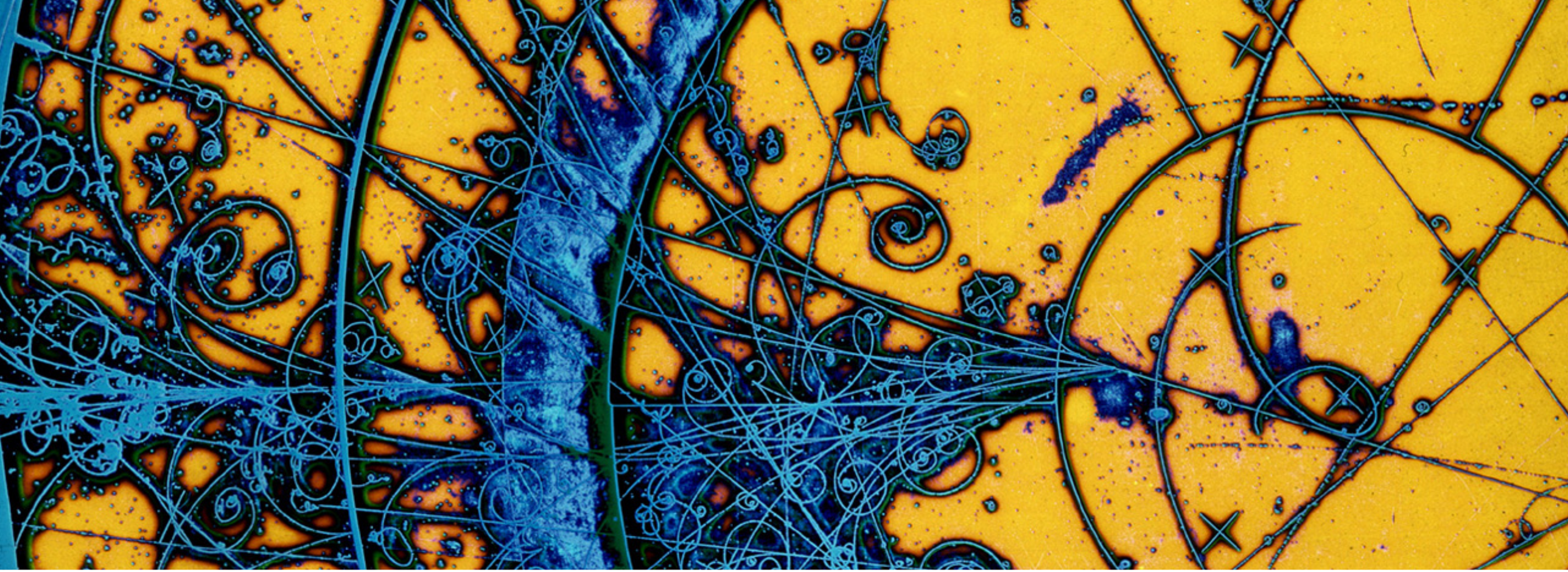
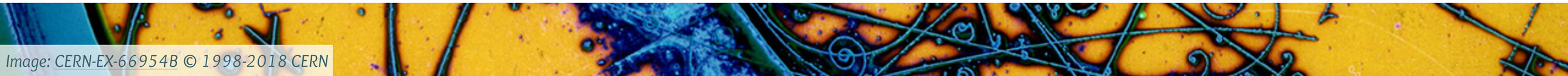


CVMFS usage and performance for LHCb

Chris Burr and **Ben Couturier**

12th September 2022

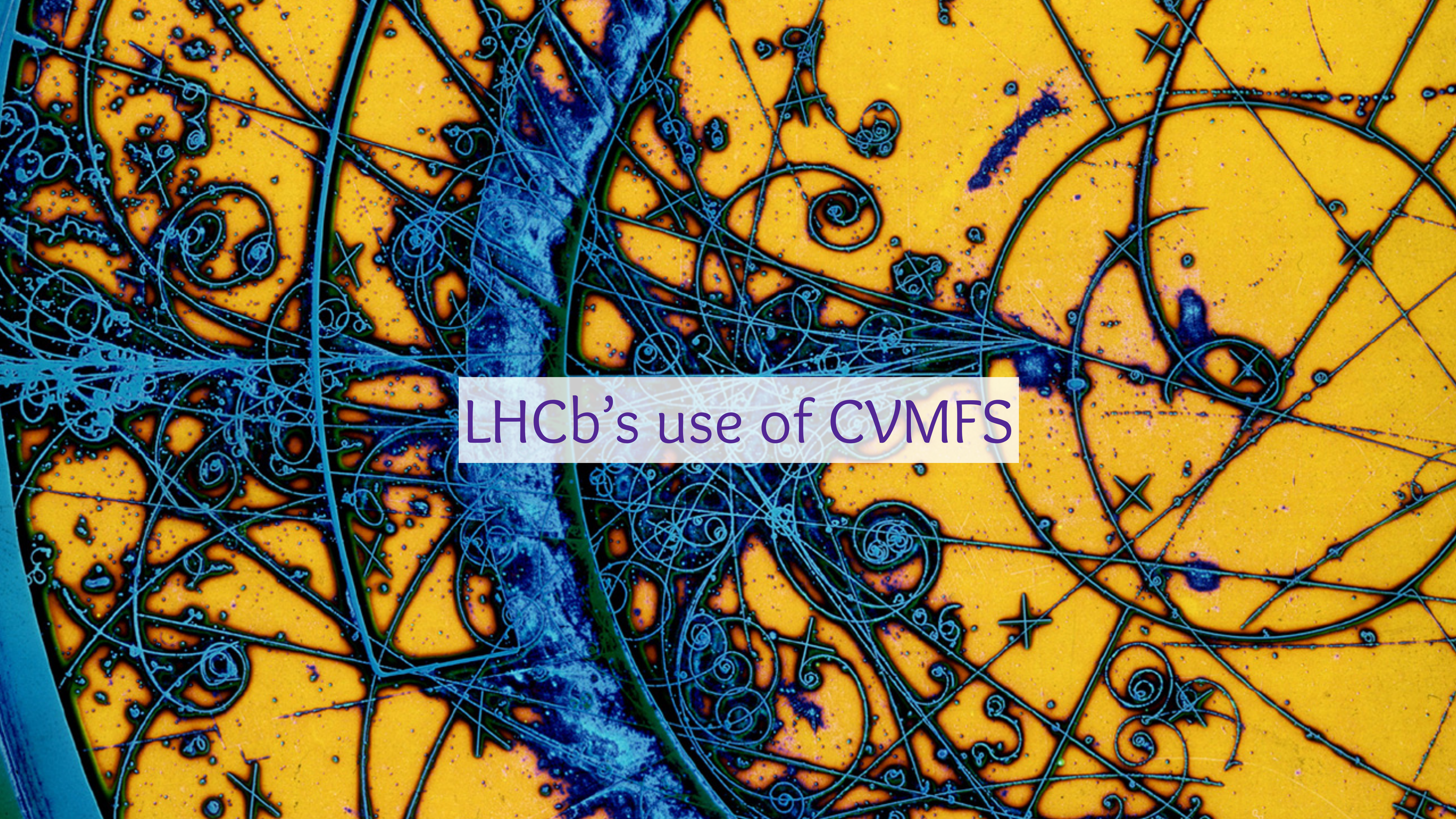


Overview

➤ Three CVMFS repositories managed by LHCb*

Repository	Size	GC?	Purpose	Comments
`/cvmfs/lhcb.cern.ch`	8.9 TB	✘	Production software (online and offline)	Used by Grid and LHCb Trigger farm A few transactions/day in average **Crucial for long term software preservation**
`/cvmfs/lhcbdev.cern.ch`	3.4 TB	✔	Nightly builds Analysis environments Miscellaneous tests	Deduplication is really helpful High turnover Mostly short term interest Crucial for development team
`/cvmfs/lhcb-condb.cern.ch`	2.0GB	✔	Detector conditions	Frequent releases Many small files Decouple the release of the GIT conditions from main repository

**Also make use of cernvm.cern.ch and implicitly grid.cern.ch, see backup for details*



LHCb's use of CVMFS

Interactive LHCb environment

- Use CVMFS to provide a Conda environment with LHCb software installed
  - Relatively lightweight, primarily provides LHCb-specific Python packages
  - Specific software environments can be launched with `lb-*` commands
  - Uniform starting environment for all LHCb users
  - Supports (almost) any Linux on `x86_64`, `aarch64` and `ppc64le`

```
-bash-4.2$ source /cvmfs/lhcb.cern.ch/lib/LbEnv
*******************************************************************************
*                              ---- LbEnv ----                                  *
*******************************************************************************
 --- User_release_area is set to /afs/cern.ch/user/c/cburr/cmtuser
 --- CMAKE_PREFIX_PATH is set to:
    /cvmfs/lhcb.cern.ch/lib/lhcb
    /cvmfs/lhcb.cern.ch/lib/lcg/releases
    /cvmfs/lhcb.cern.ch/lib/lcg/app/releases
    /cvmfs/lhcb.cern.ch/lib/lcg/external
    /cvmfs/lhcb.cern.ch/lib/contrib
    /cvmfs/lhcb.cern.ch/lib/var/lib/LbEnv/2541/stable/linux-64/lib/python3.9/site-packages/LbDevTools/data/cmake
-------------------------------------------------------------------------------
-bash-4.2$ which lb-run
/cvmfs/lhcb.cern.ch/lib/var/lib/LbEnv/2541/stable/linux-64/bin/lb-run
-bash-4.2$ which lb-conda
/cvmfs/lhcb.cern.ch/lib/var/lib/LbEnv/2541/stable/linux-64/bin/lb-conda
-bash-4.2$ which lb-dirac
/cvmfs/lhcb.cern.ch/lib/var/lib/LbEnv/2541/stable/linux-64/bin/lb-dirac
-bash-4.2$ which lb-dev
/cvmfs/lhcb.cern.ch/lib/var/lib/LbEnv/2541/stable/linux-64/bin/lb-dev
-bash-4.2$ which ganga
/cvmfs/lhcb.cern.ch/lib/bin/ganga
-bash-4.2$ which singularity
/cvmfs/lhcb.cern.ch/lib/var/lib/LbEnv/2541/stable/linux-64/bin/singularity
```

Physics software

- All LHCb software is installed on `/cvmfs/lhcb.cern.ch`
  - Continuously used for running both latest and legacy versions (back to the start of LHC data taking)
- Used by 200,000+ running grid jobs
  - Number of clients continues to grow exponentially
- Also used as the only way of running LHCb software
  - Even local full stack builds depend on taking externals from LCG via CVMFS

Detector conditions

- CVMFS is also used for distributing detector conditions data
  - Committed by the detector control system to a NFS hosted Git repository
  - Synchronised to CVMFS for long term archive and offline access
- Dedicated CVMFS instance
  - Not essential but has some nice-to-have benefits
- Has been working very smoothy for many years
  - Back-ported to all (useful) historical software versions
  - Simple with no infrastructure to maintain
- See “A Git-based Conditions Database backend for LHCb” for info

Nightly installations

- Heaviest use of CVMFS is distributing LHCb nightlies
  - Often deploy many hundreds of gigabytes of binaries during the night (only a “popular” slots are installed)
  - Distributed setup with 6 publishers
- The LHCb software is split into ~20 projects
  - Built and tested independently (often on different machines in parallel)
  - Many legacy branches for processing Run 1+2 datasets
- When building nightlies machines struggle with disk IO
  - Currently installing on local disks
  - Even fast NVMe drives struggle to keep up
- Will soon move to building against CVMFS installed packages
  - Will significantly increase the volume of binaries
  - Publication performance and propagation delay within CERN will become critical

Deploying non-x86_64 software

- Increasing interested in non-x86_64
  - Most of our installation jobs are independent of the target architecture
  - Some jobs are much easier if you can run for the target architecture (e.g. `pip install`)
  - Also useful for testing before publishing transactions
- Linux supports automatically launching executables with emulation
  - Uses QEMU + `binfmt_misc`, similar to Rosetta(2) on macOS
- Works very well for avoiding time spent supporting edge cases

```
[lhcb.cern.ch] $ python -c 'import platform; print(platform.machine())'
x86_64
[lhcb.cern.ch] $ /cvmfs/lhcb.cern.ch/lhcbdirac/prod/Linux-aarch64/bin/python -c 'import platform; print(platform.machine())'
aarch64
```

- Though of course there is a performance cost (not representative, typically ~5x slower)

```
[lhcb.cern.ch] $ /cvmfs/lhcb.cern.ch/lhcbdirac/prod/Linux-x86_64/bin/dirac-wms-cpu-normalization
Estimated CPU power is 18.9 HS06
[lhcb.cern.ch] $ /cvmfs/lhcb.cern.ch/lhcbdirac/prod/Linux-aarch64/bin/dirac-wms-cpu-normalization
Estimated CPU power is 0.9 HS06
```

Conda environments

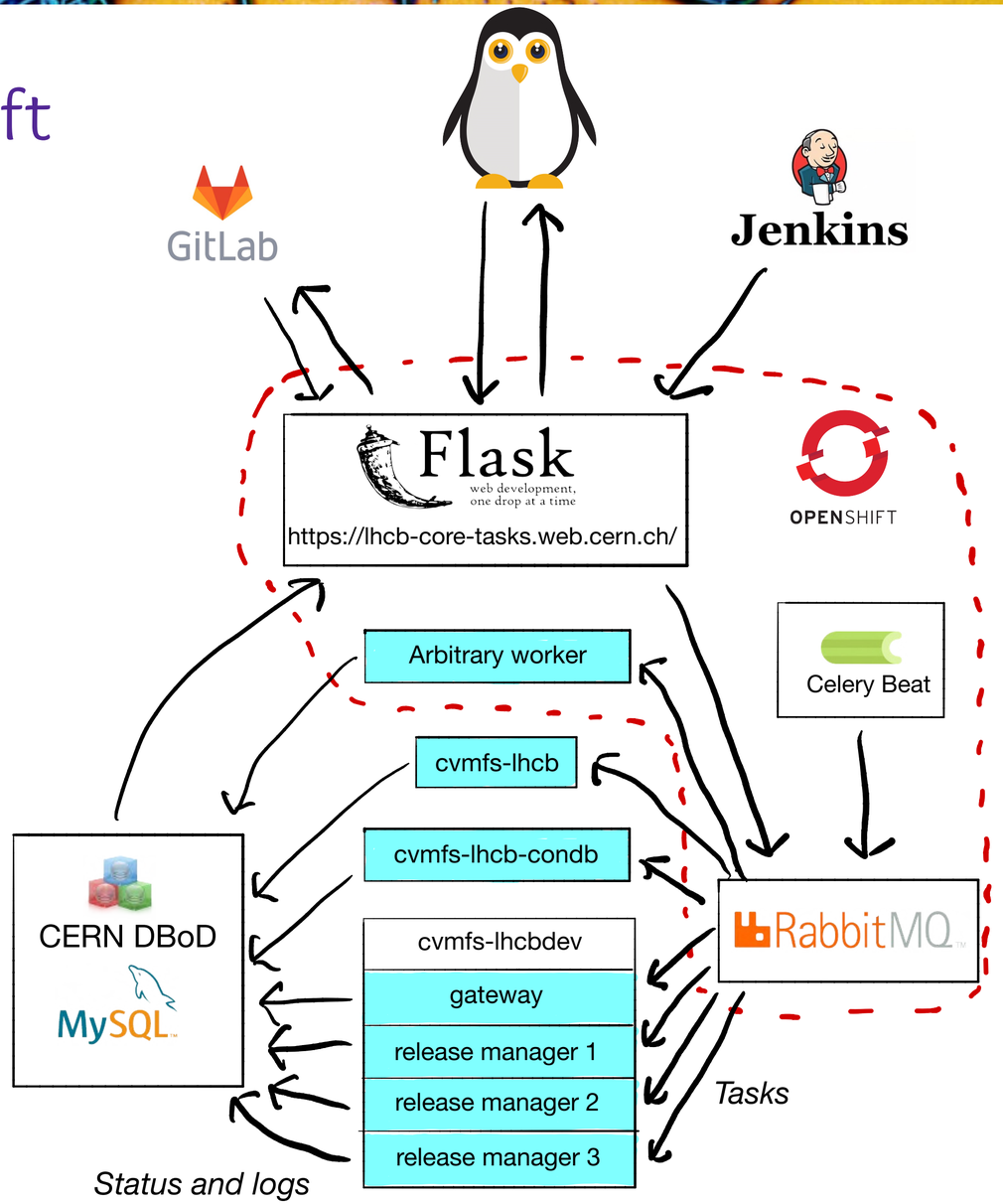
- Provide conda environments on lhcbdev
  - “default” environment contains ROOT + gcc + cmake + boost + many common Python packages
  - Option for analysts to deploy their own by committing a environment.yaml file to GitLab
- Lightweight command* in the default LHCb environment to launch a subshell
  - Also the option to create a virtual environment (venv) on top
  - `cvmfs_server` enter could be interesting if it can be packaged independently
- Avoids issues with poor filesystem performance with AFS or EOS
- Portable between most Linux distributions
  - Being used on SLC6, CC7, Ubuntu, Arch Linux and more
- Similar to using unpacked.cern.ch to get a reproducible environment
  - Easier to define (list of packages) but less flexible than a full container

*https://gitlab.cern.ch/lhcb-core/lbcondawrappers/

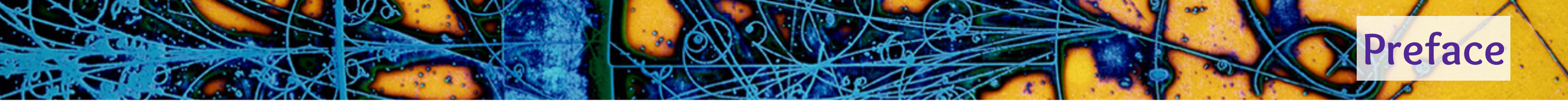
How we manage CVMFS installations

- LHCb managed services running on OpenShift
- Celery application “lbtaskrun”
  - Uses RabbitMQ to distribute “tasks” to workers
- Web application “lbtaskweb”
  - CERN SSO access for experts
  - Token based authentication for receiving jobs
- Cron-like jobs triggered by Celery Beat
- Interactive wrapper for manual transactions
  - Takes care of stopping automated activity and locking

See **LHCb talk at the 2021 workshop** for details



Performance observations

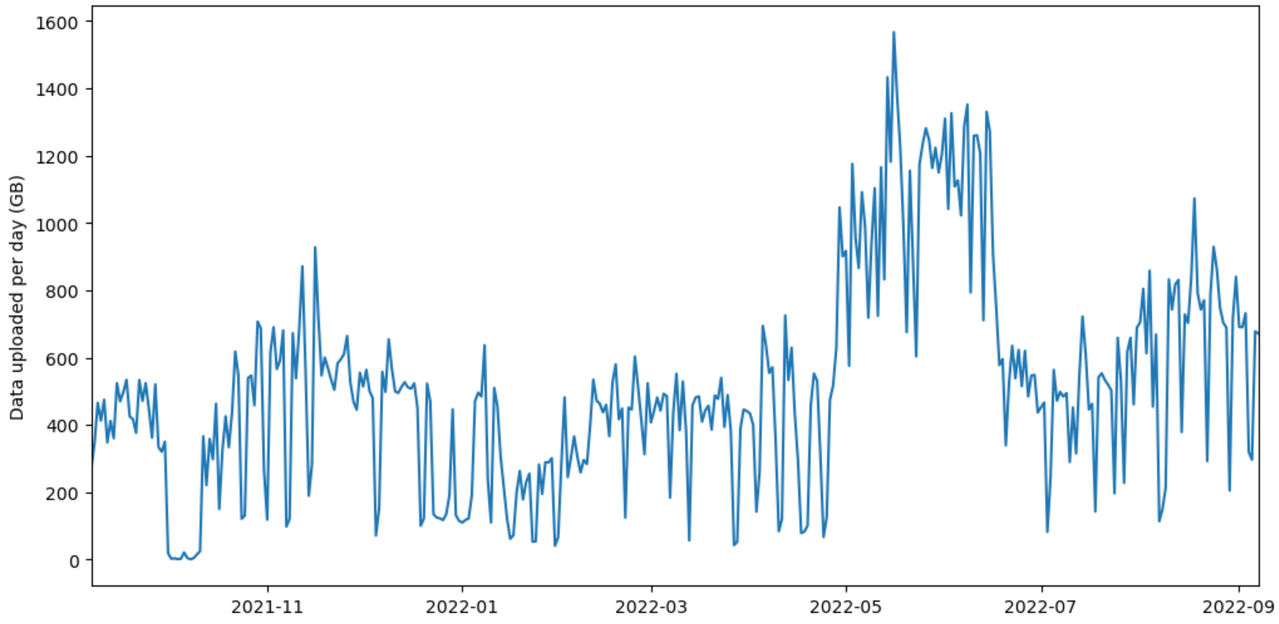


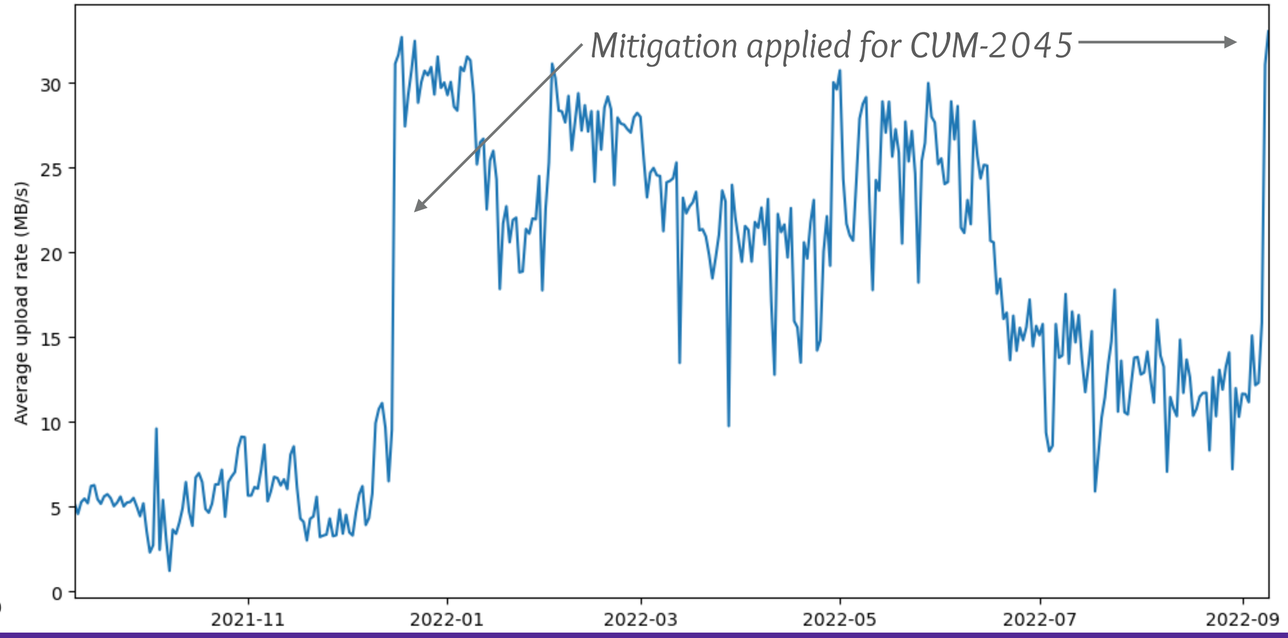
Preface

- CVMFS is central to all LHCb activities and works very well
  - Hard to imagine how we ever worked without it (and unpleasant to remember 😉)
- We've very happy with CVMFS performance
  - Manual transactions publish promptly
  - lhcbdev keeps up with the nightly installation load
  - garbage collection runs are predictable
  - Clients are stable and rarely noticed in production workloads
- But of course nothing (useful) is so perfect that it can't be critiqued

Publication performance

- The `lhcb` and `lhcb-condb` instances perform well since moving to S3
- The lhcbdev use case is more demanding
  - Typically publish ~500 GB a day and plan to significantly increase this “soon”
  - Can easily average publishing 25 MB/s on each of the 6 release managers
  - Though performance degrades over time (easy to workaround, see CVM-2045)
  - Monitoring is important 🙂





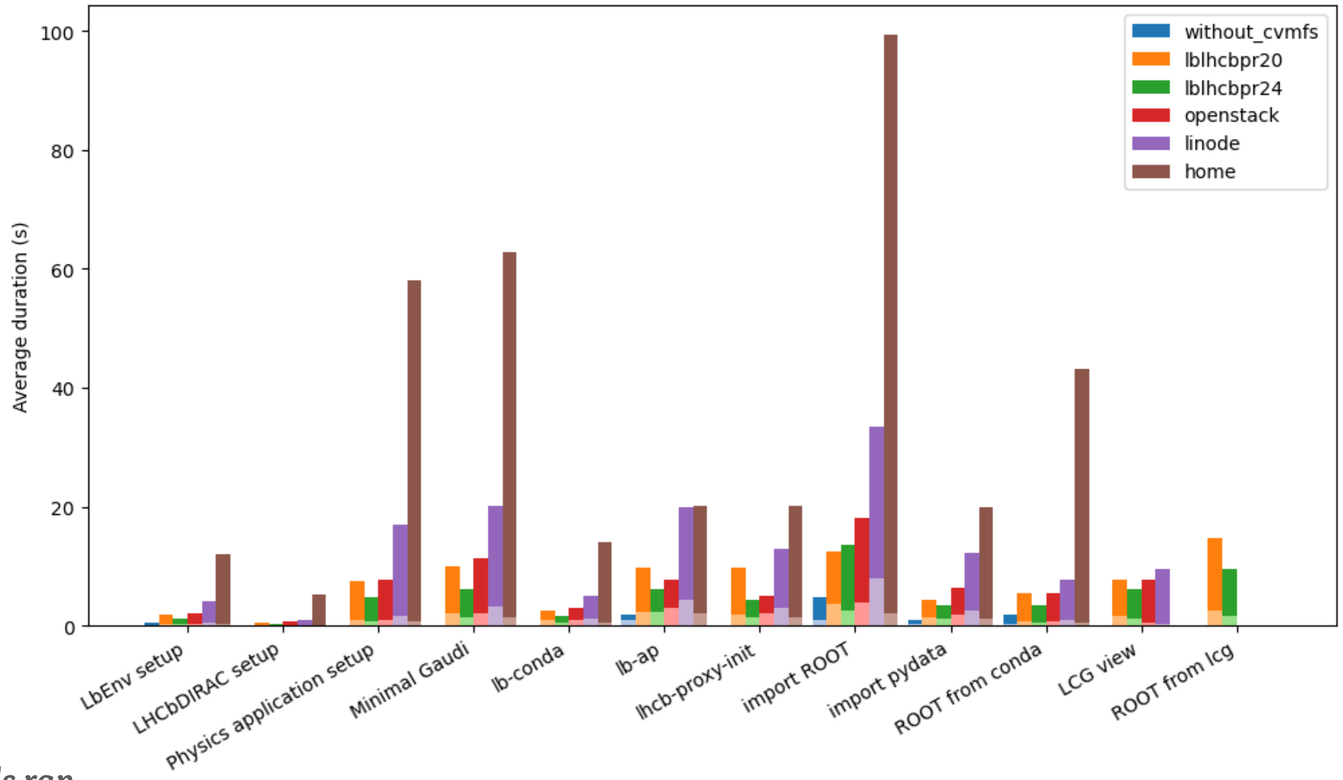
Client performance

- We often notice slowness in interactive terminal sessions
  - Underlying cause isolated to CVMFS

- Profile common LHCb tasks on various machines:
  - without_cvmfs (installation of software on local drive)
  - lblhcbpr20/lblhcbpr24 (bare metal)
  - openstack (VM at CERN)
  - linode (VM in London, UK)
  - home (desktop PC in Geneva)

- Dark region runs `wipecache` first
  - Clear kernel caches for “without_cvmfs” tests

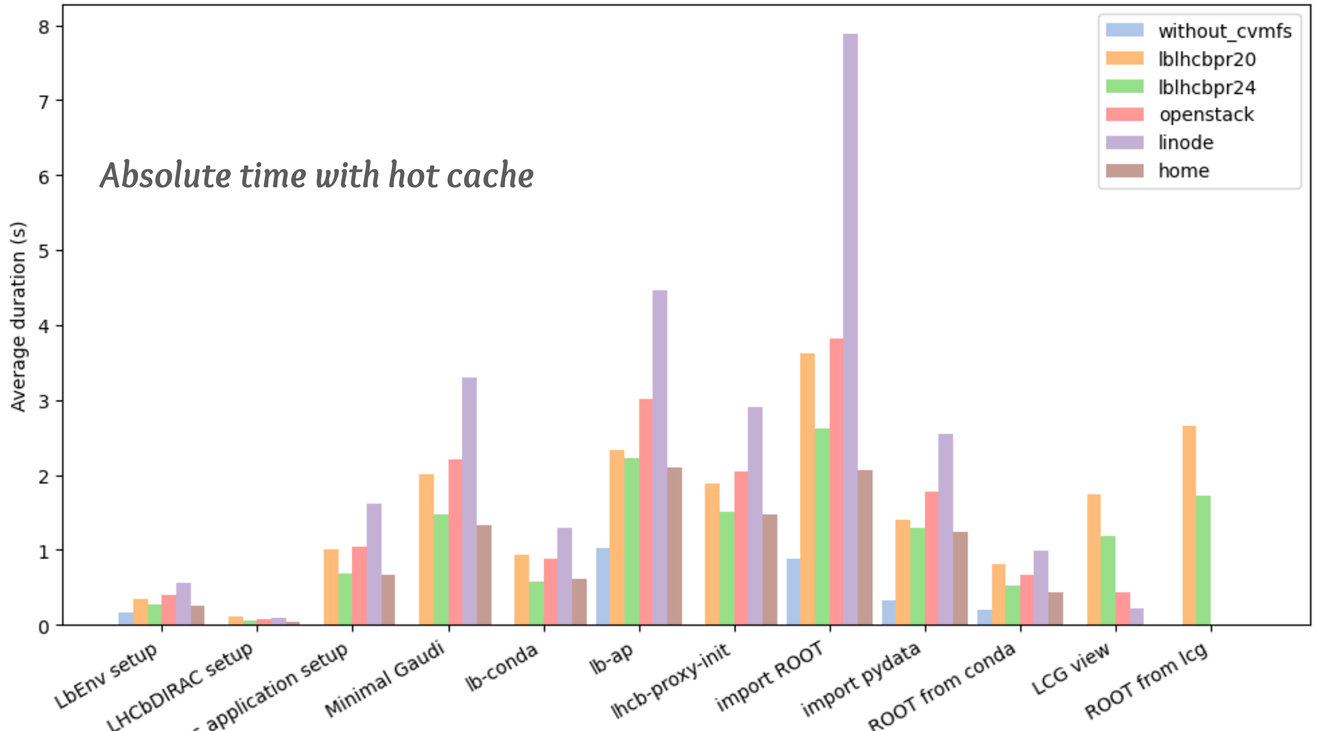
- Light regions skip `wipecache`

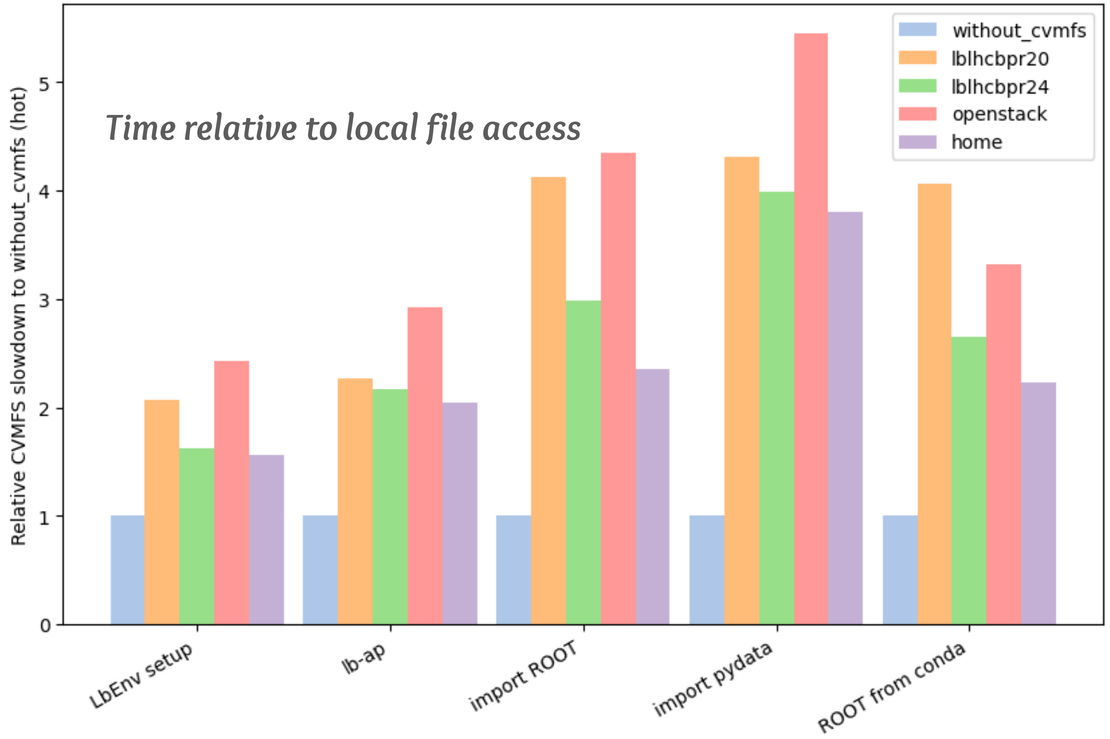


See backup for the exact commands ran

Client performance: Hot cache

- When running with a hot cache the performance is similar regardless of node
  - Linode is notably slower (small instance, likely lack of CPU)
- Still 3-5x slower than local file access
  - Common interactive tasks can take multiple seconds even when ran repeatedly

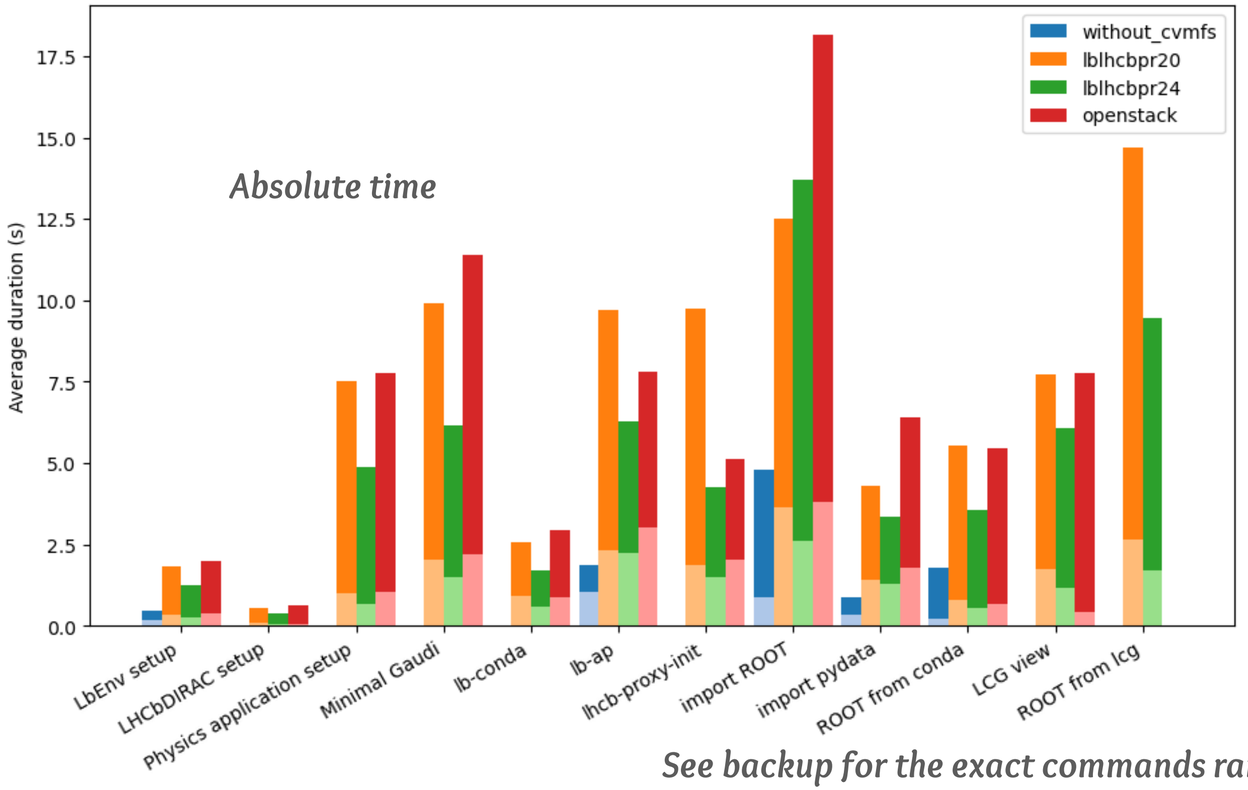




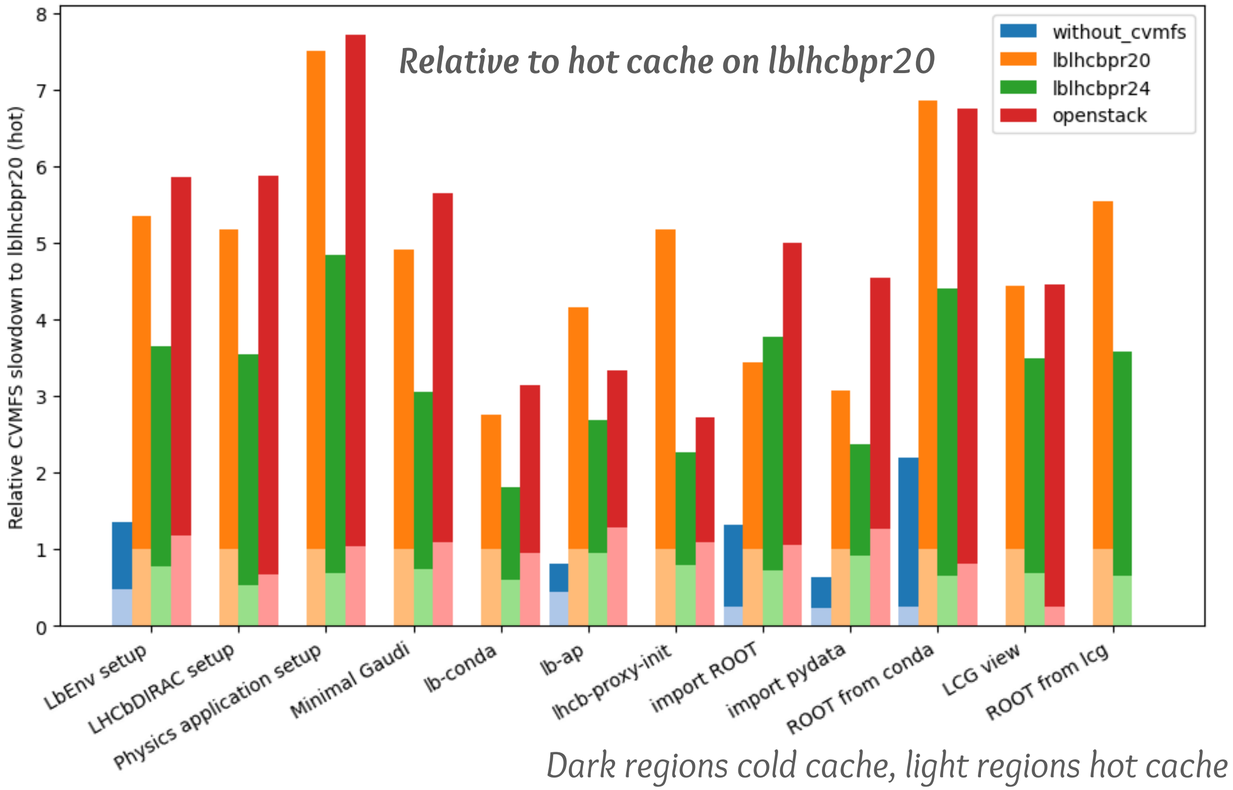
See backup for the exact commands ran

Client performance: Cold cache at CERN

- If the local machines cache is cold common tasks take 10+ seconds at CERN
  - Note these are averages, tail latency is much much worse



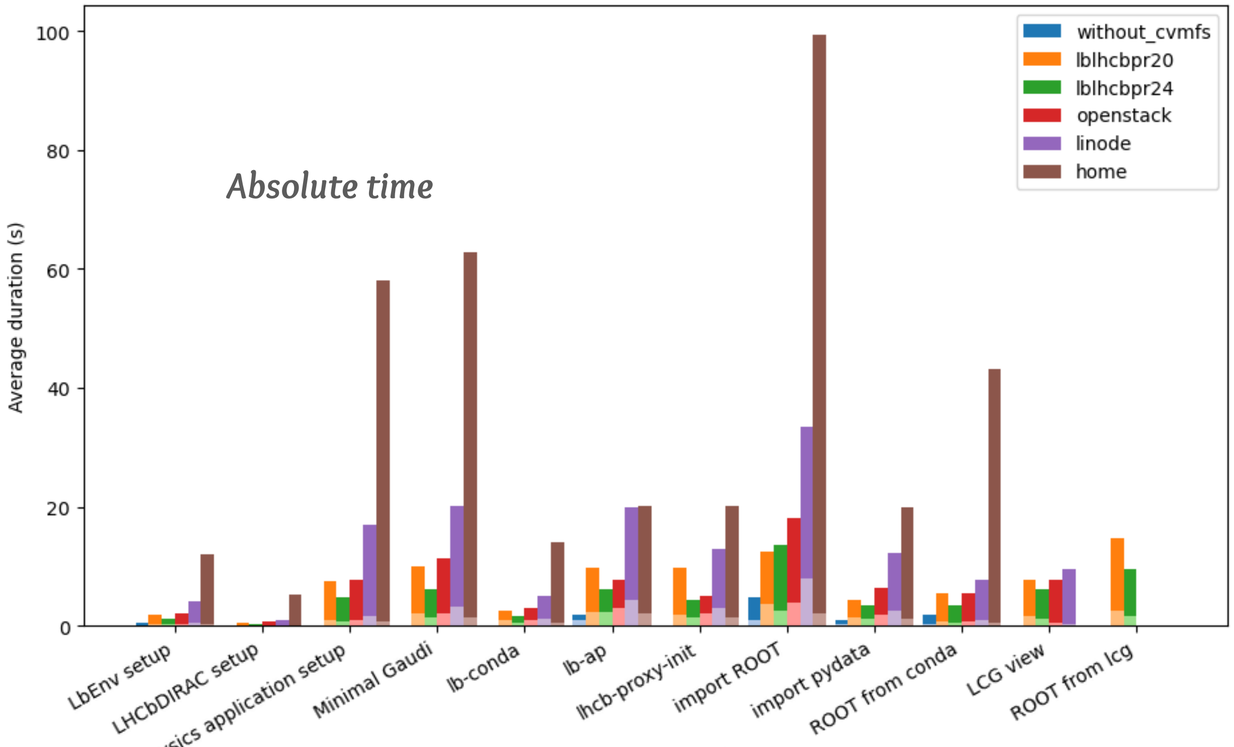
See backup for the exact commands ran



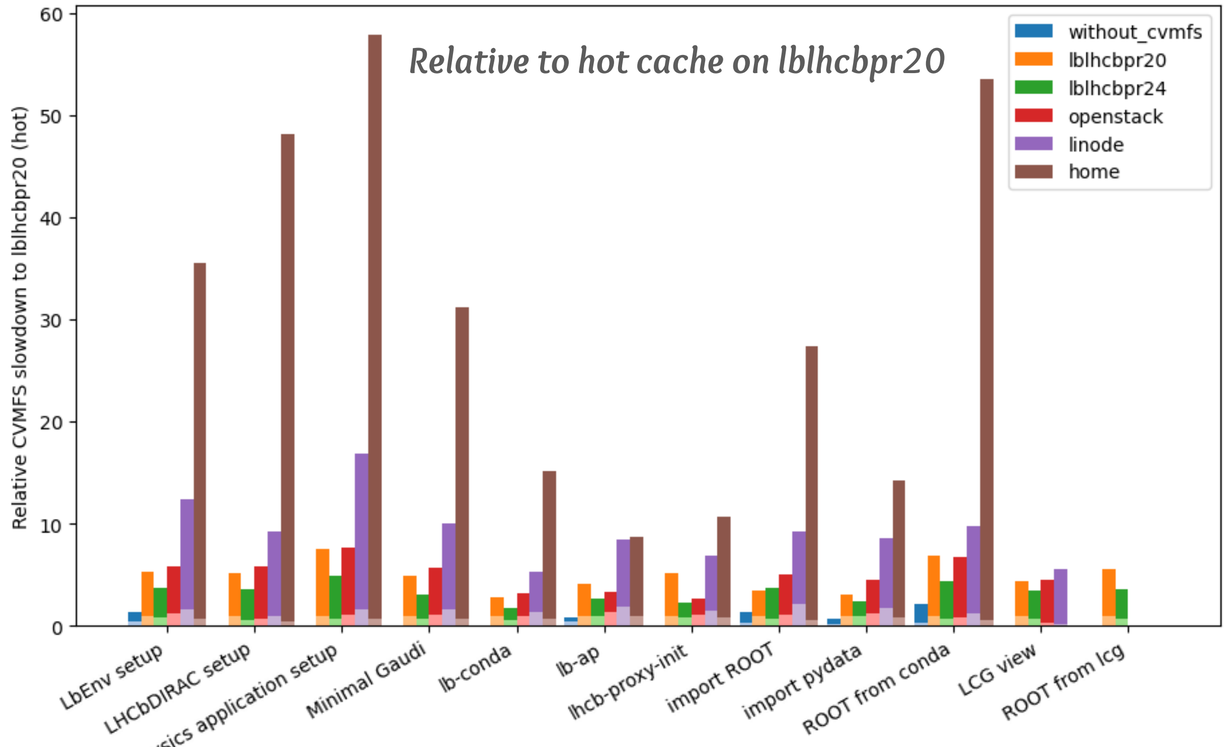
Dark regions cold cache, light regions hot cache

Client performance: Cold cache at CERN

- If the local machines cache is cold common tasks take 10+ seconds at CERN
  - Note these are averages, tail latency is much much worse
- Outside of CERN many tasks take multiple minutes with a cold cache



See backup for the exact commands ran



Dark regions cold cache, light regions hot cache

Conclusion

- CVMFS continues to scale well for everything we throw at it
- Generally happy with server performance
  - Occasional glitches with the distributed lhcbdev setup recovery is normally automatic
  - Will stress it considerably harder soon and propagation delay will become critical
- Quality of life improvements for clients would be welcome
- Thank you to everyone in the CVMFS team
  - Communication and support continue to be excellent


Questions?

Commands

Name	Commads
LbEnv setup	$LBENV_BIN/python -I -m LbEnv --sh --siteroot $LHCBSITE_ROOT
LHCbDIRAC setup	source /cvmfs/lhcb.cern.ch/lhcbdirac/lhcbdirac
Physics application setup	PATH=$LBENV_BIN:$PATH $LBENV_BIN/lb-run --disallow-containers --siteroot $LHCBSITE_ROOT Gaudi/v36r7 echo
Minimal Gaudi	PATH=$LBENV_BIN:$PATH $LBENV_BIN/lb-run --disallow-containers --siteroot $LHCBSITE_ROOT Gaudi/v36r7 gaudirun.py
lb-conda	$LBENV_BIN/lb-conda default echo
lb-ap	$LBENV_BIN/lb-ap --help
lhcb-proxy-init	$LBENV_BIN/lhcb-proxy-init --help
import ROOT	PATH=$CONDA_ENV_BIN:$PATH python -c 'import ROOT'
import pydata	$CONDA_ENV_BIN/python -c 'import numpy, matplotlib, pandas'
ROOT from conda	PATH=$CONDA_ENV_BIN:$PATH root -l -b -q -e '1-1'
LCG view	source $LCG_VIEW_ROOT/setup.sh
ROOT from lcg	source $LCG_VIEW_ROOT/setup.sh; root -l -b -q -e '1-1'

- Where:

```
LBENV_BIN=/cvmfs/lhcb.cern.ch/lib/var/lib/LbEnv/2541/stable/linux-64/bin
CONDA_ENV_BIN=/cvmfs/lhcbdev.cern.ch/conda/envs/default/2022-08-06_21-15/linux-64/bin
LHCBSITE_ROOT=/cvmfs/lhcb.cern.ch/lib
LCG_VIEW_ROOT=/cvmfs/sft.cern.ch/lcg/views/LCG_102/x86_64-centos7-gcc11-opt
```