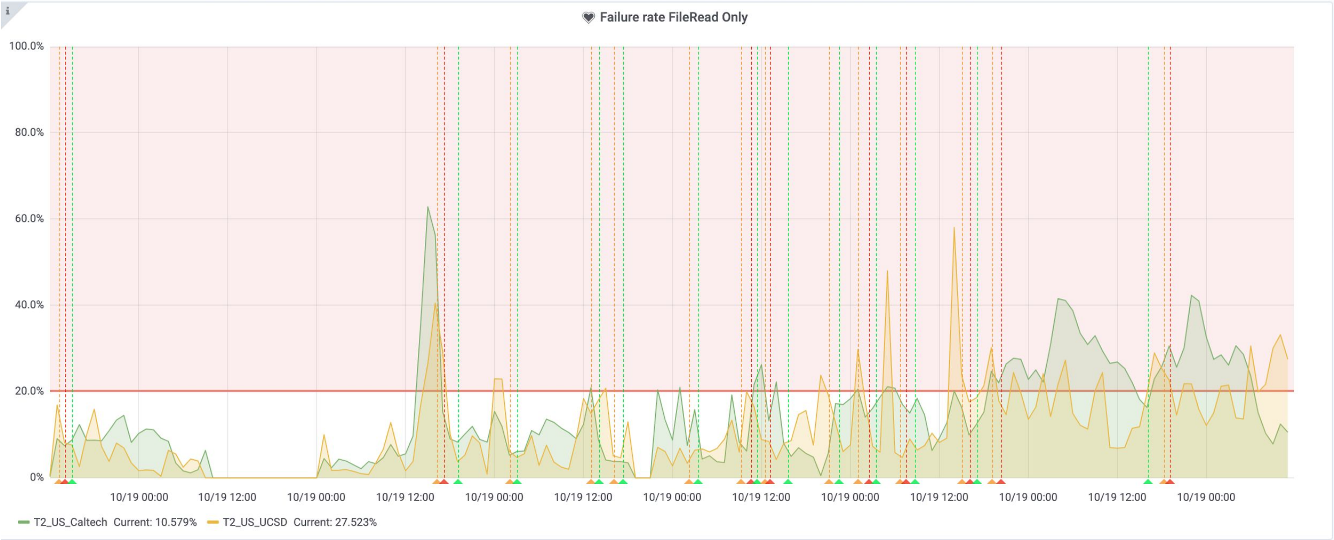
# Telegraf + InfluxDB + Grafana (Bare Metal)

Allows us to approximate instantaneous hit rate



# Monit job information (failure rate)




Failure rate for CMS Global pool jobs for only MINI* for SoCal sites. But also: CPU Efficiency, AvgReadTime, AvgInputSize, AvgOutputSize.

# Experiment with some changes

At the end of October we made a switch to have MINI* jobs running at **Caltech** read from **AAA and local hadoop** while **UCSD** jobs use the **XCache**.

**UCSD**

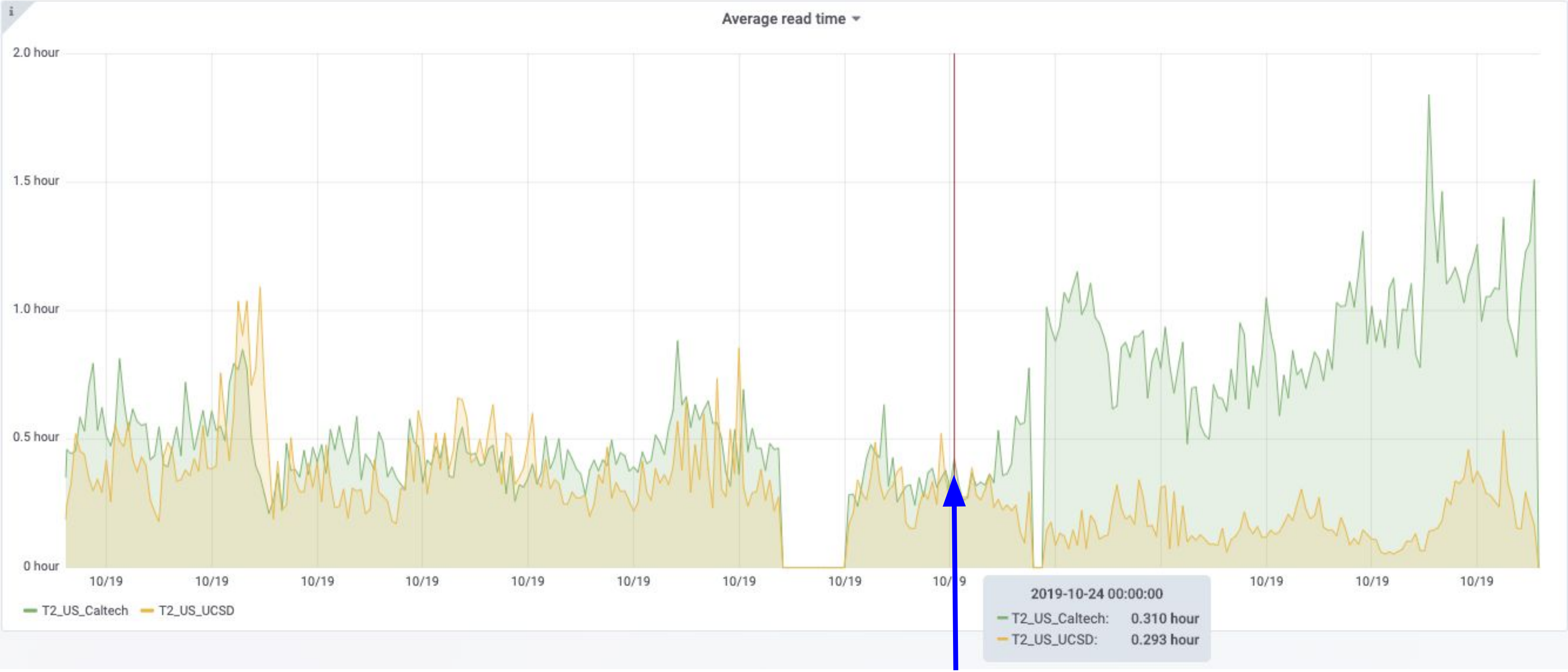
```
# Portions of /store in xcache
  <lfn-to-pfn protocol="direct" destination-match=".*"
    path-match="/+store/(data/.*/.*/MINIAOD/.*)"
    result="root://xrootd.t2.ucsd.edu:2040//store/$1"/>
  <lfn-to-pfn protocol="direct" destination-match=".*"
    path-match="/+store/(mc/.*/.*/MINIAODSIM/.*)"
    result="root://xrootd.t2.ucsd.edu:2040//store/$1"/
```

**Caltech**

```
   <lfn-to-pfn protocol="hadoop" destination-match=".*"
path-match="(.*)" result="$1"/>

   <!-- Xrootd fallback rules -->
   <lfn-to-pfn protocol="xrootd" destination-match=".*"
path-match="/+store/(.*)"
result="root://cmsxrootd.fnal.gov//store/$1"/>
```

# Compare: Avg Read Time




Caltech made the change here (Oct 24 - 2019)

# Future Work

- Investigate other deletion paradigms beyond LRU.
- Have caches serve as data origins to other caches.
- XCache to advertise to DDM's their capabilities rather than their state:
  *I can gather /foo rather than I have /foo*
- We will be installing a second set of caches for NanoAOD with a joint project with ESNet.
- We will be installing NanoAOD (100TB) caches in some US Tier 2 sites.
- Combine all monitoring sources in a sensible way so one can look only at a few plots and alerts and know the status of the cache.

# Conclusions

- The current SoCal cache has been successful at efficiently using the disk space allocated to only store datasets that are actually used.
    - Drastically reduces Average Read Time per job
- UCSD and Caltech successfully operated a service for the region that can grow horizontally based on needs.
- Working closely with other efforts in CMS:
    - INFN regional caching initiative
    - NANOAOD caches on all US CMS sites
- Influences data-storage & distribution strategies for future computing models and data lakes.

# Acknowledgements

- The authors would like to thank the funding agencies for this work, in particular the following grants:
    - PRP: NSF OAC-1541349
    - OSG: NSF MPS-1148698
    - IRIS-HEP: NSF OAC-1836650
    - US CMS Ops: NSF MPS-1624356
    - TNRP: NSF OAC-1826967
- To the CMS experiment and central submission infrastructure team
- To the PRP kubernetes federation admins
- XRootD Developers