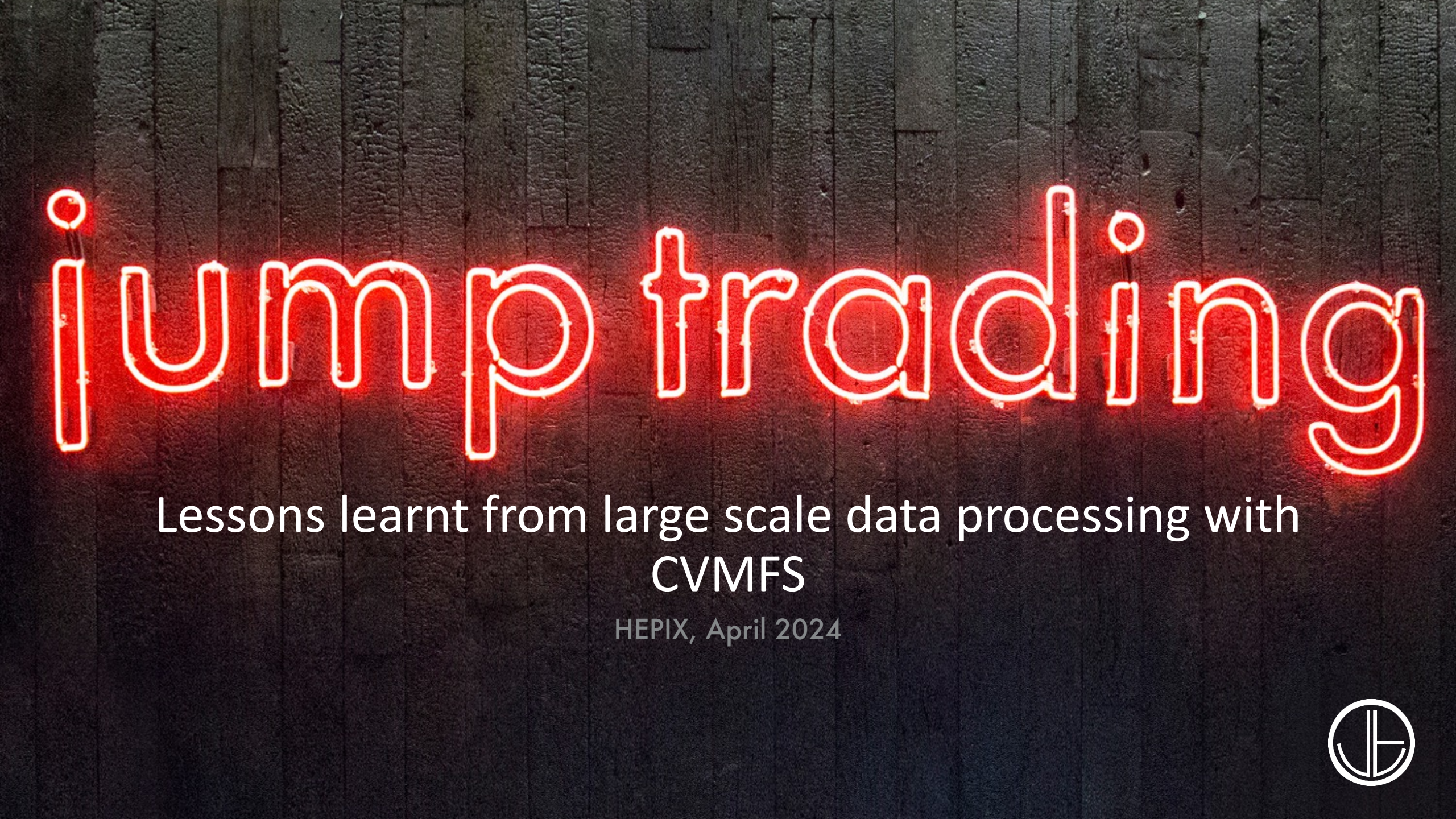


Lessons learnt from large scale data processing with CVMFS

HEPIX, April 2024

Matt Harvey

HPC Production Engineer

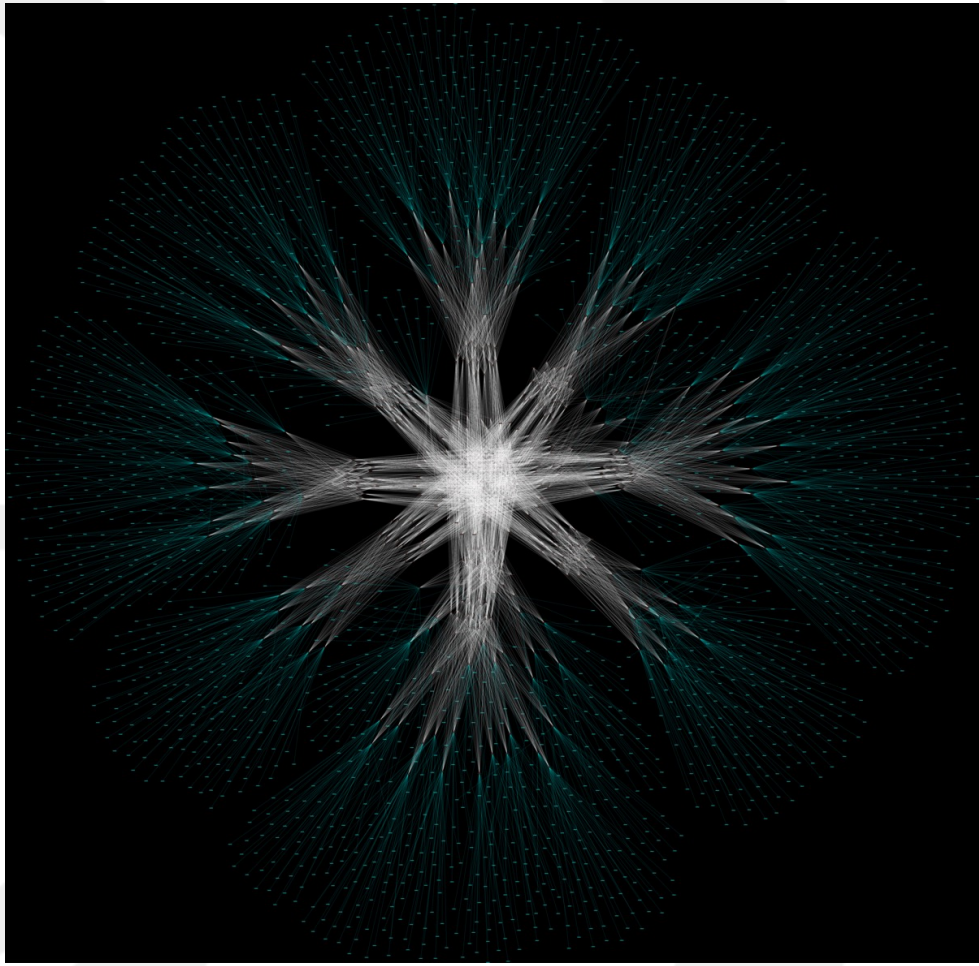
JUMP TRADING

- Privately-owned proprietary trading firm, established 1999
- Focus on algorithmic and high-frequency trading
- World-wide operations
  - 12 offices across US, EU, Asia, Pacific
- >700 employees

HPC at Jump

Jump's Research Environment (HPC / "The Grid")

- The platform where we develop and optimize trading strategies
- Sophisticated data-intensive and compute-intensive research workflows
- Technologically competitive with some of the largest publicly known research systems in the world
  - Thousands of servers
  - Hundreds of petabytes of storage
  - Fast network interconnects
  - Keeps growing: more hardware every year

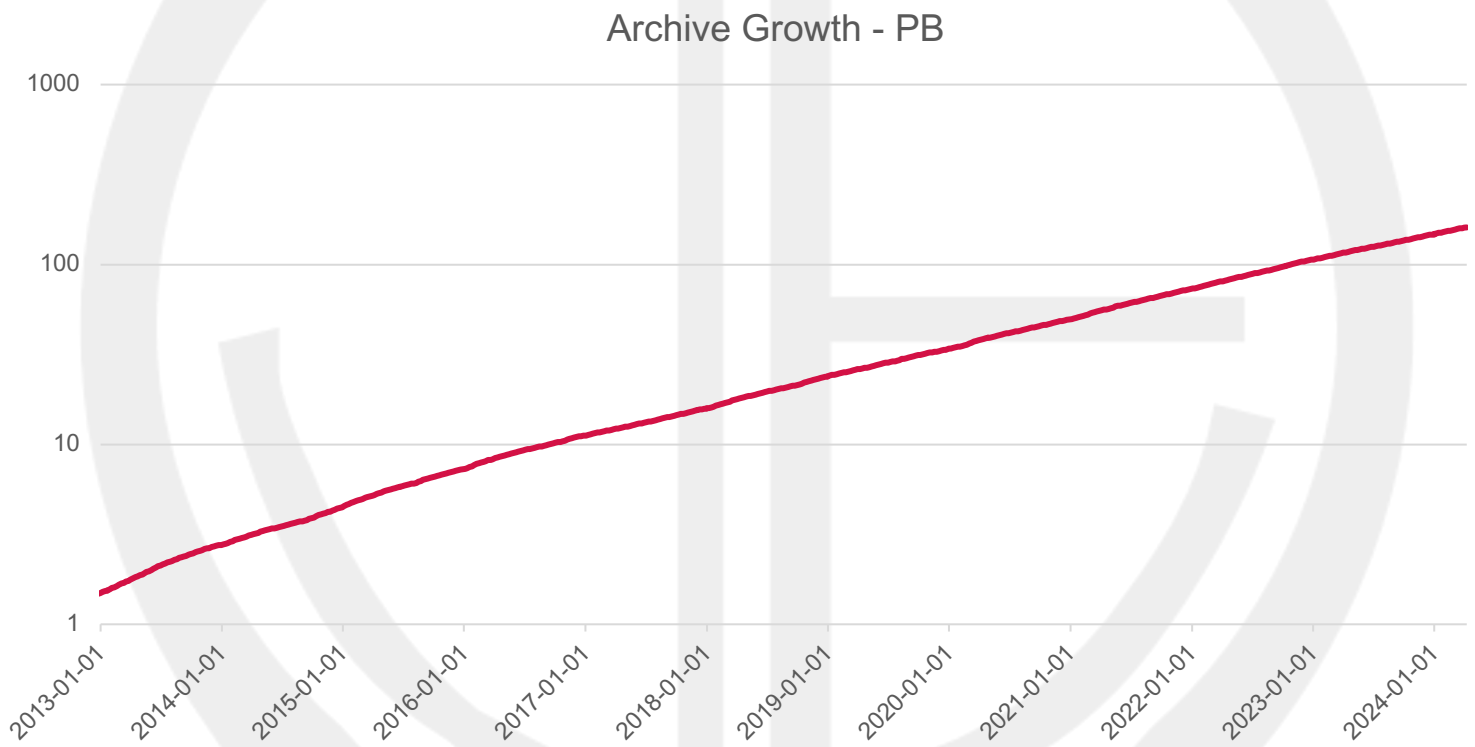


Fabric logical diagram
Image Credit: Olli-Pekka Lehto

Data Archive

- Realtime-updated repository of all market data
- Data captured from exchanges around the globe
- Derivative products for end-users

Ten Years of Archive Growth



Doubling ~2 years, currently growing at ~1.5PB/week
substantial day-on-day volatility

Archive infrastructure

- Before 2022, stored on conventional on-prem GPFS filesystem
  - Continual capacity management issues
  - Cost-inefficient
  - Could not scale to multiple colos
  - **The Legacy Archive was an obstacle to growth**
  - Re-architect to use commercial object storage

A New Design: Tiered Archive

- Able to run existing work-loads unmodified
  - POSIX filesystem presentation
- Decoupled from HPC fabric / filesystems
  - Accessible outside of HPC environment
- Able to accommodate >10x growth in capability
  - capacity, bandwidth
- Non-requirements:
  - Read-write mounts on compute nodes
  - Concurrent writes on the same file
  - Global consistency and file locking

The Tiered Archive

Three puzzle pieces:

1. **Filesystem presentation:** POSIX-like to allow existing apps to keep working
2. **Read Path:** Cloud storage backed by many layers of cache
3. **Write Path:** Pipeline to add new data to the system at scale

The Tiered Archive

Three puzzle pieces: CVMFS?

1. ✅ **Filesystem presentation:** POSIX-like to allow existing apps to keep working
2. ✅ **Read Path:** Object storage backed by many layers of cache
3. ❌ **Write Path:** Pipeline to add new data to the system at scale

Tiered Archive: CVMFS

Accessing Data Federations with CVMFS

Derek Weitzel[1], Brian Bockelman[1], Dave Dykstra[2], Jakob Blomer[3], Ren Meusel[3]

[1] University of Nebraska - Lincoln Holland Computing Center, US
[2] Fermilab, Batavia, IL, US
[3] CERN, Geneva, CH

E-mail: dweitzel@cse.unl.edu

Abstract. Data federations have become an increasingly common tool for large collaboratior such as CMS and Atlas to efficiently distribute large data files. Unfortunately, these typicall

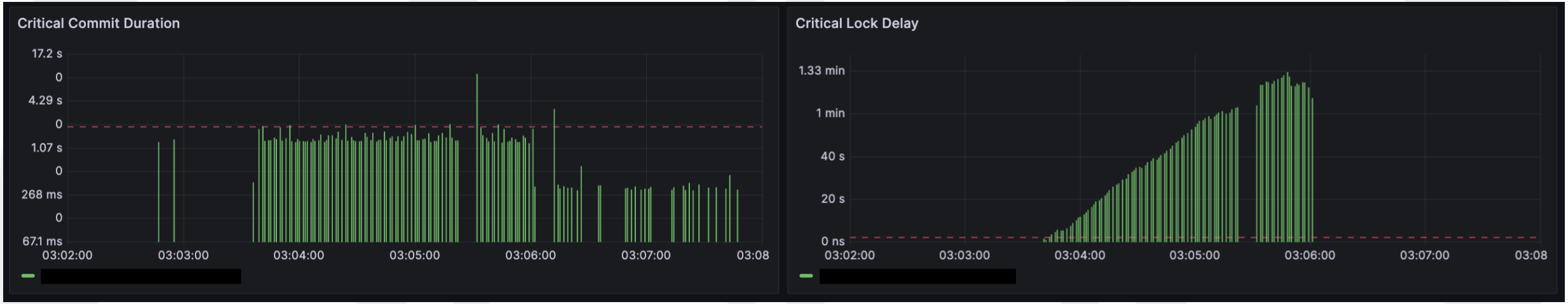
Describes how to use CVMFS with “external” data
ie not stored in content-addressable form

High performance write path

- CVMFS “grafting” to separate data and metadata write paths
  - Data is uploaded directly to cloud storage
  - Doesn’t use CVMFS content-addressable storage
- cvmfs-rsync
  - diff src and dest, upload changes to object store
  - Capture file metadata
- cvmfs_swissknife ingestsql
  - Send metadata to gateway for addition to repo

Problems and Solutions

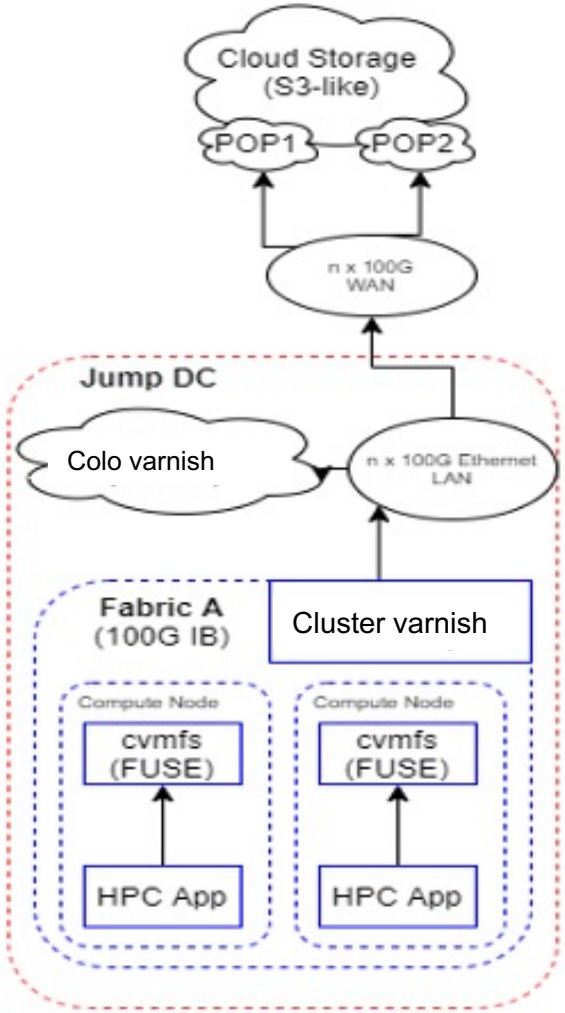
Grafting throughput



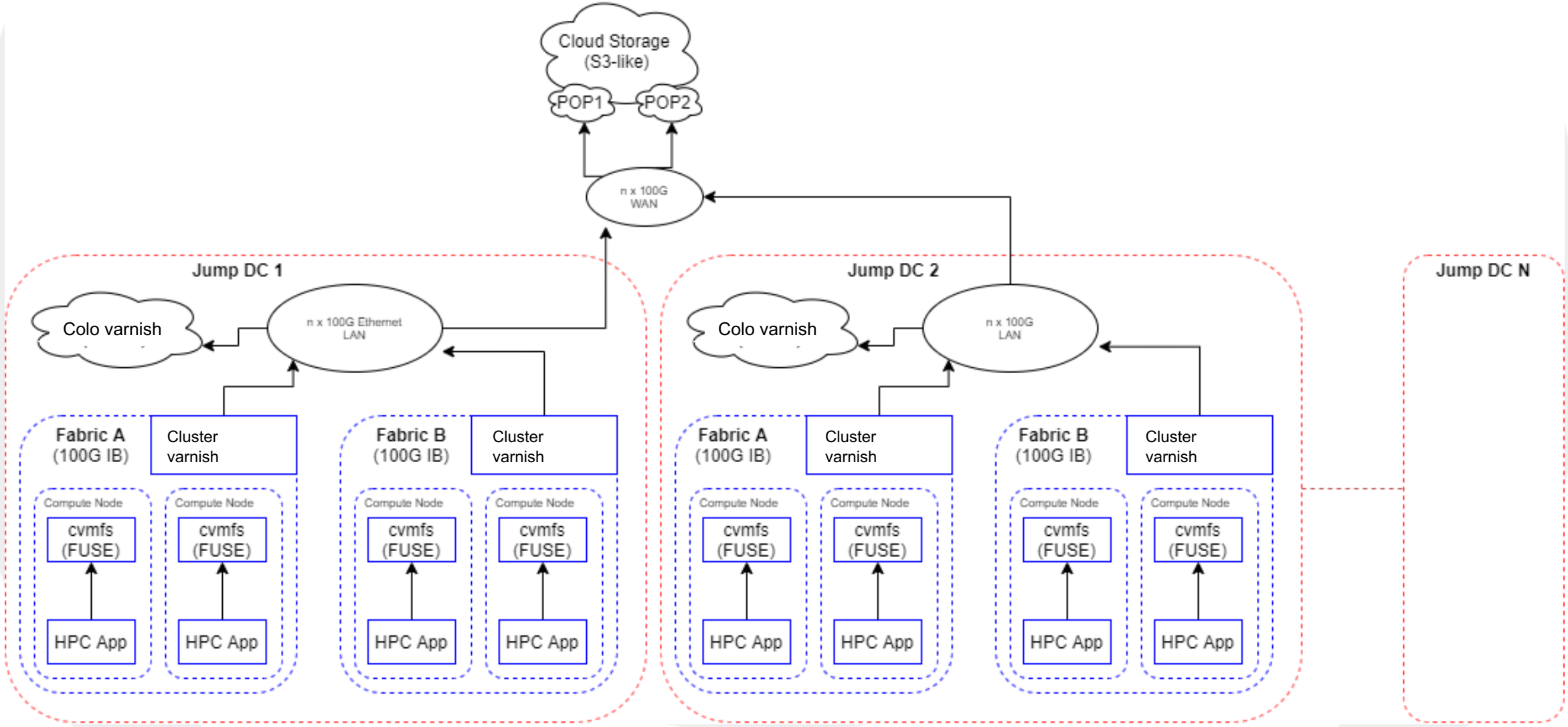
- Grafting is a point of serialization
  - Need to optimise critical path performance
  - Can now achieve a floor time of < 300ms, down from 2+ sec.
  - Fixed scenarios leading to grafts of minutes duration

Read cache hierarchy

- Use Varnish HTTP cache
  https://varnish-cache.org/
  - NVME caches in each HPC cluster
    - Bridges cluster network and DC ethernet
    - Measured 30GiB/s per server
  - NVME+HDD caches in each DC
  - Cache tiers horizontally scalable as required



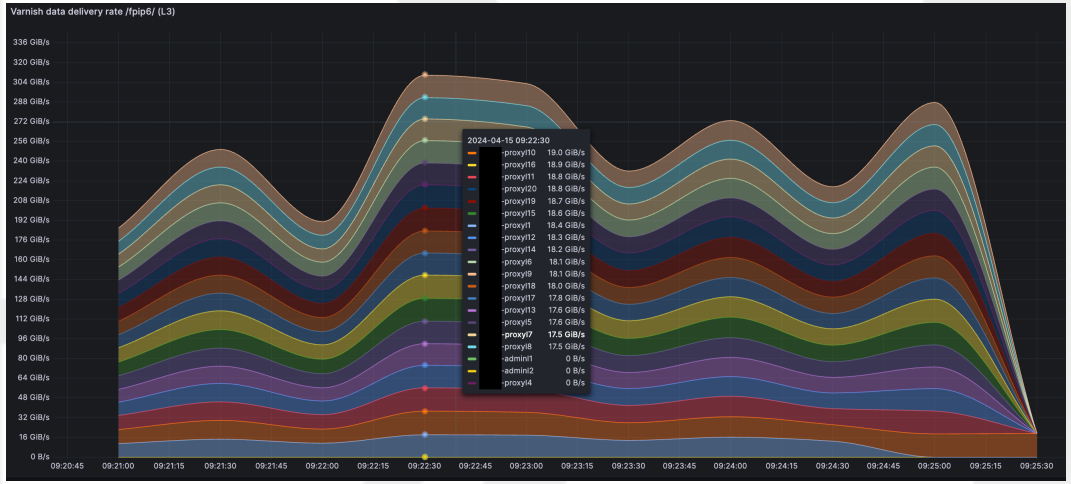
Scaling out


Cloud Storage
(S3-like)
POP1
POP2
n x 100G
WAN
Jump DC 1
Colo varnish
n x 100G Ethernet
LAN
Fabric A
(100G IB)
Cluster
varnish
Fabric B
(100G IB)
Cluster
varnish
Compute Node
cvmfs
(FUSE)
HPC App
Jump DC 2
Colo varnish
n x 100G
LAN
Fabric A
(100G IB)
Cluster
varnish
Fabric B
(100G IB)
Cluster
varnish
Jump DC N

Read cache efficiency

- Shard data across all caches
  - Use rendezvous hashing https://en.wikipedia.org/wiki/Rendezvous_hashing
  - Failure of a cache causes traffic to be equally redistributed
  - Shard at 24MiB chunks, not whole object

  * Supported in CVMFS as compile time addition
- Added health-checking code to accommodate cache instance failures

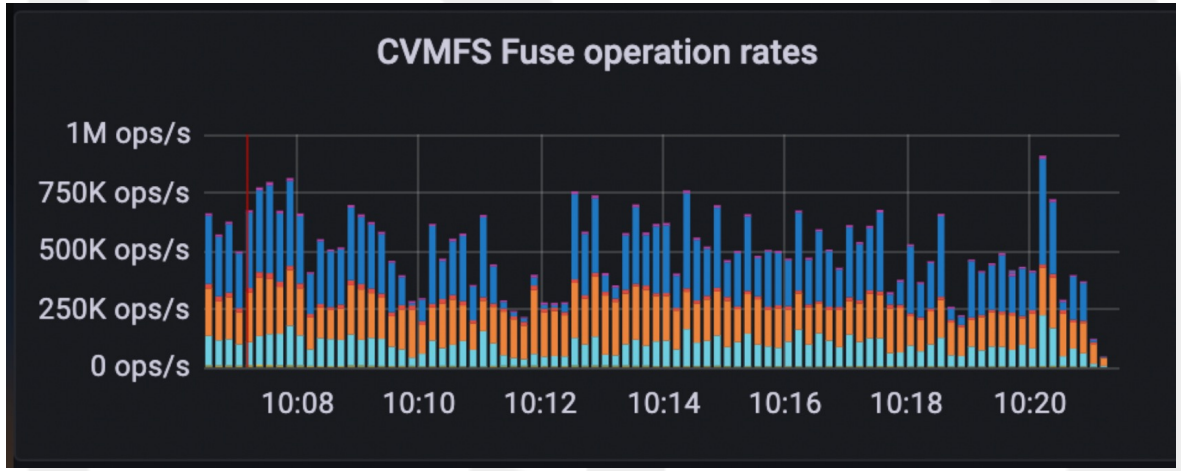


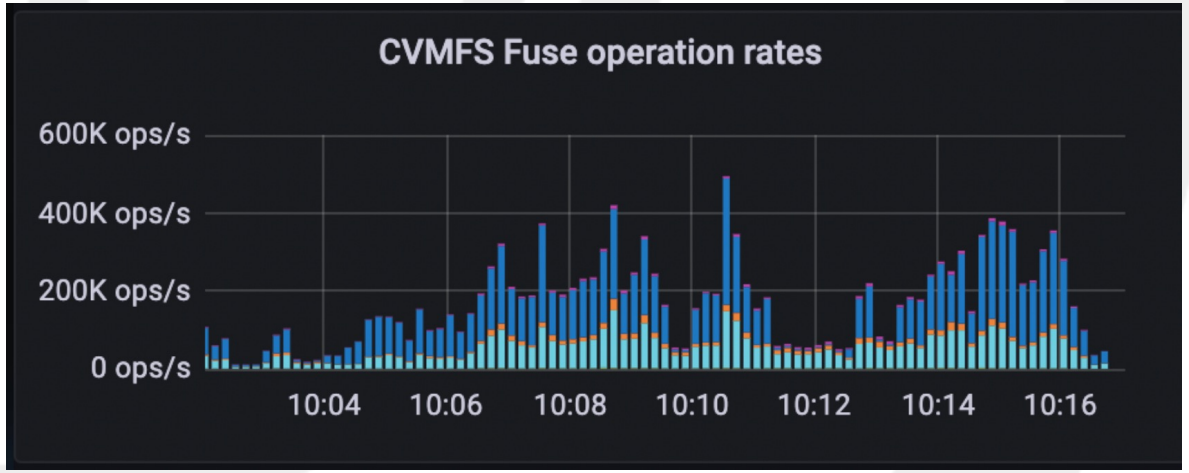
Object immutability

- CVMFS has direct mapping from filepath -> object key
- Changing a file requires cache invalidation
  - Expensive, race-prone
- Solution:
  - Object key = `path/.filename.<content shasum>`
  - Add file `path/filename` to CVMFS as:
    - File: `path/.filename.<content shasum>`
    - Symlink: `path/filename -> .filename.<content shasum>`
- Changing a file -> new dot file, flipped symlink (atomic)

Symlink lookups

* Dot-scheme is symlink heavy
- FUSE does not cache symlink lookups by default
- Enabling it exposed bug when symlink target changes – kernel cached value not expired
- Fixed with CVMFS, libfuse, kernel fuse patches





Low read performance

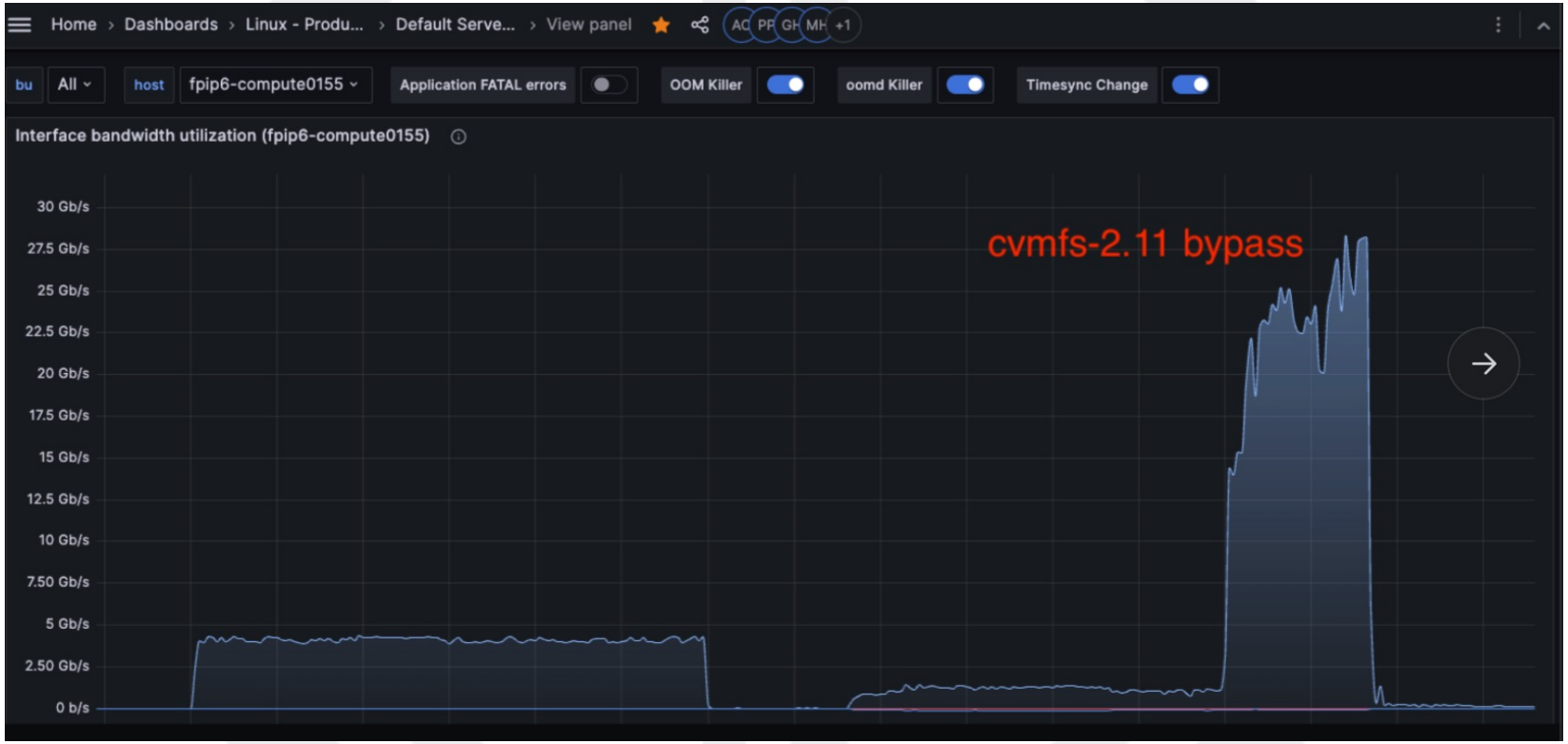
Reads are slow: ~500MB/s

- Reads in chunks of 24MiB
- Downloads, sha1sums multiplexed onto a single thread
- Local disk cache must be populated before returning read
  - Write-limited on local disk
- Data ends up in page cache twice
- In our case, hit rate of local caching very low

Cache bypass “direct” read

- Completely bypass CVMFS download path
- Map each fuse_read() to an HTTP GET
  - Multithreaded
  - No sha1summing
- No local cache
  - rely on page cache, fast network and varnish
- Tune kernel readahead and increase fuse message size
  - 128 kiB ->1MiB
  - Requires minor kernel patches (RHEL8)

Cache bypass “direct” read

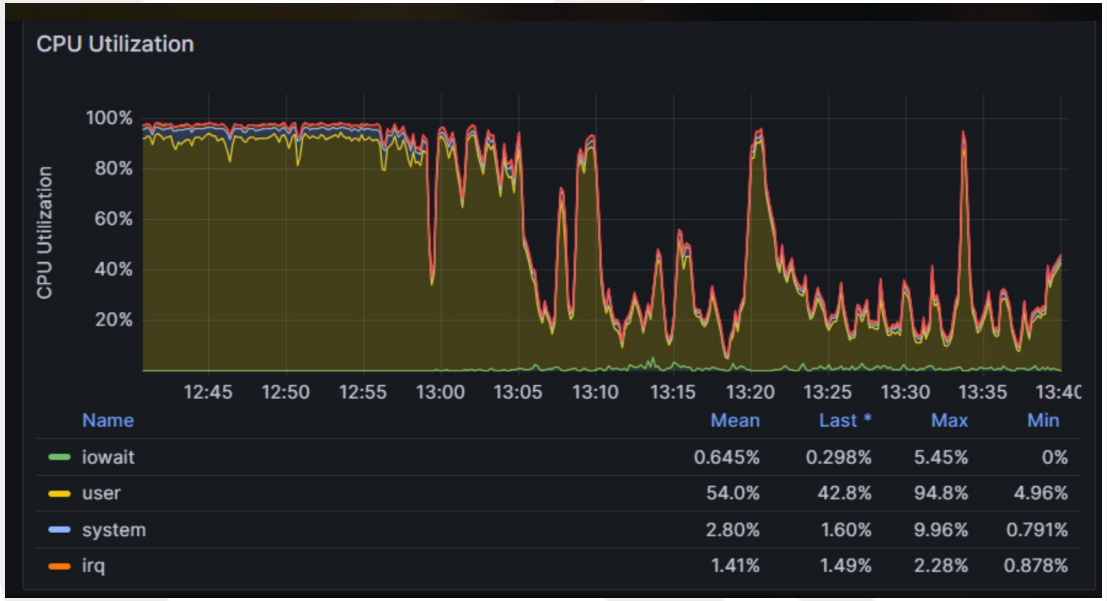


8x parallel reads, randomly selected files between 0 and 2GiB, 1MiB reads
CVMFS cache in shmem
500MiB -> > 3GiB/s

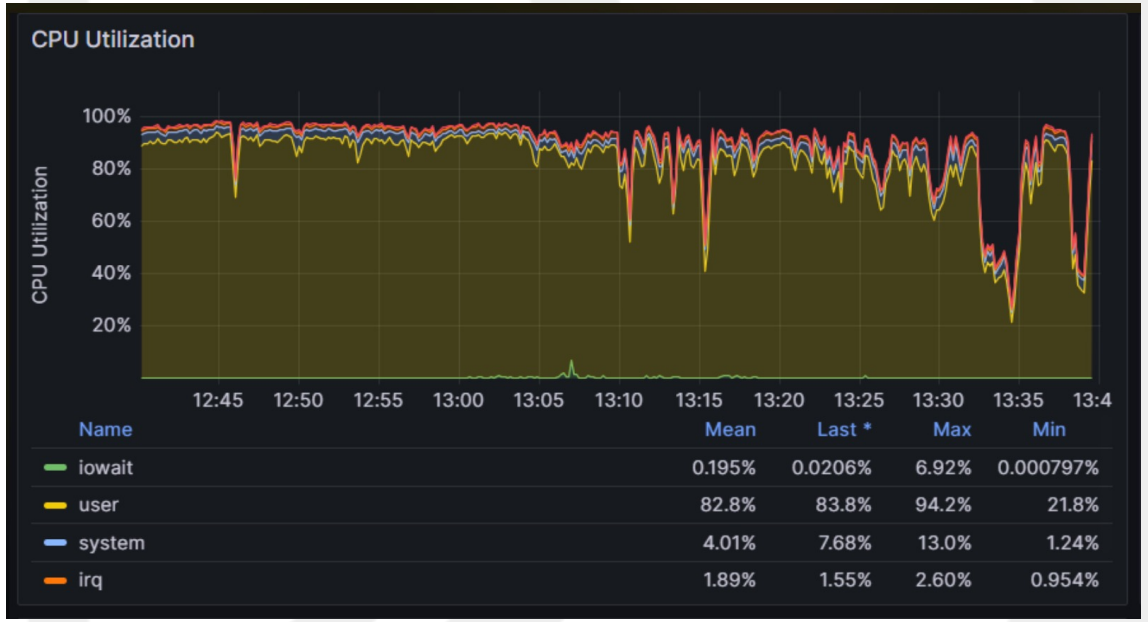
Catalogue load inefficiencies

- Every directory is represented by a sqlite database stored in an object.
- Catalog loads could stall all fuse operations
  - Very bad if historical data where catalog not in cache
  - Wide compute nodes very likely to encounter the issue
- Single-threaded catalog decompression could peg a core

Catalogue load improvements



128 core AMD system

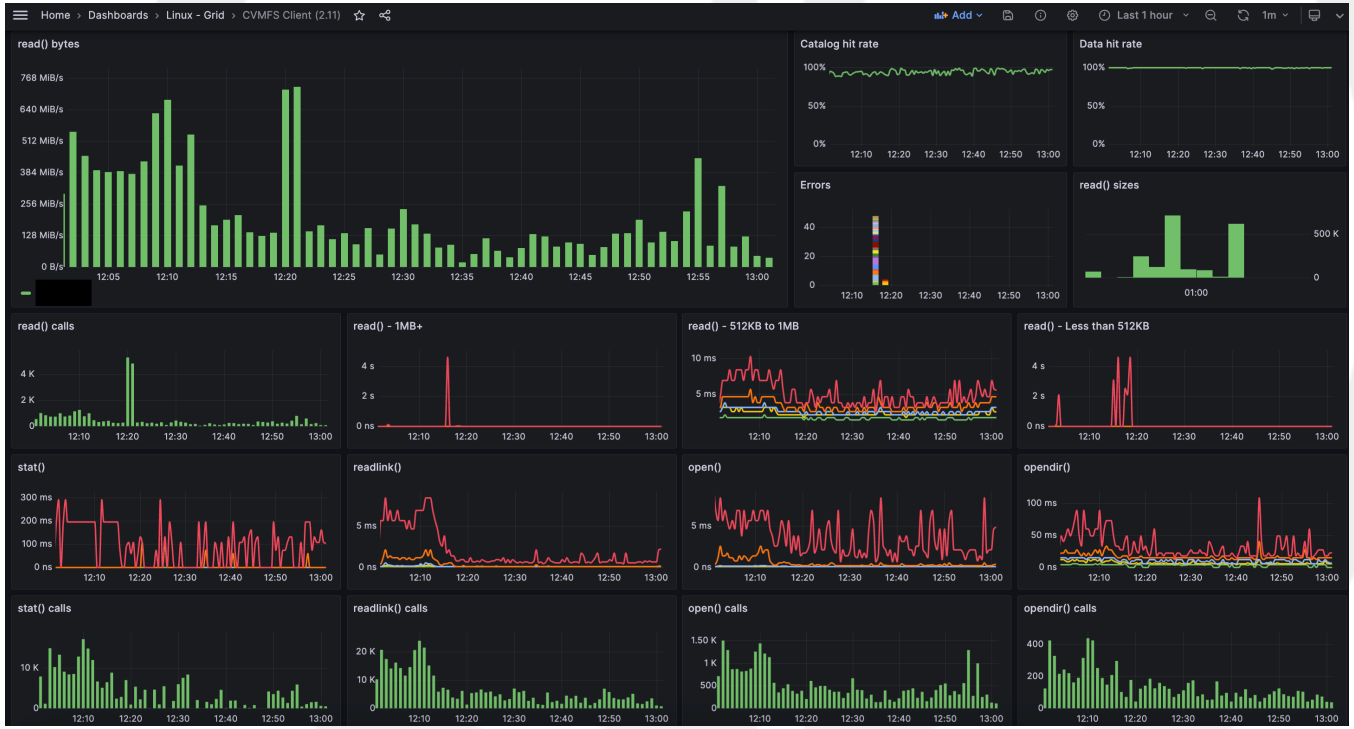


Fixed locking, multithreaded decompression

Startup time down from ~90s to ~5s

Observability

- Export internal counters, operation latencies via telegraf



Thank you

Questions?

mharvey@jumptrading.com

Infiniband fabric-aware routing

Varnish caches have multiple HBAs
Shard over all caches, but client routes via the best interface
Reduces P-values, fabric congestion