Preparing for HL-LHC

Increasing the LHCb software publication rate to CVMFS by an order of magnitude

Enrico Bocchi, Jakob Blomer, Christopher Burr,

Benjamin Couturier, Dan van der Ster

vCHEP 2021, 19th May

Introduction

- **Software** is crucial for HEP
  - Operation of experimental facilities
  - Events filtering during data acquisition
  - Reconstruction jobs for analysis
- … as well as its efficient **distribution at scale**
  - Software to be provided to thousands of clients
  - Heterogeneous resources at different sites

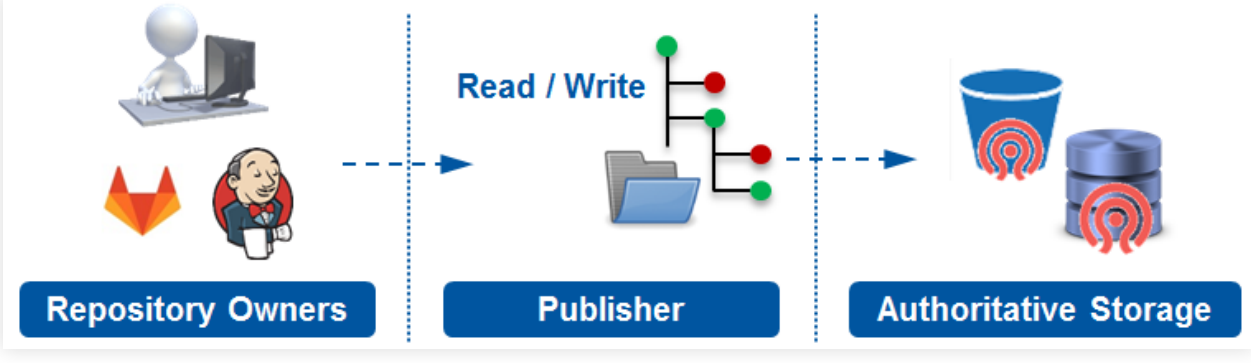
CVMFS for software delivery at scale

Increase publication rate on CVMFS

- Use case: **LHCb nightly builds**
  - Challenging use-case with strict requirements
  - Nightly builds and partial stack builds
  - 3.8 M files, 200 GB to be installed daily
- Goal: **Faster publication of LHCb software**
  1. Parallelize publication process
  2. More performant storage backend
  3. Efficient management of installations

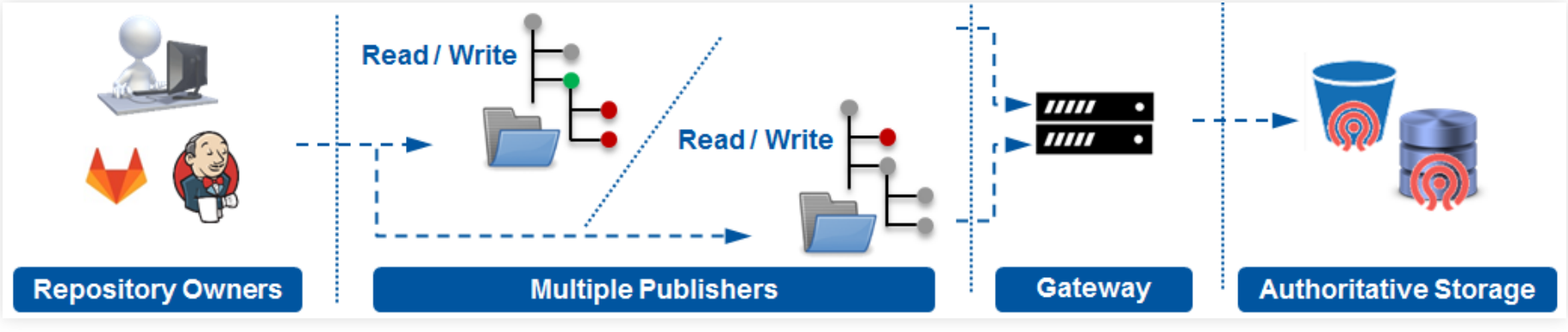
Traditional CVMFS setup

- One publisher applies changes to the repository
  - Acquires global write lock
  - Accesses authoritative storage directly
  - Only one publication at a time possible



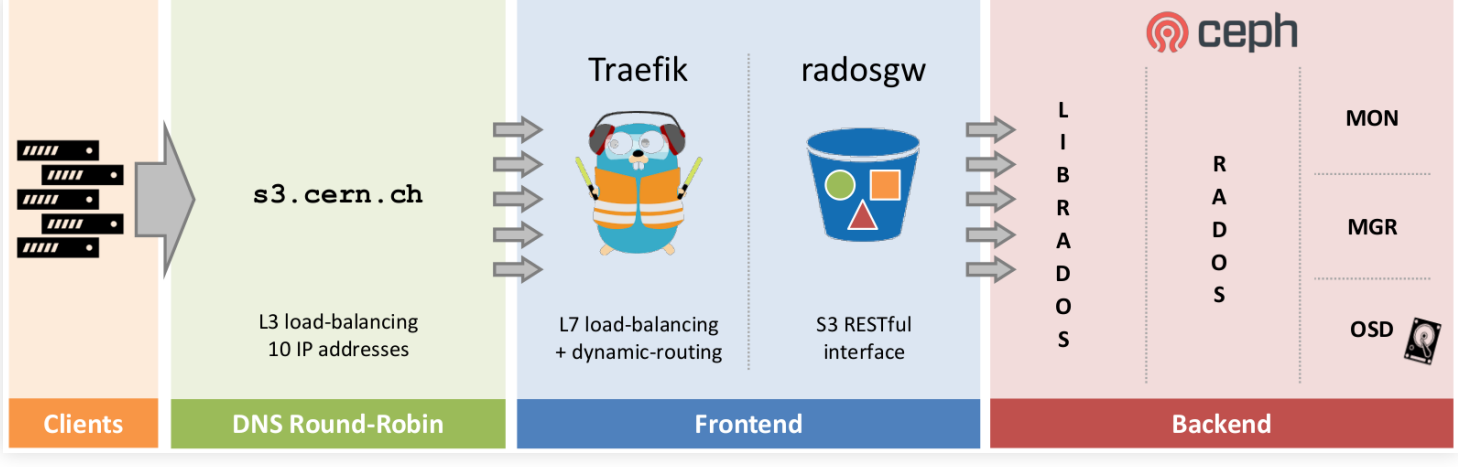
CVMFS setup with Gateway

- Allows for **concurrent publications**
  - Issues time-limited leases for specific sub-paths
  - Provides API to coordinate across publishers
- Coordinates multiple publishers
  - Each publisher locks a repository sub-tree
  - Parallelism depends on a good repository structure

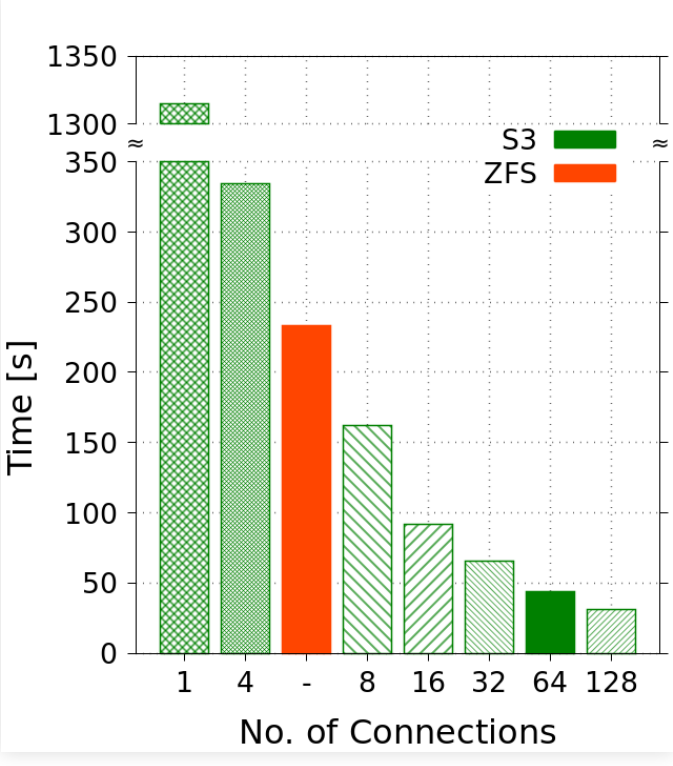


CVMFS migration to S3 storage

- All repositories hosted at CERN migrated to S3
- `s3.cern.ch`:
  - Single-region Ceph cluster, 5.8 PB raw
  - Load-balanced across 16 Traefik frontends
  - 4x dedicated RGWs for CVMFS

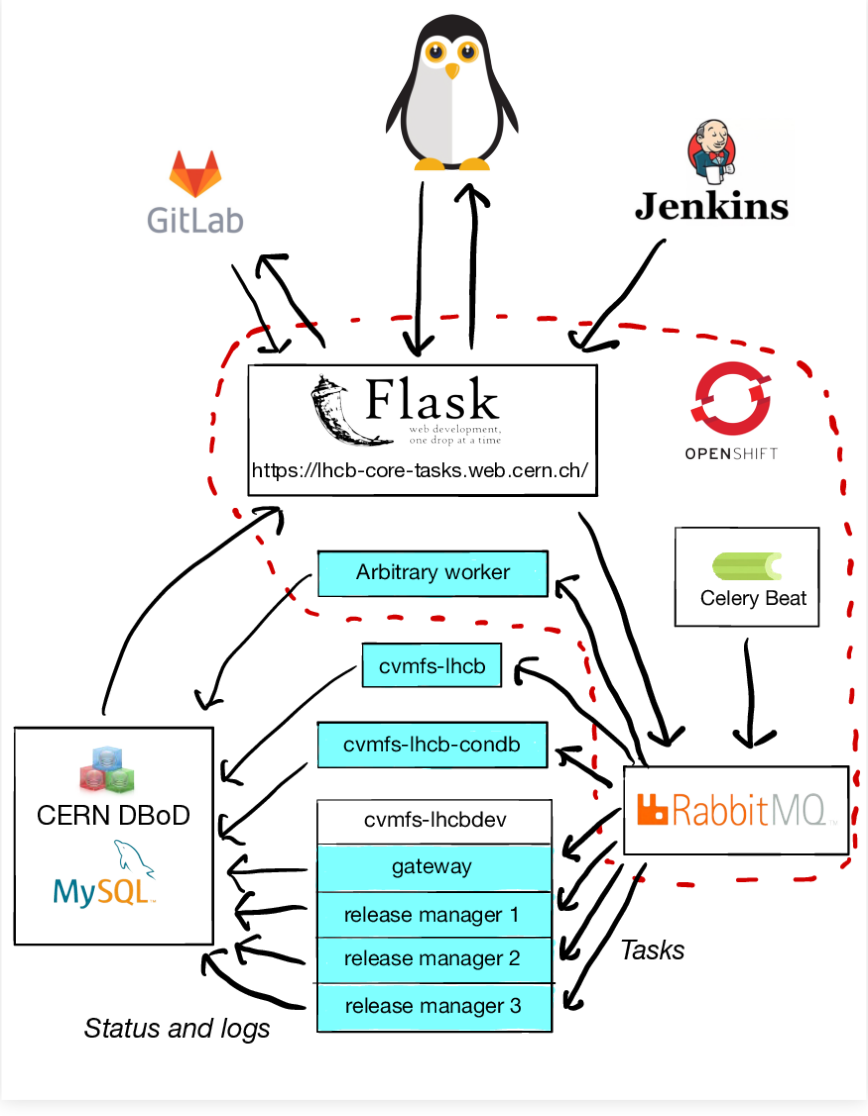


Performance improvements for CVMFS

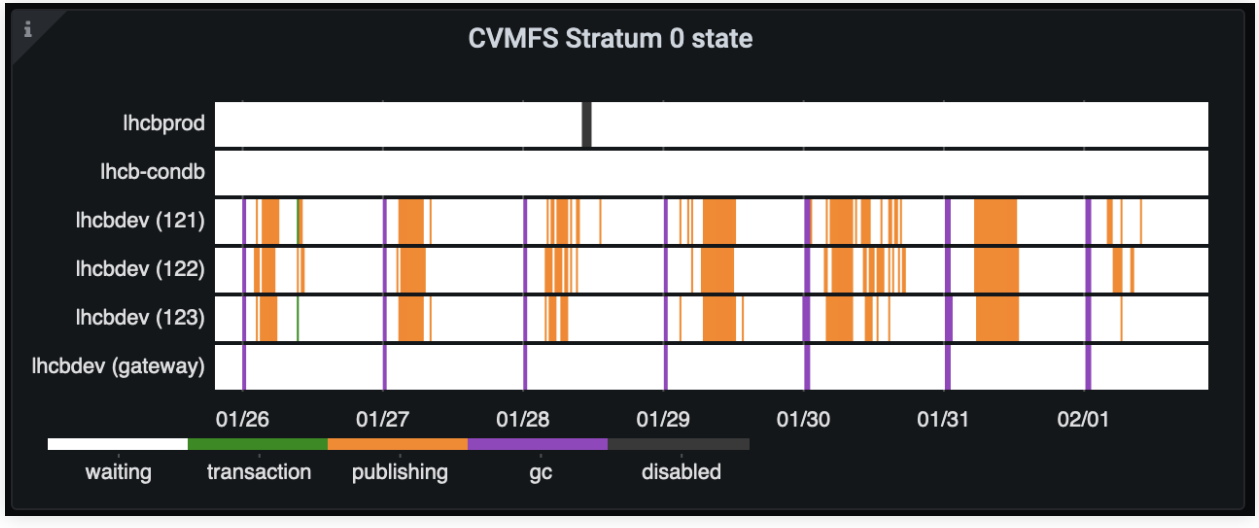


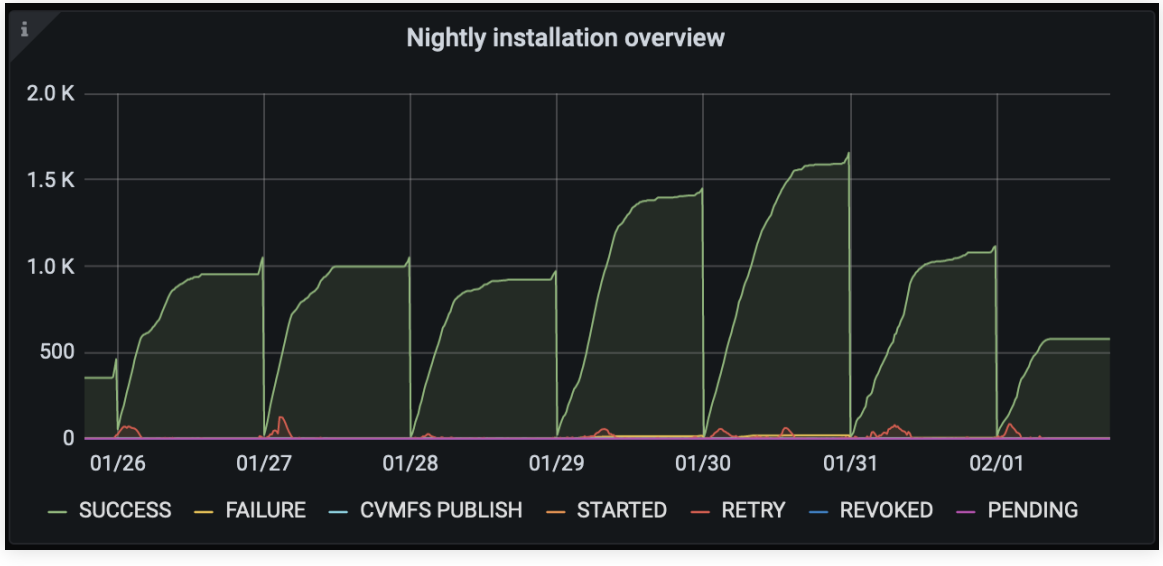
- Publication time benchmarking:
  - Workload: 250k files, 4 kB each
  - Files spread across 250 folders each with a cvmfs catalog
  - Full publication through CVMFS
- Default parallel connections: 64
- S3 outperforms volume storage: Publication on S3 is 5x faster

LHCb installations orchestration



Monitoring of CVMFS publications





Conclusions

- CVMFS and Storage teams brought major improvements
  - New gateway component for parallel publications
  - S3 object storage improved performance (5x faster)
- LHCb implemented advanced orchestration framework
  - Take advantage of the infrastructure underneath
  - Task queue, retry logic, error handling, …
- In production since November 2020
  - Meets today’s requirements for stability and speed
  - Room for scalability by adding publishers

Backup

CVMFS repositories for LHCb

- **`lhcb.cern.ch`**
  - Production software
  - Low churn rate, no garbage collection
- **`lhcb-condb.cern.ch`**
  - Conditions data
  - Slowly growing, no garbage collection
- **`lhcbdev.cern.ch`**
  - Nightly builds and partial stack builds
  - 3.8 M files, 200 GB to be installed daily
  - Requires frequent garbage collection

LHCb installations orchestration

- Installations are managed by Celery agent
  - Running on each publisher (3 at the moment)
  - Fetches jobs from RabbitMQ + MySQL (CERN DBoD)
- Each installation task
  1. Infers the destination directory on CVMFS
  2. Checks if lock exists on the directory
  3. If free, installs || Otherwise, waits and retries

Locking for garbage collection

- Deleted files on CVMFS are not purged from storage
  - To remove them completely, GC is needed
  - High churn rate repositories require frequent GC
- Garbage collection uses a **mark and sweep** approach
  - Requires exclusive access to the repository
  - Gateway is instrumented to take global lock
    - Daily, in the evening on `lhcbdev.cern.ch`
- Both mark and sweep are now parallelized
  - Take advantage of multicores and parallel connections
  - ~ 1 hour on `lhcbdev.cern.ch` to delete 100+ k files

Distribution to clients

- Typically, clients read from Stratum 1
  - Introduces (small) replication delay
  - Garbage collection inhibits replication
- With S3, clients could read from storage
  - No replication delay
  - Garbage collection is non-blocking
  - Still using intermediate caches