CERN

# Analysis of Data Access Patterns in ATLAS and CMS

Andrea Sciabà, Markus Schulz, Shreya Krishnan, Olga Chuchuk [CERN] Carlos Pérez Dengra [PIC/CIEMAT]

# Introduction

- Data access/storage by production and analysis jobs drive the design and cost of data management systems for LHC computing
- At the scale of HL-LHC the current storage and access strategies will not be affordable
- Several concepts are studied:
  - Fewer persistent storages and cache/buffers at smaller sites
  - Tape/Disk dynamic usage
  - See WLCG-DOMA talks
- A first step is to study access/storage patterns and use them to model different architecture choices

# What have we used?

Many thanks to ATLAS and CMS for the data!

- C
  - S... park cluster
  - Cover... roduction and analysis data
  - No data o... gement
- ATLAS Panda a... logs
  - JSON and AVRO file... n laptop)
  - Production and analysis ... /ntuple generation
  - Data access and manageme...
  - No logs on the individual local anal... les
- For PIC and CERN we use logs from ... e systems
  - dCache and EOS

# What have we looked at?

- Cache simulations
  - Sizes, retention strategies, data tiers
  - Differences between sites
  - Details on selected sites

- Access patterns over time
  - Access frequencies
  - Time structure of accesses
  - Data lifetime and active lifetime
  - Details on selected sites

- Too much to show in 15 minutes .....

# Why use a site cache?

- What is it?
  - A storage system that creates replicas of files on demand and appropriately deletes them to reclaim space
    - To hide latency a cache can read forward
  - The most used implementation is XCache, which supports xrootd
- What are the advantages?
  - By caching a file:
    - Subsequent accesses to it will not cause additional WAN traffic
    - Impact of latencies can be reduced
    - The application needs not to be aware of where a file is located
    - No prior staging is needed, no deletion either
  - A cache is simpler to manage than a classic storage system
    - (dCache, DPM, etc.) and allows for a lower and cheaper QoS
  - By using storage space only for “hot” data, much less storage needs to be purchased

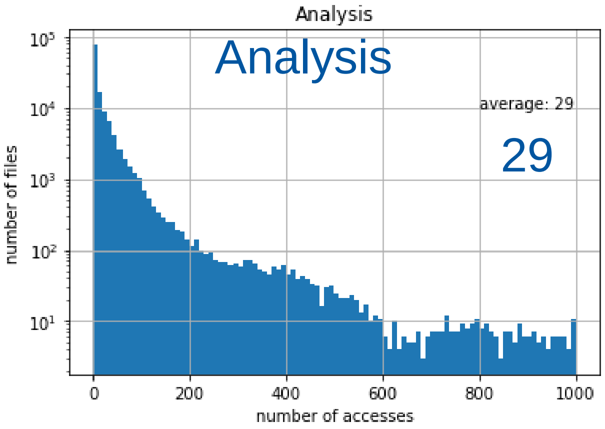
CERN

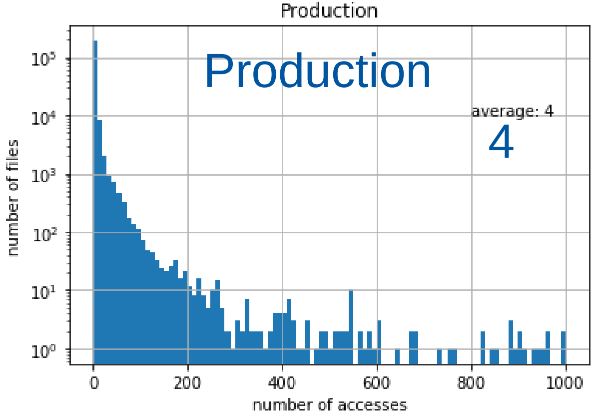
# File access patterns and caches

log scale

CMS Data from Jan 2018 to Jul 2019, all sites

- The usefulness of a cache is strongly dependent on the access patterns
    - Ideally, files will be processed only once by production jobs, hence they can stay cached (if at all) for just a short time
    - Files read by analysis jobs will might be accessed several times over the course of a long interval of time, hence they should stay cached for longer times
    - Therefore, caches are most relevant for Tier-2 sites
- Access patterns are also very correlated to the data tier



# Managing the cache space

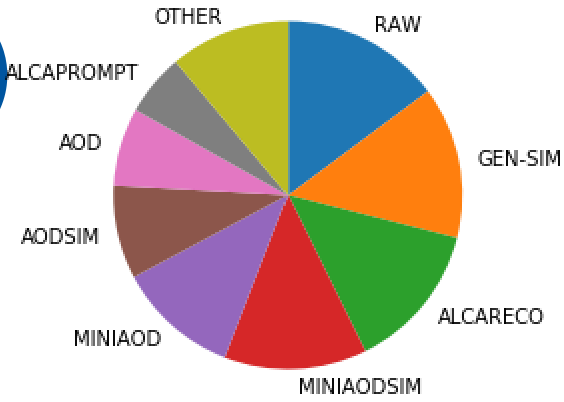
- Caches are unmanaged, but different strategies to free up space can be used
    - When? For example
        - When the cache occupancy reaches a certain threshold
        - When the cache decides that a file is not needed (at any time)
    - How? For example:
        - By deleting the least recently used (LRU) files until the cache occupancy falls below a certain threshold
        - When, given the history of the file accesses, the cache decides that it is unlikely that a file will be needed again
- One objective of this work is to compare strategies

# Simulating a site cache

- Some sites run production caches
  - CMS SoCal Tier-2 does, see Matevz' talk just after this one
- Data popularity records may allow to "simulate" the effect of having a site cache
- CMS collects and stores, for each CMSSW job, information about the files during its execution
  - File name, size, access time, user, site, type, host name of the client and the server, bytes read and written, etc.

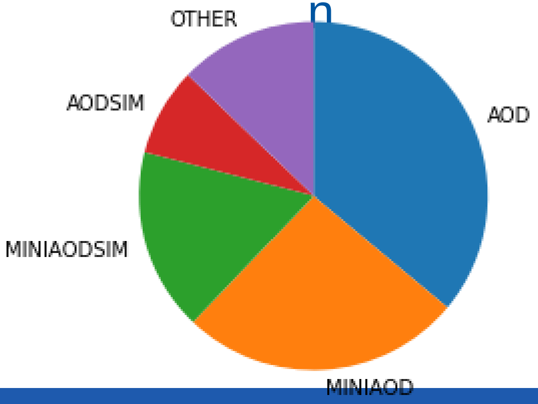
# File types ("data tiers")

- We can extract to which "data tiers" belong the files accessed
  - RAW, GEN-SIM: ~ 1 MB/event
  - AOD(SIM): ~300 kB/event
  - MINIAOD(SIM): ~30 kB/event
  - ALCA*: for alignment and calibration tasks
- Very different distributions for production and analysis
  - Assuming that caches are needed only for analysis, the only relevant data tiers are (MINI)AOD(SIM)
  - NANOAOD's volume makes caching very inexpensive




Data from Jan 2018 to Jul 2019 for all sites

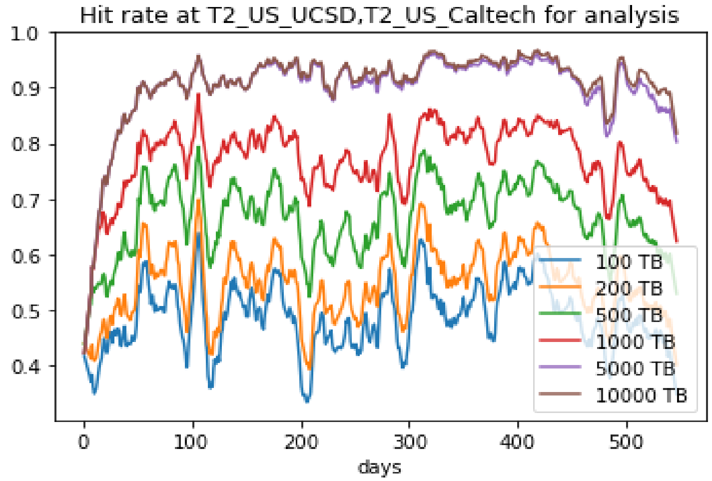
productio
n

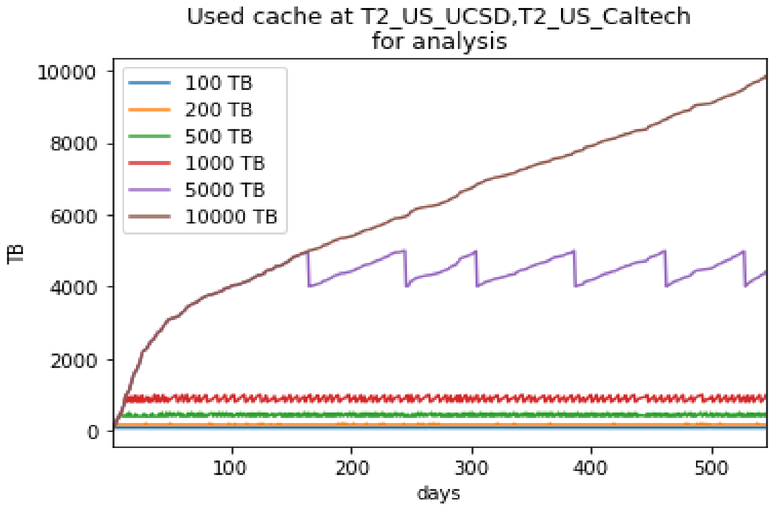



analysis

# “LRU” caches

- Estimated the hit rate vs time and vs cache size of a hypothetical site cache from actual access data
- LRU algorithm for cache management
  - Triggered when cache occupancy exceeds a certain limit (the “cache size”)
  - Purge files from the least recently accessed on, until the cache occupancy drops below 80% of the cache size
- Done for several sites
  - Will concentrate on the Southern California Tier-2 (Caltech + UCSD) for being the target of several studies in CMS on caching, for analysis jobs and for (MINI)AOD(SIM) files

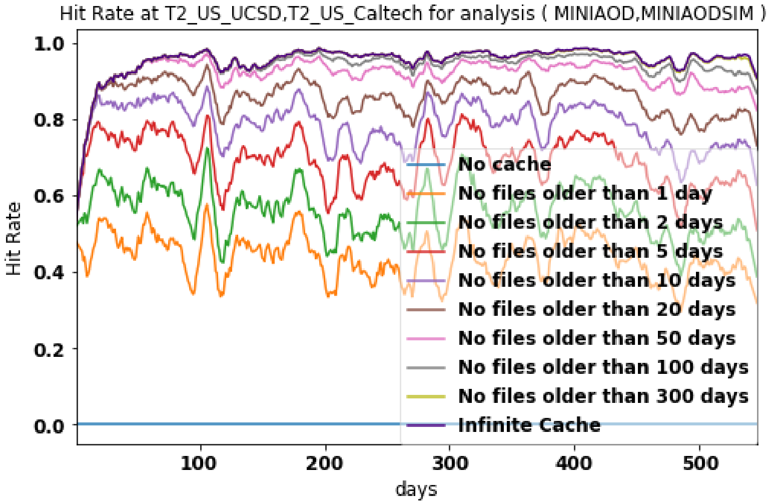


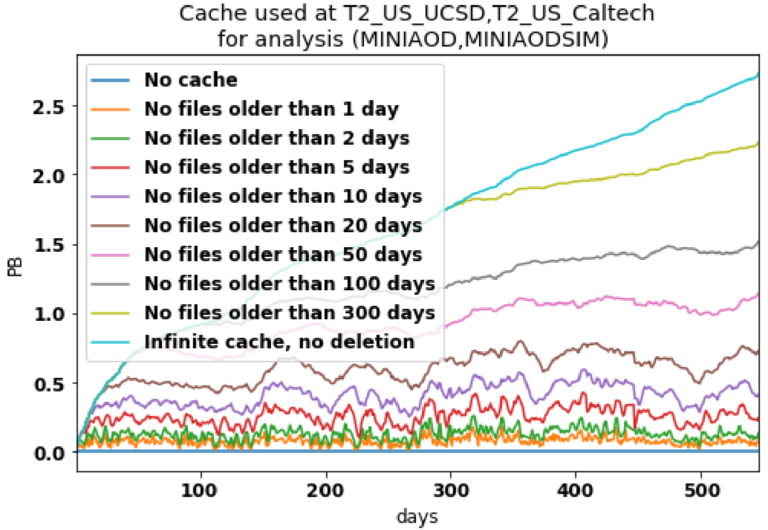



# “N-days” caches

- Age-based algorithm for cache management
  - File “age”: time passed after latest access
  - Purge files “older” than N days
  - N = 0 means “no cache at all”
  - N = 1 means that files are cached for just one day unless accessed again, etc.

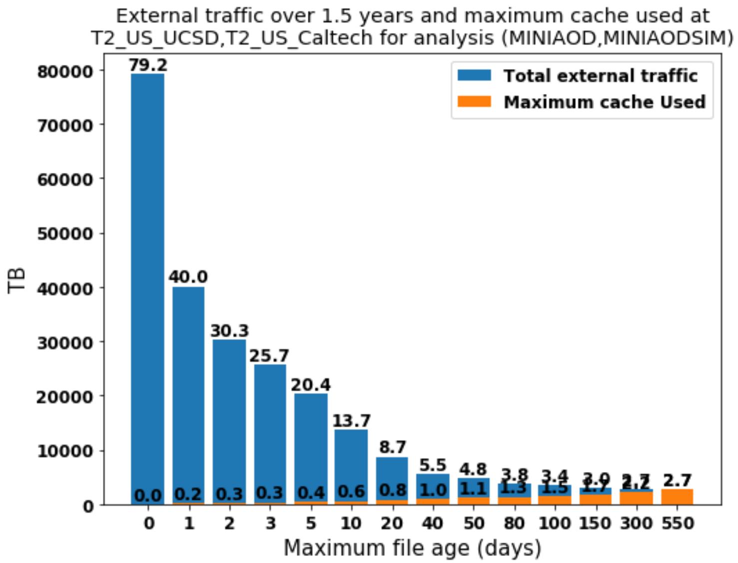
- Given a value of N, measure:
  - Average cache hit rates
  - Average cache occupancy
  - Average external traffic
  - Value of “cost function”

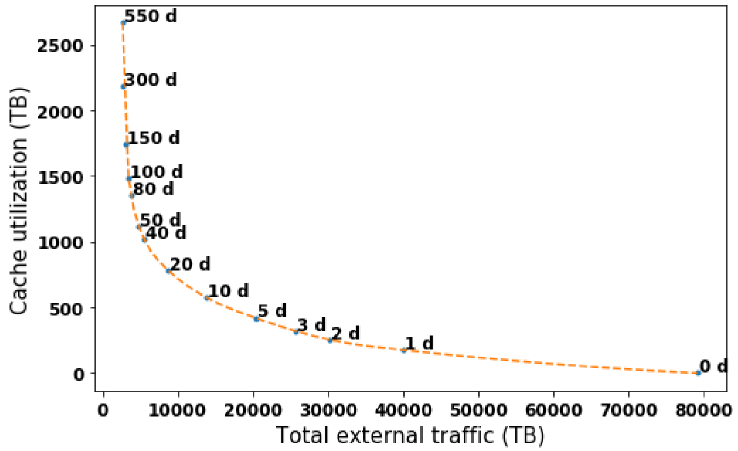


# Cost function

- Total cost = network cost + storage cost
- Storage cost = max(cache occupancy) × cost / disk storage
    - Relatively straightforward, caches have low QoS, so cheap HDDs in JBOD configuration are sufficient
- Network cost = avg(external traffic / time) × cost / bandwidth
    - Much more difficult to estimate, as it is not proportional to usage



# Unitary cost estimation

- Disk
  - Cost estimated in the WLCG/HSF cost model working group
    - 1 HDD: 8 TB, 400 EUR, 4 years lifetime
    - Disk server cost / TB ~ twice disk cost
    - ⇒ cache cost ~ 25 EUR/TB/year
  - Baseline HDD scenario: 25 EUR/TB/year
  - Pessimistic HDD scenario: 50 EUR/TB/year
  - SSD scenario: 100 EUR/TB/year
- Network
  - Cost estimated very roughly from a couple of WLCG entities
  - Baseline: 1 EUR/TB
  - Pessimistic: 10 EUR/TB

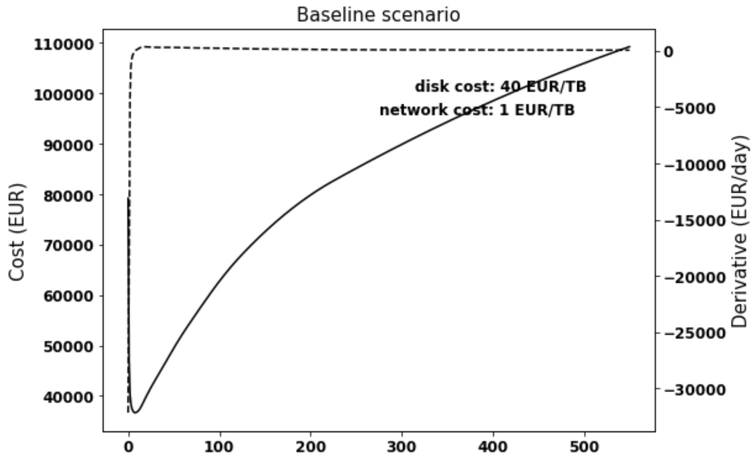
# Cost optimisation

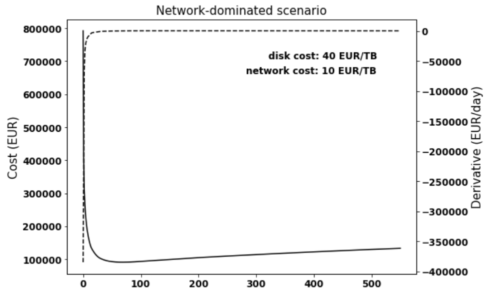
| | | Disk cost (EUR/TB) | | |
|---|---|---|---|---|
| | | 40 | 100 | 150 |
| Network cost (EUR/TB) | 1 | 8 | 2 | 1 |
| | 10 | 68 | 36 | 26 |

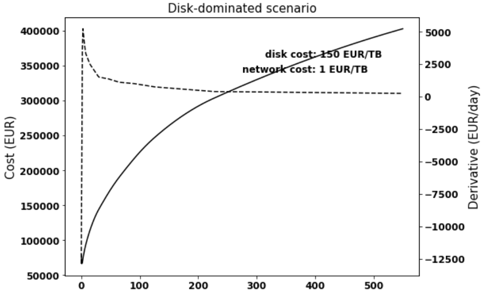
Optimal maximum file age

| | | Disk cost (EUR/TB) | | |
|---|---|---|---|---|
| | | 40 | 100 | 150 |
| Network cost (EUR/TB) | 1 | 522 TB | 252 TB | 175 TB |
| | 10 | 1262 TB | 980 TB | 870 GB |

Optimal cache size



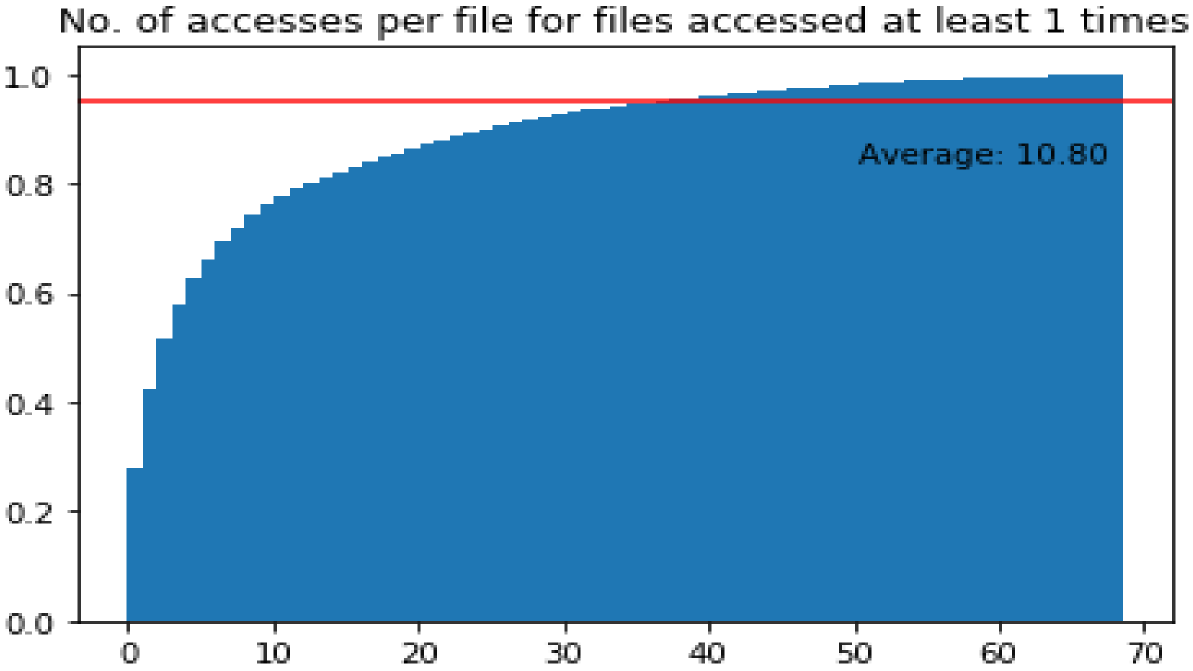



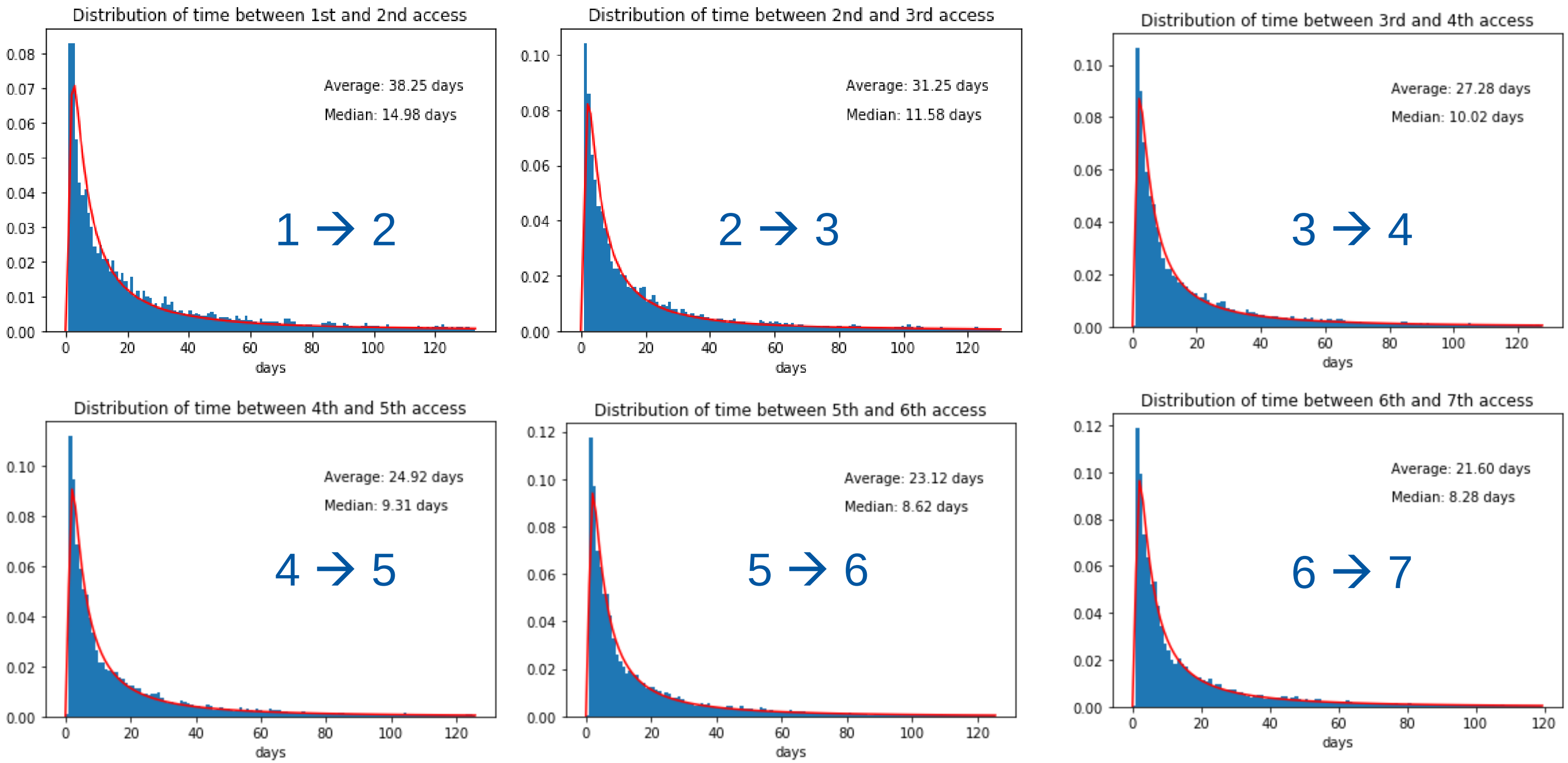
# Access time patterns

- For each file, it is possible to analyse the access history
  - The idea is to devise a way to estimate the probability that a file will be accessed again within a given time
  - This can be use to decide when it is “best” to expunge a file from the cache
  - Supposedly better than to simply expunge the least recently used file, or any file not accessed since N days
- Often files are accessed multiple times within a very short time
  - Due to concurrent jobs reading from the same file
  - Such accesses can be considered as single accesses if happening within the “minimum” lifetime of a file in the cache (e.g. one day)
    - But hit rates should be calculated using all accesses
- Possible strategy
  - (file accessed N times, time passed since last access) ⇒ probability it will be accessed at least another time
    - It ignores the fact that such probability can change a lot if the N accesses happened in a short or a long time span
    - On the other hand, if we assume that access patterns are identical for all files, the time span distribution for N accesses depends only on N

# Access time patterns

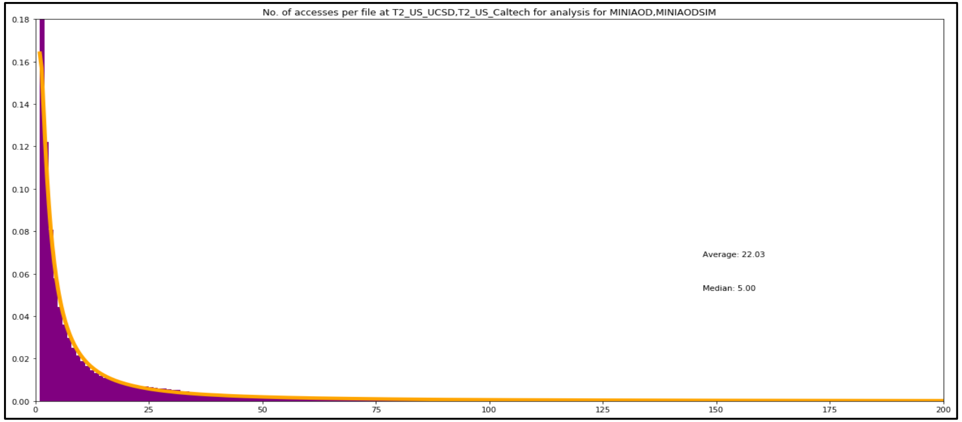


# Distribution of time to next access



# Access Probabilities

- The probability density function for a file being accessed the n-th time can be described by a simple function
- The probability for an access after a given time can be described by the same function
- → Monto Carlo Model for data access with very few parameters




$$f(x, \mu) = \frac{1}{\sqrt{2\pi x^3}} \exp\left(-\frac{(x-\mu)^2}{2x\mu^2}\right)$$

# What to do with this?

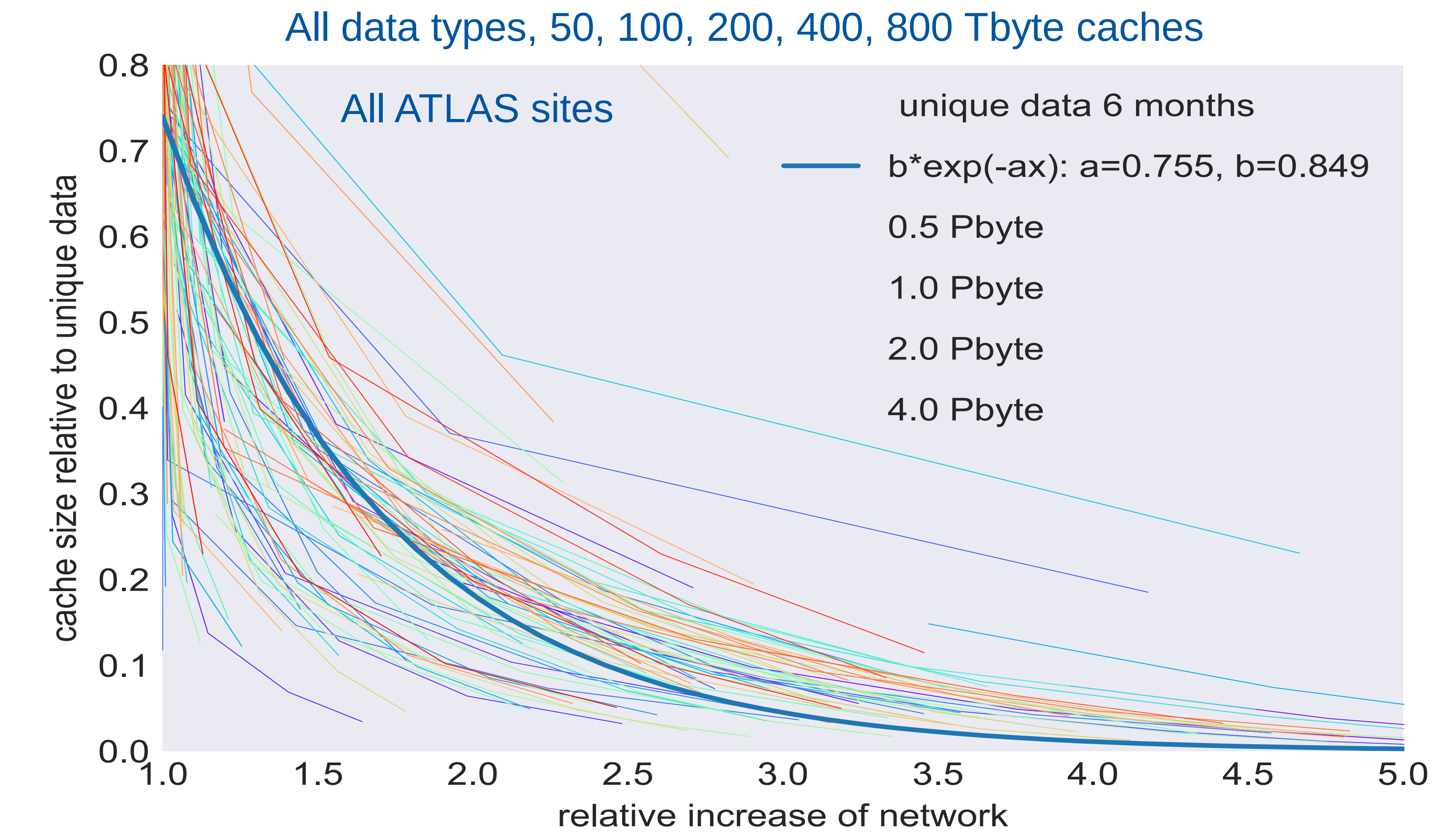
- Condense many measurements into a very compact representation
- Compare sites/workflows/changes over time
- Build a model for MC studies
- Spot anomalies (maybe)

- Try to understand why many independent Physicists doing independent research, synchronised by a few conferences create these patterns…..

# Comparing Caches on Different Sites

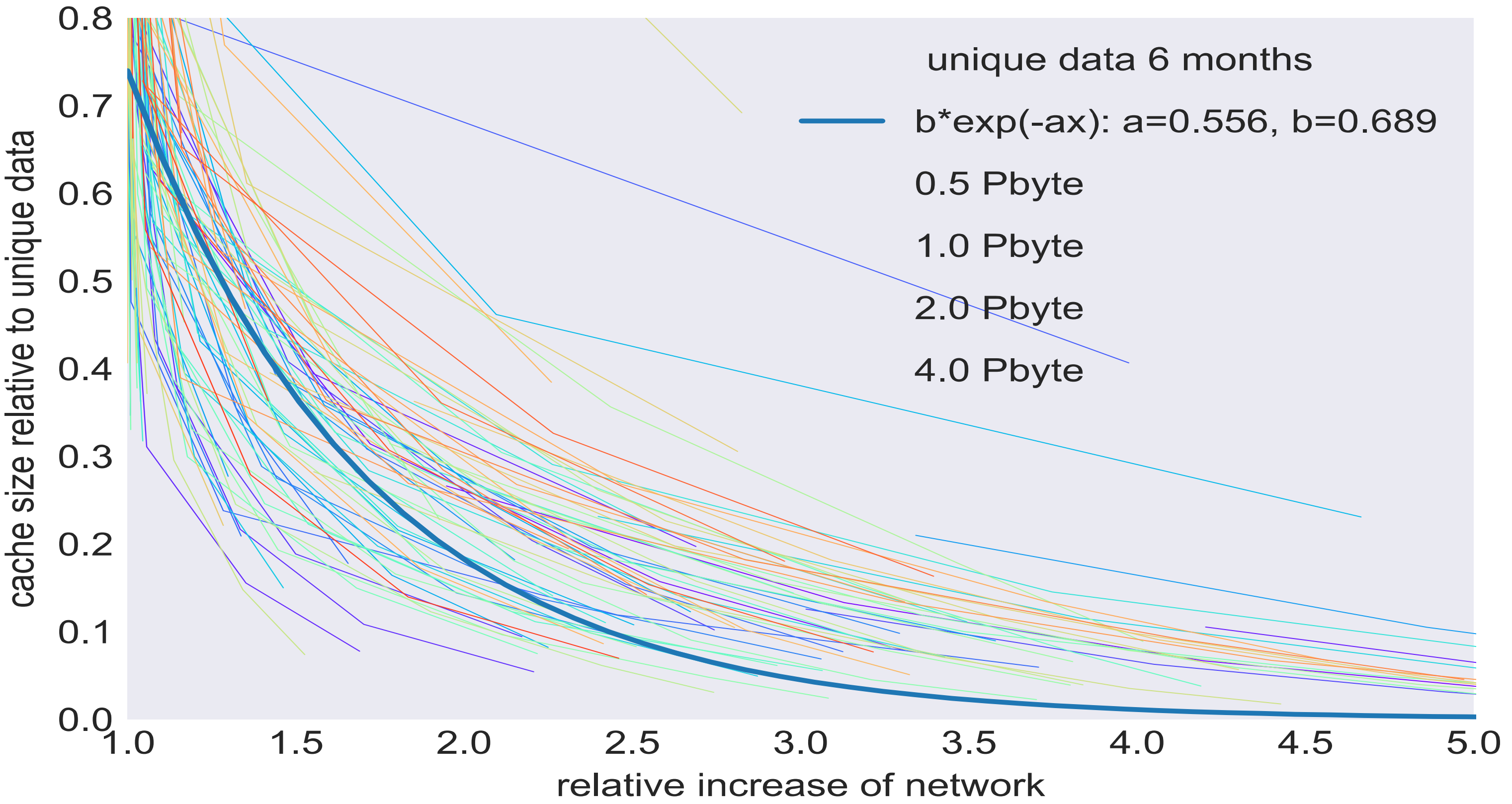
- Using ATLAS Panda logs
- LRU caches with different sizes
- Relation between cache size and network increase
    - Cache size is relative to unique data in ½ year
    - Network is relative to network use with infinite cache

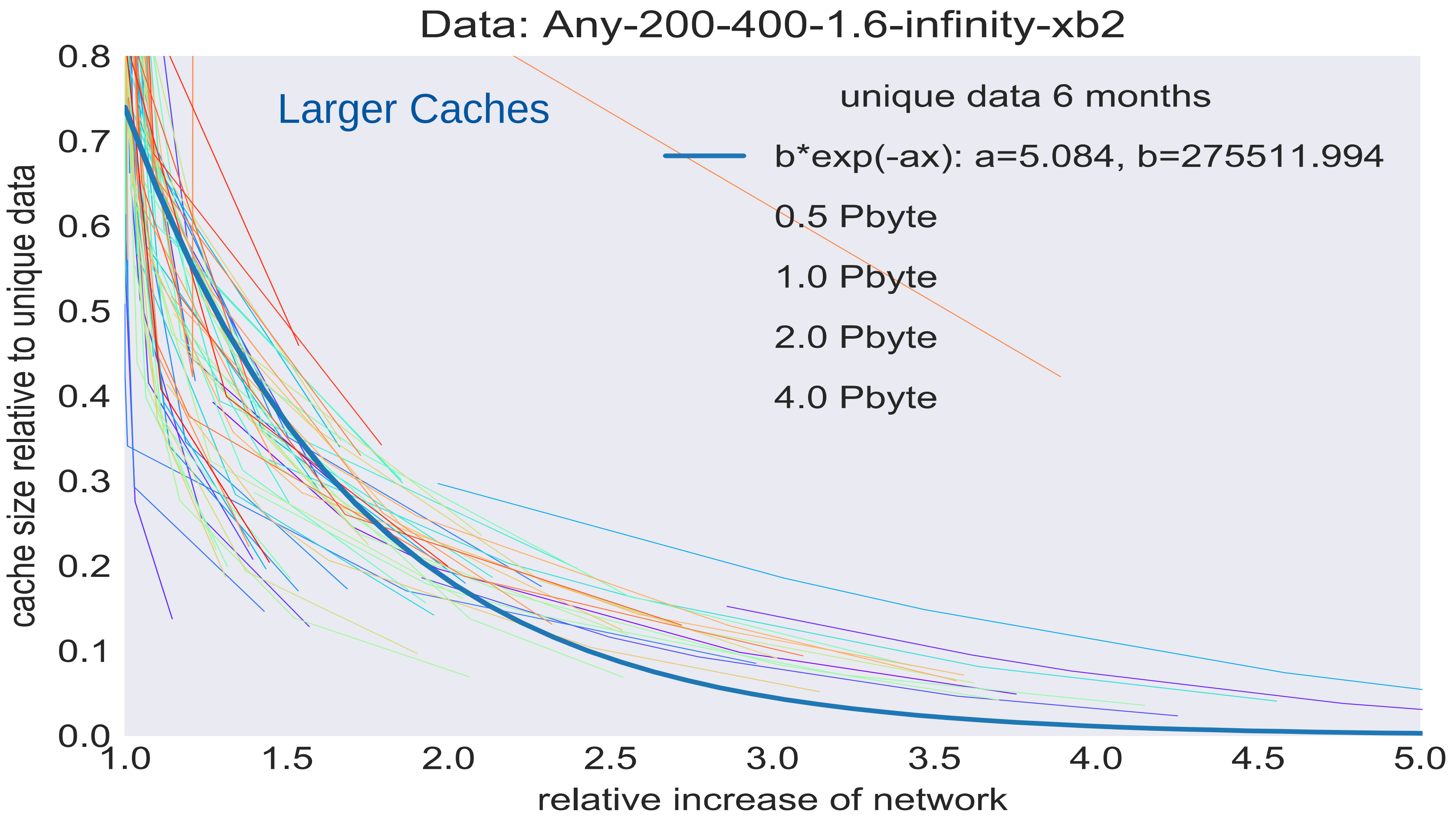
- This allows to look at large and small sites at the same time
- Can be combined with the cost function before
- Can be used to decide on trade-off between network and storage



All data types, 50, 100, 200, 400, 800 Tbyte caches
All ATLAS sites
unique data 6 months
b*exp(-ax): a=0.755, b=0.849
0.5 Pbyte
1.0 Pbyte
2.0 Pbyte
4.0 Pbyte
cache size relative to unique data
relative increase of network
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

# AOD & DAOD & NTUP , 50, 100, 200, 400, 800 Tbyte caches



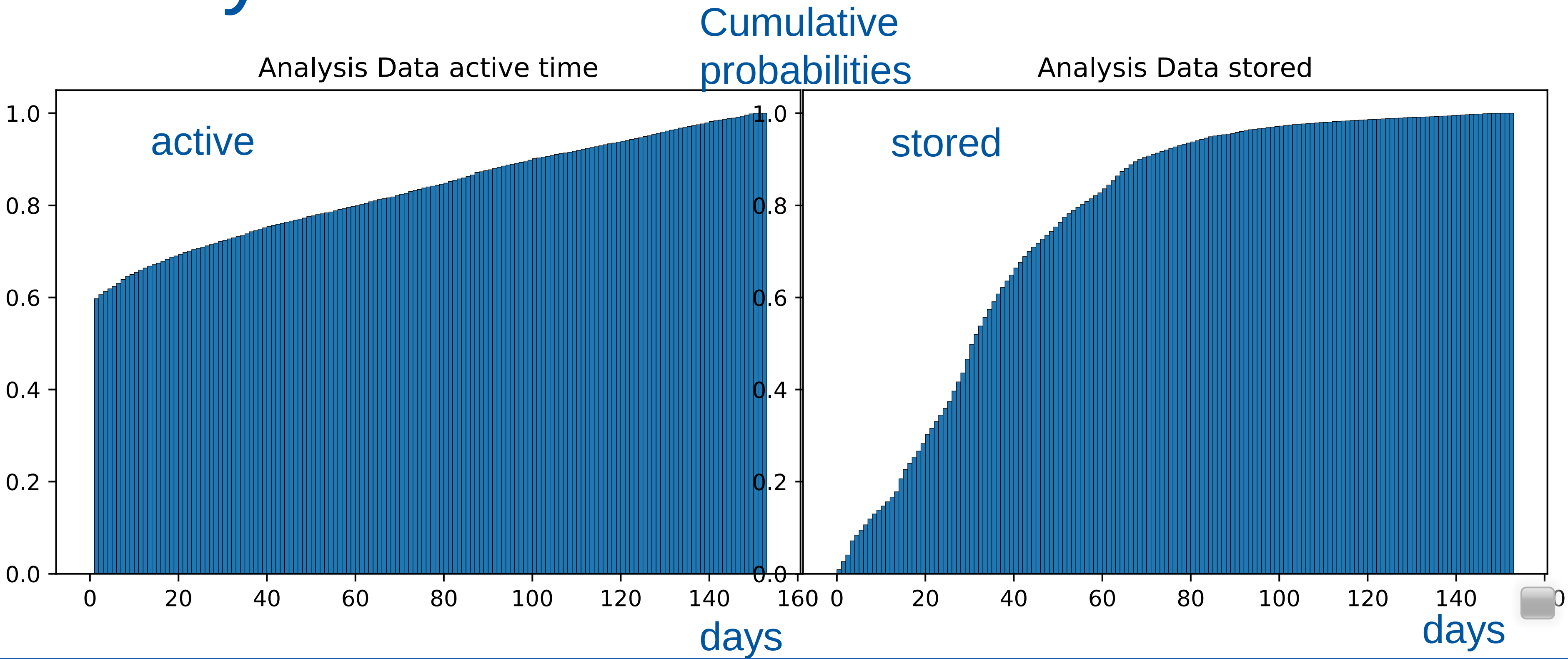


Data: Any-200-400-1.6-infinity-xb2
Larger Caches
unique data 6 months
b*exp(-ax): a=5.084, b=275511.994
0.5 Pbyte
1.0 Pbyte
2.0 Pbyte
4.0 Pbyte
cache size relative to unique data
relative increase of network
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

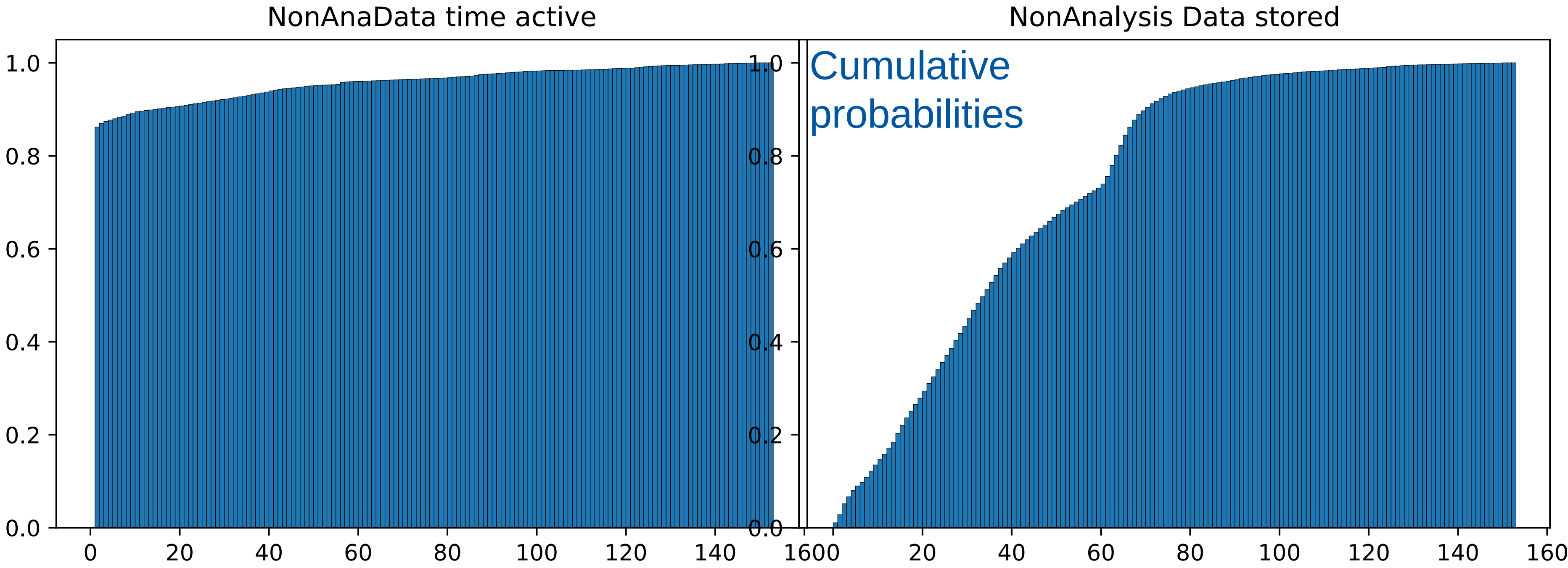
# How long is data used?

- Lifetime ($T_{deleted} – T_{move}$)
- Active-Lifetime ($T_{last\ access} – T_{first\ access}$)
- “Cost” = Lifetime x Size [GByte days]
- “Value” = Active-Lifetime x Size [GByte days]
- Analysis vs. Prod Data

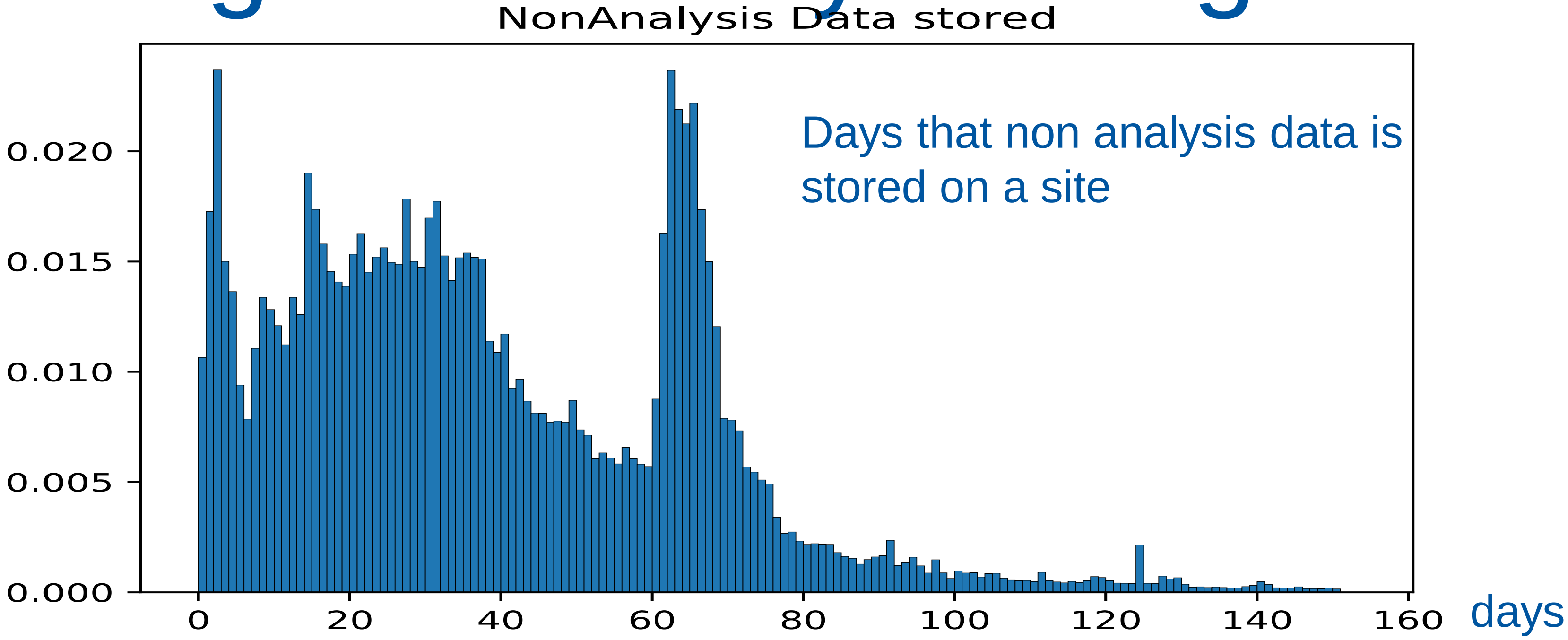
# Analysis data active/stored



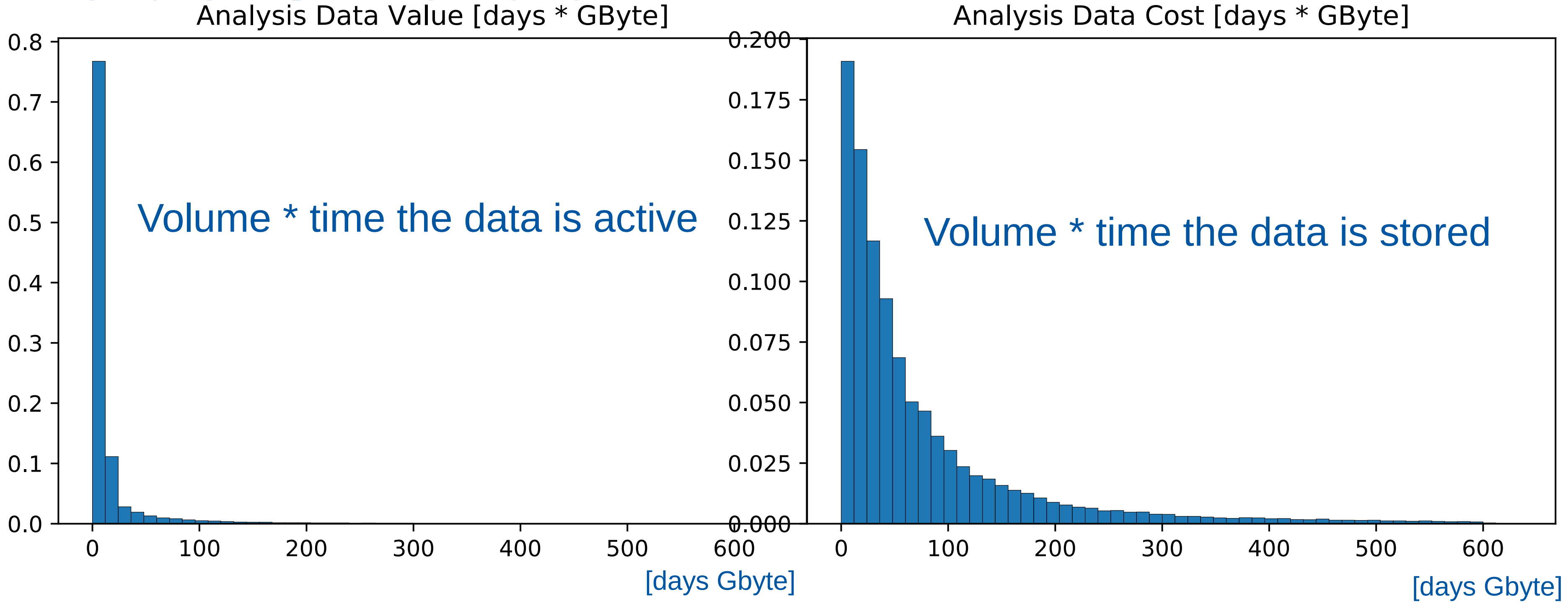
# Non Analysis data active/stored



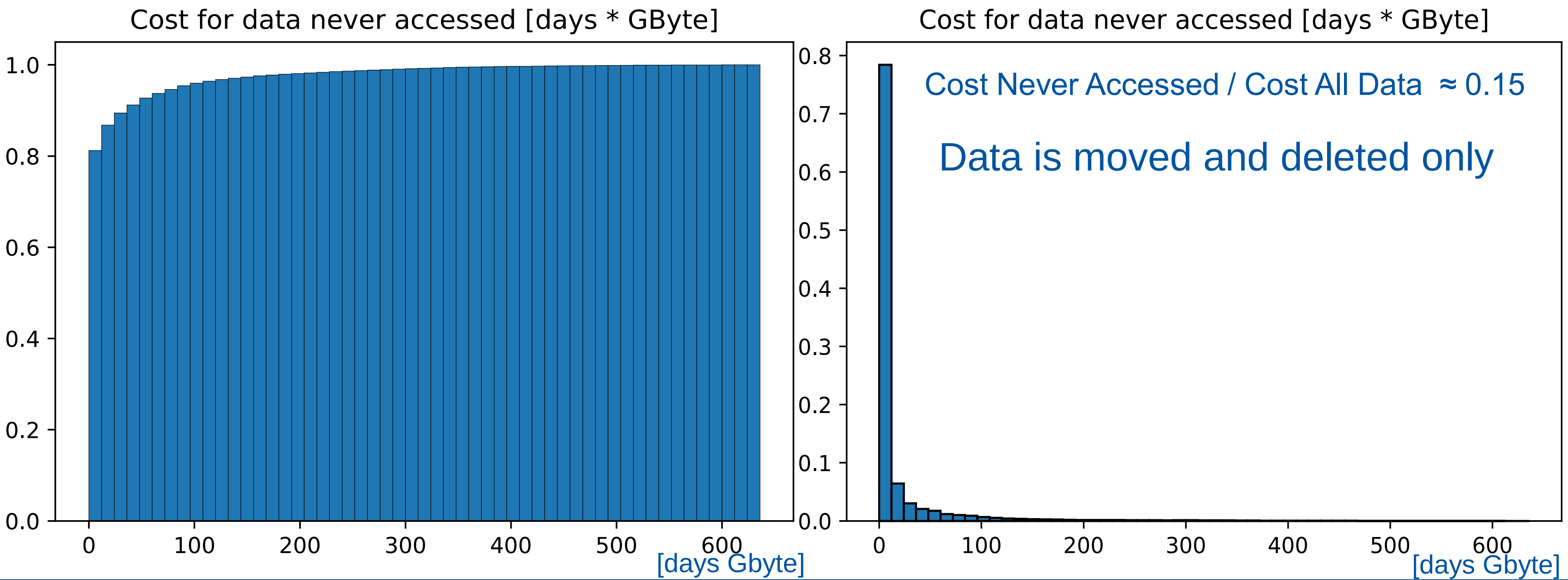
# Storage is actively managed



# Cost / Value


Analysis Data Value [days * GByte]
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Volume * time the data is active
0
100
200
300
400
500
600
[days Gbyte]
Analysis Data Cost [days * GByte]
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
Volume * time the data is stored
0
100
200
300
400
500
600
[days Gbyte]

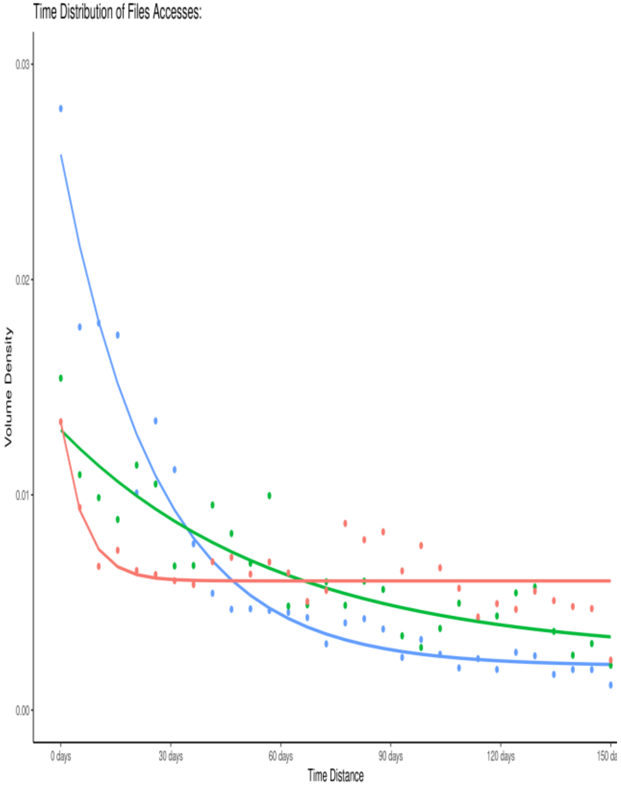
# Cost of data not accessed

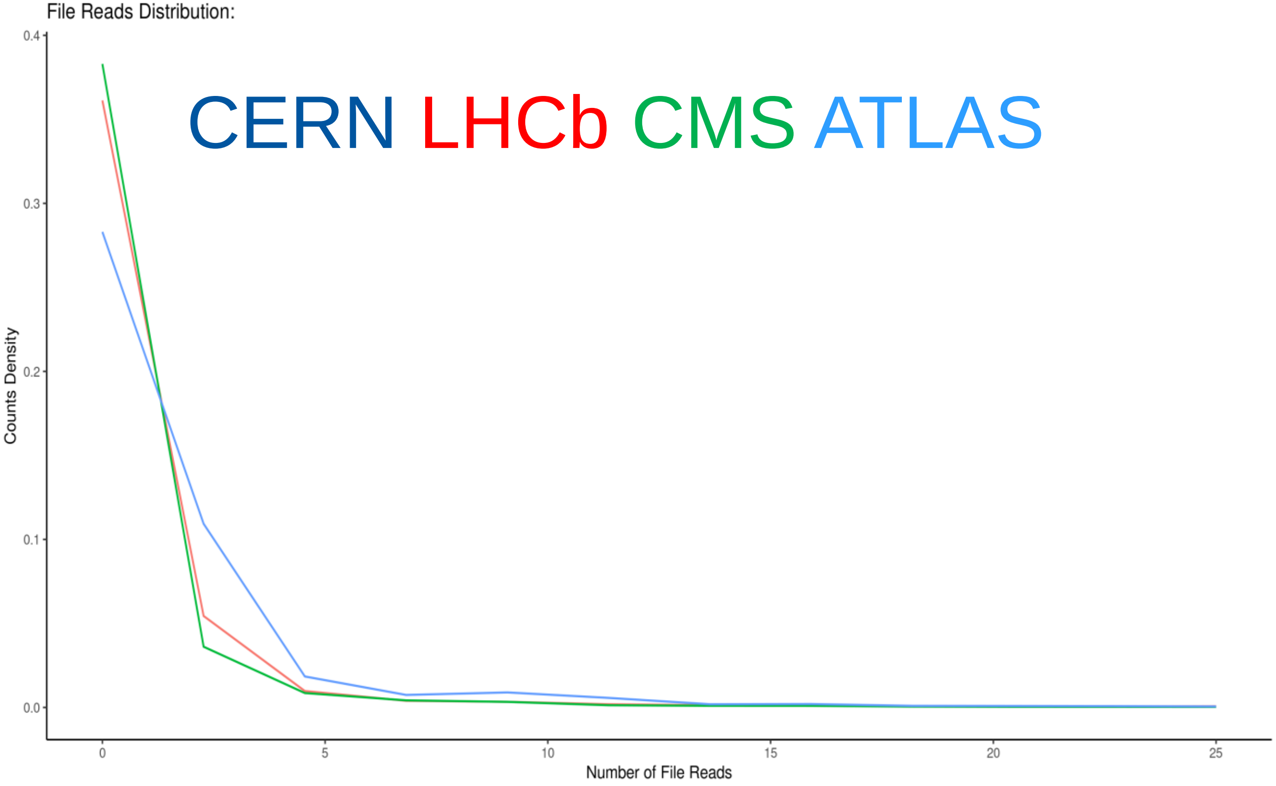


# Site Perspective

# Site Perspective

- Not all data access is traced by the experiments → ask the storage systems
- PIC T1
  - dCache traces
- CERN T0
  - EOS traces


Time Distribution of Files Accesses:
Volume Density
0.03
0.02
0.01
0.00
0 days
30 days
60 days
90 days
120 days
Time Distance


File Reads Distribution:
CERN LHCb CMS ATLAS
Counts Density
0.4
0.3
0.2
0.1
0.0
0
5
10
15
20
25
Number of File Reads

# PIC/CIEMAT Carlos Pérez Dengra

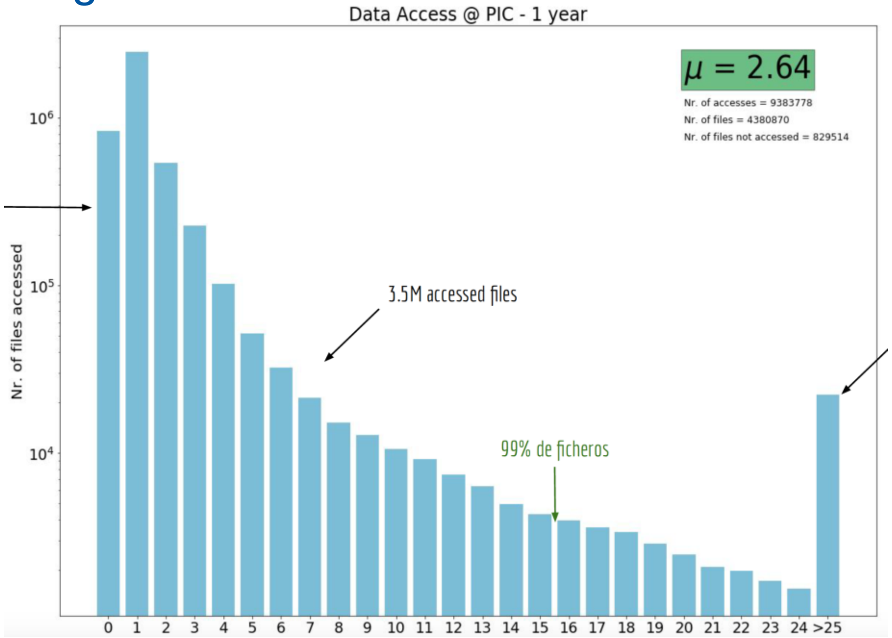
- PIC Tier-1 [CMS]
- Sept-2017→ Sept-2018
- Average disk usage ~2.3 PB
- 8.8 PB writes (10.5M files)
- 24.0 PB read (3.5M distinct files)
- 8.8 PB removed (11.0M files)

Covered by: CMS data access and usage studies at PIC Tier-1 and CIEMAT Tier-2

# Number of accesses: PIC

All Data/MC Tiers

Log scale

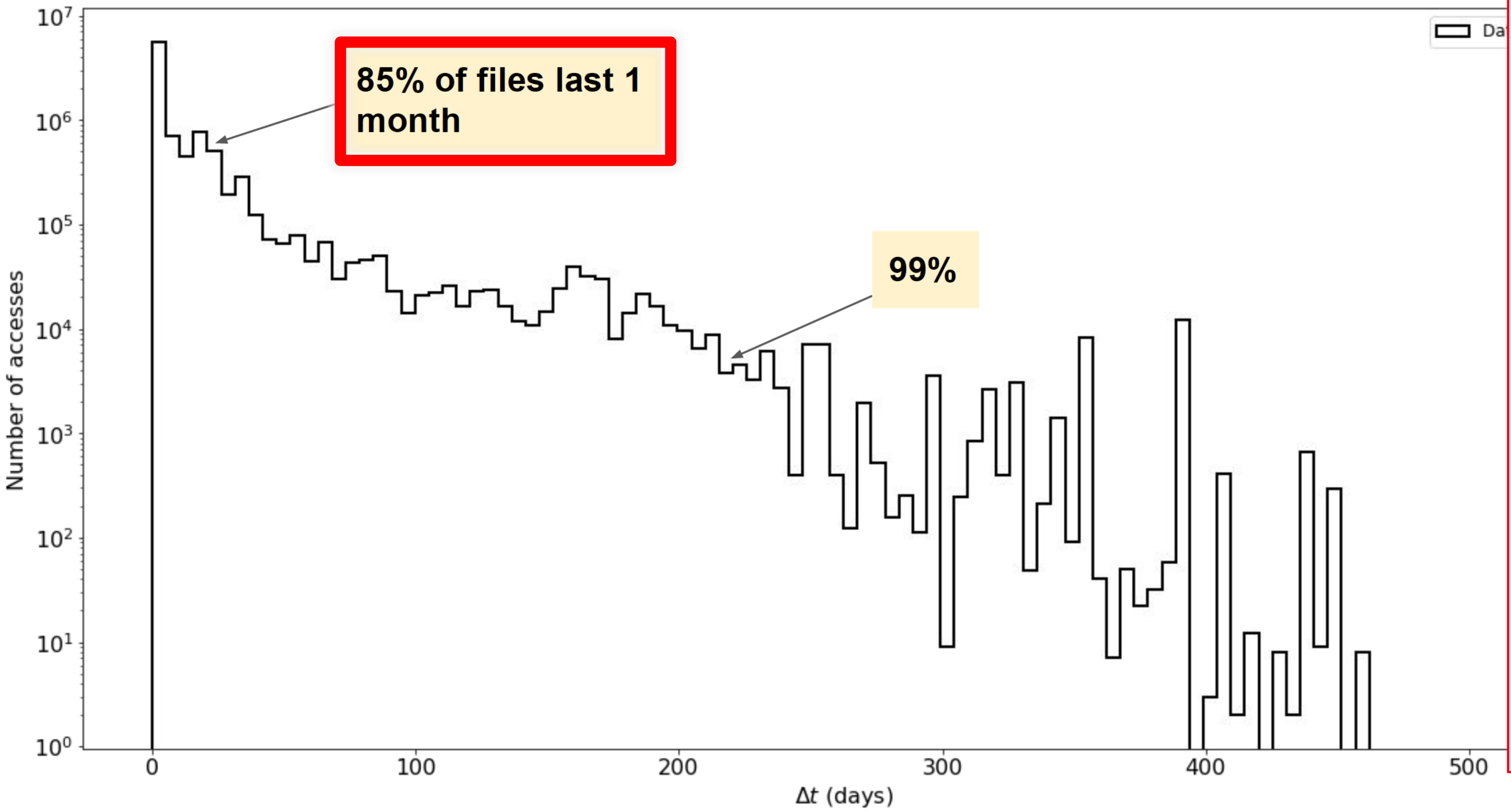



800k written and unaccessed files

22k files accessed > 25 times

6.7M written files, unaccessed and deleted are Not considered

Covered by: CMS data access and usage studies at PIC Tier-1 and CIEMAT Tier-2




Covered by: CMS data access and usage studies at PIC Tier-1 and CIEMAT Tier-2

# Summary

- With access logs the impact of caches can be simulated
  - The trade off between storage and network (cost)
- Large and small sites show very similar behaviour (ATLAS)
- The bulk of the data isn't accessed very often (90% < 15 times)
- Time between accesses is short (days)
- Lifetime is short (<80days for >90% of the data )
- Active lifetime is shorter than storage lifetime
  - ATLAS (some potential gain)
- The probability density for access in time and frequency can be described by a simple model (inverse gaussian with one shape factor)
  - Handle for mathematical description
- Access patterns within sites show similar patterns as those traced by experiment frameworks

# What would be nice ;-)

- Same format for access logs (binary format)
  - Between experiments
  - Between storage systems
  - Agreed subset of information
    - File-name/id, size, data-tier,
    - time, operation, source, destination
    - Data media (Archive (tape), Active (disk))

# Related work

- M. Tadel, Moving the California distributed CMS xcache from bare metal into containers using Kubernetes, track 4
- D. Spiga, Smart caching at CMS: applying AI to XCache edge services, track 4
- J. Flix, CMS data access and usage studies at PIC Tier-1 and CIEMAT Tier-2, track 4

# Backup slides

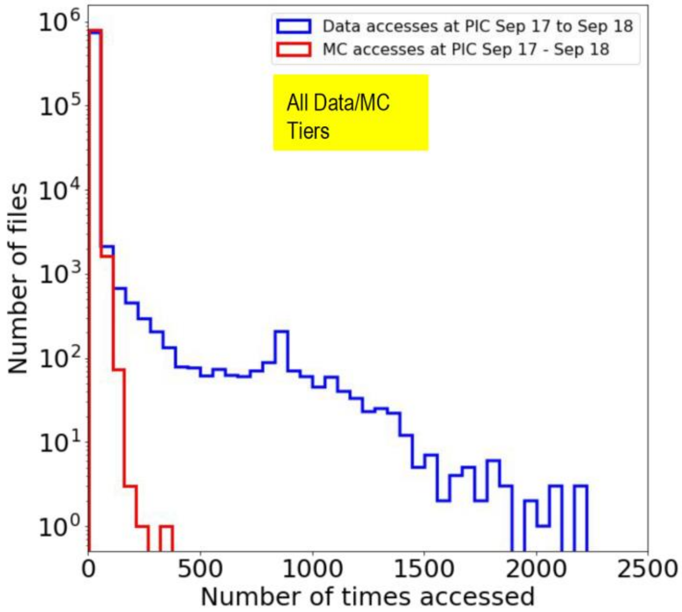
# Probability of another access

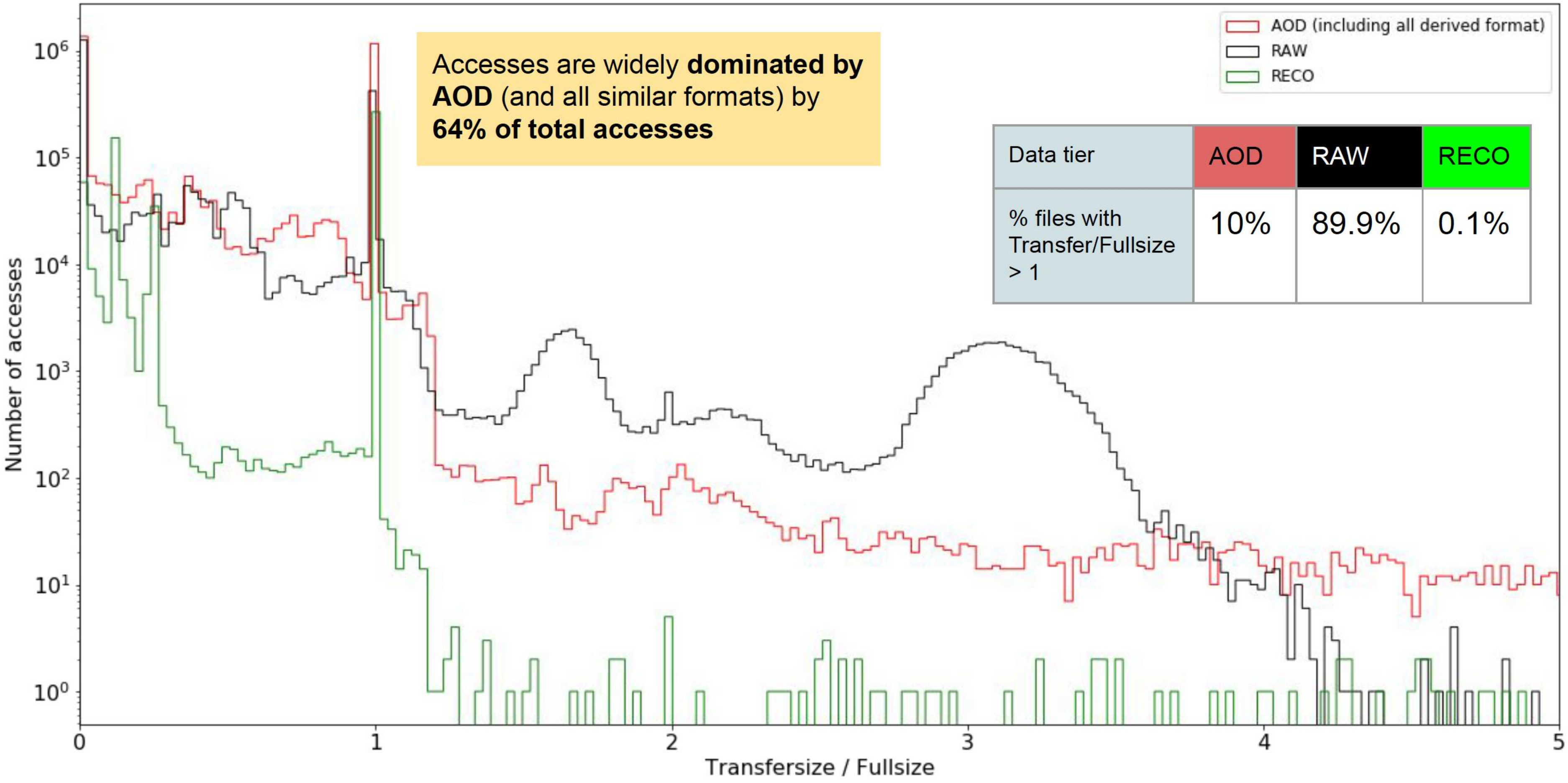
- $P(x; x_0, \ldots, x_n)$ = probability density of the file being accessed at time x if it was accessed already at times $x_0, \ldots, n_n$
- Let's assume time invariance, which leads to
  $P(x; x_1\text{-}x_0, \ldots, x_n\text{-}x_{n\text{-}1})$ = probability density of the file being accessed at time x after the last access if it was accessed already at times $x_0, \ldots, n_n$
- Special case
  $P(x)$ = probability density of the file being accessed at time x after it was accessed once
- $R_2 = \int P(x)dx$ = probability of the file being accessed at least a second time
- $R_3(x_1\text{-}x_0) = \int P(x; x_1\text{-}x_0)$ = probability of the file being accessed at least a third time if $x_1\text{-}x_0$ seconds passed between the 1st and 2nd access
- $R_{n+1} = \int P(x; x_1\text{-}x_0, \ldots, x_n\text{-}x_{n\text{-}1})$ = probability of the file being accessed at least an (N+1)-th time if etc.

# Probability of another access

- $P(x)=R_2P'(x)$
  $P'(x)$ = normalised PDF of time between 1st and 2nd access
- $P(x;\ x_1-x_0)=R_3P'(x;\ x_1-x_0)$
  $P'(x;\ x_1-x_0)$ = normalised PDF of time between 2nd and 3rd access if $x_1-x_0$ seconds passed between the 1st and the 2nd access

# PIC

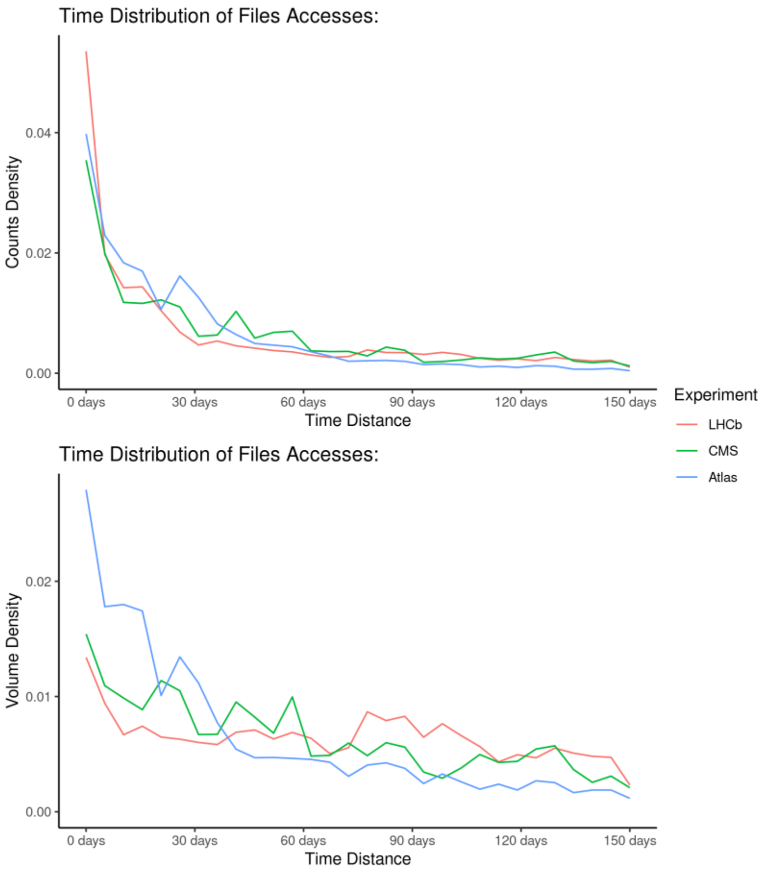

Data accesses at PIC Sep 17 to Sep 18
MC accesses at PIC Sep 17 - Sep 18
All Data/MC
Tiers
Number of files
Number of times accessed
10^6
10^5
10^4
10^3
10^2
10^1
10^0
0
500
1000
1500
2000
2500




Accesses are widely **dominated by AOD** (and all similar formats) by **64% of total accesses**

| Data tier | AOD | RAW | RECO |
|---|---|---|---|
| % files with Transfer/Fullsize > 1 | 10% | 89.9% | 0.1% |

# CERN

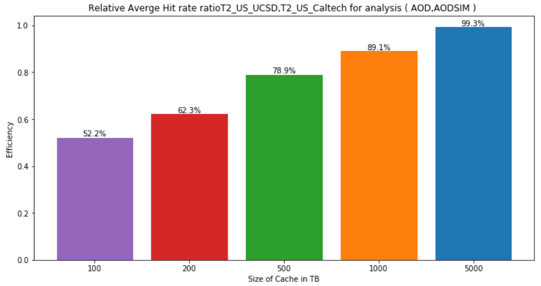


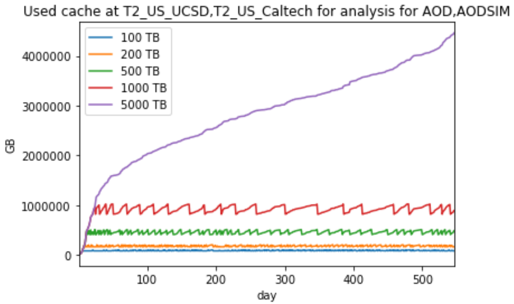
Time Distribution of Files Accesses:
Counts Density
0.04
0.02
0.00
0 days
30 days
60 days
90 days
120 days
150 days
Time Distance
Experiment
LHCb
CMS
Atlas
Time Distribution of Files Accesses:
Volume Density
0.02
0.01
0.00
0 days
30 days
60 days
90 days
120 days
150 days
Time Distance

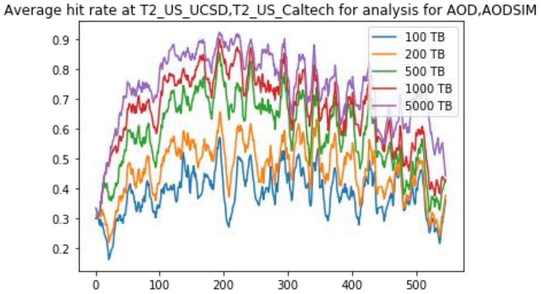
# AOD(SIM) vs. MINIAOD(SIM)

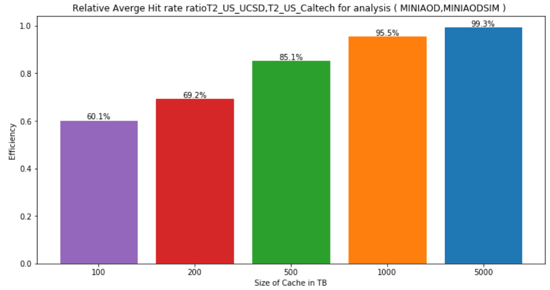
## No qualitative difference between AOD and MINIAOD

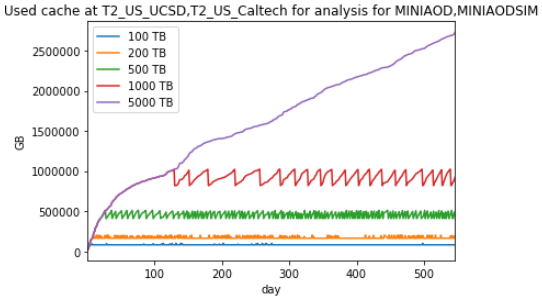
- Choosing to work on MINIAOD as AOD usage for analysis will reduce in the future

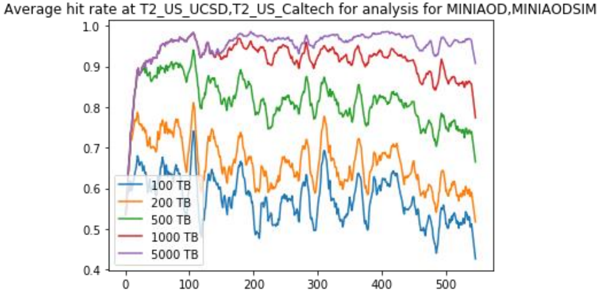


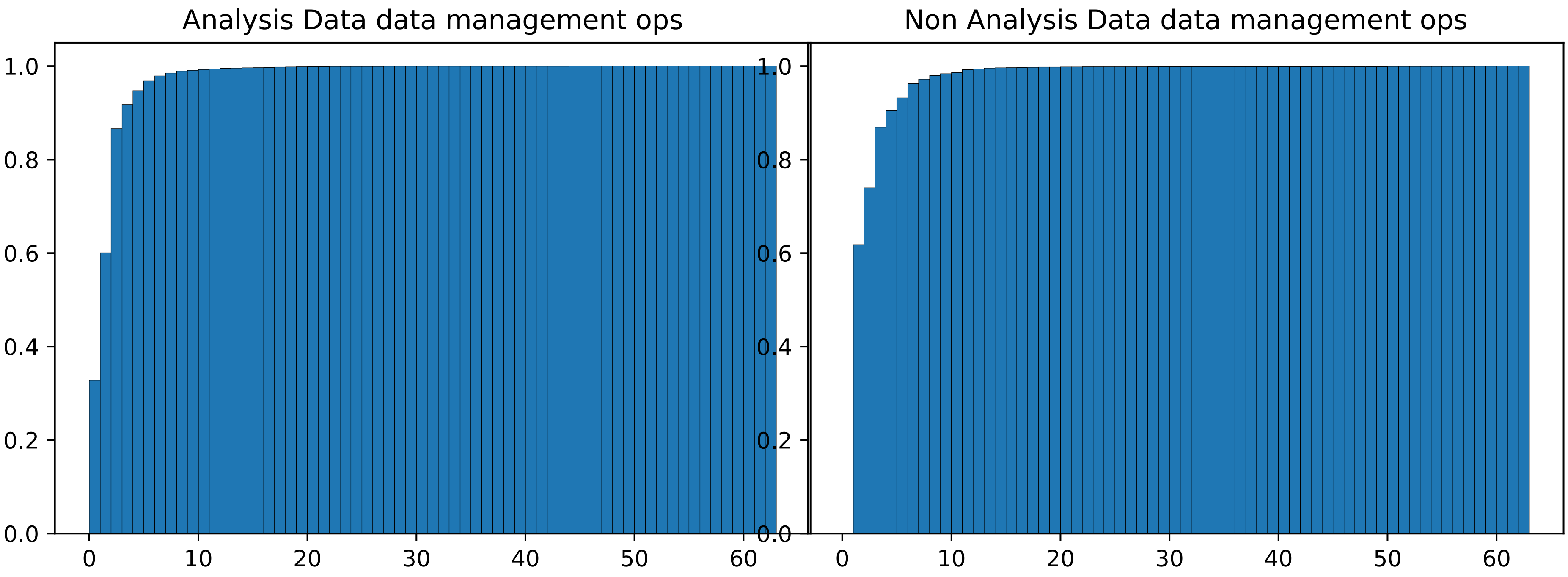



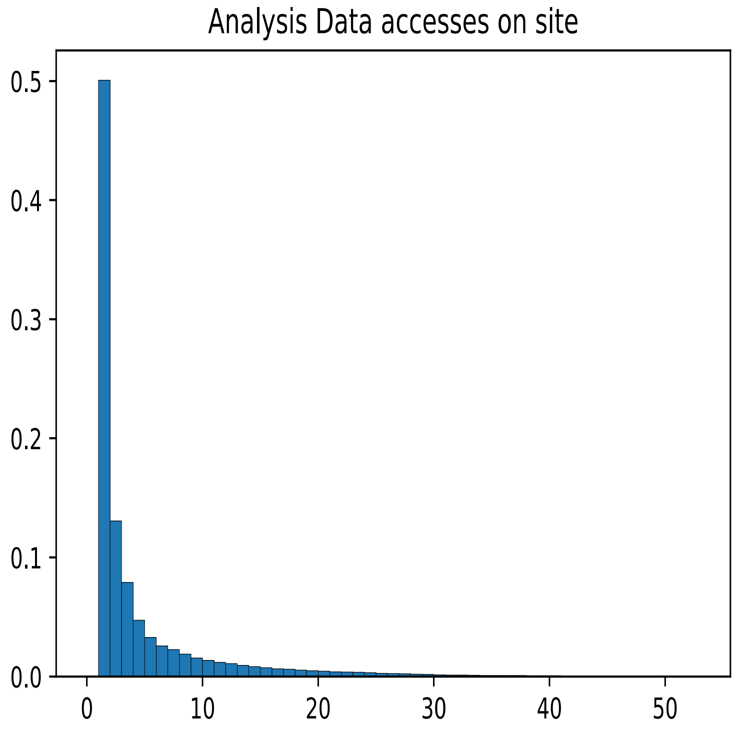



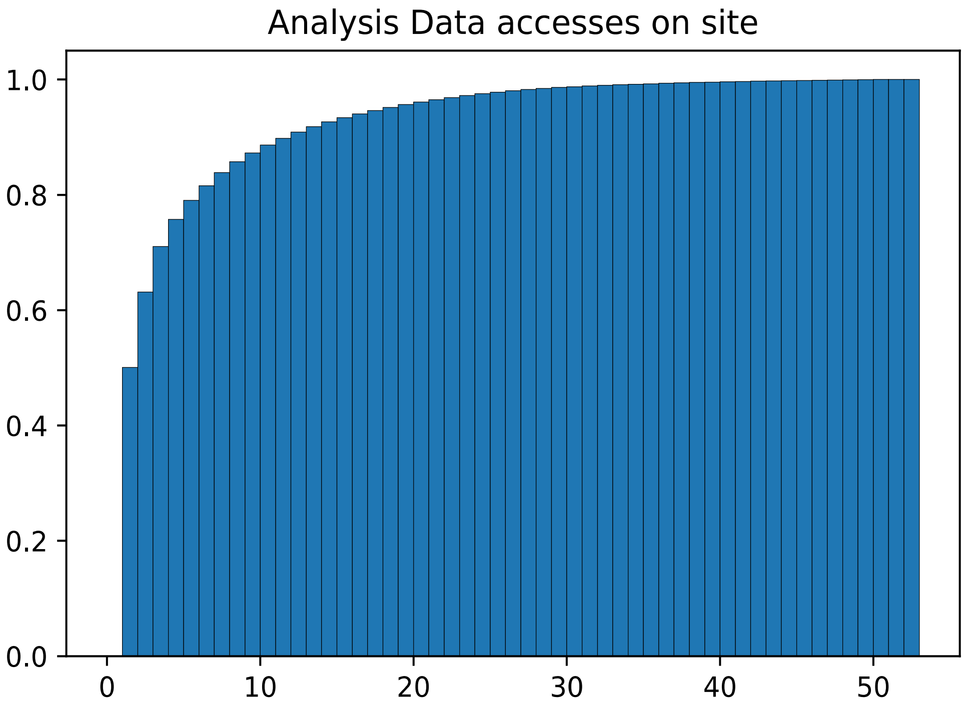
# Data management operations: copy and delete




Small differences between analysis and prod data:

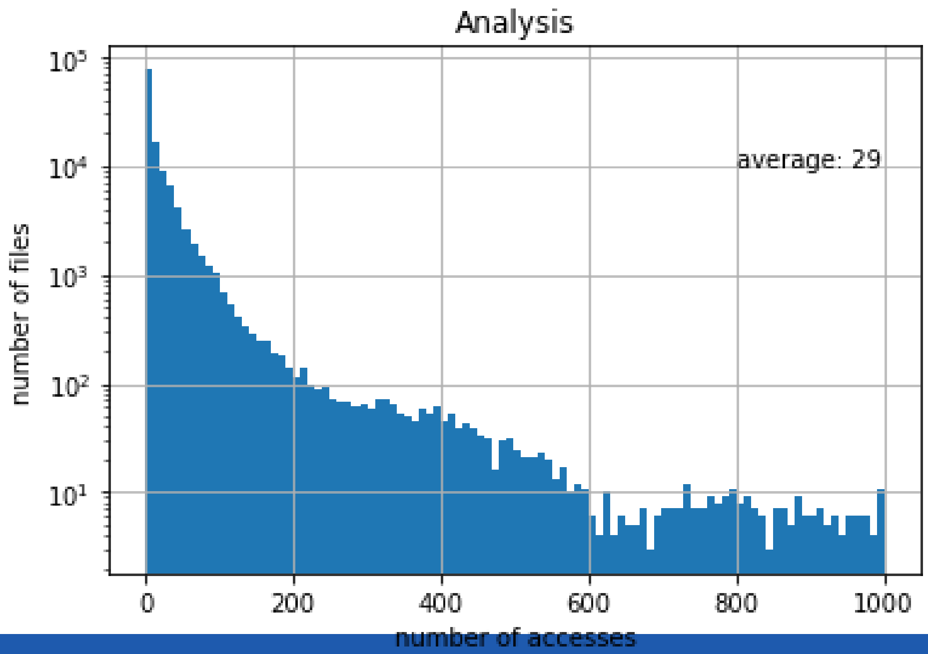


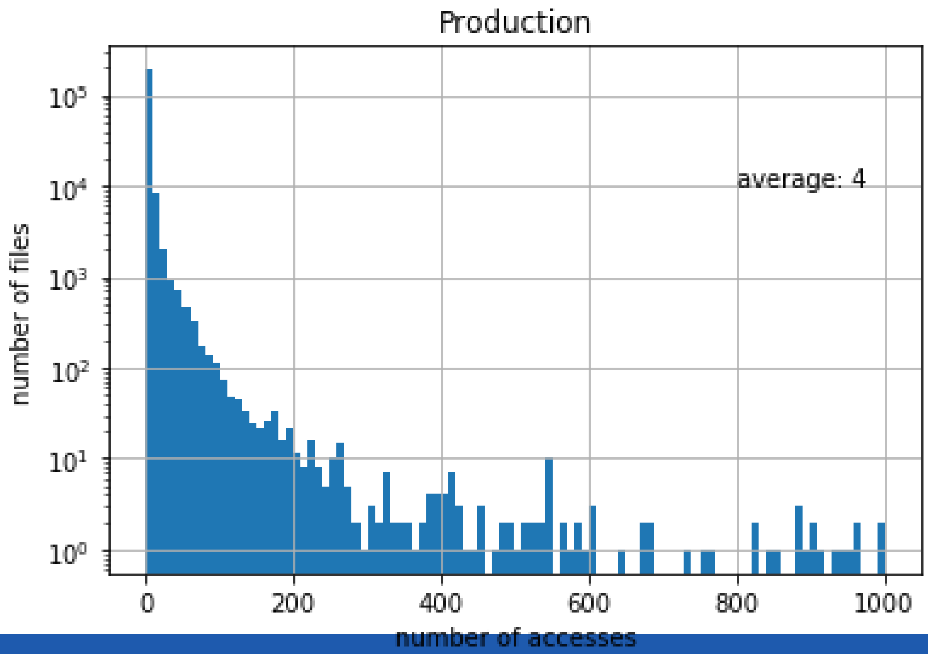
Analysis Data accesses on site
0.5
0.4
0.3
0.2
0.1
0.0
0
10
20
30
40
50

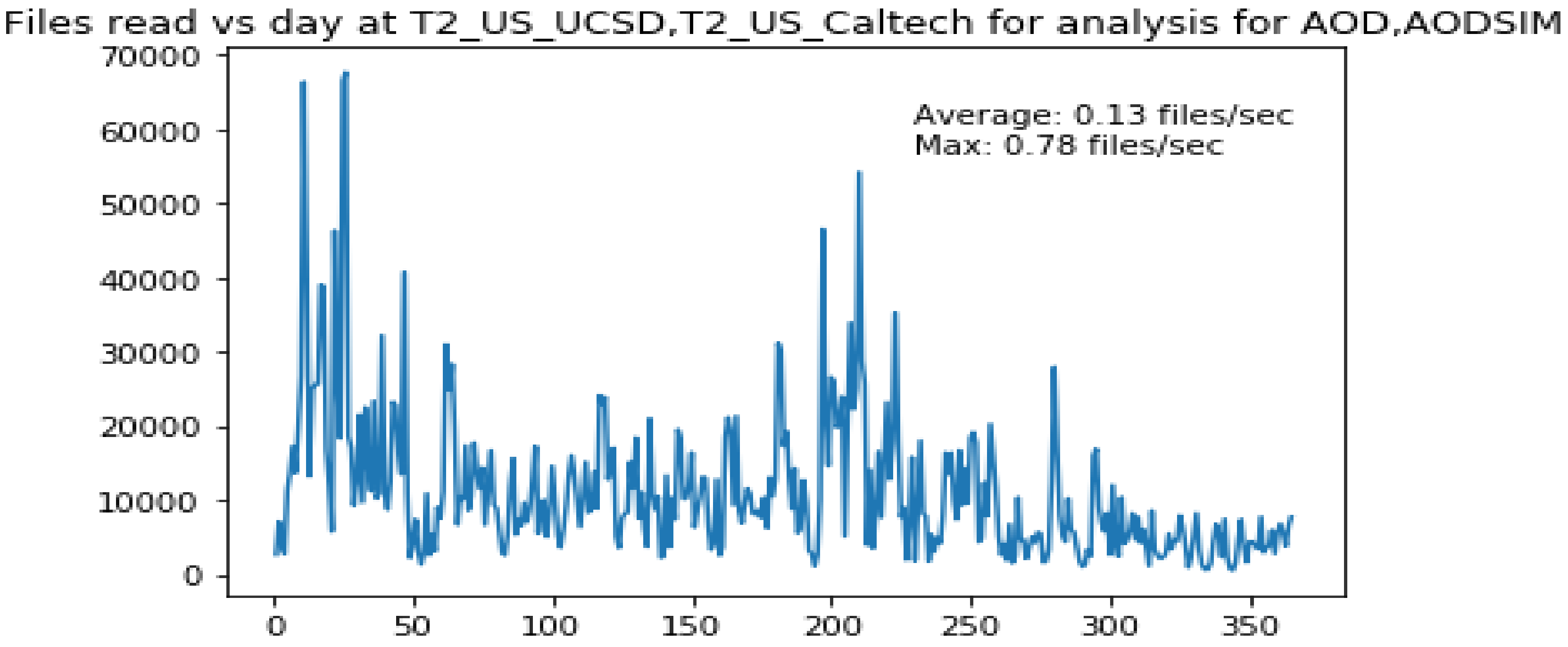
Analysis Data accesses on site
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50

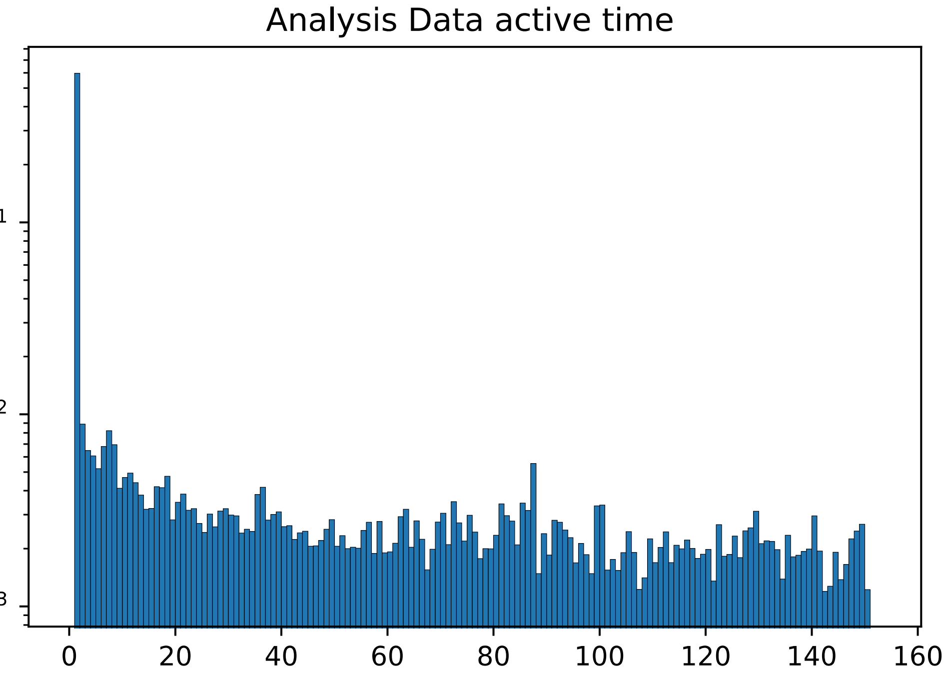
# File access patterns



# File access rates at a T2




Analysis Data active time
0
20
40
60
80
100
120
140
160