CernVM-FS Software Installation Workflows

Jakob Blomer, Valentin Volkl (CERN, EP-SFT)

CERN/EESSI Exchange, 18 September 2020

Agenda

HEP Software Stacks

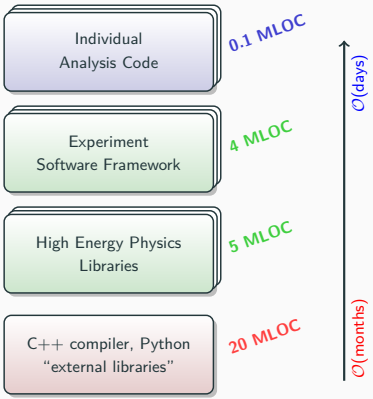
4 Installation Workflows

Spack Package Manager

HEP Software Stacks

Anatomy of the HEP Software Stack

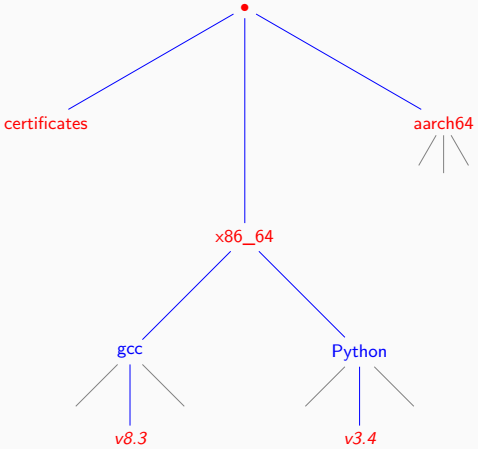
Typically 1-3 CernVM-FS repositories per experiment collaboration:
production software, nightly builds, conditions data



- $>$ 100 000 files per release
- Stack bootstrapped by C++ compiler
- Typical name includes information about OS, compiler, and build mode, e. g. `x86_64-centos7-clang10-opt`
- Tens of combinations for nightly builds
- Some leeway at the platform/stack boundary (glibc + X)

Partitioning of Meta-Data

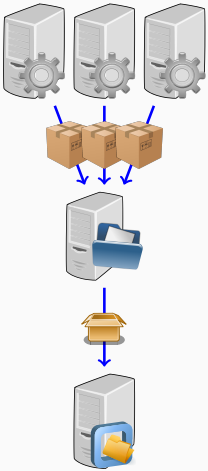
Well-designed directory layout critical for scalability and performance



- Locality by software version
- Locality by frequency of changes
- Partitioning up to software librarian, steering through `.cvmfscatalog` magical marker files
- Coarse-grained structure fixed by top-level `.cvmfsdirtab`

4 Installation Workflows

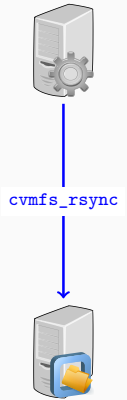
1. The Postscript Relocation Approach



1. Builders mount /cvmfs read-only
2. Builders produce new packages in scratch directory
3. Builders tar up (package) built products
4. Publisher untars packages into /cvmfs
5. A post installation script relocates packages
   - sed replacement, rpath adjustments, symlink relinking, ...
   - in principle package-specific but in practice a generic script works well

Example: /cvmfs/sft.cern.ch (also ATLAS, LHCb)

2. The rsync Approach



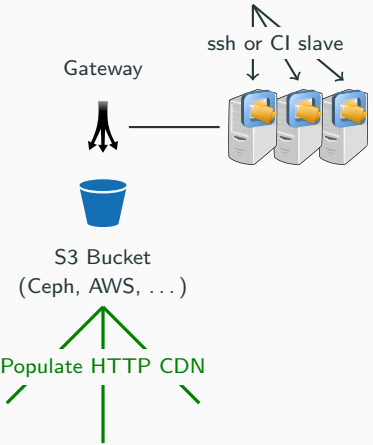
1. Builder mounts a local read-write copy of the /cvmfs tree
2. Builder changes / installs software in place
3. Publisher uses rsync to pull changes from the builder; the `cvmfs_rsync` command supports this workflow

- Non-trivial to maintain multiple, synchronized publishers

Example: /cvmfs/sw.hsf.org (also ALICE)

This workflow shall be simplified by the use of the CernVM-FS gateway, where every builder is a publisher at the same time

3. The Gateway Approach

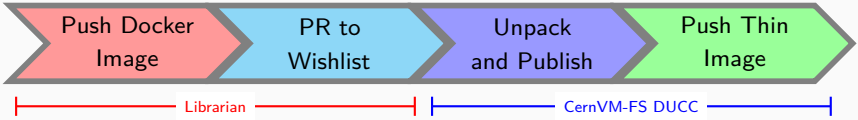


Coordinating Multiple Publisher Nodes

- Concurrent publisher nodes access storage through gateway services
- Gateway services:
  - **API** for publishing
  - Issues **leases** for sub paths
  - Receives change sets as set of **signed object packs**
- Bleeding edge deployment

Example: /cvmfs/projects.cern.ch
(multi-tenant repository)

3. The Container Approach



Wishlist https://github.com/cvmfs/images-unpacked.cern.ch

```
version: 1
user: cvmfsunpacker
cvmfs_repo: 'unpacked.cern.ch'
output_format: >
  https://gitlab-registry.cern.ch/unpacked/sync/$(image)
input:
   - 'https://registry.hub.docker.com/library/fedora:latest'
   - 'https://registry.hub.docker.com/library/debian:stable'
   - 'https://registry.hub.docker.com/library/centos:latest'
```

/cvmfs/unpacked.cern.ch

```
# Runtimes supporting flat root fs
/registry.hub.docker.com/fedora:latest -> \
  /cvmfs/unpacked.cern.ch/.flat/d0/d0932...
# Layer based container runtimes
/.layers/f0/1af7...
```

Runtime	CernVM-FS Support
Singularity	**native**
runc	**native**
podman	**native (pre-release)**
docker	**plugin**
containerd	**pre-release plugin**

Spack Package Manager

Backup Slides

Notification Service

Fast distribution channel for repository manifest: useful for CI pipelines, data QA

- Optional service supporting a regular repository
- Subscribe component integrated with the client, automatic reload on changes

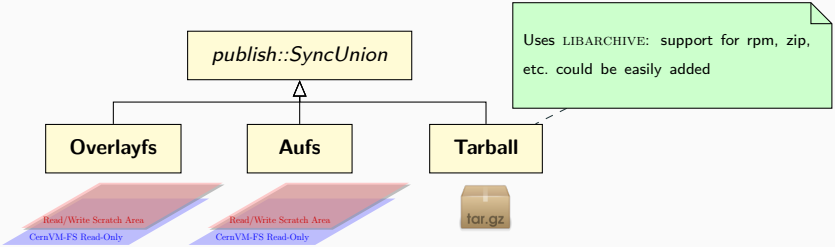
→ **CernVM-FS writing remains asynchronous but with fast response time in $\mathcal{O}$(seconds)**

Tarball Ingestion

Direct path for the common pattern of publishing tarball contents

```
$ cvmfs_server transaction
$ tar -xf ubuntu.tar.gz
$ cvmfs_server publish
```

```
$ zcat ubuntu.tar.gz | \
    cvmfs_server ingest -t -
```



Performance Example

Ubuntu 18.04 container – 4 GB in 250 k files: **56 s untar + 1 min publish** vs. **74s ingest**

Future plan: CernVM-FS Conveyor

A high-level abstraction of writing based on interdependent publication jobs.

```
$ ssh cvmfs-sft.cern.ch
$ cvmfs_server transaction sft.cern.ch /lcg/ROOT
$ tar -xf ROOT-6.18.tar.gz
$ post-install.sh
$ cvmfs_server publish
```

Current approach

```
{
  "repository": "sft.cern.ch",
  "path": "/lcg/ROOT",
  "payload": "https://root.cern.ch/ROOT-6.18.tar.gz",
  "script": "https://spi.cern.ch/post-install.sh",
  "uuid": "e7b67a2...",
  "dependencies": ["f61d...", "a00e...", "..."]
}
```

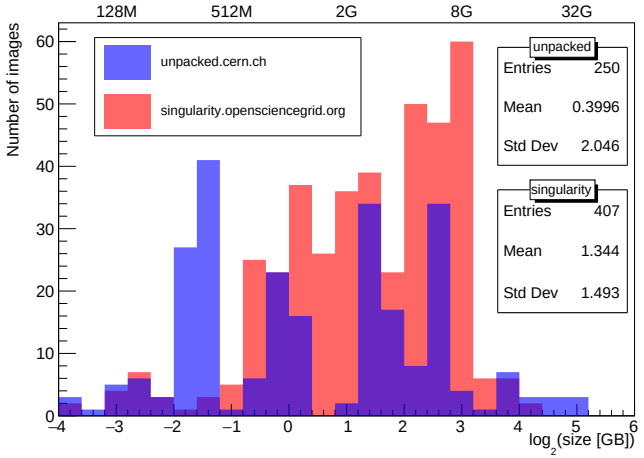
- Send jobs to Conveyor API
- Conveyor distributes work to multiple publisher nodes

Goal: liberate CI pipeline from handling `cvmfs_server` intrinsics.
Draws heavily from LHCb nightly build publishing system.

Container Image Sizes

Distribution of container images sizes in
`/cvmfs/unpacked.cern.ch` and `/cvmfs/singularity.opensciencegrid.org`

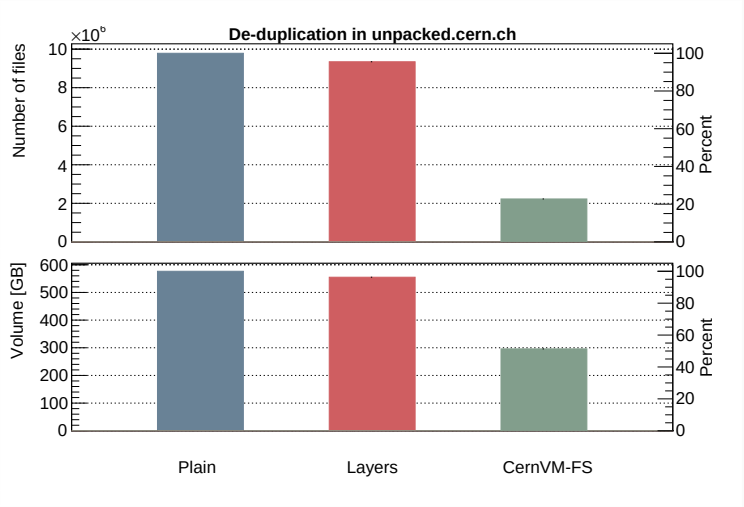
- **Likely to overflow the worker node scratch space with multi-gigabyte images**
- Interesting follow-up: distribution by image category (base, user, . . .) and creation date



De-duplication

Comparison of de-duplication efficiency between layers and file-based storage (CernVM-FS)

- **De-duplication works properly only on file-level granularity**
- Duplication occurs more often for smaller files
- Interesting follow-up: de-duplication in worker node caches



Worker Node Software Space

Observations from CERN lxbatch farm:

- Looked at 3 different images (ATLAS and CMS base images)
- Found on $>$ 250 worker nodes
- For each image:
  - 2 % to 9 % of the **image volume** in the worker node cache
  - Site-wide: $\sim$ 15 % of the volume cached
- Not yet including de-duplication effects in the worker node cache

$\rightarrow$ **$\times$10 to $\times$50 higher image distribution efficiency from /cvmfs hosted images compared to** `docker pull ...`